ЛАБОРАТОРНАЯ РАБОТА № 4

ЭКСПЕРТНЫЕ СИСТЕМЫ НА ПРОЛОГЕ

Цель работы

Целью работы является изучение принципов построения и организации экспертных систем, базирующихся на логике и правилах.

1. Задание на лабораторную работу

На материале лекционного курса и рассмотренного ниже примера экспертной системы изучить методики реализации обоих типов экспертных систем средствами языка Пролог. Реализовать экспертную систему по заданной предметной области в соответствии с номером задания в Таблице 7.1. При этом количество описываемых объектов должно быть не менее 12, а характеризующих их атрибутов – не менее 8. Рассмотреть реализацию экспертной системы как базирующуюся на логике и как базирующуюся на правилах.

Таблица 7.1

Предметная область для экспертной системы

Вариант	Предметная область
1, 5	Микропроцессоры
2, 6	Мобильные устройства
3, 10	Операционные системы
4, 11	Языки программирования
7, 12, 19	Компьютерные игры
8, 15, 20	Компьютерные вирусы
9, 16	Компьютерные сети
13, 17	Алгоритмы сортировки
14, 18	Алгоритмы поиска

2. Краткие теоретические сведения

2.1 Назначение и общая структура экспертных систем

Под Экспертной Системой (ЭС) понимается программа (комплекс программ), моделирующая в некоторой степени работу человека-эксперта в конкретной предметной области. Причем эта область строго ограничена.

Основное назначение ЭС – проведение консультаций в той предметной области, для которой спроектирована данная ЭС.

В составе ЭС выделяется три компонента (рис.7.1):

1. База Знаний (БЗ). БЗ – центральная часть Экспертной Системы. Она содержит совокупность фактов и знаний (правил) для вывода других знаний. Содержащаяся в БЗ информация используется используется ЭС для вывода экспертного заключения во время консультации. Обычно БЗ располагают отдельно от программы, на диске или другом носителе.
2. Механизм Вывода (МВ). МВ содержит описания способов применения содержащихся в БЗ знаний. Во время консультации МВ запускает ЭС в работу, выполняет правила, определяет приемлемость найденного решения и передает результаты в СПИ.
3. Система Пользовательского Интерфейса (СПИ). СПИ есть та часть ЭС, которая взаимодействует с пользователем. В функции СПИ входит: прием информации от пользователя, передача результатов пользователю в наиболее удобной для него форме, объяснение полученных ЭС результатов (выдача справочной информации по выводу результатов).

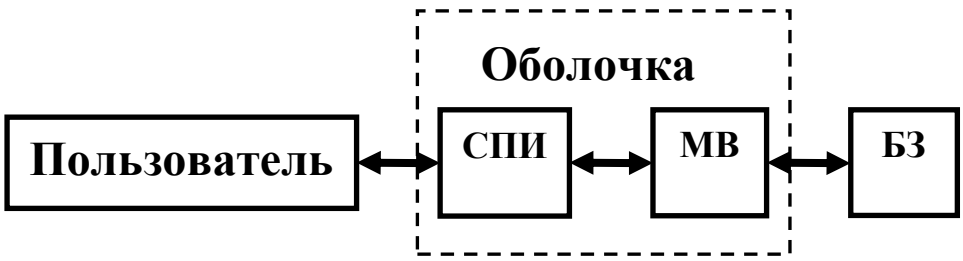

Рис. 7.1. Общая структура ЭС

В зависимости от способов классификации и размещения информации Базы Знаний различают: продукционную, сетевую и фреймовую модели представления знаний.

Сетевая модель основана на представление знаний в виде сети, вершины которой соответствуют понятиям, а дуги — отношениям между ними.

В основе фреймовой модели лежит логическая группировка атрибутов объекта, при этом для хранения и обработки логические группы описываются во фреймах.

Продукционная модель основывается на правилах вида «если – то» и позволяет помещать фрагменты фактического знания в правила языка

Пролог. Именно так строятся ЭС, базирующиеся на правилах. При реализации ЭС, базирующейся на логике, БЗ представляется совокупностью утверждений в виде фактов. Вывод экспертного заключения при этом строится на основе стандартных средств работы со списками.

2.2 Вывод экспертного заключения

Под выводом в ЭС понимается доказательство того, что из множества предположений следует некоторое заключение. Принятая логика получения заключения специфицируется правилами вывода. Вывод осуществляется посредством поиска и сопоставления по образцу.

