ГОСУДАРСТВЕННЫЙ НАУЧНЫЙ ЦЕНТР РОССИЙСКОЙ ФЕДЕРАЦИИ

ИНСТИТУТ ФИЗИКИ ВЫСОКИХ ЭНЕРГИЙ

98–32

На правах рукописи

Егошина Татьяна Васильевна

МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ФОРМИРОВАНИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ ОБРАБОТКИ ДАННЫХ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

05.13.11 - Математическое и программное обеспечение вычислительных машин, комплексов, систем и сетей

Автореферат
диссертации на соискание ученой степени
кандидата физико-математических наук

Протвино 1998

УДК 681.3.068

Работа выполнена в Институте физики высоких энергий (г.Протвино).

Научный руководитель – кандидат технических наук А.Д. Начинкин.

Официальные оппоненты: доктор технических наук В.А. Гадасин, кандидат физико-математических наук В.В. Сиколенко.

Ведущая организация – ЛВТА Объединенного института ядерных исследований (г. Дубна).

Защита диссертации состоится “_____” _____ 1998 г. в _____ часов на заседании диссертационного Совета К 034.02.01 при Институте физики высоких энергий по адресу: 142284, г.Протвино Московской обл..

С диссертацией можно ознакомиться в библиотеке ИФВЭ.

Автореферат разослан “_____” _____ 1998 г.

Ученый секретарь
диссертационного Совета К 034.02.01 В.Н.Ларин

Общая характеристика работы

Актуальность проблемы. Работа современных информационных систем (ИС), представляющих собой комплекс программных, информационных и технических средств, связана с множеством процессов сбора, хранения, обработки, передачи и использования данных. Действия человека, участвующего в работе такой информационной системы, требуют знания не просто перечня входных и выходных документов, программных модулей, таблиц или файлов базы данных (БД), но и знания внутренних взаимосвязей всех системных объектов. Иными словами, от человека требуется знание и соблюдение определенной технологии работы ИС. В связи с этим возникает необходимость решения, по крайней мере, двух проблем: 1) автоматизированного построения описания технологических процессов обработки данных, 2) использования полученного описания для анализа работы ИС и принятия решений на этапе эксплуатации системы.

Хотя в последнее время на рынке инструментальных средств появились мощные программные продукты *(Delphi, Oracle CASE* и др.), в силу экономического фактора немногие предприятия могут позволить себе выполнять разработку информационных систем с использованием Oracle CASE и СУБД Oracle. Кроме того, на предприятиях и в организациях до сих пор функционирует много информационных систем, разработанных в среде desktop СУБД, таких как *Clipper, FoxBase, FoxPro, Access* и т.п. на ПЭВМ с процессорами 286/386/486. Эффективность использования информационной системы зависит от того, насколько оперативно удается переделать, заменить или адаптировать систему к новым условиям работы. Возможным путем решения проблемы является формализованное отображение в ЭВМ профессиональных знаний разработчиков и автоматизация формирования знаний экспертов.

В связи с этим создание математического и программного обеспечения формирования технологических процессов обработки данных в информационных системах, дающего возможность разработчику формально описывать технологию работы ИС,

а администратору получать интенсивные сведения о структуре базы данных и технологии работы системы, является актуальной.

Цель диссертационной работы заключалась в создании инструментальных средств описания технологии работы информационной системы экспертом — разработчиком для организации эффективного сопровождения информационных систем.

Научная новизна работы:

1. Разработан метод формализации описания технологических процессов обработки данных в информационных системах:
   а) предложен способ разбиения информационных систем на технологические компоненты, для которых введены понятия класса, типа и кластера объектов;
   б) разработан продукционный язык описания технологических процессов (ЯОТП) обработки информации;
   в) реализован компилятор ЯОТП, выходом которого является объектная FRL–программа.
2. Предложен автоматизированный способ построения сети фреймов, являющейся своеобразной технологической базой знаний (БЗ).
3. Реализованы процедуры интерпретатора, работающие на сети фреймов и позволяющие получать ответы на запросы администратора ИС по технологии работы системы.

Практическая ценность:

1. Предложенное в работе решение формирования технологических процессов обработки данных, являющееся постстадией разработки информационных систем, ориентировано на ИС, реализованные в среде desktop СУБД типа Clipper, FoxBase, FoxPro, Access и т.п. и функционирующие на ПЭВМ с процессорами 286/386/486, что дает возможность использовать в работе весь парк ПЭВМ.

