Министерство образования и науки Республики Беларусь
БЕЛОРУССКАЯ ГОСУДАРСТВЕННАЯ ПОЛИТЕХНИЧЕСКАЯ АКАДЕМИЯ

Кафедра "Системы автоматизированного проектирования"

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В САПР

Лабораторные работы (практикум)
для студентов специальности 22.03 -
"Системы автоматизированного проектирования"

В 3-х частях

Часть 1

ЭКСПЕРТНЫЕ СИСТЕМЫ

Минск 1994

Министерство образования Республики Беларусь
БЕЛОРУССКАЯ ГОСУДАРСТВЕННАЯ ПОЛИТЕХНИЧЕСКАЯ АКАДЕМИЯ

Кафедра "Системы автоматизированного проектирования"

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В САПР

Лабораторные работы (практикум)
для студентов специальности 22.03 -
"Системы автоматизированного проектирования"

В 3-х частях

Ч а с т ь I

ЭКСПЕРТНЫЕ СИСТЕМЫ

М и н с к 1 9 9 4

УДК 683.3

В первой части лабораторного практикума предлагается комплекс лабораторных работ, необходимых для приобретения знаний и практических навыков инженером, проектирующим экспертные системы. Предлагается изучить инструментальные средства для построения экспертных систем, таких как система логического программирования TURBO PROLOG 2.0, оболочка экспертной системы ИНТЕР-ЭКСПЕРТ и объектно - ориентированный TURBO PASCAL. Серия лабораторных работ охватывает вопросы представления знаний и работы со знаниями. Рассматривается вопрос приобретения знаний. В процессе выполнения лабораторных работ студенты разрабатывают отдельные компоненты экспертных систем и проводят их исследование.

Лабораторные работы выполняются на ПЭВМ PC/AT в диалоговом режиме с использованием в качестве базового программного обеспечения системы TURBO PROLOG 2.0, оболочки экспертной системы ИНТЕР-ЭКСПЕРТ , TURBO PASCAL 6.0. При выполнении лабораторных работ используются справочные руководства, в том числе и на машинных носителях. Разрабатываемые студентами программы размещаются на сменных дискетах.

Составители:
Л.С.Овчинников, С.В.Горбачева

Рецензент А.А.Прихожий

Лабораторная работа № I

TURBO PROLOG 2.0 КАК ИНСТРУМЕНТАЛЬНОЕ СРЕДСТВО РАЗРАБОТКИ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Цель работы:

Приобретение студентами практических навыков разработки интеллектуальных систем в среде TURBO PROLOG 2.0.

I.I. Справочные сведения. Представление знаний в области искусственного интелекта

Представление знаний является важной областью исследований по искусственному интеллекту. В этой области наиболее важные типы знаний классифицируются следующим образом:

1. Объекты (object) и их свойства (property, attribute).

Объекты - это существующие в прикладной области универсальные понятия и их представители.

2. События (even).

События описывают участие объектов в деятельности и ситуациях. События характеризуют, например, время, место и причинно - следственные отношения.

3. Действия (performance, function, process).

Обычно интеллектуальная деятельность предполагает также способность совершать действия, т.е. процедурные знания о том, каким образом что-то делается.

4. Метазнания (meta - knowledge).

Метазнания - это знания о знаниях и их использовании.

1.2. Работа в интегрированной среде TURBO PROLOG 2.0

TURBO PROLOG 2.0 имеет все признаки развитой интегрированной среды программирования: компилятор, редактор, отладчик объединены в единую систему, в которой можно создавать, отлаживать и выполнять программы.

TURBO PROLOG 2.0 - это компилятор ПРОЛОГа с контролем типа данных. Содержит все характеристики интерпретатора ПРОЛОГ , но обладает гораздо более высоким быстродействием.

1. Компилятор позволяет создавать отдельные программы, работающие вне интегрированной среды программирования.

2. Поддерживают текстовой, графический и многооконный режимы.

3. Обеспечивает быстрый доступ к памяти и портам ввода-вывода, а также средствам подключения программ в машинных кодах к программам на ПРОЛОГе.

4. Компилятор поддерживает имена переменных, задаваемых пользователем, поэтому имеется возможность управлять исходной программой после процедуры компиляции. Для отладки программ используется трассировка.

5. Модули на ПРОЛОГе, СИ, ПАСКАЛе, АССЕМБЛЕРе могут быть объединены на этапе редактирования.

6. Встроенные предикаты обеспечивают прямой доступ к файлам.

7. Обеспечивается поддержка динамической базы данных.

8. Имеется полный набор математических операций и функций.

После запуска системы TURBO PROLOG 2.0 на экране появляются четыре окна:

Editor - окно редактирования (для ввода исходной программы);

Dialog - окно диалога (для ввода запросов и выдачи результатов);

Message - окно сообщений (для выдачи сообщений);

Trace - окно трассировки (для выдачи информации о выполнении программы).

В самой нижней строке будет выведена информация о возможностях, предоставляемых функциональными клавишами. Выше располагается строка подсказки , в которой появляются служебные указания.

Главное меню содержит семь команд:

FILES - команды управления файлами;
EDIT - редактирование программы;
RUN - запуск программы;
COMPILE - компиляция программы;
OPTIONS - изменение параметров компиляции;
SETUP - изменение системных параметров.

Внимание: При помощи клавиши ESC вы можете выйти из любого меню или подменю и вернуться в Главное меню.

Для организации трассировки в начало программы добавляется команда trace или shorttrace. Возможна также трассировка отдельных предикатов, если поставить их имена, разделенные запятыми, за командами trace или shorttrace.

Например: trace p, g.

В окне трассировки высвечивается информация.

Используйте клавишу F10. Для возврата в основное меню система просит нажать клавишу пробела.

1.3. Структура программы

Программа на TURBO PROLOG состоит из нескольких программных секций, каждой из которых предшествует ключевое слово, представленное в табл. 1.1.

Таблица I.I

Ключевые слова

Ключевое слово	Содержание
DOMAINS	0 и более объявлений DOMAIN
GLOBAL DOMAINS	0 и более объявлений GLOBAL DOMAIN
DATABASE	0 и более объявлений предикатов базы данных
PREDICATES	1 и более объявлений предикатов
GLOBAL PREDICATES	0 и более объявлений глобальных предикатов
GOAL	цель
CLAUSES	0 и более предложений, фактов или правил

Обычно в программе должны быть, по крайней мере, разделы PREDICATES и CLAUSES. Содержимое раздела DOMAINS может отсутствовать, но для больших программ этот раздел сокращает время компиляции и позволяет экономить размер выходного кода.

В TURBO PROLOG 2.0 при разработке больших программ целесообразно использовать модульное программирование. Для модульного программирования ключевые слова DOMAINS и PREDICATES могут иметь префикс GLOBAL, обозначающий, что последующие объявления DOMAINS и PREDICATES относятся к нескольким программам.

Программа может содержать несколько разделов DOMAINS и PREDICATES и CLAUSES. При этом необходимо соблюдать следующие ограничения:

1. Программная секция должна начинаться с соответствующего ключевого слова (DOMAINS, DATABASE, PREDICATES, CLAUSES, GOAL).

2. Во время компиляции может использоваться только одна цель.

3. Все CLAUSES, описывающие один и тот же предикат, должны записываться друг за другом.

4. Только один раздел GLOBAL PREDICATES может быть использован в процессе компиляции. При этом раздел PREDICATES не должен находиться перед ним.

5. Разделы, содержащие предикаты базы данных (DATABASE), должны предшествовать объявлениям GLOBAL PREDICATES и PREDICATES.

Рассмотрим содержание каждой из секций.

а) Секция DOMAINS (рус. термин"СЕКЦИЯ ПОЛЕЙ")

Для объявления этой секции используются четыре формата.

1. Первый формат

```
name = d,
```

где name - содержит элементы стандартного типа;

d - может быть целым (integer), знаковым (char), действительным (real), символьным (symbol), знаковым (character).

Это объявление используется для объектов, которые синтаксически схожи, а семантически различны.

2. Второй формат

```
mylist = element_of_Dom* ,
```

где mylist - список элементов element_of_Dom;
element_of_Dom - элемент, ранее описанный пользователем в DOMAINS либо один из стандартных типов DOMAINS;
* - означает "список"

3. Третий формат

```
myCompDom = f1( d11, ... , d1n );
            f2( d21, ... , d2n);...
            fn( dn1, ... , dnn).
```

DOMAINS, включающие сложные объекты, объявляются путем установления функтора и описания всех входящих в него компонентов.

4. Четвертый формат

```
file = name1; name2; ... ; nameN .
```

DOMAINS файлов используется в том случае, когда пользователю необходимо обратиться к файлу по символическому имени. В программе может быть только один DOMAINS этого типа, который называется file . Символические имена файлов задаются в качестве альтернатив

```
file = input; output; base .
```

б) Секция PREDICATES (рус. термин "СЕКЦИЯ ПРЕДИКАТОВ")

В этом разделе указывается имя предиката и область его аргументов или стандартных типов DOMAIN:

```
predname( domain1, domain2, ... , domainN ).
```

Предикат может состоять только из одного имени. Допускается многократное объявление предиката:

```
member( name, namelist );
member( number,numberlist )...
```

Альтернативы должны иметь одинаковое количество аргументов.

в) Секция CLAUSES (рус. термин"СЕКЦИЯ ПРЕДЛОЖЕНИЙ")

В этой секции определяются факты или правила. Факт представляется именем предиката, за которым следуют аргументы, заключенные в круглые скобки. Заканчивается запись факта точкой. Все имена предикатов и константы должны начинаться со строчной буквы.

Правило состоит из заголовка правила и тела правила. Заголовок представляет собой предикат, тело состоит из термов, которые могут быть связаны между собой словами OR или AND . Между телом и заголовком стоит слово IF. Символы - синонимы в правилах могут записываться двояко :

IF	как	:-
AND	как	,
OR	как	;

Каждое правило должно заканчиваться точкой. Имена переменных должны начинаться с прописной буквы и могут содержать только буквы, цифры или знак подчеркивания - '_ '.

Особую роль играют анонимные переменные. Анонимная переменная используется в том случае, когда значение этой переменной неважно. Она обозначается одиночным знаком подчеркивания.

1.4. Рекурсия

В ПРОЛОГе заложено мощное средство, позволяющее определять некоторые отношения и называемое рекурсией. Рекурсия используется в двух случаях:

1) когда отношения описываются посредством самих отношений;

2) когда сложные объекты являются частью других сложных объектов (т.е. являются рекурсивными объектами).

1.5. Содержание задания по лабораторной работе

I.5.I. Описание задачи

Рассмотрим описание логических схем с помощью TURBO PROLOG 2.0. Простейшими составляющими этих схем являются элементы " И ", " ИЛИ ", " НЕ ". Логические элементы широко используются

в цифровых интегральных микросхемах. Они выполняют определенные преобразования и обработку сигналов в двоичном коде. Интегральные микросхемы на основе логических элементов " И ", " ИЛИ ", " НЕ " находят применение в различных цифровых приборах, электронно-вычислительных машинах и т. д.

Рассмотрим логическую схему " И ", изображенную на рис. 1.1.

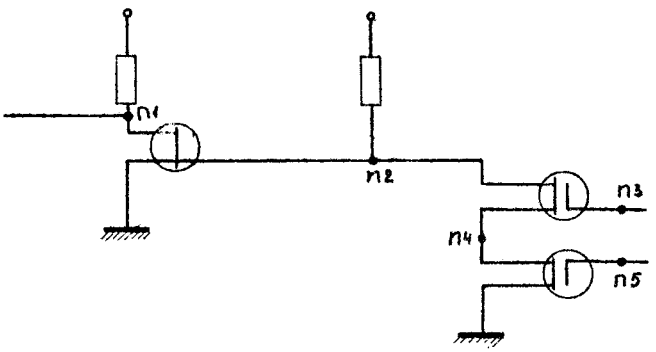

Рис. 1.1. Логическая схема " И "

Выходной сигнал является логическим " И " двух входных сигналов. На схеме последовательно соединены элемент " И - НЕ " и инвертор.

Инвертор состоит из транзистора с заземленным истоком и резистора, один вывод которого присоединен к источнику питания. Затвор транзистора является входом инвертора, а свободный вывод резистора соединен со стоком транзистора. Это соединение образует выход инвертора.

Реляционная схема: 1) резистора -
резистор (Вывод 1, Вывод 2);
2) транзистора -
транзистор (Затвор, Исток, Сток).

Для описания логической схемы " И " введем следующие отношения:

```
резистор  ( питание, n1 ).
резистор  ( питание, n2 ).
транзистор ( n2, земля, n1 ).
транзистор ( n3, n4, n2 ).
транзистор  ( n5, земля, n4 ).

инвертор ( Вход, Выход ) :-
            транзистор ( Вход, земля, Выход ),
            резистор ( питание, Выход ).
Схема_и_не ( Вход1, Вход2, Выход ):-
            транзистор ( Вход1, X, Выход ),
            транзистор ( Вход2, земля, X ),
            резистор ( питание, Выход ).
Схема_и ( Вход1,Вход2,Выход ):-
            схема_и_не ( Вход1, Вход2, X ),
            инвертор ( X, Выход ).
```

С учетом введенных отношений программа на TURBO PROLOG 2.0 имеет вид:

```
domains
    c=symbol
predicates
    resistor(c,c)
    transistor(c,c,c)
    invertor(c,c)
    scheme_and_no(c,c,c)
    scheme_and(c,c,c)
clauses
    resistor(power,n1).
    resistor(power,n2).
    transistor(n2,ground,n1).
    transistor(n3,n4,n2).
    transistor(n5,ground,n4).
    invertor(Input,Output):-
        transistor(Input,ground,Output),
        resistor(power,Output).
```

```
scheme_and_no( Input1, Input2, Output):-
        transistor( Input1, X, Output),
        transistor( Input2, ground, X),
        resistor(power, Output).
scheme_and( Input1, Input2, Output):-
        scheme_and_no( Input1, Input2, X),
        invertor( X, Output).
```

I.5.2. Задание

На рис. 1.2. представлена логическая схема " ИЛИ ":

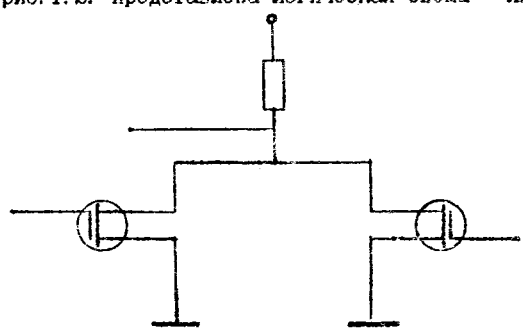

Рис. I.2. Схема логического элемента "ИЛИ"

Задание по лабораторной работе заключается в следующем:

1) к логическому элементу " ИЛИ " добавить инвертор;
2) разработать программу на TURBO PROLOG 2.0 для схемы " ИЛИ - НЕ ".

1.6. Содержание отчета

В отчете представляются:

1) логическая схема " ИЛИ - НЕ ";
2) листинг программы на TURBO PROLOG 2.0.

Лабораторная работа №2

ПРЕДСТАВЛЕНИЕ ИНЖЕНЕРНЫХ ЗНАНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ ЛОГИКОЙ ПРЕДИКАТОВ

Цель работы:

Изучение студентами способа представления знаний логикой предикатов. Самостоятельное решение модельной инженерной задачи с использованием логического программирования как средства для реализации логики предикатов. Изучение механизма вывода в логическом программировании.

2.1. Справочные сведения

2.1.1. Абстрактный интерпретатор логических программ

Непосредственное использование логики в качестве языка программирования называется логическим программированием. Логическая программа - это множество аксиом и правил, задающих отношения между объектами. Вычислением логической программы является вывод следствий из программы. Программа задает множество следствий, которое и представляет собой значение программы.

Искусство логического программирования состоит в построении ясных и изящных программ с требуемым значением.

Основу вычислительной модели логических программ составляет алгоритм унификации. Унификация является основой автоматической дедукции и логического вывода в задачах искусственного интеллекта.

Алгоритм унификации приведен ниже.

Вход: Два терма T1 и T2, которые надо унифицировать.

Результат: Подстановка Θ-наибольший общий унификатор (н.о.у.) термов T1 и T2 или отказ.

Алгоритм: Начальное значение подстановки θ пусто, стек содержит равенство T1 =T2 .

failure : - false, пока стек не пуст, и не failure,
выполнить
считать X = Y из стека
ветвление X - переменная, не входящая в Y:
заменить все вхождения X в стеке и
в θ на Y, добавить X = Y к θ
Y - переменная, не входящая в X:
заменить все вхождения Y в стеке и
в θ на X добавить Y = X к θ
X и Y - одинаковые константы или переменные:
продолжить
X есть f (X1,...,Xn) и Y есть
f (Y1,...,Yn) для некоторого функтора
f и n > 1:
поместить Xi= Yi для всех i = 1,...,n в
стек
в остальных случаях :
failure:-true
если failure, то результат - отказ;
иначе результат θ.

Абстрактный интерпретатор логических программ приведен ниже.