В ЭС, базирующейся на правилах, запросы пользователя трансформируются в форму, сопоставимую с формой правил БЗ. Механизм вывода инициализирует процесс сопоставления, начиная с «верхнего» правила. Обращение к правилу называется «вызовом». Вызов соответствующих правил в процессе сопоставления продолжается до тех пор, пока не произошло сопоставление или не исчерпана вся БЗ, а значение не найдено. Если МВ обнаруживает, что можно вызвать более одного правила, то запускается процесс разрешения конфликта. При разрешении конфликта приоритет отдается обычно тем правилам, которые более конкретны, либо правилам, которые учитывают больше текущих данных.

В ЭС, базирующейся на логике, трансформированные запросы являются значениями, которые сопоставляются со списками значений, находящихся в утверждениях БЗ.

Внутренние программы унификации Турбо-Пролога и Visual Pro-log’a позволяют решать задачи поиска и сопоставления по образцу без написания дополнительных правил. Как в системе, базирующейся на правилах, так и в системе, базирующейся на логике, пользователь получает ответы на свои запросы в соответствии с заложенной в ЭС логикой. Для программной реализации механизма вывода экспертного заключения достаточно только написать необходимые спецификации.

2.3 ЭС, базирующаяся на правилах, и ее реализация

ЭС, базирующаяся на правилах, позволяет проектировщику строить правила, которые естественным образом объединяют в группы связанные фрагменты знаний. При этом взаимная независимость продукционных правил делает базу правил семантически модульной и способной к развитию.

Реализованная на Турбо-Прологе (или Visual Prolog’e) ЭС на правилах содержит множество правил, которые вызываются посредством входных данных в момент сопоставления. Наряду с множеством правил

МВ ЭС имеет в своем составе интерпретатор, который выбирает и активизирует различные модули системы. Работа этого интерпретатора описывается последовательностью трех шагов:

1. Интерпретатор сопоставляет образец правила с элементами данных в БЗ.
2. Если можно вызвать более одного правила, то для выбора правила используется механизм разрешения конфликта.
3. Выбранное правило применяется для поиска ответа на поставленный вопрос.

Этот трехшаговый процесс является циклическим и называется циклом распознавание-действие. Утвердительный ответ ЭС является результатом выполнения одного из продукционных правил, выбор правила производится в соответствии с входными данными.

Написание на Турбо-Прологе (или Visual Prolog'e) ЭС, базирующейся на правилах, начинается с декларации БД. БД хранит ответы пользователя на вопросы СПИ. Эти данные являются утвердительными или отрицательными ответами. Далее строятся продукционные правила, описывающие фрагменты фактического знания. Пример (для ЭС идентификации и выбора породы собак):

```
dog_is («английский бульдог»): –
   it_is («короткошерстная»),
   positive («имеет», «рост меньше 22 дюймов»),
   positive («имеет», «свисающий хвост»),
   positive («имеет», «хороший характер»), !.

/* Аналогично описываются знания о других породах собак */

...

dog_is («коккер-спаниэль»): –
    it_is («длинношерстная»),
    positive («имеет», «рост меньше 22 дюймов»),
    positive («имеет», «свисающий хвост»),
    positive («имеет», «длинные уши»),
    positive («имеет», «хороший характер»), !.
```

Причем с целью ограничения пространства поиска описывающие связанные фрагменты знаний правила могут бать сгруппированы посредством введения в базу вспомогательных правил для идентификации подкатегорий. К примеру, в ЭС выбора породы собаки это будет правило

it_is, которое идентифицирует породу собаки по признаку принадлежности к группе длинношерстных или короткошерстных:

```
it_is («короткошерстная»): –
  positive («собака собака имеет», «короткую шерсть»), !.

it_is («длинношерстная»): –
  positive («собака собака имеет», «длинную шерсть»), !.
```

В рассматриваемом здесь примере основные данные для выбора породы собаки являются общеизвестными, поскольку выбраны распространенные породы собак. Набор признаков для классификации пород выбран на основании следующих соображений:

- все используемые атрибуты являются необходимыми для идентификации породы;
- ни один из атрибутов не характерен для всех пород одновременно.