2. Созданный язык описания технологических процессов обработки информации и разработанный компилятор ЯОТП позволяют строить из конструктивных элементов описания технологий работ информационных систем.

3. Показана возможность разделения процессов описания технологии информационной системы экспертами–разработчиками и работы интерпретатора. Создание описания технологических процессов обработки данных можно осуществить на более мощной вычислительной технике. Скомпилированный файл описания технологии работы системы является мобильным и может быть перенесен на любую, менее мощную ПЭВМ, где установлен FRL–интерпретатор.

4. Результаты диссертации были апробированы при описании технологии и получении ответов на запросы по работе автоматизированной системы ЦМСЧ–66 “Учет и анализ заболеваемости на основе сбора статистических данных с предприятий города”.

На защиту выносится созданный комплекс инструментальных средств для организации эффективной работы информационной системы, включающий:

1. Разработку и реализацию способа формирования технологических процессов обработки данных в информационных системах.
2. Создание языка описания технологических процессов обработки информации.
3. Разработку компилятора ЯОТП.
4. Разработку и реализацию в процедурах интерпретатора алгоритмов выполнения запросов администратора информационной системы.

Апробация работы. В диссертации изложены результаты исследований, выполненных в Институте физики высоких энергий и аспирантуре ЦНИИАтоминформ. Результаты докладывались на семинарах в ОАСУ, ОАПЭС и ОМВТ ИФВЭ, в ЦНИИАтоминформ, на Международной конференции "Нечеткая логика, интеллектуальные системы и технологии", на III Международной конференции "Развитие и применение открытых систем", на Всесоюзном научно–техническом семинаре "Использование программных средств ПЭВМ для автоматизации учрежденческой деятельности", на семинаре в МДНТП и на конференции "Информационные системы в науке–95", организованной РФФИ и РАН.

Основные результаты диссертации опубликованы в работах [1–6].

Структура диссертации. Диссертация состоит из введения, трех глав, заключения, приложения и библиографии, включающей 64 работы. Общий объем диссертации 90 страниц, включая 24 рисунка и 1 таблицу.

Содержание работы

В первой главе рассматриваются существующие методы автоматизации проектирования технологических процессов. Обсуждаются технологические аспекты функционирования информационных систем. Подразумевается, что ИС обладает стандартной схемой обработки данных с вполне детерминированными шагами. Процесс эксплуатации системы тесно связан с программированием через технологию работы информационной системы.

Одной из основных проблем определенного круга работающих ИС является понимание и знание информационных потоков в системе. Построение описаний путей прохождения данных в работающих ИС можно рассматривать и как составление инструкций для персонала, участвующего в эксплуатации информационной системы или получающего результаты ее работы. Под технологией работы информационной системы будем понимать взаимосвязанную по входам и выходам последовательность технологических операций обработки данных, выполнение которых приводит к достижению требуемого результата — бессбойному функционированию ИС. В результате выполнения некоторой технологической операции могут быть получены компоненты, являющиеся промежуточными, используемыми в качестве входа следующей технологической операции.

Любая сложная информационная система содержит несколько точек ввода и вывода информации из ЭВМ. И, конечно, ЭВМ или комплекс ЭВМ является центральным звеном организации технологии информационных процессов. Кроме того, любая сложная ИС — это совокупность информационно–согласованных подсистем. Информационная согласованность означает, что работа подсистем обеспечивается информационно–согласованными программами, т.е. результаты работы одних программ будут исходными данными для других.

Технологический информационный процесс определяет действия, их последовательность, исполнителей, программы, средства и ресурсы, необходимые для выполнения этих действий.

Для информационных систем важно знать, какие данные откуда берутся, куда попадают и с помощью чего. Поэтому разбиение информационных систем на технологические компоненты происходит по пути следования данных. Основная задача при этом состоит в выделении стабильных составляющих информационной системы, разбиении технологии работы системы на компоненты, их объявлении и описании информационных связей компонентов с помощью правил.