Вход : Логическая программа P цель G.
Результат: G Θ, если это пример G, выводимый из P, или "отказ", если возник отказ.
Алгоритм: Начальная резольвента равна входной цели G.
Пока резольвента не пуста, выполнить.
Выбрать такую цель A из резольвенты и такое переименованное предложение $A' \leftarrow B_1, B_2, \ldots, B_n$,

n>0 из P, что A и A' унифицируемы с н.о.у. θ (если нет таких цели и правила, то выйти из цикла).

Удалить A и добавить B1, B2,..., Bn к резольвенте. Применить θ к резольвенте и к G. Если резольвента пуста, то результат - G, иначе результат - отказ.

2.1.2. Семантика

Формулы логики предикатов строятся как символьные системы совершенно безотносительно к понятиям описываемого мира, что наглядно иллюстрируют рассмотренные выше алгоритмы.

Для установления соответствия необходимо выполнить следующие действия:

1. Установление соответствия между константами логики предикатов и сущностями этого мира (константы принимаются за имена объектов).

2. Установление соответствия между формулами логики предикатов и функциональными отношениями описываемого мира.

3. Установление соответствия между атомарными предикатами и концептуальными отношениями этого мира.

Иными словами, в язык привносится конкретное смысловое содержание. Это смысловое содержание является семантикой логики предикатов первого порядка.

Декларативная семантика логических программ основана на стандартной теоретико-модельной семантике логики первого порядка.

В идеале программист должен записать аксиомы, определяющие требуемые отношения, полностью игнорируя, каким образом эти аксиомы будут использоваться в процессе выполнения.

Имеющиеся языки логического программирования и, в частности, ПРОЛОГ, все еще далеки от этого идеала декларативного программирования. Нельзя игнорировать конкретный, четко определенный способ моделирования абстрактного интерпретатора в реализации каждого языка. Эффективное логическое программирование требует знания и использования этого способа. Программист должен не

только уделять внимание корректной и полной аксиоматизации отношения, но и рассматривать его выполнение в соответствии с вычислительной моделью.

Поэтому ниже рассматривается вычислительная модель ПРОЛОГа, раскрывающая механизм доказательства в процессе решения определенной цели.

2.1.3. Вычислительная модель ПРОЛОГа

Выполнение программ на ПРОЛОГе заключается в работе абстрактного интерпретатора, при которой вместо произвольной цели выбирается самая левая цель, а недетерминированный выбор предложения заменяется последовательным поиском унифицируемого правила и механизмом возврата.

Вычисление цели G относительно программы P, написанной на ПРОЛОГе, состоит в порождении всех решений цели G относительно программы P. В терминах логического программирования вычисление цели G в ПРОЛОГе является полным обходом в глубину конкретного "дерева" поиска цели G, в котором всегда выбирается самая левая цель.

Протокол вычисления в ПРОЛОГе является некоторым расширением протокола логической программы при использовании абстрактного интерпретатора. В протоколе описываются отказы и возвраты.

Встретив недоказуемую цель, вычисление переходит к цели указанной механизмом возврата.

Для обозначений отказов и возвратов используем следующие символы. Символ после цели означает, что цель недоказуема, т.е. в программе отсутствует предложение, заголовок которого унифицируют с целью. Встретив недоказуемую цель, вычисление переходит к цели, указанной механизмом возврата. Соответствующая цель уже появлялась в качестве предыдущей цели в протоколе на том же самом уровне и может быть идентифицирована именами переменной.

Символ " ; " после решения означает, что вычисление продолжается для поиска других решений. Унификатор записывается справа.

2.1.4. Трассировка программы на TURBO PROLOG 2.0

Интерпретатор TURBO PROLOG 2.0 обладает мощным средством трассировки, позволяющим отобразить протокол вычисления. Для этого необходимо вначале программы указать опцию компилятора trace или shorttrace, и тогда в окне трассировки будут отображаться трассируемые предикаты и информация, приведенная в табл. 2.1.

Таблица 2.I

Ключевые слова

Ключевое слово	Комментарий
CALL	Каждый раз при вызове предиката имя предиката и значение его параметров отображаются в окне трассировки
RETURN	RETURN отображается в окне трассировки, если clause выполнено и предикат возвращается к вызывающему предикату. Если имеются другие clauses, удовлетворяющие входным параметрам, то выводится " * ", обозначающая, что это предложение предполагает возврат к предыдущему состоянию
FAIL	Если предикат не удовлетворяется, то отображается слово FAIL, за которым следует имя невыполнимого предиката
REDO	REDO обозначает, что имеется возврат к предыдущему состоянию. Имя предиката, которое повторно выполняется, вместе со значением его параметров отображается в окне трассировки

2.2. Задание к лабораторной работе

Для достижения поставленной цели рассматривается проблема влияния процесса наростообразования на качество обрабатываемой лезвийным инструментом поверхности деталей машин. В задачу, ре-

шаемую при выполнении работы, входит формализация инженерных знаний о проблеме с использованием методики, описанной в п. 2.1.2 предложениями ПРОЛОГа.

Полученное семантическое описание задачи включается в программу на ПРОЛОГе, анализирующую влияние технологических параметров процесса резания на шероховатость обработанной поверхности с учетом явления наростообразования.

На основании трассировки программы в отчете по лабораторной работе строится протокол вычисления с использованием символов и соглашений, описанных в п. 2.1.3.

2.2.1. Описание явления наростообразования при ревании материалов

а) Общие сведения

При образовании сливной стружки часто наблюдается задерживание обрабатываемого металла на передней поверхности непосредственно около режущего лезвия. Это наслоение в сечении имеет треугольную форму (рис. 2.1). Впервые это явление обнаружил в 1915 г. Я. Г. Усачев, а само образование назвал наростом.

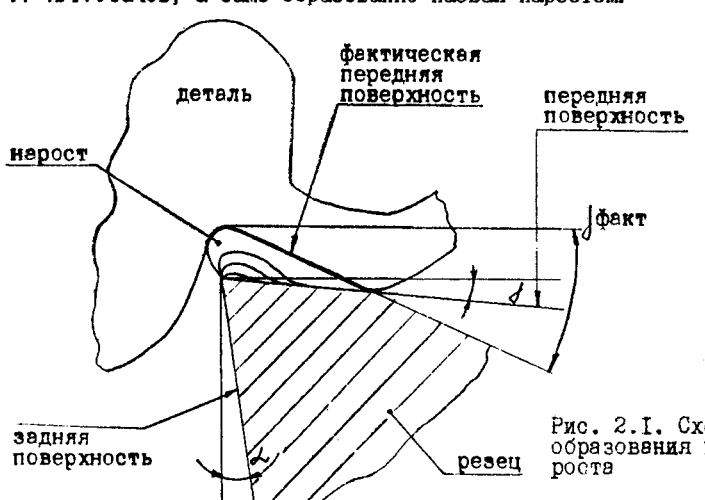

Рис. 2.1. Схема образования нароста

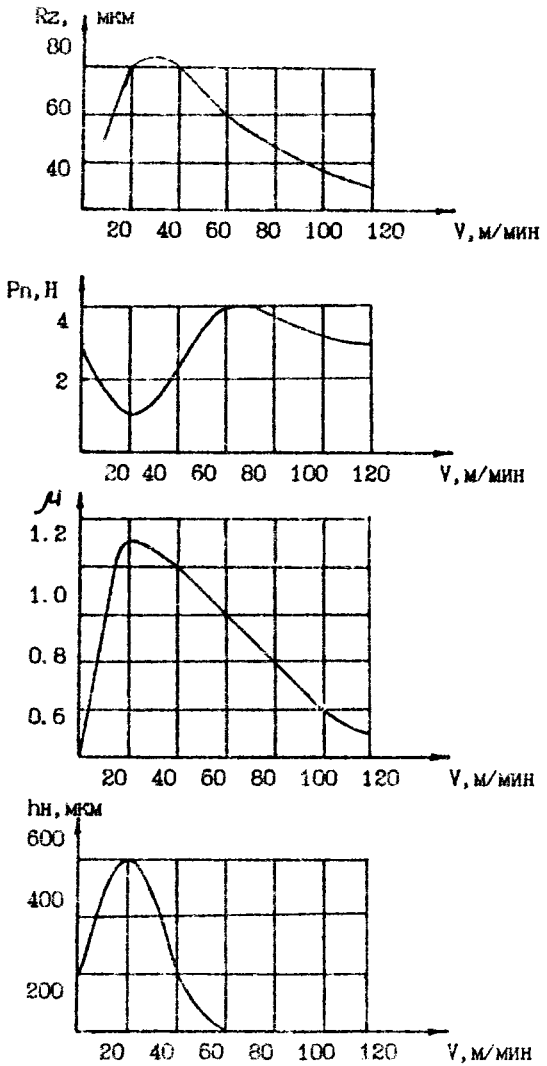

Рис. 2.2. Влияние нароста (hн) на коэффициент трения (μ), силу резания (Pн), величину шероховатости (Rz) при точении стали 40Х на различных скоростях

Он обнаружил, что структура нароста представляет собой тонкие слои металла, которые наложены друг на друга и вытянуты вдоль передней поверхности инструмента. Существует несколько точек зрения о причине образования нароста.

Слои металла наращиваются друг на друга, пока нарост не достигнет размеров, максимально возможных при данных технологических условиях. Вследствие чрезмерно сильной деформации слои металла, образующие нарост, упрочняются. Их твердость в 2,5 ...3,5 раза больше твердости исходного металла. Таким образом, нарост как бы принимает на себя функции режущего лезвия. Однако он не стабилен. Достигнув максимальной величины, нарост разрушается, частично уносится стружкой, частично-поверхностью детали.

б) Влияние нароста на процесс резания

Явление наростообразования имеет большое значение в практике обработки резанием:

нарост изменяет передний угол, а следовательно, изменяет сопротивление резанию P_N и условия трения;

ухудшает чистоту обработанной поверхности, увеличивает R_z;

защищает заднюю поверхность инструмента от разрушения.

Для каждого материала существует определенный диапазон малых скоростей резания, в котором величина нароста h_N максимальна (рис. 2.2). Периодические срывы нароста приводят к возникновению вибраций, ухудшающих качество обработки.

Одним из способов борьбы с нарастом, в особенности при чистовой обработке, является правильный выбор скорости резания.

Следует учитывать, что наросты могут образовываться при резании твердосплавными, стальными, минералокерамическими и алмазными инструментами различных материалов. Но наибольшей величины наросты достигают при резании пластичных материалов с образованием сливных стружек, инструментами, изготовленными из углеродистых и быстрорежущих сталей.

Уровни максимально допустимых скоростей резания для различных инструментальных материалов приведены в табл. 2.2.

Таблица 2.2

Максимально допустимые скорости резания

Инструментальный материал	Теплостойкость	Макс.скорость резания, м/мин
1	2	3
Углеродистые У10А...У13А	200...250 С	8...10
Быстрорежущие стали		
Р18, Р12, Р9, Р6М3, Р6М5	620...630 С	20...40
Р10К5Ф5, Р18Ф2К8М,		
Р9М4К8, Р6М5К5, Р9К5	630...650 С	20...40
В11М7К23, В18М7К25	700...720 С	40...60
Твердые сплавы		
ВК4, ВК6, ВК6-М, ВК6-ОМ, ВК8	800...900 С	100...120
Т5К10, Т15К6, Т30К4	900...1000 С	120...150
Минералокерамика		
ЦМ-332, В-3, ВОК-60	1100...1200 С	150 ...200
Сверхтвердые материалы		
алмазы (баллас, карбонадо)	800 С	90 ...100
Нитрид бора	1500 С	200 ...250

2.2.2. Содержание отчета

В отчете представляются:

1) программы на ПРОЛОГе с комментариями;
2) протокол вычисления.

Лабораторная работа №3

ПРЕДСТАВЛЕНИЕ ИНЖЕНЕРНЫХ ЗНАНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ ПРОДУКЦИОННЫМИ ПРАВИЛАМИ

Цель работы:

Изучение студентами способов представления знаний продукциями.

3.1. Справочные сведения

3.1.1. Компоненты экспертной системы

У полностью сформированной экспертной системы имеются четыре существенные компоненты:

1) база знаний;
2) машина вывода;
3) модуль извлечения знаний;
4) система объяснения (интерфейс).

Все четыре модуля, показанные на рис. 3.1, являются важными, и хотя система, основанная на знаниях, может обойтись без одного - двух из них, истинно экспертная система обязана иметь их все.

3.1.2. База знаний

База знаний содержит факты (или утверждения) и правила. Факты представляют собой краткосрочную информацию в том отношении, что они могут изменяться, например, в ходе консультации. Правила представляют более долговременную информацию о том, как порождать новые факты или гипотезы из того, что сейчас известно.

Факты в базе данных обычно пассивны: либо они там есть, либо их нет. База знаний, с другой стороны, активно пытается пополнить недостающую информацию.

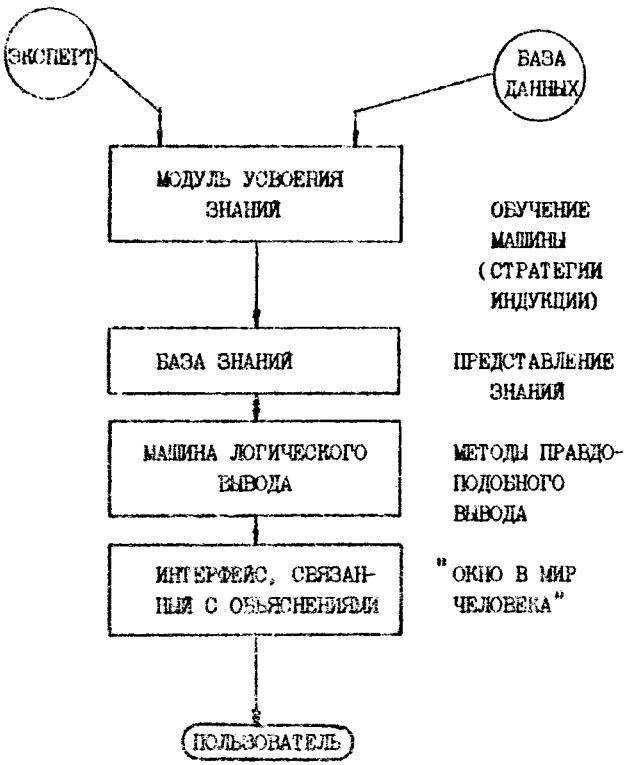

Рис. 3. 1. Типичная экспертная система

3.I.3. Правила продукций и машина вывода продукционной системы

Правила продукций являются одним из средств отображения неформальных знаний. Такие правила имеют формат ЕСЛИ - ТО, например:

ЕСЛИ хозяева поля проиграли последнюю игру у себя дома И гости выиграли последнюю игру на своем поле, ТО вероятность ничьей следует умножить на 1,075; вероятность выигрыша гостей умножается на 0,96.

Но следует помнить, что эти знания не воплощены в какую - то программу, а представляют собой данные для высокоуровневого интерпретатора, а именно, машины вывода. В машине вывода применяются три стратегии для логического вывода в целом. Прямая цепочка рассуждений связана с рассуждениями, ведущимися от данных к гипотезам, тогда как обратная цепочка - с попыткой найти данные для доказательства или опровержения некоторой гипотезы. Чисто прямая цепочка рассуждений ведет к неуправляемому режиму задания вопросов в диалоге, тогда как обратная цепочка будет, как правило, приводить к настойчивому повторению вопросов, касающихся цели. Поэтому третья стратегия предусматривает комбинацию этих цепочек.

3.I.4. Средства для конструирования экспертных систем

В настоящее время доступны следующие средства для создания экспертных систем, которые мы расположим в порядке убывания содержащихся в них заранее скомпонованных элементов экспертных систем :

оболочки экспертных систем (такие как EMICYN, SAVE, REVAL, "МИКРОЭКСПЕРТ ", " ИНТЕРЭКСПЕРТ " и т.д.):

специальные программные среды (типа POPLOG и т.п.);

символьные языки (ПРОЛОГ, ЛИСП);

алгоритмические языки (СИ , ФОРТРАН, ПАСКАЛЬ , БЕЙСИК).

3.2. Реализация экспертной системы на ПРОЛОГе

Воспользуемся языком ПРОЛОГ, считая его системной продукцией.
1. ПРОЛОГ дает нам базу данных, в которой можно расположить правило продукций в виде:

заключение ЕСЛИ предусловие 1
И предусловие 2

В логической интерпретации языка ПРОЛОГ это означает:

Чтобы доказать заключение
доказать предусловие 1 и
доказать предусловие 2

Для этого в языке ПРОЛОГ используется весьма простой синтаксис, в котором " :- " обозначает ЕСЛИ, " , " обозначает И.

2. Пролог позволяет хранить также и факты в форме правил, показанных выше, но у которых нет предусловий.

3. Управляющий механизм, обеспечиваемый системой ПРОЛОГ, известен как поиск в глубину или рассуждение в обратном направлении.

