Курс лекций «Введение в ИИ.». Часть I. От психофизиологической проблемы до экспертных систем. Лекция 6. ЯЛП ПРОЛОГ. Часть II

О.Г. Чанышев

Содержание

1 Общие сведения о языке на примере Turbo Prolog v.2.0.

Лучший известный мне учебник по программированию на Turbo Prologe, переведенный на русский язык - это уже цитированная книга “Использование Турбо-Пролога” [4]. В ней есть все, за исключением использования системы управления внешней “бинарной” БД. Исходные модули, написанные на Turbo Prolog'е, могут быть включены и в состав приложений Visual Prolog'а. Полный комплект документации на английском языке можно получить в составе Visual Prolog v. 5.2 Personal Edition. Она содержит также удобную и полную Help- систему.

Изучение версии 5.2. визуального пролога рекомендую начать с написания программ, работающих в текстовом режиме. Для этого:

1. Стартуйте систему.
2. В общем меню выбираете Projec.Затем New Project.
3. Появлятся меню Эксперта приложения (Application Axpert). В этом меню в подменю GENERAL (оно уже выбрано) печатаете наименование проекта и имя файла-проекта (VPR-файла). Печатаете или выбираете директорию проекта.
4. Выбираете подменю TARGET. В позиции UI Strategy заменяете VPI на Text Mode.
5. Нажимаете кнопку Create.

6. Создаются все необходимые файлы. В настоящий момент Вас интересуют только два из них - файлы с расширениями .pro и pre.Последний содержит секции GLOBAL DOMAINS и GLOBAL DATABASE. Напишите в этих секциях необходимые определения, не убирая уже имеющийся текст.Файл с расширением .pro содержит секции predicates, goal, clauses. Пишите необходимый код. Можно редактировать вообще вне системы. Для вызова системы установите курсор на файл с раширением .vpr и нажмите Enter.

7. После того, как вы написали текст программы, откомпилируйте Ваш основной файл (<имя_проекта>.pro). Для этого либо Ctrl-F9 либо соответствующая позиция в меню Projec. Если есть ошибки, появится соответствующее окно с сообщениями. Установите курсор на сообщение и нажмите Enter (либо двойное нажатие левой клавишей мышки) и вы окажетесь в исходном модуле в позиции ошибки.

8. Устранив ошибки, выберите позицию Rebuild All или нажмите сочетание клавиш Ctrl-Alt-F9 для получения исполняемого файла.

1.1 Структура Пролог-программы

Вобщем случае, программа на Прологе состоит из следующих секций:

```
CONSTANTS /* раздел определения констант */
    const1 = definition
  const2 = definition
%Пример:
str_main = "Это строка"
[GLOBAL] DOMAINS /*раздел определения структур данных*/
 dom[,dom]=[reference]declaration1;declaration2
Примеры:
listdom = dom* /*определение списка целых чисел*/
dom, nb_jbject = integer
compaund_dom = cmpd(dom,string,symbol,name)
name = string
file = inputfile;outputfile
[GLOBAL]DATABASE[-<databasename>]/*раздел определения данных,
хранящихся в оперативной памяти*/
   [determ] pred1(....)
            pred2(.....)
%Пример:
term(integer,string,real,name)
GLOBAL PREDICATES /* раздел объявления глобальных "процедур"*/
[determ|nondeterm] pred1(.........)
-(i,i,o,..)(i,o,i,..)[language c|pascal|fortran]  [ as
"name" ]
  pred2(........)
%Пример:
main_calc(integer,real,real,string) - (i,i,i,o)
my_predict
make_NewString(string,string,string) - (i,i,o)
```

```
PREDICATES /* раздел объявления локальных “процедур”*/
[determ|nondeterm] pred1(.........)
pred2(........)
%Пример:
calc2(integer,real,real,string)
Goal /*аналог процедуры main в языке C*. Вся программа может состоять
только из раздела Goal /
%Пример:
Goal
InpStr=”Отредактировать эту строку”,
edit(InpStr,OutStr),
file_str(“primer.txt”,InpStr2),
concat(OutStr,InpStr2,S3),
edit(S3,_).
CLAUSES  /*раздел правил или определения “процедур”*/
 p(....):-p1(...), p2(.....), ... .
%Пример1: Только конъюнкция
edit_3(InpStr1,InpStr2,OutStr):-
concat(InpStr1,InpStr2,InpStr),
edit(InpStr,OutStr).
%Пример2 : Конъюнкция и дизъюнкция
edit_3(InpStr1,InpStr2,OutStr):-
InpStr2<>”Не присоединять”,!,
concat(InpStr1,InpStr2,InpStr),
edit(InpStr,OutStr);
!,edit(InpStr1,OutStr);

include "filename"  включение файла во время компиляции
Пример:
include “modul1.pro”
```