Изучение большого круга информационных систем, работающих на основе desktop СУБД, позволило выявить несколько классов объектов. Каждый класс в свою очередь допускает несколько типов объектов. Понятие *типа* введено для объектов, сходных по описанию, и в тип вынесен перечень основных правил. Таким образом, с типом связано описание обобщенных правил. Метод обработки объектов одного класса не зависит от типа. А это значит, что *класс* определяет фундаментальный метод работы с объектами нескольких типов. Множество, объединяющее близко расположенные объекты разных типов с одним и тем же именем, но из разных классов, будем называть *кластером*. Кластеры имеют большое значение при выборе имени нового объекта в правилах перехода. При обобщенном описании правил возникает проблема именования следующего объекта, имя которого нельзя назвать точно, иначе не получится обобщенного правила в типе. Поэтому новое имя объекта образуется по умолчанию на основании имени кластера.

Можно назвать, например, такие классы объектов: класс объектов “Программы” разных типов, класс объектов “Базы данных” разных типов, класс объектов “Рабочие места” разных типов, класс объектов “Принтеры” разных типов и др. Причем класс “Программы” включает в себя не только программные модули, но экранные формы и отчеты.

При графическом изображении связи каждый объект будет представляться узлом, а ссылки между объектами — дугами, направленными от начального объекта к конечному.

Граф на рис.1 показывает направление движения информации от первого объекта (object_1) ко второму (object_2). Запишем переход информации из object_1 в object_2 в виде

object_1 —> object_2

Тогда для каждого класса объектов графически можно изобразить все этапы обработки данных в информационной системе.

Рис. 1. Граф связи объектов.

Объекты — это конкретные компоненты технологии работы системы. Правила — некие состояния информации во время ее движения от объекта к объекту. Переход из одного состояния в другое можно записать в виде

$$\mathbf{r_1} \text{ —> } \mathbf{o.r_2}$$

Здесь **o** — объект, $\mathbf{r_1}$ и $\mathbf{r_2}$ — правила. При отсутствии $\mathbf{r_2}$ в правой части выражения считается, что состояние информации не меняется; при отсутствии **o** в правой части происходит движение информации в рамках одного объекта.

При переходе информации в конкретный кластер немаловажным становится однозначное определение имени объекта. Имя объекта может быть составным typename$objectname, включающим указание типа объекта typename и собственное имя объекта objectname. Связи, описанные для определенного типа объектов одного класса, верны для любого объекта данного типа из этого класса.

Предложенный способ записи правил перехода служит отображением структурных и динамических свойств объекта автоматизации.

Итогом разработки средств формирования технологических процессов обработки данных должны стать инструментальные средства для администратора информационной системы, с помощью которых он может получать сведения и осуществлять анализ структуры информационной базы и технологии работы системы. В качестве конечного программного продукта, с которым предстоит работать администратору, выступает интерпретатор запросов, осуществляющий поиск ответа на запрос в некотором описании информационной системы. Приведем примеры возможных запросов администратора в неформализованном виде: в какую таблицу БД заносится информация из заданного входного документа? Или, каков путь (трасса) следования порции информации из конкретной таблицы БД?

Для создания базы данных интерпретатора — описания структуры информационной системы и технологии ее работы, которую можно назвать технологической базой знаний информационной системы, — необходимо разработать язык описания технологических процессов обработки информации и компилятор ЯОТП, позволяющий преобразовывать формализованное описание работы информационной системы в объектную программу для построения сети фреймов.

На рис.2 приводится схема работы инструментальных средств формирования технологических процессов обработки данных в ИС.

Рис. 2. Схема работы средств формирования технологических процессов обработки данных.

В разделе 1.6 первой главы формулируется постановка задачи, включающая:
1. Разработку грамматики, синтаксиса и компилятора ЯОТП.
2. Создание интерпретатора запросов пользователей.

Во второй главе описываются грамматика и синтаксис предлагаемого языка описания технологических процессов обработки информации. При разработке языка описания технологических процессов обработки информации учитывалось, что он должен быть удобен и требовал бы при его использовании минимальных усилий не только от квалифицированных программистов, но и от конечных пользователей. С другой стороны, ЯОТП должен быть прост для реализации на стандартных средствах ЭВМ и не должен предъявлять жестких требований к другим ресурсам вычислительных систем.