Правила МВ сопоставляют данные пользователя с данными в продукционных правилах (правила positive и negative в рассматриваемой ЭС), а также сохраняют «трассу» утвердительных и неутвердительных (отрицательных) ответов (правило remember для добавления в БД утверждений с ответами 1 (да) и 2 (нет)), которые используются при сопоставлении с образцом:

```
/* Механизм вывода экспертного заключения.
xpositive (X, Y) и xnegative (X, Y) – предикаты динамической БД, хранят, соответственно, утвердительные и неутвердительные ответы пользователя на вопросы, которые СПИ задает на основе значений аргументов предиката positive в теле очередного варианта правила dog_is */

positive (X, Y): –
   xpositive (X,Y), !.
positive (X, Y): –
   not(negative (X,Y)), !, ask (X,Y).
negative (X, Y): –
   xnegative (X,Y), !.

/* Консультация. Предикат dlg_Ask создает стандартное диалоговое окно получения ответа пользователя на вопрос Да/Нет,
смотри лабораторную работу № 4 */

  ask (X, Y): –
```

```
concat («Вопрос : », X, Temp),
concat (Temp, « », Temp1), concat (Temp1, Y, Temp2),
concat (Temp2, «?», Quest),
Reply1=dlg_Ask («Консультация», Quest, [«Да», «Нет»]),
Reply=Reply1+1,
remember (X, Y, Reply).
```

/* Занесение ответов пользователя в динамическую БД */

```
remember (X, Y, 1): – !,
   assertz (xpositive (X, Y)).
remember (X, Y, 2): – !,
   assertz (xnegative (X, Y)), fail.
```

2.4 ЭС, базирующаяся на логике, и ее реализация

Здесь БЗ состоит из утверждений в виде предложений логики предикатов. При этом часть утверждений описывают объекты, другая часть – условия или атрибуты, которые характеризуют различные объекты. Количество признаков определяет степень точности классификации. Интерпретатор внутри системы выполняет свои функции на основе следующей схемы:

- система содержит в БЗ предложения, которые управляют поиском и сопоставлением. Интерпретатор сопоставляет эти предложения с элементами данных в БД;
- если существует возможность вызова более одного правила, то для разрешения конфликта система использует возможности механизма внутренней унификации Пролога;
- система получает результаты унификационного процесса автоматически, поэтому они направляются на нужное (логическое) устройство вывода информации.

Так же, как и в ЭС, базирующейся на правилах, данный циклический процесс является процессом распознавание-действие.

Основное отличие структуры ЭС, базирующейся на логике, состоит в описании объектов и атрибутов в виде фактов:

/* Условия-характеристики различных пород.*/

```
cond (1, «короткошерстная порода»).
cond (2, «длинношерстная порода»).
cond (3, «рост меньше 22 дюймов»).
```

```
cond (4, «рост больше 30 дюймов»).
cond (5, «свисающий хвост»).
cond (6, «длинные уши»).
cond (7, «хороший характер»).
cond (8, «вес более 100 фунтов»).

/* Данные о типах породы */

topic («короткошерстная»).
topic («длинношерстная»).

/* Данные о конкретных породах */

rule (1, «собака», «короткошерстная», [1]).
rule (2, «собака», «длинношерстная», [2]).
rule (3, «короткошерстная», «английский бульдог», [3,5,7]).
rule (4, «короткошерстная», «гончая», [3,6,7]).
rule (5, «короткошерстная», «датский дог», [5,6,7,8]).
rule (6, «короткошерстная», «американский фокстерьер», [4,6,7]).
rule (7, «длинношерстная», «коккер-спаниэль», [3,5,6,7]).
rule (8, «длинношерстная», «ирландский сеттер», [4,6]).
rule (9, «длинношерстная», «колли», [4,5,7]).
rule (10, «длинношерстная», «сенбернар», [5,7,8]).
```

Третий аргумент предиката rule – список целых чисел-номеров условий из предложений типа cond. Каждое предложение типа cond задает свое условие-признак из используемых здесь для классификации пород собак. В ЭС, базирующейся на логике, интерпретатор использует эти номера условий, чтобы делать соответствующий выбор.

МВ содержит правила обработки списков атрибутов в описаниях объектов. Посредством применения правила go МВ просматривает утверждения БЗ rule и cond для выяснения с помощью правила check существования или отсутствия подходящих значений данных:

```
/* Начальное правило механизма вывода */

go (_, Mygoal): –
   not (rule (_, Mygoal, _, _)), !,
   concat («Рекомендуемая порода : », Mygoal, Temp),
   concat (Temp, «.», Result),
   dlg_Note («Экспертное заключение : », Result).

go (History, Mygoal): –
```

```
    rule (Rule_number, Mygoal, Type_of_breed, Conditions),
    check (Rule_number, History, Conditions),
    go ([Rule_number|History], Type_of_breed).
```