3.3. Содержание задания по лабораторной работе

3.3.1. Описание задачи

В настоящем разделе рассматривается простая учебная экспертная система, предназначенная для различения отдельных деталей машин по набору конструктивных признаков (совокупности элементов). Структура разрабатываемой экспертной системы , основанной на продукционных правилах, имеет форму сети, показанной на рис. 3.2. В прямоугольниках размещаются утверждения; кружочки

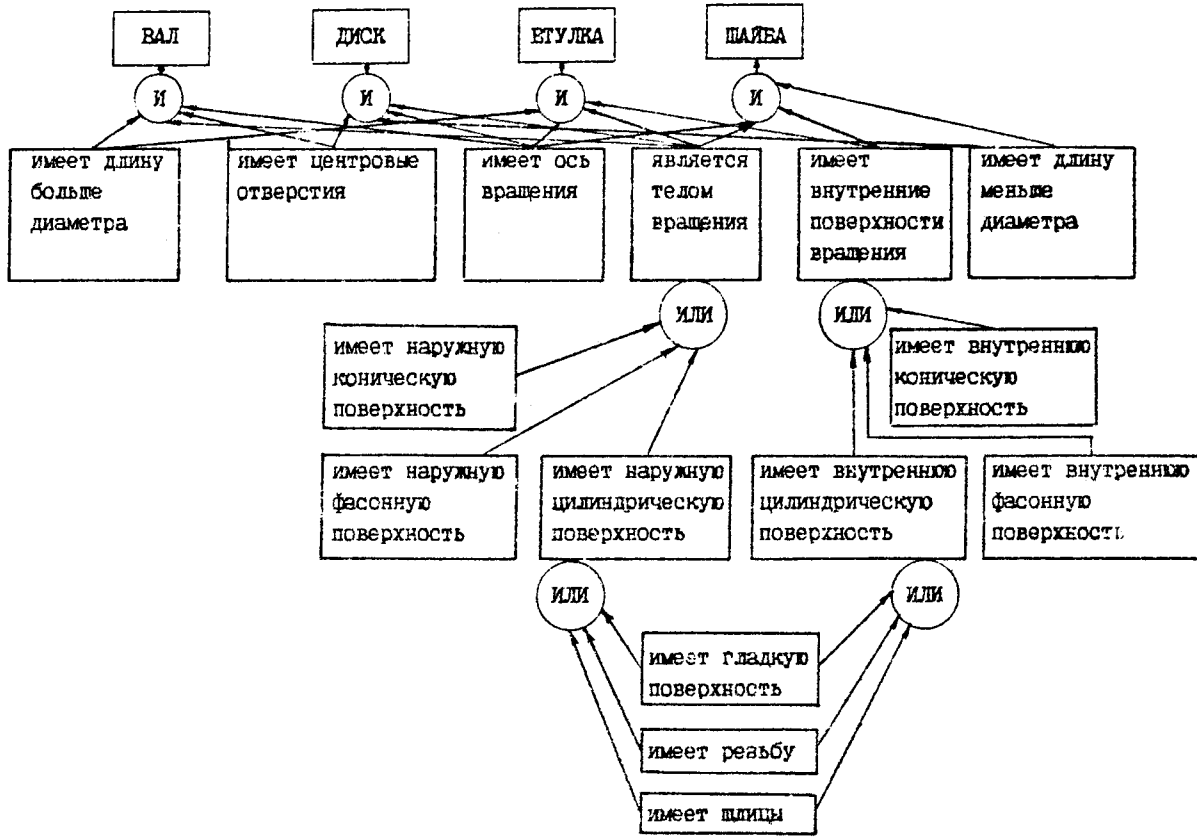

Рис. 3. 2. Пример сети для разрабатываемой системы, основанной на правилах

представляют способы их комбинирования. Следует отметить, что эксперты могут дополнить систему и другими правилами. Здесь мы ограничились приведенным набором правил исключительно из соображений упрощения задачи.

Экспертная система, написанная на языке ПРОЛОГ, имеет следующий вид:

```
деталь ( вал ):-
        имеет ( длину больше диаметра ),
        имеет ( центровые отверстия ),
        имеет ( ось вращения ),
        является ( телом вращения ).
деталь ( диск ):-
        имеет ( длину меньше диаметра ),
        имеет ( центровые отверстия ),
        имеет ( ось вращения ),
        является ( телом вращения ).
деталь ( втулка ):-
        имеет ( длину больше диаметра ),
        имеет ( ось вращения ),
        имеет ( внутренние поверхности вращения ),
        является ( телом вращения ).
деталь ( шайба ):-
        имеет ( длину меньше диаметра ),
        имеет ( ось вращения ),
        имеет ( внутренние поверхности вращения ),
        является ( телом вращения ).
является ( телом вращения ):-
        имеет ( наружную цилиндрическую поверхность ).
является ( телом вращения ):-
        имеет ( наружную коническую поверхность ).
является ( телом вращения ):-
        имеет ( наружную фасонную поверхность ).
имеет ( внутренние поверхности вращения ):-
        имеет ( внутренние цилиндрические поверхности ).
имеет ( внутренние поверхности вращения ):-
        имеет ( внутренние конические поверхности ).
имеет ( внутренние поверхности вращения ):-
```

```
        имеет ( внутренние фасонные поверхности ).
имеет ( наружную цилиндрическую поверхность ):-
        имеет ( гладкую поверхность ).
имеет ( наружную цилиндрическую поверхность ):-
        имеет ( резьбу ).
имеет ( наружную цилиндрическую поверхность ):-
        имеет ( шлицы ).
имеет ( внутреннюю цилиндрическую поверхность ):-
        имеет ( гладкую поверхность ).
имеет ( внутреннюю цилиндрическую поверхность ):-
        имеет ( резьбу ).
имеет (внутреннюю цилиндрическую поверхность ):-
        имеет ( шлицы ).
```

В базе данных, основанной на системах продукций, вначале нет никаких фактов о текущей ситуации. Поэтому необходимо включить в нашу программу простой подраздел, позволяющий в ходе беседы получить информацию от пользователя:

```
имеет ( Y ):-
     написать ( " имеет ли деталь" ),
     написать ( Y ), nl,
     прочесть ( Ответ ),
     положительный ( Ответ ),
     внести ( имеет (Y) ).
является ( Y ):-
     написать ( " является ли деталь " ),
     написать ( Y ), nl,
     прочесть ( Ответ ),
     положительный ( Ответ ),
     внести ( является ( Y ) ).
положительный ( yes ).
```

Наконец, нам нужно определение всей задачи в целом:

```
запуск:-
     деталь (Y),
     написать ( " я думаю, что это"),
     написать (Y), nl, !.
запуск:-
     написать ( " я не знаю такую деталь " ), nl.
```

Все, что нужно теперь пользователю, - это набрать на терминале команду " запуск ", а остальное сделает сама система.

3.3.2. Задание по лабораторной работе

Оно заключается в следующем:

1) объяснить работу программы;
2) дополнить программу правилами, распознающими корпусную деталь;
3) написать программу на TURBO PROLOG 2.0; при этом следует учесть, что предикаты написать ( ), прочесть ( ), внести ( ) являются встроенными предикатами TURBO PROLOG, и их описание содержится в руководстве по системе программирования.

3.4. Содержание отчета

В отчете представляются:

1) листинг разработанной программы;
2) консольный протокол работы системы.

Лабораторная работа № 4

ПРЕДСТАВЛЕНИЕ ИНЖЕНЕРНЫХ ЗНАНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ. РАЗРАБОТКА ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ ЭКСПЕРТНОЙ СИСТЕМЫ ПРОДУКЦИОННОГО ТИПА СРЕДСТВАМИ TURBO PROLOG 2.0

Цель работы:

Изучение студентами форм организации диалога в экспертной системе продукционного типа и получение навыков разработки программ поддержки интерфейса в рамках модульного программирования.

4.I. Справочные сведения. Представление знаний экспертной системы

Знания, приобретенные экспертной системой при ее взаимодействии с экспертом, обычно заносятся в программу в виде фактов и правил, описываемых предикатами базы знаний.

Средствами ПРОЛОГа обеспечивают хранение этих знаний в естественной терминологии, что в некоторых случаях позволяет производить корректировку базы знаний, используя обычный текстовой редактор.

Интерфейс эксперта и пользователя должен быть ориентирован на неподготовленных в области программирования людей, владеющих только системой профессиональных понятий и соответствующей терминологией.

Для удовлетворения этого требования в ПРОЛОГе используется развитая оконная система, а также программные модули, реализующие построение разнообразных меню.

Знания в экспертной системе представляют собой правила, имеющие условную конструкцию типа ЕСЛИ ... ТО:

```
rule ( N, CAT1, CAT2, COND ).
```

В структуре правила можно выделить четыре элемента :

N - порядковый номер правила;

CAT1 - выражение (ключевое слово или фраза), находящееся в левой части правила и характеризующее признак класса (подкласса и т.д.), к которому относится данное правило;

CAT2 - содержание правила, расположенное в его правой части и содержащее вывод или утверждение, вытекающее из этого правила;

COND - список условий, определяющих правило.

4.2. Описание базовой программы, осуществляющей ввод новых правил

Проект, выполненный в лабораторной работе, опирается на базовую программу, выполняющую следующие функции и имеющую следующий вид:

```
    /* Формирование окон для создания нового правила */

 update:-
  makewindow(4,7,7,"НОВОЕ ПРАВИЛО",9,0,4,80),
  makewindow(5,7,7,"НОВОЕ ПРАВИЛО",13,0,4,80),
  makewindow(6,7,7,"НОВОЕ ПРАВИЛО",17,0,6,80),
  repeat,sost(K),K="",!,
  shiftwindow(6),removewindow,
  shiftwindow(5),removewindow,
  shiftwindow(4),removewindow.
 update.

/* Формирование и присоединение к базе знаний предикатов,
   описывающих созданное правило */

 sost(KAT):- shiftwindow(6),
   clearwindow,shiftwindow(5),
   clearwindow,shiftwindow(4),
   clearwindow,
   cursor(0,1),
   write("Состояние:"),
   readln(KAT),KAT<>"",!,
   asserts(KATNUM,KAT),
   shiftwindow(5),
   cursor(0,1),
   write("Его суть: "),
   cursor(0,14),
   readln(SUB),SUB<>"",!,
   asserts(SUBNUM,SUB),
   readcondl(CONDL),
   getrnr(1,RNO),
   assert(ru(RNO,KATNUM,SUBNUM,CONDL)).
  sost(K):-K="".
```

```
 /* Формирование и присоединение к базе знаний предикатов,
    описывающих условия, при которых будет выполняться
    созданное правило */

readcondl( [BNO|R] ):-
  shiftwindow(6),
  clearwindow,
  write("Условие: "),readln(COND),
  COND><"",!,
  write("Комментарий: "),readln(M),
  getcond(BNO,COND,M),readcondl(R).
readcondl([]).

 /* Получение очередного порядкового номера предиката
    базы знаний, описывающего правило, условие или
    категорию */

 getrnr(N,N):-not(ru(N,_,_,_)),!.
 getrnr(N,N1):-H=N+1, getrnr (H,N1).
 getbnr (N,N):-not(cd(N,_)),!.
 getbnr(N,N1):-H=N+1,getbnr(H,N1).
 getcatnum(N,N):-not(cg(N,_)),!.
 getcatnum(N,N1):-H=N+1,getcatnum(H,N1).


 /* Присоединение к базе знаний предикатов, описывающих ус-
    ловия, комментарии и категории , которые соответствуют
    введенному правилу */


 getcond(BNO,COND,_):-cd(BNO,COND),!.
 getcond(BNO,COND,Message):-getbnr(1,BNO),clearwindow,
 assert(cd(BNO,COND)),assert(ct(BNO,Message)).
 asserts(SUBNUM,SUB):-cg(SUBNUM,SUB),!.
 asserts(SUBNUM,SUB):-getcatnum(1,SUBNUM),
                      assert(cg(SUBNUM,SUB)).
```

Приведенный фрагмент программы дает возможность пользователю организовать новые правила. При этом на экране появляются три окна. В первом окне на запрос " Состояние : " вводится выражение, характеризующее класс, подкласс или группу, к которым относится правило. Затем дается формулировка самого правила - содержимое окна " Его суть ".

Перечень условий и комментариев к условиям соответственно заносится в третье окно на запросы " Условие : " и " Комментарий ".

Правила и факты, описывающие введенные данные, автоматически присоединяются к базе знаний.

4.3. Содержание задания по лабораторной работе. Описание задачи

Задачей лабораторной работы является реализация базовой программы в среде TURBO PROLOG 2.0 в форме project с использованием внутренней динамической базы данных и внесение в нее следующих усовершенствований:

1) разработка модуля просмотра правил в базе данных;
2) разработка модуля, обеспечивающего удаление правил из базы данных по выбору пользователя с организацией системы запрета от ошибочного стирания, обеспечивающей следующее:
   а) запрос типа " Вы действительно хотите удалить правило ? ";
   б) при удалении правила оно должно заноситься во временную базу данных, формируемую в памяти на время работы программы;
3) разработка модуля восстановления удаленного правила в базу данных из временной базы данных.

П р и м е ч а н и е. Набор кода к базовой программе получить у преподавателя на дискете.

4.4. Содержание отчета

В отчете представляются:

1) листинг разработанной программы;
2) консольный протокол работы программы.

Лабораторная работа №5

ПРЕДСТАВЛЕНИЕ ИНЖЕНЕРНЫХ ЗНАНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ ФРЕЙМАМИ СРЕДСТВАМИ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО TURBO PASCAL 6.0

Цель работы:

Приобретение студентами навыков представления знаний фреймами средствами объектно-ориентированного TURBO PASCAL 6.0.

5.1. Справочные сведения. Сущность теории фреймов

В основе систем, использующих фреймы для представления знаний, лежит теория фреймов. Теория фреймов относится к психологическим понятиям, касающимся понимания того, что мы видим и слышим.

Согласно автору теории фреймов М. Мински [6], суть этой теории заключается в следующем. Когда человек попадает в новую ситуацию (или радикально изменяет свое отношение к текущим обстоятельствам), он вызывает из своей памяти основную структуру, именуемую фреймом. Фрейм (рамка) - это единица представления знаний, запомненная в прошлом, детали которой при необходимости могут быть изменены согласно текущей ситуации. Каждый фрейм может быть заполнен различной информацией. Эта информация может касаться способов применения данного фрейма, последствий этого применения, действий, которые необходимо выполнить, если не оправдался прогноз, и т. п. Каждый фрейм можно рассматривать как сеть, состоящую из нескольких вершин и отношений. На самом верхнем уровне фрейма представлена фиксированная информация: факт, касающийся состояния объекта, который обычно считается истинным. На последующих уровнях расположено множество так называемых терминальных слотов (терминалов), которые обязательно должны быть заполнены конкретными значениями и данными. В текущем слоте задается условие, которое должно выполняться при установлении соответствия между значениями (слот либо сам устанавливает соответствие, либо обыч-

но это делает более мелкая составляющая фрейма). Простое условие указывается меткой; оно может содержать требования, - например, чтобы соответствие устанавливал пользователь; чтобы было достаточно полно описанное значение; чтобы был указатель специальных составляющих фреймов и т. п. Сложные условия указывают отношения между фактами, соответствующими нескольким терминалам. Соединив множество фреймов, являющихся отношениями, можно построить фреймовую систему. Наиболее важный результат такого построения проявляется в возможности преобразования фреймов в одной системе.

При анализе видимого объекта различные фреймы одной системы описывают его с различных углов зрения, и преобразование от одного фрейма к другому показывает результат перехода от одного пункта наблюдения к другому. В одной системе различные фреймы могут иметь общие терминалы. Это - серьезный момент, на который следует обратить внимание, поскольку благодаря ему вовможно связывание информации, полученной с различных точек зрения.

Несколько терминалов одного фрейма обычно заранее определяются значениями по умолчанию. Следовательно, даже когда не задана подробная информация, касающаяся некоторого места, данный фрейм все равно будет информативен.

Фреймовые системы связаны с информационно-поисковыми сетями. Если фрейм-кандидат не соответствует текущей проблеме, другими словами, если установление соответствия с терминалом не вполне отвечает условию метки, такая сеть задает другой фрейм. С помощью подобной межфреймовой структуры можно представлять знания, касающиеся фактов, сходств и другой информации, полученной для понимания.

5. 2. Структуры данных фрейма

Фрейм представлен определенной структурой данных, показанной на рис. 5. 1.

1. **Имя фрейма.** Это идентификатор, присваиваемый фрейму. Фрейм должен иметь имя, единственное в данной фреймовой системе (уникальное имя). Каждый фрейм состоит из произвольного числа слотов,

причем несколько из них определяются самой системой для выполнения специфических функций, а остальные определяются пользователем. В их число входят: слот IS-A, показывающий фрейм - родитель данного фрейма; слот указателей дочерних фреймов; слот для ввода имени пользователя, даты определения, даты изменения, текста комментария и другие слоты. Каждый слот, в свою очередь, также представлен определенной структурой данных.

Имя фрейма

Слот 1	Указатель наследования	Указатель атрибутов слота	Значение слота	Демон
Слот 2	----	----	----	----
.....				
Слот n	----	----	----	----

Рис. 5.I. Структура данных фрейма

2. Имя слота. Уникальное имя во фрейме. Обычно не носит никакой смысловой нагрузки. Но в некоторых случаях оно может иметь специфический смысл.