Грамматика ЯОТП относится к классу контекстно–свободных грамматик (КС–грамматик) и формализует большую часть правил, предназначенных для описания синтаксической структуры ЯОТП. В разделе 2.1.2 приводится синтаксис ЯОТП в форме Бэкуса–Наура (БНФ).

<operators> ::= <operator» | <operators> <operator>

<operator> ::= </*empty*/> | <spc> | <typeheader> |
<objectheader> | <ruleline>
<typeheader> ::= TYPE <name> <description>
<objectheader> ::= <objecthdr> <objref> <description>
<objecthdr> ::= OBJECT <name> TYPE <name>
<objectref> ::= </*empty*/> | REF <name>
<ruleline> ::= <spc> <name> <GT> <dest> <description>
<spc> ::= SPACE | <spc> SPACE
<dest> ::= <fname> <drname> | '(' <result> ')'
<fname> ::= </*emty*/> | <name> | <name> '/' <name>
<drname> ::= </*empty*/> | ' . ' <name>
<result> ::= </*empty*/> | <name>
<description> ::= </*empty*/> | <spc> <text> | <text>
<name> ::= <буква> | <буква> <цифра> |
<name> <буква> | <name> <цифра> |
<name> <спец_символ> <буква> |
<name> <спец_символ> <цифра>
<text> ::= ' ; ' <текст>
<GT> ::= —>
<SPACE> ::= <TAB>
<буква> ::= A | B | C | ... | Z | A | Б | В | . . . |
Я | a | b | c | ... | z | а | б | в | . . . | я
<цифра> ::= 1 | 2 | 3 | ... | 9 | 0
<спец_символ> ::= _ | –

Грамматика ЯОТП включает следующие виды операторов:

1. *Операторы описания типа объекта*

TYPE <идентификатор типа> [; комментарий]

2. *Операторы описания объекта*

OBJECT <имя объекта> TYPE <идентификатор типа>
[REF <идентификатор типа>] [; комментарий]

3. *Операторы перехода*

<имя правила> —> <имя нового правила> [; комментарий]

Ключевые слова TYPE, OBJECT, REF (ссылка на другой тип объекта) являются зарезервированными и не могут быть использованы в качестве имен. Комментарий, начинающийся знаком “точка с запятой”, включает любые символы русского и латинского алфавитов, скобки, табуляторы, пробелы, слеши и другие знаки, за исключением знака “новая строка”, которым заканчивается каждый оператор языка. Знак —> (“стрелка”) используется в записи правил для обозначения

перехода информации из одного состояния в другое. В составном имени объекта задействован знак / (“слеш”). В идентификаторах типов и объектов, в именах правил могут использоваться любые буквы русского и латинского алфавитов, цифры и специальные символы (– , _).

В разделе 2.1.3 описывается компилятор ЯОТП. В основе всех процессов обработки языков лежит теория автоматов. Процесс компиляции рассматривается как взаимодействие небольших процессов. Согласно упрощенной модели компиляция осуществляется тремя последовательно соединенными блоками: лексическим блоком, синтаксическим блоком и генератором выхода. Эти три блока имеют доступ к общему набору таблиц, куда можно помещать долговременную или глобальную информацию о программе. Лексический блок разбивает цепочку символов на слова (лексемы), из которых она состоит. Синтаксический блок переводит последовательность лексем, построенную лексическим блоком, в последовательность атомов, которая более непосредственно отражает порядок выполнения операций.

Немаловажным вопросом при построении компилятора является определение, что компилятор должен выдавать в качестве выхода. Поэтому на первом этапе построения компилятора должен быть получен ответ на этот вопрос. Так как для получения сведений по технологии информационной системы был выбран фреймовый метод представления знаний, то на выходе компилятора ЯОТП будут получены команды для генерации фреймов, содержащих описание технологии работы информационной системы.

Для однопроходного компилятора ЯОТП управление передается из блока в блок всякий раз, когда требуется или когда производится лексема или атом.

При построении компилятора ЯОТП использованы сервисные программы построения компиляторов, такие как генератор синтаксических анализаторов BISON — прототип сервисной программы YACC, и генератор программ лексического анализа FLEX — прототип генератора лексических анализаторов LEX.

