Logic Programming– Laboratory 10

Efficiency in Prolog; Strings

Isabela Drămnesc

December 12, 2012

1 Notions

- Declarative thinking;
- Procedural thinking;
- Tail recursion;
- Efficiency;
- Predicates for strings;
- I/O operations with files;
- Indexing

2 Examples–Tail recursion

Some examples starting with page 46

Other examples:

```
test1:-write(hello),nl,test1. % the predicate is tail recursive,
                              % BUT it will run out of memory.

test2:-test2, write(hello),nl. % the predicate is not tail recursive
                               % because the recursive call is not the last
                               % it has a continuation

test3:-write(hello),nl,test3.  % the predicate is not tail recursive
test3:-write(goodbye).         % because it has a backtrack point

test4:-a, write(hello),nl,test4.  % the predicate is not tail recursive
                            % because it has alternatives for predicates in
                             % the recursive clause preceding the recursive call,
                             % so backtracking may be necessary
a:-write(starting).
a:-write(beginning).
```

TAIL RECURSION:

A predicate is tail recursive if:

- the recursive call is the last subgoal in the clause,
- there are no untried alternative clauses,
- there are no untried alternatives for any subgoal preceeding the recursive call in the same clause.

If a recursive predicate has no continuation, and no backtracking point, Prolog can recognize this and will not allocate memory.

Such recursive predicates are called tail recursive (the recursive call is the last in the clause and there are no alternatives).

They are much more efficient than the non-tail recursive variants.

3 Exercises

1. Find in the lecture notes the examples test3 and test4 and rewrite the predicates such that will be tail recursive.

2. Give at least 2 examples of predicates which are not efficient in Prolog and at least 2 predicates which are efficient in Prolog. Explain!

3. Write a program in Prolog and use indexing (details at page 50).

4. Some predicates for strings:

Examples of predefined predicates in Prolog:

```
?- string_to_list(String, [99,101,118,97]).
String = "ceva".

% the predicate which appends two strings
?- string_concat(timi,X,timisoara).
X = "soara".

?- string_concat(Y,X,timisoara).
Y = "",
X = "timisoara" ;
Y = "t",
X = "imisoara" ;
Y = "ti",
X = "misoara" ;
Y = "tim",
X = "isoara" ;
Y = "timi",
X = "soara" ;
Y = "timis",
X = "oara" ;
Y = "timiso",
X = "ara" ;
```

```
Y = "timisoa",
X = "ra" ;
Y = "timisoar",
X = "a" ;
Y = "timisoara",
X = "".


% the predicate which returns the length of a string
?- string_length('the string length',M).
M = 17.

?- string_length('lala', M).
M = 4.


% the predicate which returns the subset of a string
% between two specified index flags
?- sub_string('Today is a beautiful day',0,10,HowManyPositionsAre,Substring).

HowManyPositionsAre = 14,
Substring = "Today is a".

% the predicate which returns the current time and the current date
 ?- get_time(Time),convert_time(Time, Data),nl,write('Today is '),write(Data).

Today is Tue Jan 04 16:10:12 2011
Time = 1.29415e+009,
Data = "Tue Jan 04 16:10:12 2011".

?- get_time(Time),convert_time(Time,Year,Month,Day,Hour,Minute,Second,Milisecond).

Time = 1.29415e+009,
Year = 2011,
Month = 1,
Day = 4,
Hour = 16,
Minute = 11,
Second = 41,
Milisecond = 515.
```

5. Read from the keyboard N numbers until introducing a character.

6. Randomize M numbers and write them into a file. Apply at least 10 operation on them! (e.g. sorting, delete the first element, etc.), and print the result into another file.

If you apply 10 operations then you have:

- One file that contains the input data (the numbers randomly generated)

- 10 files: sort.txt, deletefirst.txt,....etc.

Create efficient programs!

Verify the running time for each predicate created.

4 Homework:

Next time present all your homework!!! Study.