3. Указатель наследования. Эти указатели касаются только фреймовых систем иерархического типа, основанных на отношениях " абстрактное - конкретное "; они показывают, какую информацию об атрибутах слотов во фрейме верхнего уровня наследуют слоты с такими же именами во фрейме нижнего уровня.

4. Указатель типа данных. Указывается, что слот имеет численное значение либо служит указателем другого фрейма.

5. Значение слота. Пункт ввода значения слота. Значение слота должно совпадать с указанным типом данных этого слота; кроме того, должно выполняться условие наследования.

6. Демон. Демоном называется процедура, автоматически запускаемая при выполнении некоторого условия. Демоны запускаются при обращении к соответствующему слоту. Демон может являться разновидностью присоединенной процедуры.

7. Присоединенная процедура. В качестве значения слота можно использовать программу процедурного типа. В данном случае присоединенная процедура запускается по сообщению, передаваемому из другого фрейма. Поскольку состояние выполнения в этом случае такое же, как и в объектно-ориентированном языке, то язык фреймового типа называют еще объектно-ориентированным языком.

Когда мы говорим, что в моделях представления знаний фреймами объединяются процедурные и декларативные знания, то демоны и присоединенные процедуры представляются процедурными знаниями. Кроме того, в языке представления знаний фреймами отсутствует специальный механизм управления выводом, поэтому пользователь должен реализовать данный механизм с помощью присоединенной процедуры.

Известны примеры систем, допускающих применение правил продукций в качестве типа данных. Это обусловлено, с одной стороны, тем, что большинство систем, ориентированных на решение сложных проблем, содержит в качестве составляющей продукционную систему, а с другой стороны,- снижением нагрузки на пользователя. Кроме того, известны примеры систем, допускающих применение функций ПРОЛОГа в качестве присоединенной процедуры.

5.3. Способы управления выводом

Во фреймовых системах используются три способа управления выводом: два - с помощью присоединенных процедур - демона и служебной процедуры (или метода) и один-с помощью механизма наследования. Последний способ можно назвать единственным основным механизмом управления выводом, которым оснащаются фреймовые системы.

Другими словами, с помощью механизма управления наследованием, базирующегося на отношениях " абстрактное - конкретное " (или отношениях типа IS - A) осуществляется автоматический поиск и определение значений слотов фрейма верхнего уровня и присоединенных процедур служебного типа.

5.4. Содержание задания по лабораторной работе

5.4.I. Расчет зубчатых передач

Расчет зубчатых передач (расчет конструктивных параметров передач) для различных передач имеет свои особенности и инвариантные компоненты. Методика расчета (расчетная задача определения основных размеров зацепления) может быть представлена в виде фреймовой системы, имеющей структуру, представленную на рис. 5.2.

В рамках лабораторной работы ограничимся фрагментом системы (1.1.1, 1.1.1.1 и 1.1.1.2), выделенным на рис. 5.2 штриховой линией.

Формулы для определения основных размеров зацепления некорригированных передач с цилиндрическими зубчатыми колесами наружного зацепления представлены в табл. 5.1.

В таблице-ссылки на страницы из [6].

5.4.2. Содержание задания по лабораторной работе

В лабораторной работе необходимо:

1. Разработать структуру фреймов, описывающих расчеты 1.1.1, 1.1.1.1 и 1.1.1.2 (см. рис. 5.2).
2. Представить фреймы в форме объекта в системе объектно-ориентированного программирования TURBO PASCAL 6.0 [10].

Предполагается, что система инициализируется при поступлении исходных данных для расчета.

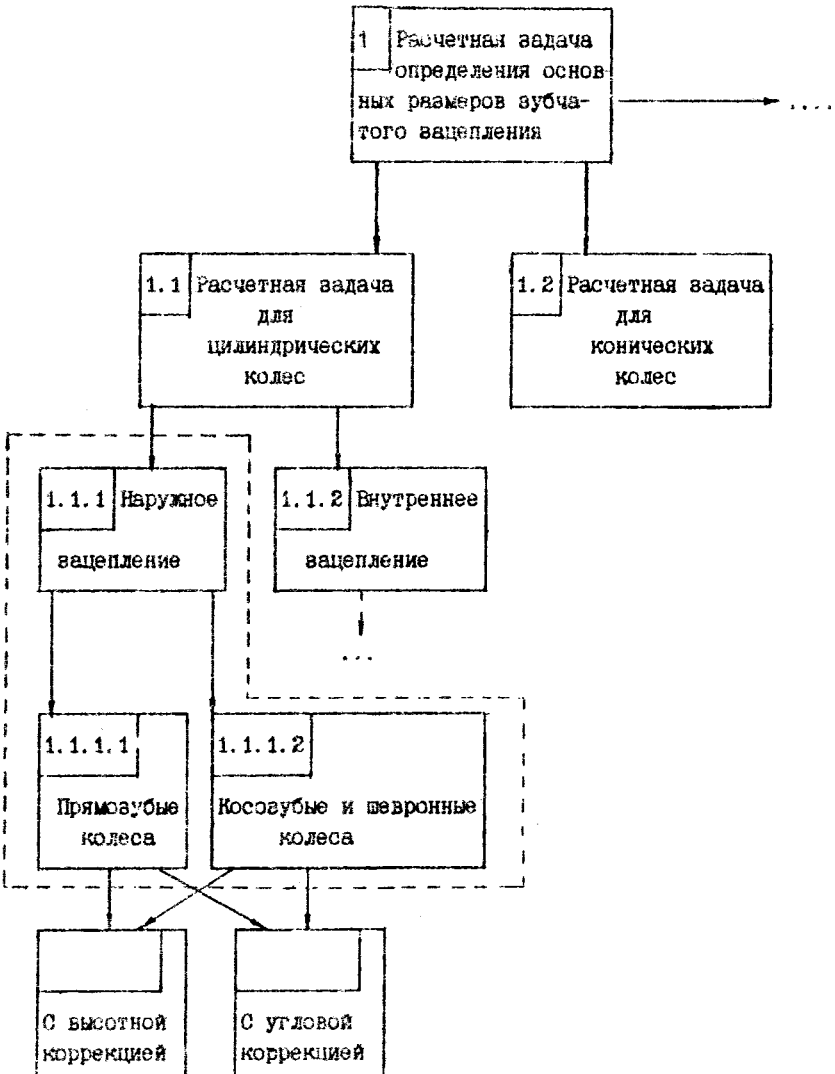

Рис. 5. 2. Фреймовая система задачи расчета зубчатой передачи

Таблица 5.I

Формулы определения основных размеров зацепления некорригированных передач с цилиндрическими зубчатыми колесами наружного зацепления

Обозначение	Прямозубые колеса	Косозубые и шевронные зубчатые колеса
1	2	3
A	$A=A_0=\frac{Z_1+Z_2}{2}=0,5*Z_\Sigma*m_s$	
m_n	$m_n=m$	
m_s	$m_s=m$	$m_s=\frac{m_n}{\cos\beta_\partial}$
C	$C\approx 0,25\times m_s$	$C\approx 0,25\times m_s$
h	$h\approx 2,25\times m_s$	$h\approx 2,25\,m_n$
$d_{\partial 1}$	$d_{\partial 1}=d_1=m_s Z_1$	
$d_{\partial 2}$	$d_{\partial 2}=d_2=m_s Z_2$	
D_{e1}	$D_{e1}=d_{\partial 1}+2*f_0 m_s=(Z_1+2*f_0)m_s$	$D_{e1}=d_{\partial 1}+2*f_0 m_n$
D_{e2}	$D_{e2}=d_{\partial 2}+2*f_0 m_s=(Z_2+2*f_0)m_s$	$D_{e2}=d_{\partial 2}+2*f_0 m_n$
D_{i1}	$D_{i1}=d_{\partial 1}-2*f_0 m_s-2*C$	$D_{i1}=d_{\partial 1}-2*f_0 m_n-2*C$
D_{i2}	$D_{i2}=d_{\partial 2}-2*f_0 m_s-2*C$	$D_{i2}=d_{\partial 2}-2*f_0 m_n-2*C$
f_0	Согласно ГОСТ 3058-54, величина $f_0=1$	
r_{o1}	$r_{o1}=0,5*d_{\partial 1}*\cos\alpha_\partial$	$r_{o2}=0,5*d_{\partial 1}*\cos\alpha_{on}$

Продолжение табл. 5.1

I	2	3
r_{os}	$r_{b2}=0,5 \cdot d\partial_2 \cdot \cos\alpha\partial$	$r_{o2}=0,5 \cdot d\partial_2 \cdot \cos\alpha_{os}$
α_{on}	$\alpha_{on}=\alpha\partial=20^\circ$	$\alpha_{on}=\alpha_n=20^\circ$
α_{os}	$\alpha_{os}=\alpha\partial=20^\circ$	$tg\alpha_{os}=\frac{tg\alpha_{on}}{\cos\beta\partial}=\frac{tg\alpha\partial}{\cos\beta\partial}$
α_s	$\alpha_s=\alpha\partial=20^\circ$	$\alpha_s=\alpha_{os}$
$\beta\partial$	$\beta\partial=0^\circ$	$\beta\partial \approx 7^\circ \div 25^\circ$ - для косозубых $\beta\partial \approx 25^\circ \div 35^\circ$ - для шевронных
β	—	если $\beta\partial \neq 0$, необходимо выполнение условия $\beta \geqslant 0,9\pi m_s ctg\beta\partial$
ξ_s	Определение ξ_s дано на с. 651	
Z_c	$Z_c=Z_1+Z_2$	
Z_{min}	$Z_{min}=17$	$Z_{min}=\frac{2f_o\cos\beta\partial}{\sin\alpha_{os}}$

Примечание. Указания к нахождению размеров, определяющих толщину зубьев, даны на с. 652.

Лабораторная работа №6

ИЗВЛЕЧЕНИЕ ЗНАНИЙ ЭКСПЕРТА И ПРЕДСТАВЛЕНИЕ ИХ ВО ФРЕЙМОВОЙ СИСТЕМЕ, РАЗРАБОТАННОЙ СРЕДСТВАМИ TURBO PROLOG 2.0

Цель работы:

Приобретение студентами навыков структурирования инженерных знаний, извлечение экспертных знаний и представление знаний в экспертных системах фреймового типа.

6.1. Методы приобретения знаний

Приобретение знаний - это процесс передачи знаний и опыта по решению определенного класса задач от источника информации (которым может быть эксперт, специальная литература, справочно-нормативные сведения, набор данных и др.) в базу знаний экспертной системы.

Знания для загрузки в базу знаний ЭС могут приобретаться несколькими способами :

1) инженер знаний получает знания эксперта в процессе интенсивного интервью и механически переносит их в базу знаний;

2) инженер знаний сам становится экспертом полагаясь на собственный анализ фраз эксперта, выполняемый им в процессе общения с экспертом (кодирует знания эксперта на выбранном им языке описания);

3) эксперт самостоятельно загружает базу знаний ЭС с помощью специальной программы, в какой-тото мере исполняющей функции инженера знаний;

4) специальная программа анализирует источник знаний (например, базу данных) и сама преобразует их в форму, пригодную для хранения в базе знаний ЭС;

5) специальная программа, способная воспринимать и анализировать текстовый источник (например, книгу) , структурирует полученную информацию и помещает ее в базу знаний ЭС.

Первые три метода отнесены к ручным методам . Их классификация приведена на рис. 6.1.

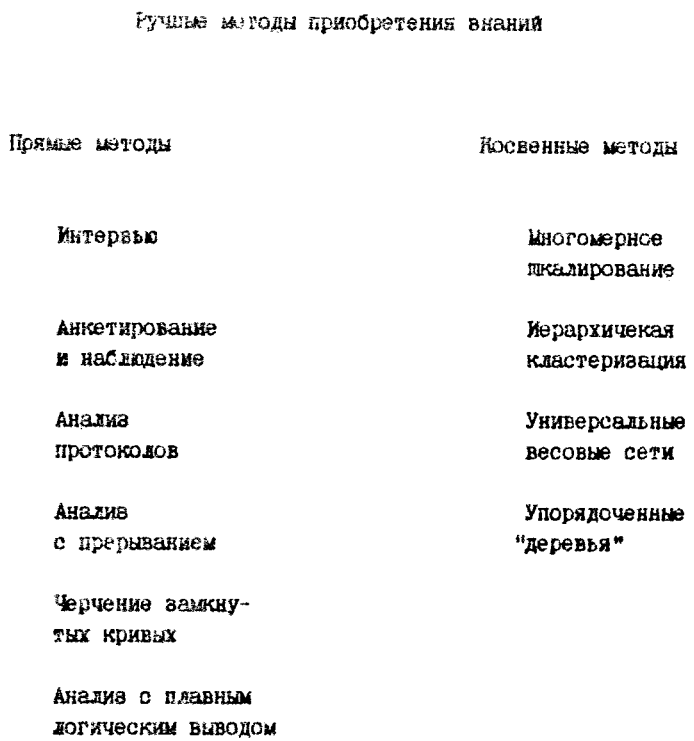

Рис. 6.I. Классификация ручных методов приобретения знаний

В современных экспертных системах разрабатывается " Интерфейс эксперта " - программа, которая предназначена заменить инженера знаний с максимально возможным переносом его положительных качеств и получением ряда новых свойств, среди которых можно отметить следующие:

1. В значительной степени исключаются отрицательные психологические аспекты в общении эксперта и инженера знаний.
2. В тех случаях, когда эксперт выполняет диагностику только на основе опыта и интуиции, он получает возможность в какой-то степени систематизировать свои знания и построить модель, несущую, возможно, новую для него информацию о предметной области. Иногда интерфейс эксперта помогает эксперту формулировать чисто интуитивные понятия и строить из них систему , пригодную для описания.
3. Эксперт получает возможность самостоятельно загружать базу знаний неограниченное количество раз и выполнять пробную экспертизу с помощью каждого варианта базы знаний до тех пор, пока результат не будет полностью его удовлетворять.

Существуют десятки различных подходов к разработке интерфейса эксперта, однако почти все они используют прямые или косвенные методы получения знаний и их различные комбинации.

Интерфейс эксперта в общем случае состоит из трех основных частей, реализующих соответственно:

1) загрузку знаний в базу знаний;
2) корректировку (исправление ошибок или расширение базы знаний);
3) верификацию (т.е. проверку на правильность работы) загруженной базы .

Интерфейс эксперта обеспечивает в то же время связь эксперта с базой знаний. Поэтому алгоритм работы интерфейса в значительной степени определяется формой представления знаний в базе знаний экспертной системы. Во многих случаях интерфейс эксперта является составной частью системы управления базой знаний.

В лабораторной работе используется база знаний экспертной системы фреймового типа и система управления базой знаний, поддерживающая фреймовую структуру.

6.2. Представление знаний фреймами

Фреймовые функции похожи на таблицы или вопросники (анкеты). Они часто представляются как таблично-данные структуры, в которых вся информация в данных категориях сгруппирована вместе.

Подобно входам в таблицу фреймы могут иметь номера (имена) слотов или мест, где информация может быть сохранена. Например, чтобы представить описание самолета, можно иметь слоты для всех категорий, таких как цвет, длина, тип двигателя и крыла, скорость и т.д.

Другой важной особенностью представления фреймом является то, что слоты могут иметь неполное описание, т.е. это способ представления знаний, при котором нет необходимости описывать в деталях все факты о данном объекте. Польза фреймового описания заключается в том, что оно поддерживает работоспособность интеллектуальной системы, когда не все знания имеются в наличии.

В слотах могут упоминаться другие фреймы. Поэтому один фрейм может ссылаться на другой фрейм, который, в свою очередь, может ссылаться на следующий фрейм.

Фреймы могут быть динамически соединены с другими фреймами; таким образом представляется иерархия знаний о проблеме.

Фрейм состоит из поля для хранения основной информации о самом фрейме, такой как название, автор и употребление . " Родительское " поле содержит имя фрейма, которое ссылается на это поле. Поле уровня показывает уровень иерархии этого фрейма. В добавок каждый фрейм имеет слоты, и нет предела номерам слотов, которые фрейм может иметь. Слоты также имеют поддерживающие поля или атрибуты. Эти поля помогают определить или описать значение (величину) слота.

6.3. Построение фреймов в TURBO PROLOG 2.0

На рис .6.2 показано иерархическое представление компьютерной системы. Рассмотрим на этом примере построение простого фреймового представления.

Первый шаг заключается в определении необходимого сохраняемого формата. Один из путей - это использование большого сложного предложения, чтобы создать все фреймы и слотовые части. Этот путь ис-

пользует эффективно память. Второй путь облегчает сохранение и применение описания и заключается в использовании двух предложений: " frame_list "-для сохранения фреймов и " slot_list "-для сохранения слотов. Тогда предложение для создания фреймов " new_frame " можно определить следующей пролог-программой:

```
database
    frame_list(string,string,integer,string)
predicates
     new_frame(sting,string,string)
clauses
  new_frame(F_name,_,_) :-
      frame_list(F_name,_,_,_),
       write("\nError - Frame: ",F_name,
          "Already Exists ").
   new_frame(F_name,Par_name,Comment) :-
       frame_list(Par_name,_,Top_level,_),
       F_level = Top_level + 1,
       assertz(frame_list(F_name,Par_name,
       F_level,Comment)).
   new_frame(F_name,_,_) :-
       write("\nError - Possibly invalid Parent
           Name: ",F_name).
```

Аргументами этого предложения являются: имя фрейма, имя родительского фрейма и поле комментария, которое может быть использовано, чтобы описать автора, пользователя или другую информацию о фрейме.