Компилятор ЯОТП реализует так называемый LALR(1)–разбор, являющийся модификацией одного из основных методов разбора “снизу вверх” — LR(k)–разбора, что означает чтение входных символов слева направо и использование правостороннего вывода. Индекс в скобках показывает число предварительно просматриваемых лексических единиц.

Любой разбор по принципу “снизу вверх” (или восходящий разбор) состоит в попытке приведения всей совокупности входных данных (входной цепочки) к начальному символу грамматики путем последовательного применения правил вывода.

В каждый момент грамматического разбора компилятор находится в некотором СОСТОЯНИИ, определяемом предысторией разбора, и в зависимости от очередной лексемы предпринимает то или иное действие для перехода к новому состоянию. Различают два типа действий: СДВИГ означает чтение следующей входной лексемы, и СВЕРТКА, т.е. применение одного из правил подстановки для замещения нетерминалом последовательности символов, соответствующей правой части правила. Процедура синтаксического анализа осуществляется на основе таблиц, которые

для каждого из состояний определяют тип действий компилятора и номер следующего состояния в соответствии с каждой из входных лексем.

Лексический блок компилятора, выполняющий первую стадию компиляции, читает строки компилируемой программы, выделяет лексемы и передает их на дальнейшие стадии компиляции. Хотя лексический анализ по своей идее прост, тем не менее эта фаза работы компилятора часто занимает больше времени, чем любая другая. Частично это происходит из–за необходимости просматривать и анализировать исходный текст символ за символом. Иногда даже бывает необходимо вернуть прочитанный символ во входной поток с тем, чтобы повторить просмотр и анализ. Происходит это потому, что часто бывает трудно определить, где проходят границы лексемы. Единственный способ преодолеть это затруднение — просмотр полученной цепочки символов назад и вперед. Процесс просмотра входного потока можно рассматривать как движение влево и вправо рамки над цепочкой символов. При этом анализируется только тот символ, который охвачен рамкой (рис.3).

TYPE PREDPRI - PRINTER

⇐ ⇒

Рис. 3. Пример просмотра входного потока ЯОТП–программы.

Анализ заключается в определении соответствия рассматриваемой последовательности символов некоторому регулярному выражению.

Идея, что в большинстве языков программирования общая структура символов достаточно одинакова, и поэтому можно иметь один общий алгоритм сканирования для всех языков, позволяет обратиться к генератору FLEX, который по описанию символов языка программирования готовит таблицы, соответствующие этому языку (рис.4). Как только генератор построил таблицы, сам он становится ненужным. Поэтому можно отбросить все, что на рисунке расположено слева от двойной линии.

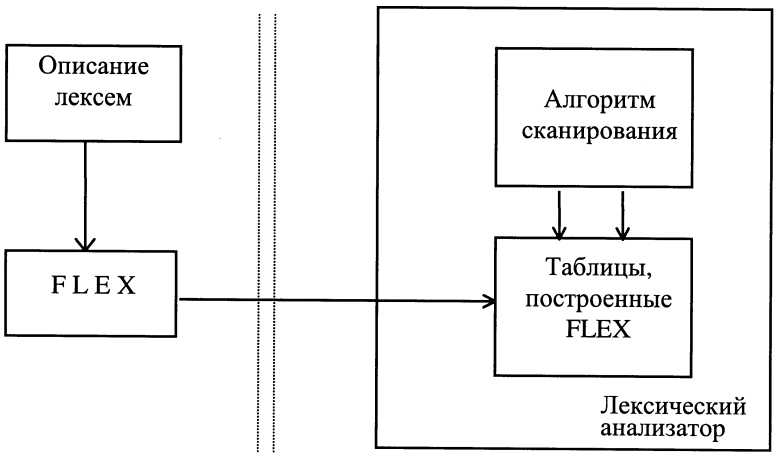

Рис. 4. Генератор FLEX и лексический анализатор.

Описание символов задается определенными правилами, каждое из которых состоит из двух частей:

<регулярное выражение> <действие>

По регулярным выражениям, содержащимся в левой части правил, FLEX строит детерминированный конечный автомат.

Действия в правилах FLEX–программы выполняются, если правило активно, и если автомат распознает цепочку символов из входного потока как соответствующую регулярному выражению данного правила. Таким образом, лексический анализатор осуществляет ввод из входного ряда очередных символов до выявления конструкции, соответствующей одной из лексем, и возвращает номер типа этой лексемы и, когда это необходимо, значение лексемы.