1.2 Предопределенные термы

char integer real ref symbol string

1.3 Предопределенные домены

`bt_selector` - указатель на бинарную БД-ключей
`bynary tree db_selector` - указатель на внешнюю БД
`place in_memory; in_ems; in_file` - указатель на место размещения внешней БД
`db_selector=dba1;dba2`

```
file keyboard; screen; printer; com1; stdin;
stdout .... userdefined
file=filein;fileout
..
%Пример:
```

```
openread(filein,"text1.txt"),readdevice(filein),
openwrite(fileout,"outs.txt),writedevice(fileout)
```

Отметим, что Turbo Prolog располагает достаточной для решения большинства задач библиотекой математических функций.

Свободные и связанные переменные

Вызывая некоторый предикат с N переменными, части из них вы можете задать значения, другим же – нет. Первые – это будут параметры, или связанные переменные, последние – свободные переменные, которые в процессе вычисления должны получить значения.

Пример:

```
..
delenie(17,Rezult),
..
delenie(D,Rezult):-Rezult=(27-10)/D.
```

Или:

```
..
delenie_plus(17,Rezult),
..
Delenie_plus(D,Rezult):-
    P1=27-10,
    delenie2(P1,D,Rezult).
    delenie2(P1,D,Rezult):-Rezult=P1/D.
```

Когда какие-то значения Вас не интересуют, всесто них следует ставить знак подчеркивания:

```
database - b1
    nekij_fact(string,integer,symbol,integer)
clauses
    ..,
     nekij_fact(A,_,_,_),
     write(A),
     ..
```

1.4 Внутреняя БД Пролога

Используя аналогию с языком С, внутренюю БД пролога (раздел database) можно рассматривать как множество массивов структур. Например, анализируя построчно входной текст, информацию о строках Вы можете хранить в структурах типа:

```
domains
..
number_str,type_str, length_str=integer
..
database - ab_strings
..
input_string(number_str,type_str,length_str)
..
```

Для размещения фактов используются встроенные предикаты assert, asserta, assertz. Извлекается факт по его имени (функтору). Удаляются факты при помощи встроеннных предикатов retract или retractall. Рассмотрим подробнее процессы размещения и извлечения фактов. asserta всякий новый факт помещает в начало однименных фактов, assertz – в конец. Т.е., для работы с множеством фактов как со стеком фактов следует использовать asserta, а для работы с очередью фактов – assertz. Например, если во входном файле было 1000 строк и факты зазмещались при помощи assertz(`input_string(NbS,TS,LS)`), то используя рекурсивную процедуру

Пример 3.1.

```
get_all_DefStrings:-
    input_string(NbS,TS,LS),!,
     retractall(inp_strings(NbS,_,_)),
     write(NbS,',',),
     get_all_DefStrings,!;!.
```

или же более корректный, с точки зрения программирования на Прологе, вариант, в котором используется бэктрекинг (да и факты не приходится удалять):

Пример 3.2.

```
get_all_DefStrings:-
    input_string(NbS,TS,LS),
     write(NbS,',',),fail.
get_all_DefStrings.
```

Вы получите на выходе последовательность чисел 1,2,3,..,1000

Если использовался при загрузке предикат `asserta(input_string(NbS,TS,LS))`, то те же процедуры выдатут: 1000,999,998,..,1. Если факты были загружены в БД впроизвольном порядке, а нужно извлечь в определенной последовательности, то для индексированного массива, каким является `input_string`, можно поступить так:

Пример 3.3.

```
get_all_DefStrings(I):-
    input_string(I,TS,LS),!,
     write(I,',',TS,',',LS),nl,
     Inew=I+1,
     get_all_DefStrings(Inew),!;!.
```

Использование assert эквивалентно assertz.