Поле комментария используется как помощь для обслуживания базы знаний. Имя родительского фрейма обеспечивает обратную связь к предыдущему фрейму, таким образом, создается иерархия структуры знаний.

Первое предложение " new_frame " определяет, что фрейм уже существует.

Второе предложение " new_frame " первоначально показывает, что родительский фрейм существует, и затем оно увеличивает фреймовый уровень. Эта информация используется, чтобы внести новый фрейм.

Последнее предложение " new_frame " в описании компьютерной сети (см . рис .6.2):

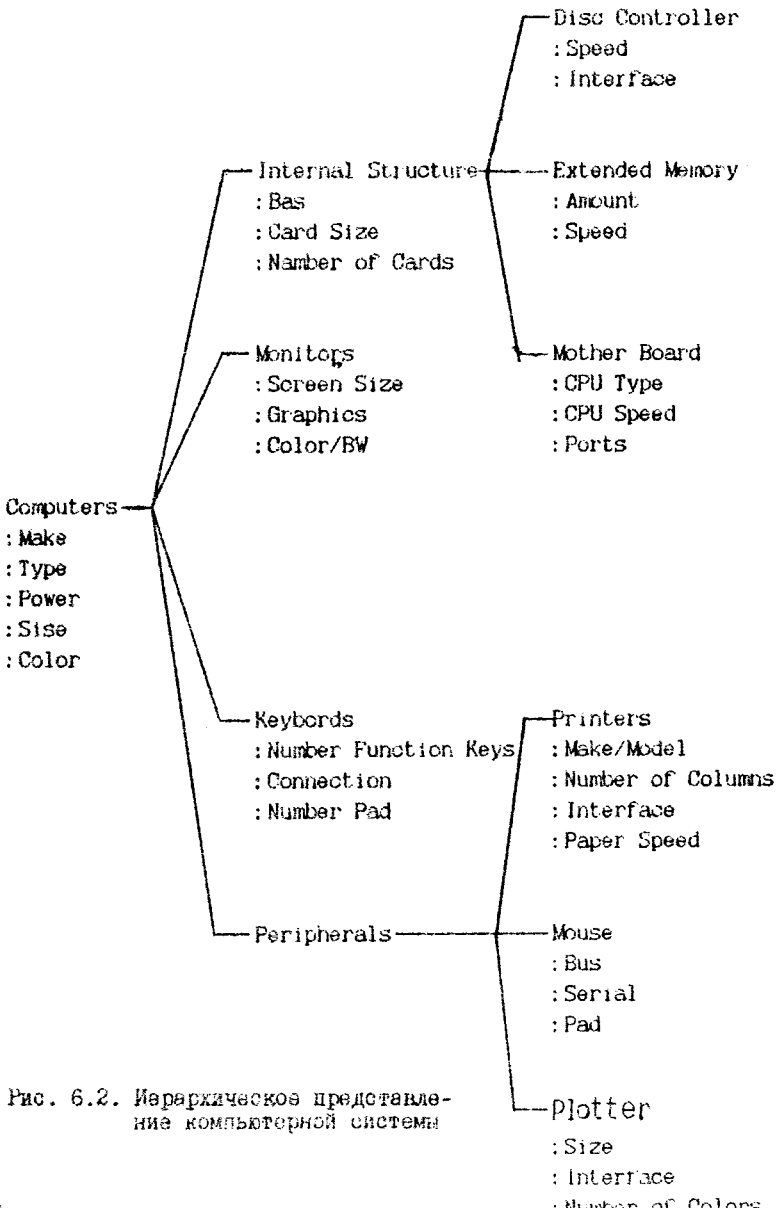

Рис. 6.2. Иерархическое представление компьютерной системы

```
new_frame(computers,top,"Top level of hierarchy").
new_frame(monitors,computers,"CRT Screen").
new_frame(peripherals,computers,"There are many.").
new_frame(mouse,peripherals,"Not on all systems.").
```

Первое заявление создает фрейм с названием " computers ". Этот фрейм является головой фреймовой системы . Для него имя родителя принято " top ". Аргумент комментария используется, чтобы внести примечания об этом фрейме.

Чтобы поддержать фреймовую иерархическую структуру, используются следующие два предложения. Одно инициализирует базу знаний, пока другое вносит в список структуру:

```
predicates
    init
    list(symbol)
clauses
 init :-
     retract(frame_list(_,_,_,_)),
     fail.
 init :-
     assertz(frame_list(top,myself,0,
             "Top of Structure")).
 list(frames) :-
     makewindow(1,1,7, "FRAME LIST",1,1,20,6),
     write("\nFrame\t\tParent\t\tLevel"),
     frame_list(F_name,Par_name,F_level,_),
     write("\n",F_name,"\t",Par_name,"\t",
     F_level),
     fail.
 list(frames) :- nl,
     write("\nPress any key to continue."),
     readchar(_).
```

" Init " предложение передвигает и вносит в конец списка (" end_of_structure ") фрейм.

Этот первый внесенный фрейм представляет вершину фреймовой иерархии. Все другие фреймы линкуются прямо или косвенно к этому фрейму.

Предложение " list " является инструментом для просмотра всех установленных фреймов. Каждый фрейм показывается в открытом окне с именем его родителя и его фреймовым уровнем. Последнее " list " предложение инициализируется после просмотра всей фреймовой структуры.

Структура слота определяется предложением " slot_list ".

```
domains
 slist = symbol *
 vlist = string *
database
 slot_list(string,string,symbol,integer,
 integer,vlist,string,string,string).
```

Предложение " slot_list " - это структура, содержащая номера аргументов, таких как имя фрейма, имя слота, величина слота и комментарии. Аргумент комментария имеет некую цель, как цель, найденная в предложении " new_frame ". В этом примере имеются четыре типа слотовой величины: открытая, фреймовая, числовая, символьная. Одна из них сохраняется как слотовый тип. Слот типа integer показывает, что новая слотовая величина должна быть сравнена с верхним и нижним пределами, определенными для этого слота. Для слота символьного типа " legal_val " аргумент содержит список кандидатов слотовых величин (значений). Содержатся только величины, которые позволяют управлять этим слотом. Открытый слотовый тип не имеет ограничений, так что любая величина может быть расположена в этом типе слота. Следующий слотовый тип есть фрейм, который позволяет слоту указать на другой фрейм. Это обеспечивает направление линкования структуры для базы знаний:

```
predicates
          member (string,vlist)
          member (symbol,slist)
        new_slot(string,string,symbol,integer,integer,
                 vlist,string,string)
        slot_type(symbol)
database
        slot_list(string,string,symbol,integer,integer,
                  vlist,string,string,string)
predicates
         new_slot(string,string,symbol,integer,integer,
                  vlist,string,string)
clauses
        new_slot(F_name,S_name,_,_,_,_,_,_):-
           slot_list(F_name,S_name,_,_,_,_,_,_),
        write("\nError_Slot:",S_name,"Already Exist",
              "In This Frame:",F_name).
        new_slot(F_name,S_name,S_type,Up_val,Low_val,
      Legal_val,Default,Comment):-
           frame_list(F_name,_,_,_),!,
       slot_type(S_type),
       assertz(slot_list(F_name,S_name,S_type,Up_val,
            Low_val,Legal_val,Default,Default,Comment)).
       new_slot(F_name,_,_,_,_,_,_,_):-
          write("\nError_Frame:",F_name,
                "Does not Exist.").
        slot_type(S_type):-
          member (S_type,[open,frame,integer,symbol]).
        slot_type(S_type):-
          write("\nError - Invalid slot type:",S_type),
          fail.
        member(Value,[Value|_]).
          member(Value,[_|Rest]):-!,
          member(Value,Rest).
```

До внесения нового слота первое предложение " new_slot " проверяет, не существуют ли уже эти фрейм и слот. Если это-единственный в своем роде слот, тогда следующее предложение " new_slot " проверяется, чтобы подтвердить существование определенного фрейма. Если-нет, тогда второе предложение ложно, и третье предложение дает сообщение ошибки.

Если фрейм существует, тогда второе " new_slot " предложение инициализирует " slot_type ". Это предложение решает (определяет), действителен ли определенный тип слота. Если тип слота действителен, тогда " slot_list " предложение вносится в базу знаний. Если-нет, тогда просматривается сообщение ошибки, и предложение " slot_type " отсутствует.

В предложении " new_slot " оператор cut помещается после предложения " frame_list ". Этот оператор предотвращает неудачу предложения " slot_type ", вызывающую backtracking предложения " new_slot ". Если имеет место возврат, третье предложение " new_slot " покажет сообщение об ошибке .

Каждый слот содержит недостающую величину (default). Как ранее упоминалось, эта особенность изображается фреймами знаний, когда знания в некоторых областях отсутствуют или неполные.

Следующие примеры демонстрируют использование предложения " new_slot ":

```
new_slot(keyboards,function_keys,integer,32,0,[ ],
   "16","Number of Function keys").
new_slot(keyboards,number_pad,symbol,0,0,[yes,no],
         yes,"Is there one?").
new_slot(printers,make_model,open,0,0,[ ],
         toshiba_1350,"What make and model?").
new_slot(peripherals,printer_slot,frame,0,0,[ ],
          " ","Name of frame for this slot.").
```

Для изменения слотовых значений определяются предложения :

```
predicates
     new_slot_value(string,string,string)
```

```
    new_slot_value_1(symbol,string,string,string)

clauses
    new_slot_value(F_name,S_name,Value):-
        slot_list(F_name,S_name,S_type,_,_,_,_,_),
        new_slot_value_1(S_type,F_name,S_name,Value).

    new_slot_value(F_name,S_name,_):-
        write("\nError-Frame:",F_name,"or slot:",
            S_name,"Does Not Exists.").

    new_slot_value_1(open,F_name,S_name,Value):-
        slot_list(F_name,S_name,S_type,Up_val,Low_val,
            Legal_val,Default,Comment),
        retract(slot_list(F_name,S_name,S_type,_,_,_,_,_)),
        assertz(slot_list(F_name,S_name,S_type,Up_val,
            Low_val,Legal_val,Default,Value,Comment).

    new_slot_value_1(integer,F_name,S_name,Value):-
        slot_list(F_name,S_name,S_type,Up_val,Low_val,
            Legal_val,Default,_,Comment),
        str_int(Value,Int_value),
        Int_value>=Low_val,
        Int_value<=Up_val,
        retract(slot_list(F_name,S_name,S_type,_,_,_,_,_)),
        assertz(slot_list(F_name,S_name,S_type,Up_val,
            Low_val,Legal_val,Default,Value,Comment)).

    new_slot_value_1(symbol,F_name,S_name,Value):-
        slot_list(F_name,S_name,S_type,Up_val,Low_val,
            Legal_val,Default,_,Comment),
        member(Value,Legal_val),
        retract(slot_list(F_name,S_name,S_type,_,_,_,_,_)),
        assertz(slot_list(F_name,S_name,S_type,Up_val,
            Low_val,Legal_val,Default,Value,Comment)).

    new_slot_value_1(frame,F_name,S_name,Value):-
```

```
        slot_list(F_name,S_name,S_type,Up_val,Low_val,
            Legal_val,Default,_,Comment),
        frame_list(Value,_,_,_),
    retract(slot_list(F_name,S_name,S_type,_,_,_,_,_,_)),
    assertz(slot_list(F_name,S_name,S_type,Up_val,
        Low_val,Legal_val,Default,Value,Comment)).

    new_slot_value_1(_,F_name,S_name,Value):-
        write("/nError - Invalid value:",Value,"For",
        "frame:",F_name,",And Slot:",S_name),nl.
```

Первое " new_slot_value " предложение определяет, существует ли фрейм и слот. Если-нет, высвечивается сообщение об ошибке, и предложение завершается неудачно. Если слот существует, тип слота устанавливается, и инициализируется предложение " new_slot_value_1 " . В зависимости от типа слота используется одно из четырех " new_slot_value_1 " предложений.

Первый аргумент предложения " new_slot_value_1 " соответствует типу, присвоенному слоту.

Числовая версия проверяет, определена ли новая величина от верхнего до нижнего пределов.

Символьная версия использует предложение " member ", чтобы подтвердить, что символьная величина определена в списке легальных величин. Тип фрейма соответствует, если величина является именем действительного фреймового типа. Если обнаруживается ошибка, выводится сообщение. Предложение процесса открывает тип слота, и данные вносятся, если не обнаружено никакой ошибки. Некоторые образцы " new_slot_value " имеют вид:

```
new_slot_value(keyboards,function_keys,"24").
new_slot_value(keyboards,connection,"direct").
new_slot_value(keyboards,number_pad,"no").
new_slot_value(peripherals,printer_slot,"printers").
new_slot_value(printers,make_model,"Okidata").
```

Информация из слота возвращается предложением "get_slot_value". Это предложение возвращает значения величины слота определенного

фрейма и имя слота. Если фрейм и слот не существует, то выводится сообщение об ошибке. Ниже представлен код для " get_slot_value ":

```
        predicates
get_slot_value(frame_name,slot_name,value)
        clauses
get_slot_value(F_name,S_name,Value):-
    slot_list(F_name,S_name,_,_,_,_,_,Value,_),!.
get_slot_value(F_name,S_name,""):-
    write("\nError-Frame:",F_name,"And slot:",
    S_name, "Do not exist.\n").
```