Для построения синтаксического анализатора необходимо задать грамматические правила, определяющие все допустимые входные конструкции и связанные с этими конструкциями действия по обработке входного потока. Такие правила имеют вид

<имя нетерминального символа>: определение {действие};

Здесь ":" и ";" — специальные символы. Правая часть правила — определение — представляет собой упорядоченную последовательность нетерминальных символов и лексем. Нетерминальные символы заданы именами, а лексемы — именами или литералами. Именами нетерминалов считаются все имена, не объявленные именами лексем. Имя каждого нетерминала обязательно стоит в левой части хотя бы одного правила. Отметим специфические виды правил: пустое правило и леворекурсивное правило. Пустое правило вида

<имя нетерминального символа>: ;

в сочетании с другими правилами, определяющими тот же нетерминальный символ, позволяет указать на необязательность вхождения соответствующей конструкции. Приведем пример такого правила для ЯОТП:

```
objref:     /* empty*/    { $$=NULL } | REF name       { $$ = $2 } ;
```

Псевдопеременные $$ и $2, появившиеся в правой части правил примера, служат для связи между действиями, а также между действиями и лексическим анализатором. Переменная $i соответствует значению i–ого элемента правой части правила. Переменная $$ соответствует значению образованного в результате свертки нетерминала.

Леворекурсивные правила, используемые при построении ЯОТП, имеют вид

<имя нетерминала>: <имя нетерминала>
<многократно повторяемый фрагмент>;

Синтаксический анализатор вызывает лексический блок, когда необходима новая лексема, и вызывает семантическую процедуру (действие), когда конструкция разобрана. Кроме всего, семантические процедуры выполняют генерацию фрагментов объектной программы, которая получается в виде функций языка LISP и процедур FRL. В однопроходном компиляторе этапы подготовки и генерации команд растворяются в семантических программах.

В разделе 2.2 главы описывается следующий этап автоматизированного формирования технологических процессов обработки данных в информационных системах, включающий использование администратором ИС откомпилированной ЯОТП–программы для построения сети фреймов и получение с помощью интерпретатора сведений по структуре БД и технологии работы ИС.

Анализ существующих программных средств показал, что в наибольшей степени для решения задачи получения ответов на запросы подходят методы представления и извлечения знаний, используемые в интеллектуальных системах. Был выбран фреймовый подход, связанный с попытками совместить строгость формальных систем с простотой и гибкостью семантических сетей.

Для представления рассматриваемого типа данных, каким являются знания, требуется не массив информации, а многоуровневая система в виде дерева, позволяющая производить выводы, генерацию, конструирование решения, которое в явном и готовом виде не содержится в системе.

Наиболее известным, развитым и доступным средством оказался интерпретатор языка FRL, который является открытой системой с точки зрения механизмов обработки объектов, что позволяет при переходе от одной проблемной области к другой дополнять его необходимыми процедурами. Язык FRL реализован как библиотека над языком LISP. Одним из основных понятий FRL, так же как и LISP, является понятие *списка*. S–выражения LISP, составленные из атомов и скобок, характерны и для FRL. Атом или список будем называть *термом*.

Для представления списков произвольного вида в LISP используется конструкция точечной пары. В FRL введено понятие поименованного списка с ассоциативным доступом — списка, к которому возможен доступ по первому элементу — имени *фрейма*. Все подструктуры фрейма представлены такими списками, вложенными друг в друга. Соответствующие подструктуры фрейма называются: *слот, аспект, данное, комментарий, сообщение*.

Модель технологии работающей информационной системы образована системой фреймов и правил перехода. Рассматриваемая предметная область представляется сетью фреймов объектов — компонентов. Для каждого объекта справедливы правила перехода

$$\mathbf{r_1} \longrightarrow \mathbf{o.r_2}$$

Здесь $\mathbf{o}(bject)$ соответствует фрейму, а $\mathbf{r_1}(ule)$ и $\mathbf{r_2}(ule)$ — слотам фреймов. Если в выражении отсутствует правое $\mathbf{o}$, то используется имя текущего фрейма. Если в записи выражения отсутствует $\mathbf{r_2}$, то используется имя слота, соответствующего значению $\mathbf{r_1}$, причем $\mathbf{r_1}$ в выражении относится к текущему фрейму.