/* Сопоставление входных данных пользователя со списками атрибутов отдельных пород собак */

```
check (Rule_number, History, [Breed_cond|Rest_breed_cond_list]): –
   yes (Breed_cond), !,
   check (Rule_number, History, Rest_breed_cond_list).

check (_, _, [Breed_cond|_]): –
   no (Breed_cond), !, fail.

check (Rule_number, History, [Breed_cond|Rest_breed_cond_list]): –
   cond (Breed_cond, Text),
   ask_question (Breed_cond, Text),
   check (Rule_number, History, Rest_breed_cond_list).

check (_, _, [ ]).

do_answer (Cond_number, 1): – !,
   assertz (yes(Cond_number)).

do_answer (Cond_number, 2): – !,
   assertz (no(Cond_number)), fail.
```

/* Запрос и получение ответов yes и no от пользователя */

```
ask_question (Breed_cond, Text): –
    concat («Вопрос : », Text, Temp),
    concat (Temp, « », Temp1),
    concat (Temp1, «?», Quest),
    Response1=dlg_Ask («Консультация», Quest, [«Да»,«Нет»]),
    Response=Response1+1,
    do_answer (Breed_cond, Response).
```

Правило go пытается сопоставить объекты, классифицированные при помощи номеров условий. Если сопоставление происходит, то данный модуль программы должен добавить в БЗ сопоставленные значения и продолжить процесс с новыми данными, полученными от пользователя. Если сопоставление не происходит, механизм останавливает текущий процесс и выбирает для сопоставления другую трассу. Поиск и

сопоставление продолжается до тех пор, пока не исчерпаны все возможности.

Достоинством ЭС, базирующейся на логике, является возможность хранения фактов Базы Знаний в утверждениях динамической Базы Данных. При этом База Знаний может быть размещена в файле на диске, что позволяет сделать ее не зависимой от программного кода.

3. Содержание отчета по работе

Отчет по работе должен содержать:

- формулировку цели и задач проводимых исследований;
- описание характеристик разработанных Экспертных Систем;
- диаграмма потоков данных и структурная диаграмма для вариантов реализации Экспертной Системы как базирующейся на правилах и как базирующейся на логике;
- тексты программ с комментариями и обоснованиями;
- описание механизмов вывода в рассмативаемых Экспертных Системах;
- выводы по проведенным машинным экспериментам.

Список литературы

1. *Адаменко А.Н., Кучуков* А.М. Логическое программирование и Visual Prolog.–СПб.: БХВ-Петербург, 2003. – 992 с.: ил.

2. *Жигалов В.А., Соколова Е.Г.* InBASE:ТЕХНОЛОГИЯ ПОСТРОЕНИЯ ЕЯ-ИНТЕРФЕЙСОВ К БАЗАМ ДАННЫХ // Труды международного семинара Диалог’2001 по компьютерной лингвистике и ее приложениям. – 2001. – т.2, с. 123–135.

3. *Ин Ц., Соломон Д.* Использование Турбо-Пролога: Пер. с англ. – М.: Мир, 1993. – 608 с.: ил.

4. *Стерлинг Л., Шапиро Э.* Искусство программирования на языке Пролог: Пер. сангл. – М.:Мир,1990. – 235с. ил.

5. *Нильсон Н.* Искусственный интеллект. Методы поиска решений: Пер. с англ. – М.: Мир, 1973. –270 с.: черт.

6. *Васильев, В.Л. Гутенмахер, Ж.М. Раббот, А.Л. Тоом Н.Б.* Заочные математические олимпиады. – 2-е изд., перераб. /.– М.: Наука, 1987.– 176 с.

7. *Гик Е.Я.* Шахматы и математика. – М.: Наука,1983. –175с.: ил.

8. *Арсак Ж.* Программирование игр и головоломок. – М.: Наука, 1990.– 220 с.: ил.

9. *Гарднер М.* Математические досуги. – М.:Оникс,1995. – 495с.: ил.

10. *Демидович Б.П.* Сборник задач и упражнений по математическому анализу. – М.: Наука, 1977. – 527 с.

11. *Мочалов Л. П.* Головоломки: Кн. для учащихся. – М.: Просвещение: АО «Учеб.лит.», 1996. – 190 с.: ил.

12. *Вирт Н.* Алгоритмы + структуры данных = программы: Пер. с англ. – М.: Мир, 1985. – 406 с.: ил.

13. *Вирт Н.* Алгоритмы и структуры данных: Пер. с англ. – СПб.: Невский диалект, 2001.– 351 с.
14. *Ахо А. В., Хопкрофт Д. Э., Ульман Д. Д.* Структуры данных и алгоритмы.: Пер. с англ. – М.: Издательский дом «Вильямс», 2000. – 382 с.

15. *Маплас Дж.* Реляционный язык Пролог и его применение: Пер. с англ.– М.: Наука, 1990. – 464 с.