Наконец, два предложения необходимы, чтобы поддержать фреймовое представление . Предложение " delete_slot " использует те же приемы, чтобы подтвердить удаляемый слот и фрейм до удаления слота. Предложение " help " суммирует опции, имеющиеся для представления знаний фреймами :

```
                / * Frame System * /

domains

    slist=symbol*
    vlist=string*

 database

    frame_list(string,string,integer,string)
    slot_list(string,string,symbol,integer,integer,
              vlist,string,string,string)
predicates

    delete_slot(string,string)
    get_slot_value(string,string,string)
    help
    init
```

```
    list(symbol)
    member(string,vlist)
    member(symbol,slist)
    new_frame(string,string,string)
    new_slot(string,string,symbol,integer,integer,
        vlist,string,string)
    new_slot_value(string,string,string)
    new_slot_value_1(symbol,string,string,string)
    slot_type(symbol)

clauses
    init :-
        retract( frame_list(_,_,_,_)),
        fail.
    init :-
        retract( slot_list(_,_,_,_,_,_,_,_,_)),
        fail.
    init :-
        assertz(frame_list(top,myself,0,
            "Top of Structure")).
    new_frame(F_name,_,_):-
        frame_list(F_name,_,_,_),
        write("\nError - Frame: ", F_name,
            " Already Exists.").
    new_frame(F_name,Par_name,Comment) :-
        frame_list(Par_name,_,Top_level,_),
        F_level = Top_level + 1,
        assertz(frame_list(F_name,Par_name,F_level,Comment)).
    new_frame(F_name,_,_) :-
        write("\nError - Possibly invalid Parent Name: ",
            F_name).
     new_slot(F_name,S_name,_,_,_,_,_,_) :-
        slot_list(F_name,S_name,_,_,_,_,_,_,_),
        write("\nError - Slot:",S_name,"Already Exists",
            "In This Frame:",F_name).
    new_slot(F_name,S_name,S_type,Up_val,Low_val,
        Legal_val,Default,Comment) :-
```

```
    frame_list(F_name,_,_,_),!,
    slot_types(S_type),
    assertz(slot_list(F_name,S_name,S_type,Up_val,
        Low_val,Legal_val,Default,Default,Comment)).

new_slot(F_name,_,_,_,_,_,_,_) :-
    write("\nError - Frame:",F_name,
        "Does Not Exists.").
new_slot_value(F_name,S_name,Value) :-
    slot_list(F_name,S_name,S_type,_,_,_,_,_),
    new_slot_value_1(S_type,F_name,S_name,Value).
new_slot_value(F_name,S_name,_) :-
    write("\nError - Frame:",F_name," or slot:",
        S_name,"Does Not Exists.").
new_slot_value_1(open,F_name,S_name,Value) :-
   slot_list(F_name,S_name,S_type,Up_val,Low_val,
   Legal_val,Default,_, Comment ),
   retract(slot_list(F_name,S_name,S_type,_,_,_,
          _,_,_)),
   assertz(slot_list(F_name,S_name,S_type,Up_val,
       Low_val,Legal_val,Default,Value,Comment)).
new_slot_value_1( integer,F_name,S_name,Value) :-
    slot_list(F_name,S_name,S_type,Up_val,Low_val,
        Legal_val,Default,_, Comment ),
        str_int(Value,Int_value),
        Int_value>=Low_val,
        Int_value<=Up_val,
        retract(slot_list(F_name,S_name,S_type,_,_,_,
        _,_,_)),
        assertz(slot_list(F_name,S_name,S_type,Up_val,
          Low_val,Legal_val,Default,Value,Comment)).
 new_slot_value_1( symbol,F_name,S_name,Value) :-
     slot_list(F_name,S_name,S_type,Up_val,Low_val,
        Legal_val,Default,_, Comment),
     member(Value,Legal_val),
     retract(slot_list(F_name,S_name,S_type,_,_,_,
          _,_,_)),
```

```
        assertz(slot_list(F_name,S_name,S_type,Up_val,
        Low_val,Legal_val,Default,Value,Comment )).
new_slot_value_1(frame,F_name,S_name,Value) :-
    slot_list(F_name,S_name,S_type,Up_val,Low_val,
              Legal_val,Default,_,Comment),
    frame_list(Value,_,_,_),
    retract(slot_list(F_name,S_name,S_type,_,_,_,
            _,_,_)),
     assertz(slot_list(F_name,S_name,S_type,Up_val,
         Low_val,Legal_val,Default,Value,Comment)).
new_slot_value_1(_,F_name,S_name,Value) :-
    write("\nError - Invalid value:",Value,"For",
       "frame:",F_name,",And Slot:",S_name),nl.
slot_types(S_type) :-
    member(S_type,[open,frame,integer,symbol]).
slot_types(S_type) :-
    write("\nError - Invalid slot type:",S_type),
    fail.
get_slot_value(F_name,S_name,Value) :-
    slot_list(F_name,S_name,_,_,_,_,_,Value,_),!.
get slot_value(F_name,S_name,"") :-
    write(""\nError - Frame:",F_name,"And Slot:",
        S_name,"Do Not Exist.\n").
delete_slot(F_name,S_name) :-
    slot_list(F_name,S_name,_,_,_,_,_,_,_),
    retract(slot_list(F_name,S_name,_,_,_,_,
            _,_,_)).
delete_slot(F_name,S_name) :-
    write("\nError - Frame:",F_name,"And Slot:",
       S_name,"Do Not Exist.").
member(Value,[Value|_]).
member(Value,[_|Rest]) :-!,
   member(Value,Rest).
list(frames) :-
   makewindow(1,1,7,"FRAME LIST",1,1,20,60),
```

```
    write("\nFrame\t\tParent\t\tLevel"),
    frame_list(F_name,Par_name,F_level,_),
    write("\n",F_name,"\t",Par_name,"\t",F_level),
    fail.
list(frames) :-nl,
    write("\nPress any key to continue."),
    readchar(_).
list(slot) :-
    makewindow(1,1,7,"SLOT LIST",1,1,20,79),
    write("\nFrame\t\tSlot\t\tType\tUpper\tLower\t",
    "Default\tLegal"),
    slot_list(F_name,S_name,S_type,Up_val,Low_val,
              Legal_val,Default,_,_),
    write("\n",F_name,"\t",S_name,"\t",S_type,"\t",
         Up_val,"\t",Low_val,"\t",Default,"\t",
         Legal_val),
    fail.
list(slots) :-nl,
    write("\nPress any key to continue."),
    readchar(_).

list(values) :-
    makewindow(1,1,7,"SLOT VALUES",1,1,20,75),
    write("\nFrame\t\tSlot\t\tValue"),
    slot_list(F_name,S_name,_,_,_,_,_,Value,_),
    write("\n"F_name,"\t",S_name,"\t",Value),
    fail.

list(values) :-nl,
    write("\nPress any key to continue."),
    readchar(_).

list(fcomments) :-
    makewindow(1,1,7,"FRAME COMMENTS",1,1,20,60),
    write("\nFrame Name\tComment"),
    frame_list(F_name,_,_,Comment),
```

```
    write("\n",F_name,"\t",Comment),
    fail.

list(fcomments) :-nl,
    write("\nPress any key to continue."),
    readchar(_).

list(scomments):-
    makewindow(1,1,7,"SLOT COMMENTS",1,1,20,60),
    write("\nFrame Name\tSlot Name\tComment"),
    slot_list(F_name,S_name,_,_,_,_,_,_,Comment),
    write("\n",F_name,"\t",S_name,"\t",Comment),
    fail.

list(scomments) :-nl,
    write("\nPress any key to continue."),
    readchar(_).

help :-
    makewindow(1,1,7,"HELP INFORMATION",1,1,20,79),
    write("\ninit - Clears old frames,set top frame."),
    write("\nhelp - This message."),
    write("\nlist(X) - lists asserted  values for X = ",
          "\n\tfcomments,scomments,frames,slots,",
          "values."),
    write("\nload(X) - load knowledge base X from",
          "f_data."),
    write("\nnew_frame(frame_name,parent_name,",
          "comment)",
          "- Create new frame."),
    write("\nnew_slot(frame_name,slot_name,slot_type,",
          "upper_limit,lower_limit,\n\tlegal_values, ",
          "default,comment)",
          " - Create new slot."),
    write("\nnew_slot_value(frame_name,slot_name,value",
          ") - Change current slot value."),
    write("\nget_slot_value(fram_name,slot_name,value",
```

```
            ") - Obtains slot value."),
        write("\ndelete_slot(frame_name,slot_name) -",
            "Delete slot."),
        write("\n\nPress any key to continue."),
        readchar(_).

include "f_data.pro".
```

"Include"предикат ссылается на файл, который содержит данные из рис .6.2. Этот файл использует "load" предикат, который выполняет точно установленное число предложений, чтобы построить работу фрейма для этой простой базы знаний. Другие базы знаний могут располагаться в этом фрейме, используя этот же подход. Эти базы знаний могут затем загружаться, используя " load (symbol) " , где "symbol" есть имя ссылаемого другого значения .

Листинг этого файла приведен ниже :

```
predicates
   load(symbol)
clauses

   load(computer) :-init( ),
       new_frame(computers,top,"Top level of hierarchy."),
       new_frame(intern_strs,computers,
           "Internal Structure."),
       new_frame(monitors,computers,"CRT Screen."),
       new_frame(keyboards,computers,""),
       new_frame(peripherals,computers,"There are many".),
       new_frame(mouse,peripherals,"Not on all systems."),
       new_frame(printers,peripherals,"Many Types."),
       new_frame(plotters,peripherals,
           "Interface is important."),
       new_frame(disc_controller,intern_strs,"Floppy."),
       new_frame(extend_memory,intern_strs,"Extended."),
       new_frame(mother_board,intern_strs,"Contains CPU."),
```

```
new_slot(computers,make_model,open,0,0,[ ],
    "IBM","What is Make and Model."),
new_slot(keyboards,function_keys,integer,32,0,[],
    "16","Number of Function Keys."),
new_slot(keyboards,connection,symbol,0,0,[direct,
    fiber_optic],"direct","Keyboard interface."),
new_slot(keyboards,number_pad,symbol,0,0,[yes,no],
    yes,"Is there one?"),
new_slot(printers,no_columns,integer,132,0,[],"80",
    "Number of printer columns."),
new_slot(printers,interface,symbol,0,0,["RS-232",
    "IEEE-488",parallel],"RS-232",
    "Type of printer interface."),
new_slot(printers,paper_feed,symbol,0,0,
    [tractor,carriage],tractor,"Which one?"),
new_slot(printers,make_model,open,0,0,[],
    toshiba_1350,"What make and model?"),
new_slot(extend_memory,amount_mem,integer,640,0,[],
    "64","Amount of memory in K bytes."),
new_slot(peripherals,printer_slot,frame,0,0,[],
    "","Name of frame for this slot."),
new_slot_value(keyboards,function_keys,"24"),
new_slot_value(keyboards,connection,"direct"),
new_slot_value(keyboards,number_pad,"no"),
new_slot_value(peripherals,printer_slot,"printers"),
new_slot_value(printers,interface,"parallel"),
new_slot_value(printers,make_model,"Okidata").
```

6.4. Использование фреймовой системы

Для тестирования фреймов, представленных системой, вводят " load (symbol) " как цель. Это утверждение загружает все данные из файла"f_data.pro." Вводя " list (values) ", вы добудете список всех фреймов, их слотов и их уровней. На экране вы увидите:

```
                    SLOT VALUES

Frame               Slot              Value
computers           make_model        IBM
printers            no_columns        80
printers            paper_feed        tractor
extend_memory       amount_mem        64
keyboards           function_keys     24
keyboards           connection        direct
keyboards           number_pad        no
printers            interface         parallel
printers            make_model        Okidata
peripherals         printer_slot      printers

Press any key to continue.
```

Если вы хотите получить информацию об отдельном слоте, используйте цель " get_slot_value " так,как это сделано в следующем примере:

```
Goal: get_slot_value(printers,no_columns,X)
X=80
1 Solution
Goal:
```

После того, как описанная выше базисная структура установлена, новые фреймы и слоты могут добавляться или слотовые значения могут высвечиваться на дисплее или модифицироваться. Опции для предложений " list ":

values_list_all_slot_value (список всех слотовых величин)
slots_list_all_slots (список всех слотов)
frames_list_all_frames (список всех фреймов)
fcomments_list_all_frame_comments (список всех комментариев фрейма)
scomments_list_all_slot_comments (список всех комментариев слота).

Чтобы сохранить представление знаний как запись в память, используют предикаты " save " и " consult ". Используя "save (filename)", сохраняют добавленные данные. Если вы используете предложение " load ", как описано в этом примере, вы не нуждаетесь в этих предикатах. Однако, если вы динамически вводите много информации на TURBO PROLOG, они могут пригодиться. Другой путь состоит в том, что результирующий файл может быть просмотрен в окне редактора TURBO PROLOG. Этот путь обеспечивает метод для просмотра информации, которая вносилась в течение сеанса работы фреймовой системы.

6.5. Содержание задания по лабораторной работе

1. Осуществить тестирование базы знаний, описанной в п. 6.2 - 6.4, используя приведенные инструкции.

2. Разработать сценарий интервью, позволяющий построить иерархическое представление компьютерной системы, приведенное на рис. 6.2, и определить ее характеристики.

3. Разработать диалоговый монитор, образующий в совокупности с системой управления базой знаний фреймового типа, описанной в п. 6.2 - 6.4, интерфейс пользователя и позволяющий эксперту устанавливать в соответствии со сценарием интервью параметры предложений " new_frame ", " new_slot ", " new_slot_value ", помещаемых в файл "f_data.pro."

4. Разработанный диалоговый монитор подключить к базовой фреймовой системе посредством директивы компилятора "include."

6.6. Содержание отчета

В отчете по лабораторной работе представляется:

1) сценарий интервью эксперта для определения иерархической структуры компьютерной системы и ее характеристик;

2) код программы диалогового монитора эксперта с объяснением его работы в форме комментариев к программе на TURBO PROLOG 2.0.

Лабораторная работа № 7

ПРЕДСТАВЛЕНИЕ ИНЖЕНЕРНЫХ ЗНАНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ СЕМАНТИЧЕСКОЙ СЕТЬЮ

Цель работы:

Приобретение студентами практических навыков моделирования технических систем семантической сетью и ее описание средствами TURBO PROLOG 2.0.

7.1. Семантические сети, как способ представления знаний

Схема семантической сети является более мощным способом представления знаний, чем фреймовая система. В семантических сетях знания представлены набором вершин и дуг. В этой схеме вершина обозначает объект, понятие или событие, а дуга представляет отношение (соотношение) между двумя объектами. Семантические сети обычно структурированы как направленный (ориентированный) граф.

Например, на рис. 7. 1 показана семантическая сеть для представления знаний о самолете. Кружками обозначены вершины графа, а линии, соединяющие вершины, есть дуги графа. Примечательно, что дугами представлены соотношения между объектами.

Логично определить следующие наиболее часто применяемые соотношения:

is-a. "is-a" отношение указывает, что объект - подкласс другого объекта. В примере семантическая сеть показывает, что "самолет - это авиация" и "реактивный (самолет) - это самолет". В этом отношении соотношение "is-a" имеет встроенную иерархию.

has-a. "has-a" отношение используется, чтобы указать, что объект состоит из других частей или свойств. В примере "реактивный самолет имеет реактивный двигатель (мотор)", а простой двигатель имеет пропеллер.

is . "Is" отношение используется, чтобы придать объекту значение. В примере "реактивный двигатель - быстрый (скорый) ".

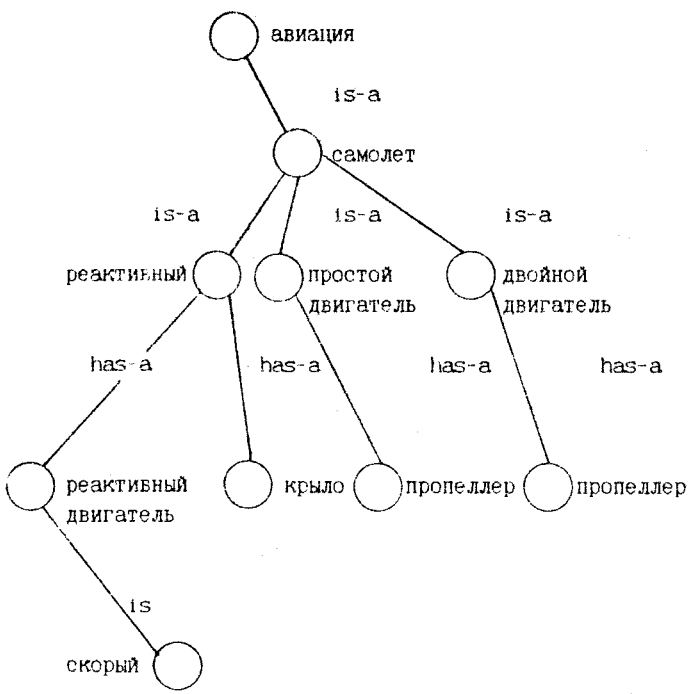

Рис. 7.I. Семантическая сеть

В зависимости от сложности задачи и объектов описываемых семантической сетью, могут быть определены и другие соотношения.

7.2. Определение объектов и отношений между ними в TURBO PROLOG

Рассмотренные в описанном выше примере объекты и отношения могут быть легко выражены в TURBO PROLOG простыми предикатами. Поскольку TURBO PROLOG использует английский алфавит, изобразим схему, представленную на рис. 7. 1 в терминах английского языка (рис. 7.2).

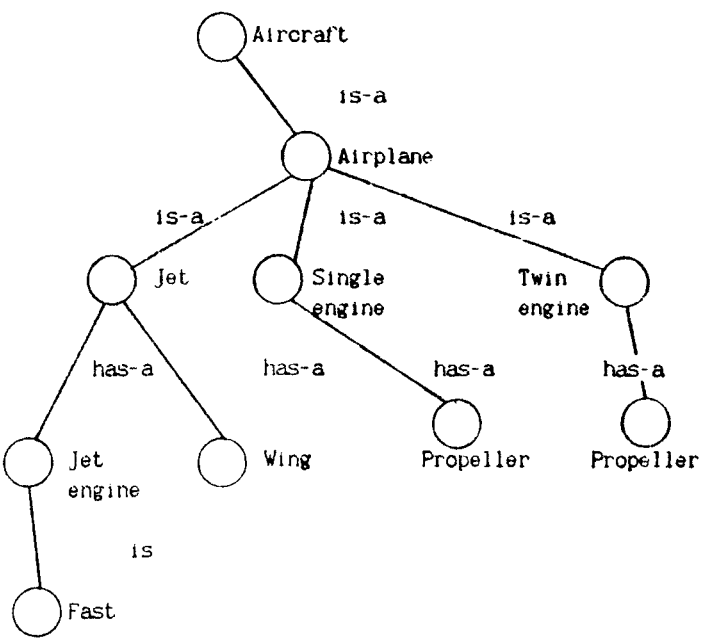

Рис. 7.2. Семантическая сеть в терминах английского языка

Тогда отношение "is-a" может быть записано в форме:

```
is_a ( airplane, aircraft ).
is_a ( jet, airplane ).
```

Отношение "has-a" также может быть написано в форме:

```
has_a ( jet, wing ).
has_a ( single_engine, propeller ).
```

Наконец, отношение "is" может быть написано:

```
is ( jet_engine, fast ).
```

В некоторой степени семантические сети подобны фреймам. Информация или знания закодированы в связи или дуги; таким образом, позволено представление сложных (составных) отношений. В этой связи представление знаний очень действенно (продуктивно) и вполне эффективно. Семантические сети также опираются на понятие иерархии.

Объекты низших классов могут указывать атрибуты объектов высших классов.

База знаний является динамической составляющей программы на ПРОЛОГе, так как знания могут изменяться, дополняться и удаляться в процессе функционирования интеллектуальной системы. Особенностью представления семантической сетью является то, что машина логических выводов непосредственно связана со знаниями, а не существует как инвариантная программная компонента системы, как, например, в продукционной системе представления знаний.

7.3. Содержание задания по лабораторной работе

1. Разработать граф семантической сети, описывающей классификацию металлорежущих станков, их узлы, характеристики узлов, используя отношения is-a, has-a, is.

2. Разработать администратор знаний для формирования базы знаний модели семантической сети, позволяющей устанавливать новые отношения, добавлять объекты и их свойства.

3. Осуществить тестирование разработанной семантической сети.

7.4. Содержание отчета

В отчете представляются:

1) граф семантической сети;
2) листинг администратора знаний;
3) консольный протокол тестирования системы.