На рис.5 схематично показано соответствие между объектами и слотами фреймов.

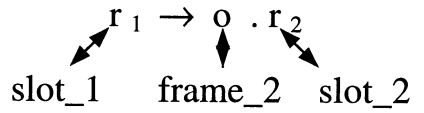

Рис. 5. Соответствие между объектами и слотами фреймов.

Полученное после компиляции описание технологии информационной системы используется для построения сети фреймов, которое представляет собой граф, образованный наложением сети фреймов объектов–компонентов, связей группирования и правил перехода. Соответственно вершины сети могут отображать объект, тип или правило.

Сеть фреймов обеспечивает:

- накопление и использование знаний и опыта разработчиков информационной системы, выступающих в качестве экспертов;
- сокращение времени получения ответа на вопрос в результате использования знаний специалистов в их отсутствии;
- увеличение точности нахождения ответа на вопрос в отсутствие разработчика;
- возможность эксплуатации системы пользователем — непрограммистом.

Сети на фреймах в FRL реализуются посредством указания имен фреймов в качестве значений терминалов. В процессоре FRL имеется ряд встроенных механизмов, называемых механизмами наследования, обеспечивающих работу с сетью.

В системе FRL возможно несколько типов наследования:

- наследование по умолчанию, когда запрос на данные из аспекта $VALUE дополняется запросом в аспект $DEFAULT;
- наследование по AKO–иерархии, когда запрос автоматически распространяется на фреймы, лежащие выше по AKO–иерархии;
- косвенное наследование, при котором запрос дополняется данными из специально указанного фрейма.

Текущая ситуация запроса описывается подграфом активных в настоящий момент узлов и связей сети фреймов.

Основные типы запросов следующие:

1. Получить конечную точку (или конечное значение) следования заданной порции информации.

2. Получить путь (трассу) следования заданной порции информации.

3. Выдать набор исходных данных, достаточных для получения требуемого результата в заданном пункте (объекте).

4. Получить конечный пункт или трассу пути следования информации при неполном задании исходных данных.

5. Получить значения функций при определении трассы следования информации.

Рассмотрение вопроса вывода решений проводится применительно к способу представления технологии информационной системы. Таким образом, после организации сети фреймов проблема извлечения из нее неявной информации при обработке запроса является основной.

Формирование правильного по смыслу ответа достигается путем механизмов сопоставления фреймов, построенных из запроса, с фреймами из сети и получения в результате этого фреймов, в которых все слоты заполнены. Поиск фрейма в системе делается с помощью ключей, выделенных из запроса. После обработки найденного фрейма через слот AKO осуществляется переход на более высокий уровень иерархии фреймов и вновь применяются слоты нового фрейма для заполнения слотов фрейма, найденного по запросу. В ходе сопоставления может возникнуть конфликт информации, для его устранения используется комментарий. Предпочтение может отдаваться информации из запроса. С помощью комментария разделяются семантический (из запроса) и прагматический (из сети фреймов) источники информации. Первый имеет больший приоритет. Однако не все компоненты слотов могут быть изменены. Аспект $VALUE в слоте фиксирует значение и делает его неизменным, а аспект $DEFAULT позволяет менять значение в слоте на основе запроса. То, что описано как $VALUE, никогда не подлежит сопоставлению со значениями из запроса и позволяет организовать, например, фиксированную связь в иерархии фреймов. Значение, описанное в $DEFAULT, всегда заменяется на соответствующую информацию из запроса, если она в нем есть и не является предикатом. Комментарии, связанные со значением, позволяют выполнить вывод на основе присоединенных процедур и наследования свойств.

В разделе 2.3 обоснован выбор средств программирования. Описываются программы, реализующие компилятор и процедуры интерпретатора. Приводится информация о времени работы и объеме программ. Выбор сервисных средств построения компиляторов, таких как BISON и FLEX, способствовал ускорению и упрощению разработки компилятора ЯОТП на языке C. Открытая архитектура FRL–интерпретатора позволила дополнить его функциями для реализации запросов.

Третья глава посвящена апробации предложенного подхода в системе сбора данных по заболеваемости.