Встроенный предикат save сохраняет БД во внешнем файле.

save(DosFileName) (string) - (i)

save(DosFileName,InternalDatabaseName) (string,DatabaseName) - (i,i)

Встроенный предикат consult загружает БД, сохраненную предикатом save, из файла в память.

consult(DosFileName) (string) - (i)

consult(DosFileName,InternalDatabaseName)

(string,InternalDatabaseName) - (i,i)

Примеры:

```
save("stringDB.dba",ab_strings)
consult("stringDB.dba",ab_strings)
```

Следует отметить, что только одна БД в программе может быть неименованной. Все остальные должны иметь имя. Вы можете сформировать факты БД в программе, написанной на любом другом языке и записать их в файл; или сформировать "вручную" при помощи текстового редактора, затем загрузить в память в программе на Прологе при помощи consult.

1.5 Обработка условий и организация циклов в Turbo Prolog'e

Два встроенных предиката, очень полезных для обработки условий и организации циклов.

1) Предикат `fail` искусственно порождает неуспех.

2) Предикат `cut` (или !) предотвращает использование механизма бэктрекинга:

`P:-p1,p2,!,p3,..` – если достигнуты цели p1 и p2, бэктрениг не осуществляется.

1.5.1 Обработка условия

Пусть a- предикат, который может быть либо успешным либо нет. В случае успеха мы хотим выполнить предикат u, в обратном случае – предикат f.

```
p:-a,!,u;f.
```

Можно и так:

```
p:-a,!,u.p.
```

Пример:

```
p(A,C,Dplus):-A>=C,!,Dplus=A-C;Dplus=C-A.
```

Или:

```
p(A,C,Dplus):-A>=C,!,Dplus=A-C.
p(A,C,Dplus):-Dplus=C-A.
```

Конечно, наиболее простое решение: `p(A,C,Dplus):-Dplus=abs(A-C)`, где `abs` – встроенная функция.

Если в нашем примере со строками требуется получить номера строк с длиной не более 45, тогда процедура, решающая эту задачу, будет такой:

Пример

```
get_all_DefStrings:-
    input_string(NbS,TS,LS),
     LS<=45,
     write(NbS,','),
     fail.
get_all_DefStrings.
```

1.5.2 Использование предиката repeat

Пусть требуется в рамках одного правила предиката `pravilo` выполнить некотое множество заведомо успешных предикатов, после чего выполнять некоторое другое множество предикатов, до тех пор, пока справедливо некоторое условие. Первую и вторую группы предикатов можно, очевидно, обозначить одним предикатом, `p1` и `p2` соответственно. Пусть `p_control` – предикат, проверяющий условие, а предикат `repeat` записывается в виде следующих двух правил:

```
 repeat.
 repeat:-repeat.
```

Тогда: `pravilo:-p1,repeat,p2,p_control`.

Предикат `p1` выполнится один раз, предикат же `p2` будет выполняться до тех пор, пока `p_control` неуспешен.

Разберемся, как это происходит. `p1` всегда успешен, затем выполняется всегда успешное первое правило предиката repeat, Пролог устанавливает указатель отката на второе правило. Выполняется `p2`. После неуспеха `p_control` выполняется второе правило repeat:-repeat, после чего – первое правило, указатель отката устанавливается на второе правило, `p2, p_control` и т.д.

Приведем простой пример использования repeat. Требуется загрузить факты БД со значениями, вводимыми с клавиатуры. Предикат `check_cont` запрашивает пользователя о разрешении ввода группы значений факта `fact`, предикат vvod принимает значения.

```
database
    fact(string,string,string)
predicates
    repeat
        check_cont
        vvod
goal
    vvod,exit.

clauses
repeat.
repeat:-repeat.
vvod:-
     retractall(fact(_,_,_)),
    write("Начинаем процесс ввода значенний"),nl,
     repeat,
     clearwindow,
    write("A?"),readln(A),nl,
    write("B?"),readln(B),nl,
    write("C?"),readln(C),nl,
     assert(fact(A,B,C)),
    check_cont.
 check_cont:-
write("ВВОДИТЬ ДАЛЕЕ? (Y|N)"),
readchar(Ans),Ans='N',!,
     write("Процесс завершен"),readchar(_).
```

Следующая лекция

Лекция 7. Представление знаний в системах ИИ