Лабораторная работа № 8

РАБОТА В ИНТЕГРИРОВАННОЙ СРЕДЕ ОБОЛОЧКИ ЭКСПЕРТНОЙ СИСТЕМЫ "ИНТЕР-ЭКСПЕРТ". ИЗУЧЕНИЕ РАБОТЫ В СИСТЕМЕ "ИНТЕР-ЭКСПЕРТ"

Цель работы:

Изучение студентами приемов работы в интегрированной системе "ИНТЕР-ЭКСПЕРТ".

8.1. Справочные сведения

ИНТЕР-ЭКСПЕРТ - это общая для искусственного интеллекта (ИИ) и деловой практики среда. Она обеспечивает комбинацию компонентов экспертной системы, четыре интерфейса на выбор, современные вычислительные средства, которые тесно связаны друг с другом. В системе реализован принцип синергизма, т.е. каждой части ИНТЕР-ЭКСПЕРТ доступны все данные, имеющиеся в системе.

8.I.I. Интерфейсы системы

Всеми четырьмя интерфейсами можно пользоваться во время одного и того же сеанса взаимодействия с системой ИНТЕР-ЭКСПЕРТ.

8.I.2. Управление по принципу меню

При использовании интерфейса на основе меню система ИНТЕР-ЭКСПЕРТ покажет всевозможные опции обработки шаг за шагом; при этом всегда доступна дополнительная помощь. Если выбирается желаемая опция, то дальше появляется меню со следующим набором опций. Это продолжается до тех пор, пока пользователь полностью не определит то, что желает, чтобы выполняла система ИНТЕР-ЭКСПЕРТ. Затем система ИНТЕР-ЭКСПЕРТ выполняет затребованную обработку.

8.I.3. Интерфейс естественного языка

Интерфейс естественного языка, который понимает повседневный английский, можно использовать для проведения консультаций с экспертной системой и для доступа информации, хранимой в других вычислительных устройствах. При использовании этого интерфейса система делает запросы на уточнение. ИНТЕР-ЭКСПЕРТ запоминает содержание разговора.

8.I.4. Командный язык

Специалисты предпочитают вступать во взаимодействие с ИНТЕР-ЭКСПЕРТ посредством коротких команд, похожих на английский язык. Если по ходу работы нужна помощь, ее можно получить оперативно.

8.I.5. Процедурная программа

Процедурные программы могут создаваться на заказ, поэтому пользователи могут отвечать на подсказки или выбирать меню. Разработки не ограничены проектированием одного экрана, но могут создавать экраны, которые отвечали бы требованиям каждого конкретного применения.

8.2. Средства разработки экспертных систем

В экспертную систему ИНТЕР-ЭКСПЕРТ входят администратор правил для разработки набора правил экспертной системы и машина логических умозаключений, которая аргументирует набор правил в ответ на запрос , на консультацию.

Администратор набора правил спроектирован таким образом, что простые наборы правил может написать любой человек, даже незнакомый с вычислительной техникой. Процесс написания правил может стать очень простым, если следовать меню и заполнять пробелы. Для создания более сложных экспертных систем требуется человек, знающий ЭВМ.

Машина логических умозаключений - это программное обеспечение, которое выполняет аргументацию, необходимую для решения задачи. Она черпает информацию из хранимой экспертизы по данной проблеме для выдачи заключений и может объяснить свой процесс аргументации.

8.3. Вычислительные средства, используемые для решения задач

В экспертные системы, созданные с помощью ИНТЕР-ЭКСПЕРТ, всегда можно ввести уже известные вычислительные средства, в то же время их можно использовать и вне экспертных систем в каждодневных вычислительных целях. Эти средства прочно объединены в пределах одной программы, что обеспечивает быстрый и простой доступ к ним в ходе работы.

8.3.1. Управление реляционными данными

СУБД ИНТЕР-ЭКСПЕРТ делает простым анализ и доступ к информации, хранящейся в разных источниках.

8.3.2. Спонтанный запрос

Спонтанный запрос в ИНТЕР-ЭКСПЕРТ позволяет выбирать данные из разных таблиц с помощью одного оператора запросов. Запросы могут относиться к ячейкам электронных ведомостей в их выражениях; условия и результаты будут доступны незамедлительно в форме определения ячеек электронных ведомостей.

8.3.3. Электронная ведомость

Обработка электронных ведомостей ИНТЕР-ЭКСПЕРТ дает возможность доступа к любой информации из базы данных и использования любой команды ИНТЕР-ЭКСПЕРТ и любой характеристики ячейки электронных ведомостей.

Правила экспертных систем имеют прямой доступ к ячейкам, или ячейка электронной ведомости может быть определена как результат консультации, выполненный экспертной системой.

8.3.4. Управление форматами

Используя ИНТЕР-ЭКСПЕРТ, можно разрабатывать специальные формы упрощения ввода в поля и вывода из них .

8.3.5. Языки программирования

Язык программирования ИНТЕР-ЭКСПЕРТ позволяет разработчикам приспособить к нуждам потребителей прикладные программы, созданные с помощью ИНТЕР-ЭКСПЕРТ.

8.3.6. Деловая графика

Информация, запрашиваемая с помощью доступных вычислительных средств, может выдаваться в форме работоспособной графики, в которой объединены цвет, типы линий , точки и области заполнения.

8.3.7. Раскрашивание форм

Используя цветовые блоки и специальные эффекты, можно быстро создать формы для конкретных целей.

8.3.8. Обработка текстов

Нет необходимости в процессоре для обработки отдельных текстов. Текстовый процессор ИНТЕР-ЭКСПЕРТ используется для :

создания или редактирования наборов правил;
составления писем и документов;
объединения информации, полученной от консультаций с экспертом, из баз данных, электронных ведомостей и других источников.

8.3.9. Генерация отчетов на заказ

При составлении отчетов используется информация, отобранная из консультаций экспертных систем.

8.3.10. Манипулирование " мышью "

ИНТЕР-ЭКСПЕРТ допускает работу с "мышью", если предпочитают обходиться без клавиатуры .

8.3.11. Дальняя связь

ИНТЕР-ЭКСПЕРТ может иметь связь с удаленными объектами, что разрешает передавать данные или обмениваться ими между машинами, находящимися в разных местах.

8.4. Основные понятия ИНТЕР-ЭКСПЕРТ

В ИНТЕР-ЭКСПЕРТе используются следующие понятия: константы, переменные, выражения, функции, макроопределения, шаблоны.

8.4.1. Константы

Константа - это фактическое значение данных, которое система может использовать во время работы. Система признает четыре типа констант: строки, числа, логические константы и неизвестные константы.

Строка содержит от 1 до 255 символов, заключенных в двойные кавычки (" "), например:

"модель станка" .

Числовая константа - это или целое (без десятичной точки), или десятичное (с точкой) число, имеющее до 14 разрядов. Самый левый разряд может иметь знак " + " или " - ". Десятичная точка считается разрядом. Все остальные разряды - цифры.

Например: - 54 321.32.

Логическая константа - это слово TRUE (истинно) или FALSE (ложь). Неизвестная константа - это неопределяемая в процессе работы константа (UNKNOWN).

8.4.2. Переменные

Основной характеристикой переменной является то, что в разное время у нее могут быть разные значения. Все переменные должны быть оригинально поименованы.

Существует пять типов переменных: символьные, числовые, целочисленные, логические, неизвестные. Класс переменной указывает на ту роль, которую эта переменная может играть во время обработки в системе. Существует четыре класса переменных: переменные полей, переменные ячеек, рабочие переменные и предварительно определенные переменные.

С помощью переменных полей ИНТЕР-ЭКСПЕРТ позволяет определять таблицы, в которых содержатся записи данных. При определении таблицы (команда DEFINE) пользователь определяет одно или несколько полей, характеризующих содержимое записей таблицы и имеющих строго определенный тип. Переменные ячеек используются при обработке электронных ведомостей, состоящих из строк и столбцов. Пересечение строки и столбца называется ячейкой. Ячейка является переменной, так как ее значение и тип могут быть изменены.

Рабочие переменные - это переменные общего назначения, которые используются для хранения результатов вычисления, процедуры, операции ввода и т. д. Рабочей переменной может быть присвоен некоторый фактор уверенности пользователя в значение переменной.

Например, команда :

```
LET BIRSDAY -23,04 CF 88
```

присваивает переменной BIRSDAY значение 23,04 с фактором уверенности 88.

Нечеткие рабочие переменные - это рабочие переменные, имеющие одно или несколько значений, каждое со своим фактором уверенности. Так, переменной SALARY в команде

LET SALARY = (110CF 95,42CF 80,31 CG9)

назначается три целочисленных значения с указанными факторами уверенности.

Предварительно определенные переменные - это переменные, определяемые системой. Они делятся на две категории: переменные, определяющие среду, и утилитные переменные.

8.4.3. Выражения

Выражение состоит из одного или нескольких переменных и (или) констант.

Кроме того, выражение может содержать операторы, показывающие системе ИНТЕР-ЭКСПЕРТ,, как следует оценивать выражения. В зависимости от типа переменных и (или) констант выражения имеет значение типа: строка, число, логическое или неизвестное значение. Выражения первых трех типов могут иметь факторы уверенности. В выражениях применяются следующие операторы: + , - , * , / , ** , MOD.

8.4.4. Функции

Функции заставляют систему выполнять специальный тип операций. Имя функции указывает на характер операции. Функция может иметь один или несколько аргументов.

Например:

arcsin (числовые значения) .

8.4.5. Макроопределения

Макроопределения - это имена, назначаемые пользователем строкам текста. Если пользователь обращается к макроопределениям по имени, то система автоматически выводит текст вместо имени макроопределения. Для макроопределения используется команда:

MAKRO < макроимя >< макротекст >,
где < макроимя > - имя макроопределителя;
< макротекст > - связанный с этим именем текст.

Макроопределения используются для часто повторяющихся вычислений с различными переменными.

8.4.6. Шаблоны

Шаблоны назначаются для полей рабочих переменных, ячеек и выражений.

Шаблоны - это краткий способ указаний на то, как должно выглядеть значение данных. Шаблоны производят форматирование вывода.

8.5. Представление знаний в системе ИНТЕР-ЭКСПЕРТ

Построение экспертной системы с помощью ИНТЕР-ЭКСПЕРТ заключается в построении набора правил. Набор правил состоит из знаний эксперта по аргументированию решения конкретного типа задачи. Набор правил можно построить и ввести посредством команды BUILD, представляющей полный диапазон вариантов управления набором правил. Кроме того, для управления набором правил может быть использован процессор обработки текстов общего назначения.

Общий вид правила ИНТЕР-ЭКСПЕРТ.

IF < условия > THEN < действия >.

Если во время консультации ИНТЕР-ЭКСПЕРТ узнает, что условия верны, то правило запускается. Это значит, что ИНТЕР-ЭКСПЕРТ выполняет действия, указанные в правиле.

Если же условия правила не удовлетворяются, то действия правила не выполняются.

Условия правила часто называются посылкой правила, а действия правила - заключением. Если посылка известна, ИНТЕР-ЭКСПЕРТ пытается определить значение ее переменных. Обычно ИНТЕР-ЭКСПЕРТ ищет другое правило, которое дает значение переменной в посылке

или запрашивает недостающее значение. Как только все переменные становятся известными, система повторно оценивает посылку.

8.6. Логический вывод

Система ИНТЕР-ЭКСПЕРТ предоставляет два метода консультации: прямой и обратной аргументации.

Метод прямой аргументации используется для нахождения решения задачи при условии, что дана реальная ситуация. Каждое правило обрабатывается в направлении от посылки к заключению. Машина логических выводов проверяет, истинна ли посылка правил. Если посылка истинна, правило вступает в действие. Если правила не запускаются, эти операции проверки продолжаются до тех пор, пока не будет найдено решение проблем пользователя или пока данная проблема не будет признана не разрешимой.

Метод обратной аргументации используется для нахождения решения задачи при условии, что дана реальная ситуация. Каждое правило обрабатывается в направлении от заключения к посылке. Машина логических выводов распознает правила, в чьем заключении находится решение проблемы пользователя, после чего определяет, истинна ли посылка. Если посылка истинна, правило может быть выполнено с целью получения решения.

8.7. Содержание задания по лабораторной работе

В данной лабораторной работе необходимо просмотреть демонстрационный пример системы ИНТЕР-ЭКСПЕРТ.

Чтобы запустить демонстрационный пример системы, необходимо набрать

D:\INEX C:\INEX < RETURN > .

Если у вас спросят имя, нажмите < RETURN >.

Сейчас вы находитесь в основном меню системы ИНТЕР-ЭКСПЕРТ.

Чтобы прогнать пример системы, используйте клавишу со стрелкой вниз и выберите из меню последовательно:

Администратор данных < RETURN >.

Операции < RETURN >.

Команды операционной системы < RETURN >.

Смена каталога < RETURN >.

Затем ввести DEMO < RETURN >.

Прогон программы ИНЭК < RETURN >.

На правой стороне экрана появляется список файлов.

Используйте клавишу со стрелкой вниз, чтобы найти имя " GDEMO "; и нажмите RETURN.

На экране воспроизведено меню со следующими вариантами:

1. Экспертные системы.
2. Естественный язык.
3. Администратор данных.
4. Выход.

Сделайте выбор из меню, чтобы предварительно просмотреть экспертные системы, естественный язык и возможности средств управления информацией.

Если вы выберете "Выход", вы вернетесь в меню операционной системы ИНТЕР-ЭКСПЕРТ.

Лабораторная работа № 9

РАБОТА В ИНТЕГРИГОВАННОЙ СРЕДЕ ОБОЛОЧКИ ЭКСПЕРТНОЙ СИСТЕМЫ "ИНТЕР-ЭКСПЕРТ". РАЗРАБОТКА УЧЕБНОЙ ЭКСПЕРТНОЙ СИСТЕМЫ В СРЕДЕ "ИНТЕР-ЭКСПЕРТ"

Цель работы:

Приобретение студентами навыков разработки экспертных систем в среде ИНТЕР-ЭКСПЕРТ.

9.1. Справочные сведения

С помощью системы ИНТЕР-ЭКСПЕРТ можно как создавать, так и использовать экспертные системы (ЭС). В связи с тем, что в систему встроены средства, позволяющие делать логические выводы, процесс программирования при создании ЭС исключается. Используется просто команда BUILD. Она служит для составления набора, содержащего сведения об определенной области применения. При дальнейшем использовании команды CONSULT с целью нахождения решения определенной проблемы ИНТЕР-ЭКСПЕРТ обратится к специальным сведениям, содержащимся в наборе правил, и получит решение. После консультации можно с помощью команд HOW и WHY заставить ИНТЕР-ЭКСПЕРТ объяснить, каким образом найдено решение. С помощью команды BUILD вы можете формализовать знания как группу правил внутри набора правил, а также для модификации существующих наборов правил. Набор правил должен быть составлен до того, как вы сможете запросить ИНТЕР-ЭКСПЕРТ обратиться к нему.

Для составления и модификации набора правил вместо BUILD можно использовать команду TEXT. Наборы правил могут компилироваться командой COMPILE.

9.1.1.Набор правил

Набор правил - это совокупность правил, к которым можно обратиться за советом. Каждый набор правил имеет имя, которое может содержать до 8 символов. Первый символ имени должен быть буквенным. Имя набора правил должно отличаться от имен таблиц данных, полей, переменных. Оно не должно быть ключевым словом ИНТЕР-ЭКСПЕРТ.

Набор правил составляется либо с помощью команды BUILD, либо TEXT.

Набор правил хранится в файле, имя которого состоит из имени набора правил плюс расширение RSS. Это называется исходной верси-

ей набора правил. После компиляции источника набора правил результат хранится в файле, имя которого составлено из имени набора правил плюс расширение RSC. Это называется компилированной версией набора правил. Процесс компиляции не влияет на исходный файл (RSS).

Во время запроса консультации используется компилированная версия (RSC) набора правил.

9.I.2. Команда BUILD

Введите: BUILD < Набор правил >, где:

< Набор правил > - это новое имя набора правил, который вам надо составить, или существующий набор правил, который необходимо модифицировать. Если необязательное имя < Набора правил > опущено, то система ИНТЕР-ЭКСПЕРТ подскажет вам имя набора правил.

ИНТЕР-ЭКСПЕРТ отвечает:

Определение
Инициализация
Правила
Переменные
Завершение
Печать
Выход ,

представляя управляющее, основное меню набора правил, состоящее из 7 опций.

Если курсор стоит на нужной вам опции, нажмите клавишу ENTER.

С помощью опции " Определение " можно определить и изменить цель набора правил, коды доступа или окно " Why ".

Опция " Инициализация " позволяет работать с командами инициализации набора.

Выбрав опцию " Правила ", можно просмотреть, создать, отредактировать и удалить правила.

Опция " Переменные " позволяет осуществить то же с переменными набора правил.

С командами завершения набора правил можно работать, используя опцию " Завершение ".

Если необходимо получить печатный вариант исходного текста набора правил,- пользуйтесь опцией " Печать ".

Опция " Выход " дает вам возможность сохранить источник набора правил, компилировать его или завершить работу с набором правил.