В первом разделе главы указывается, что созданные инструментальные средства формирования технологических процессов обработки данных применимы к автоматизированным системам управления. Во втором разделе предлагается разбиение системы сбора данных по заболеваемости на технологические компоненты. В разделах 3.3 и 3.4 рассматривается описание технологии работы системы с помощью ЯОТП и компиляция ЯОТП–программы. В пятом разделе показано использование

FRL–интерпретатора для выполнения запросов пользователя. Приводятся примеры запросов и выводы ответов на экран компьютера.

В Заключении сформулированы основные результаты диссертации, состоящие в следующем: предложены, разработаны и апробированы инструментальные средства для организации эффективной работы информационной системы.

Полученные в ходе разработок математического и программного обеспечения результаты имеют самостоятельное научное и практическое значение.

1. Предложен способ разбиения информационных систем на технологические компоненты, для которых введены понятия класса, типа и кластера объектов.

2. В результате установления взаимосвязей между объектами — технологическими компонентами ИС, разработана контекстно–свободная грамматика языка описания технологических процессов обработки информации. Синтаксис языка позволяет описывать технологию работы системы с использованием букв русского и латинского алфавитов, что безусловно удобно для русскоязычных пользователей. Реализован компилятор ЯОТП. Использование сервисных программ BISON — генератора синтаксических анализаторов и FLEX — генератора программ лексического анализа ускорило реализацию компилятора на языке С. Разработка компилятора на языке С позволила отказаться от жестких требований к вычислительной технике.

3. На основе объектной FRL–программы предложен автоматизированный способ построения сети фреймов, являющейся своеобразной технологической базой знаний по работе информационной системы.

4. Разработаны и реализованы в процедурах интерпретатора алгоритмы выполнения запросов администратора информационной системы.

5. Показана возможность разделения процессов описания технологии работы информационной системы и работы интерпретатора. Создание описания технологических процессов обработки данных эксперт–разработчик может осуществлять на более мощной вычислительной технике, а использование описания после компиляции в работе интерпретатора возможно на ПЭВМ с меньшими вычислительными мощностями.

В Приложении приводятся тексты исходной и скомпилированной ЯОТП–программы.

Список литературы

[1] Егошина Т.В., Начинкин А.Д. Язык описания работы информационной системы. – В сб.: Международная конференция “Нечеткая логика, интеллектуальные системы и технологии”. — Владимир, ВлГУ, 1997.

[2] Егошина Т.В. Формализация описания технологических процессов в АСУ: Препринт ИФВЭ 96–43. — Протвино, 1996.

[3] Егошина Т.В. Метод представления технологии информационной системы, работающей на основе базы данных. – В сб.: III Международная конференция "Развитие и применение открытых систем". — Москва, МЦНТИ, 1996.

[4] Егошина Т.В. Технология учета и анализа медицинской статистики по заболеваемости на базе ПЭВМ. – В кн.: Применение персональных ЭВМ и локальных вычислительных сетей в промышленной и непромышленной сферах. Аннотации докладов. — Москва, МДНТП, 1991.

[5] Егошина Т.В. Автоматизированная система учета и анализа заболеваемости на базе ПЭВМ. – В сб.: Всесоюзный научно — технический семинар "Использование программных средств ПЭВМ для автоматизации учрежденческой деятельности"/ Тезисы докладов. — Тверь, НПО "Центрпрограммсистем", 1990.

[6] Егошина Т.В. Учет и анализ заболеваемости на основе сбора статистических данных. – В сб.: Информационные системы в науке — 95. - Москва, РАН, 1995.

Рукопись поступила 19 мая 1998 г.

Т.В. Егошина.
Математическое и программное обеспечение формирования технологических процессов обработки данных в информационных системах.

Оригинал-макет подготовлен с помощью системы LaTeX.
Редактор Н.В.Ежела Технический редактор Н.В.Орлова

Подписано к печати 25.05.98. Формат 60 × 84/8. Офсетная печать.
Печ.л. 1,87. Уч.-изд.л. 1,44. Тираж 100. Заказ 164. Индекс 3649.
ЛР №020498 17.04.97.

ГНЦ РФ Институт физики высоких энергий
142284, Протвино Московской обл.

Индекс 3649

А В Т О Р Е Ф Е Р А Т 98–32, И Ф В Э, 1998