9.I.3. Команда TEXT

Введите TEXT < Имя файла >,
где < Имя файла > имеет расширение RSS и является именем файла, который содержит или будет содержать исходную версию набора правил.

В текстовом файле набор правил содержит 7 четких разделов:

раздел инициализации;
раздел цели;
раздел правил;
раздел переменных;
раздел завершения;
раздел окна;
раздел доступа.

В наборе правил может быть несколько разделов правил и несколько разделов переменных. Все разделы и правила должны появляться вместе, и среди них не должны появляться разделы других типов.

Набор правил заканчивается словом END.

В наборе правил не имеет значения, с какого столбца начинается слово и сколько места остается между словами. Буквы могут быть в символах верхнего и нижнего регистров. Кроме того, в любое мес-

то в тексте можно ввести комментарии, заключив их в /* и */.

Восклицательный знак может использоваться для того, чтобы оставшаяся часть строки текста игнорировалась.

За создание работающей экспертной системы несут ответственность предложения цели и правила. Каждое предложение начинается с ключевого слова, за которым следует двоеточие; после них идут операторы предложения.

9.I.4. Предложение имени набора правил (RULESET)

Предложение имени начинается с RULESET : , после чего идет действительное имя.

Например :

```
RULESET : MYEXPERT .
```

Таким образом, MYEXPERT - имя набора правил.

Предложение цели правил (GOAL)

Предложение цели представляет имя переменной цели. Оно начинается с GOAL : , после чего следует имя.

Например :

```
GOAL : MYGOAL ,
```

где MYGOAL - имя переменной цели.

Система ИНТЕР-ЭКСПЕРТ начинает консультацию с данным набором правил. изучая значение переменной цели. Если значение неизвестно, ИНТЕР-ЭКСПЕРТ ищет значение в наборе правила для того, чтобы определить значения переменной.

9.1.5. Предложение правила
(RULE)

Предложения правила содержат действительное правило; таких правил может быть в неограниченном количестве. Вместе с тем, набор правил должен включать в себя, по крайней мере, одно правило.

Каждое правило начинается с RULE , за которым следует имя правил и еще ряд предложений.

Например :

RULE : PROF ,

где PROF - имя правила.

Наиболее важными предложениями являются предложения IF, THEN, REASON.

1. Предложение IF.

В предложении IF содержится посылка правила. Оно начинается с IF, за которым следует выражение, которое может быть истинным, ложным или неизвестным.

Например :

IF : SALES > EXPEN .

Если SALES выше, чем EXPEN, тогда посылка истинна.

Если SALES ниже, чем EXPEN, посылка ложна.

Если SALES или EXSPEN неизвестны, посылка считается неизвестной.

2. Предложение THEN.

В предложении THEN содержится заключение правила. Оно начинается с THEN. После THEN может идти любое количество операторов ИНТЕР-ЭКСПЕРТ.

Например :

```
THEN : PROF = SALES - EXPEN.
```

Если посылка в предложении IF истина, ИНТЕР-ЭКСПЕРТ выполняет операторы в заключении. Это известно как запуск правила.

Если посылка ложна, операторы заключения не выполняются и правило не запускается.

Если посылка неизвестна, ИНТЕР-ЭКСПЕРТ пытается определить значение ее переменных. Обычно ИНТЕР-ЭКСПЕРТ ищет другое правило, которое дает значение переменных в посылке или запрашивает значения. Как только все переменные становятся известными, система ИНТЕР-ЭКСПЕРТ повторно оценивает посылку. Если посылка истинна, правило выполняется. Если-нет,- не запускается.

Предложения IF и THEN необходимы в каждом правиле.

9.I.6. Предложение объяснения (REASON)

В предложении REASON содержится объяснение правила. После ключевого слова REASON может идти любой текст.

REASON появляется, когда пользователь ЭС выдает команду WHY или нажимает (^Y). Так как при появлении REASON пользователь не видит самого правила, лучше всего начать предложение REASON с изложения правила и затем добавить любую информацию, которая нужна пользователю.

Длина предложения REASON определяется вашими требованиями.

Присутствие предложения REASON не обязательно, но очень рекомендуется. Без этого предложения пользователь ЭС не может проследить ход процесса аргументации.

9.I.7. Предложение переменной (VARIABLE)

Предложения переменных описывают переменные, которые использует ЭС. Если переменная применяется либо в посылке, либо в заключении правила, она должна быть определена.

Предложение переменной начинается с VARIABLE. После этого

идет имя переменной, за которым следует ряд предложений, самыми важными из которых являются FIND и LABEL.

Например:

VARIABLE : SALES,

где SALES - имя переменной, которую используют одно или несколько правил в наборе.

Предложение LABEL содержит краткое (менее 64 символов) описание переменной . Метка - это то, что видит пользователь, когда команды HOW и WHY показывают переменные, используемые конкретным правилом. Меткой должно быть описание переменной.

Например:

LABEL: Общий объем выпуска за этот год .

Предложение FIND содержит один или несколько операторов ИНТЕР-ЭКСПЕРТ, которые следует использовать тогда, когда необходим ввод значений переменной. Обычно это какая-нибудь форма оператора INPUT, хотя здесь разрешаются все операторы ИНТЕР-ЭКСПЕРТ.

Например:

FIND: INPUT SALES WITH "Введите общий объем выпуска\
за этот год".

Если системе ИНТЕР-ЭКСПЕРТ необходимо знать значение неизвестной переменной, она прежде всего пытается найти правило, которое дает значение переменной.

Если система не находит правило, она выполняет операторы в FIND для того, чтобы выяснить значение у пользователя.

9.1.8. Предложение инициализации (INITIAL)

Раздел инициализации начинается с ключевого слова INITIAL и состоит из одного или нескольких операторов ИНТЕР-ЭКСПЕРТ , где

каждый оператор - любая достоверная команда. Эти команды будут автоматически последовательно выполняться, как только к набору правил обратились за консультацией. В одной строке текста могут инициализироваться несколько операторов команды при условии, что они будут отделяться точкой с запятой. Если оператор команды на одной строке не помещается, строку необходимо закончить знаком косой черты (\ ) и продолжать команду на следующей строке текста.

Например:

```
INITIAL: STAT = FALSE
         SALES = UNKNOWN
         INPUT SALESMA WITH " Введите общий объем \
                              поставок ".
```

В этот момент можно задать любые начальные условия, которые необходимы пользователю для консультации с ЭС.

9.I.9. Предложение завершения (DO)

Предложение завершения содержит операторы ИНТЕР-ЭКСПЕРТ , которые выполняются после того, как значение переменной цели найдено и консультация закончена. Предложение завершения начинается с DO, за которым может следовать любое количество операторов ИНТЕР-ЭКСПЕРТ.

Например:

```
DO: OUTPUT " Ответ: " , MYGOAL .
```

9.I.IO. Завершение текста набора правил (END)

Набор правил заканчивается END: , который должен быть последним оператором в наборе правил.

9.2. Использование функциональных клавиш

Функциональные клавиши и их действие, выполняемое в режиме обработки текста, приведены в табл. 9.1.

Таблица 9.1

Использование функциональных клавиш

Клавиша	Действие, выполняемое в режиме обработки текста
1	2
ESCAPE	Передвигает курсор между областью состояния и областью текста
DEL (^R)	Удаляет символ под курсором
END (^T)	Удаляет текущую строку
(^X)	Смещает курсор на одну строку ниже
END (^B)	Смещает курсор в конец строки
ENTER	Сигнализирует об окончании ввода данных в области состояния
^C	Заканчивает сеанс; возвращается в операционную систему
^L	Воспроизводит на экране выполняемое сообщение
^Q	Вставляет строку перед текущей строкой
INS (^W)	Осуществляет переход от операции ввода символов к операции замены и наоборот
^O	Присоединяет следующую строку к текущей строке
- (^F)	Передвигает курсор к следующему слову в тексте
- (^A)	Передвигает курсор к предыдущему слову в тексте
^Z	Повторно начинает вводить текущую строку или данные области состояния
^HOME (^Y)	Восстанавливает последнюю удаленную строку и вводит ее перед текущей строкой
PgUp (^U)	Передвигает окно визуального отображения вверх экрана

Продолжение табл. 9.1

I	2
TAB (^I)	Передвигает курсор вправо в следующую таблицу табуляции
: (^E)	Передвигает курсор на одну строку вверх
Backspace	Возврат на одну позицию со стиранием

9.3. Содержание задания по лабораторной работе

9.3.I. Описание задачи

В лабораторной работе разрабатывается экспертная система, предназначенная для выбора типоразмера токарного станка.

При составлении плана и методов обработки детали одновременно указывают станок, на котором будет выполняться данная операция, и его характеристику (наименование станка, название завода-изготовителя, модель и основные размеры).

Выбор станка определяется его возможностью обеспечения требований, предъявляемых к обрабатываемой детали по точности размеров, формы и классу чистоты поверхности.

Если указанные требования могут быть выполнены на различных станках, то из них выбирают станок, учитывая следующие факторы:

I) соответствие основных размеров станка размерам обрабатываемой детали;

2) соответствие производительности станка количеству деталей, надлежащих обработке в течение года;

3) степень использования станка по мощности и по времени;

4) себестоимость обработки;

5) отпускную цену станка;

6) реальность приобретения того или иного станка;

7) возможность использования имеющихся станков.

Для решения вопросов по каждому из пунктов требуются технологические знания, которые могут быть представлены в форме про-

дукционных правил, с помощью которых выбирается требуемый станок из базы данных о станках токарной группы и реальных возможностей предприятия.

С целью упрощения поставленной задачи ограничимся рассмотрением серийного производства и сразу же ограничим номенклатуру станков, имеющихся в распоряжении технолога на некотором условном предприятии. В базу данных станков включены станки, сведения о которых представлены в табл. 9.2 , 9.3 , 9.4.

Таблица 9.2

Токарно-винторезные станки

N пп	Цена и техническая характеричтика	Модель станка			
		1А616	1К62	1К625	163
1	Условная цена, руб	1500	2000	2630	3420
2	Наибольший диаметр обработки над станиной, мм	320	400	500	630
3	Расстояние между центрами, мм	750	710, 1000, 1400	1000, 1500, 2000	1400, 2800
4	Наибольший диаметр обрабатываемой заготовки над суппортом, мм	175	220	260	340
5	Наибольший диаметр обрабатываемого прутка, мм	34	36	50	65
6	Пределы числа оборотов шпинделя в минуту	9 - 1800	12,5 - 2000	12,5 - 2000	10 - 1250
7	Пределы подач на один оборот шпинделя				
	1) продольных	0,07-4,16	0,07-4,16	0,11-4,67	0,20-3,05
	2) поперечных	0,035-2,08	0,035-2,08	0,035-1,5	0,07-1,04
8	Мощность электродвигателя, кВт	4	10	10	13

Таблица 9.3

Токарно-револьверные станки

N пп	Цена и техническая характеристика	Модель станка		
		1365	1П365 ★	1П371 ★
1	Условная цена, руб	4610	3800	7360
2	Наибольший диаметр обработки над станиной, мм	-	500	630
3	Наибольшее продольное перемещение револьверного суппорта, мм	725	725	1080
4	Наибольший диаметр обрабатываемой заготовки над суппортом, мм	-	320	420
5	Наибольший диаметр обрабатываемого прутка, мм	80	80	125
6	Пределы чисел оборотов шпинделя в минуту	34-1500	34-1500	20-893
7	---	---	---	---
8	Мощность электродвигателя, кВт	13	13	22

★ - патронные станки.

Таблица 9.4

Токарные многорезцовые автоматы

N пп	Цена и техническая характеристика	Модель станка	
		1А720	1А730
1	Условная цена, руб	3710	4470
2	Наибольший диаметр обработки над станиной, мм	310	410
3	Расстояние между центрами, мм	0-300	200-500
4	Наибольший диаметр обработки над суппортом, мм	200	300
5	Наибольший диаметр обрабатываемого прутка, мм	Обработка из прутка не выполняется	
6	Пределы чисел оборотов шпинделя в мин	146-1400	56-710
7	----	----	----
8	Мощность электродвигателя, кВт	7,5	13

Если параметры заготовки не позволяют осуществить ее обработку на станках из номенклатуры, представленной в табл. 9.2, 9.3, 9.4, следует предусмотреть сообщение пользователя ЭС о необходимости покупки нового станка.

В таблицах каждая из характеристик имеет номер, одинаковый для рассматриваемых трех типов станков. Каждую из характеристик можно представить как некоторую переменную, принимающую различные зна-

чения для различных моделей станков, и использовать эти переменные в продукционных правилах экспертной системы.

Сформулируем знания, требуемые для выбора станка по некоторым из основных учитываемых факторов, ограничив их число с целью упрощения задачи:

Фактор 1. Не учитывается, так как все станки обеспечивают примерно одинаковую экономическую точность и шероховатость обработки.

Фактор 2. Станки применяют:
либо только для обработки из прутка;
либо только для обработки штучных заготовок;
либо для обработки из прутка и штучных заготовок.

Габариты заготовки не должны превышать предельных значений диаметров и длины, представленных в табл. 9.2, 9.3 и 9.4.

Факторы 3 и 5. При решении вопроса о том, какой станок выбрать из ряда возможных станков, нужно произвести технико-экономическое сравнение обработки данной детали на разных станках при заданной производственной программе и принять ту модель станка, которая обеспечивает большую производительность и наименьшую себестоимость обработки.

9.3.2. Задание

Используя сведения, изложенные в руководстве к лабораторной работе, выполнить следующее задание:

1. Сформулировать продукционные правила в объеме, достаточном для определения типоразмера токарного станка.

2. Определить данные, хранимые в базе данных, и данные, вводимые по запросу экспертной системы в процессе диалога с пользователем.

3. Представить продукционные правила в форме набора правил ИНТЕР-ЭКСПЕРТ.

4. Проанализировать работу экспертной системы и составить отчет по лабораторной работе.

9.4. Содержание отчета

В отчете представляется исходная версия набора правил.

Литература

1. Братко И. Программирование на языке ПРОЛОГ для искусственного интеллекта: Пер с англ.-М.: Мир, 1990. -560 с.

2. Доорс Дн. и др. ПРОЛОГ - язык программирования будущего: Пер. с англ. (Предисловия А.Н. Волкова.)-М.: Финансы и статистика, 1990. -144с.

3. Интегрированная система для создания прикладных систем с базами данных и знаний (ИНТЕР-ЭКСПЕРТ): Руководство пользователя.- М., 1979.

4. Космачев И.Г. Карманный справочник технолога-инструментальщика -М.: Машиностроение, 1970. -264с.

5. Маталин А.А. Точность механической обработки и проектирование технологических процессов.- Л., 1970.

6. Стерлинг Л., Шапиро Э. Искусство программирования на языке ПРОЛОГ: Пер. с англ. -М.: Мир, 1990. -235с.

7. Справочник металлиста / Под. ред. С.А. Чернавского.- Т.2: - М.: Машгиз, 1985. - 629 с.

8. Таундсенд К., Фохт Д. Проектирование и программная реализация экспертных систем на персональных ЭВМ: Пер. с англ.

(Предисловия Г.С. Осипова.)-М.: Финансы и статистика, 1990. -300с.

9. Технология обработки конструкционных материалов: Учеб. для машиностр. спец. вузов / П.Г. Петруха, А.И. Марков, П.Д. Беспахотный и др.; Под ред. П.Г. Петрухи - М.: Высш. школа,1991. -512с.

10. ТУРБО-ПАСКАЛЬ 6.0. Руководство по объектно-ориентированному программированию, НПО " Центрпрограммсистем ", - Калинин,1990.

11. Экспертные системы для персональных компьютеров: методы, средства, реализации: Справ. пособие /В.С. Крисевич, Л.А. Кузьмич, А.М. Шиф и др.-Мн.: Выш. школа,1990. -197с.

12. Ящерицын П.И., Еременко М.Л., Жигалко Н.И. Основы резания материалов и режущий инструмент. -Мн.: Выш. школа, 1981. -560 с.

13. Ящерицын П.И. Основы технологии механической обработки и сборки в машиностроении.-Мн.,Выш. школа, 1974.-607с.

14. Kelly Rich and Phillip R.Robinson. Usin Turbo Prolog.- Borland - Osborne/ McGRAW - HILL, 1988.

15. Weiskamp, Keith. Artificial intelligence programming with Turbo Prolog.- John Wiley & Sons, Inc. New York, 1988.

Содержание

Учебное издание

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В САПР

Лабораторные работы (практикум)
для студентов специальности 22.03 -
"Системы автоматизированного проектирования"

В 3-х частях
Ч а с т ь I
ЭКСПЕРТНЫЕ СИСТЕМЫ

Составители: ОВЧИННИКОВ Леонид Степанович
ГОРБАЧЕВА Светлана Владимировна

Редактор Т.А. Палилова. Корректор М.П. Антонова

Подписано в печать 25.02.94.
Формат 60x84 1/16. Бумага тип. № 2. Офсет. печать.
Усл. печ. л. 5,6. Уч.-изд. л. 4,4. Тир. 300. Зак. 154.
Белорусская государственная политехническая академия
Отпечатано на ротапринте БГПА. 220027, Минск, пр. Ф.Скорины, 65.