Logic Programming – Laboratory 4
Accumulators, Open lists and Difference lists

Isabela Drămnesc

November 2, 2010

1 Questions

- What do you understand by inductive domain? How can we define a list?
- How do we define the recursion in Prolog? Give 2 examples!
- Write a predicate to append two lists $append_lists(+L1, +L2, ?Rez)$
- Write a predicate that calculates the factorial of a number $factorial(+N, ?F)$

2 Concepts

- Operations with lists
- Accumulators
- The difference between programs with accumulator and without it
- Open lists
- Difference lists

3 Accumulators

The list is decomposed until we find the boundary condition, then we obtain the solution which is captured in the accumulators, then it takes place the reverse process of decomposing the list, and the result is the solution which was captured in the accumulator.

1) The reverse of a list:

a) without accumulators:

```
reverse_list([],[]).
reverse_list([H|T],X):- reverse_list(T,T1),
                        append_lists(T1,[H],X).
```

Explain what is happening when we ask Prolog: (use the trace command)

```
?- reverse_list([1,2,3],X).
```

b) with accumulators:

```
reverse_list2(L,R):-reverseAcc(L,[],R).     /* call the variant
                                               with accumulator */
reverseAcc([],R,R).        /* boundary condition */
reverseAcc([H|T],A,R):-
            reverseAcc(T,[H|A],R).        /* recursive call
                                             with accumulator */
```

Try trace of $reverseAcc([1,2,3],[],R)$.

And trace of $reverse-list2([1,2,3],X)$. Compare the cost for $reverse-list$ and for $reverse-list2$.

2) The predicate which calculates the n'th term from the Fibonacci sequence: 1,1,2,3,5,8,13,21,..., where $f(n)=f(n-1)+f(n-2)$ for $n>2$.

(a) without accumulator:

```
fibonacci(1,1).
fibonacci(2,1).
fibonacci(N,F):-N>2,N1 is N-1,N2 is N-2,
                 fibonacci(N1,F1),fibonacci(N2,F2),
                 F is F1+F2.
```

(b) with accumulator:

```
/* F=f(M). F1=f(M-2), F2=f(M-1). */
fibonac(N,F):-fib(2,N,1,1,F).
fib(M,N,_,F2,F2):-M >= N.
fib(M,N,F1,F2,F):-M < N, M1 is M+1,
                   F1plusF2 is F1+F2,
                   fib(M1,N,F2,F1plusF2,F).
```

Use trace to see the difference between the two programs (without accumulator and with accumulator).

3) The predicate which returns the length of a list:

```
lengthList([],0).
lengthList([H|T],N):-lengthList(T,N1),N is 1+N1.
```

Modify the predicate such that it will use accumulator.

4) Modify the predicate which calculates $n!$ using accumulator.

5) Use the next predicates to see the running time for the reverse of the list (using and not using accumulator), for factorial, for the length of a list, for fibonacci:

```
current_time(Timestamp) :- get_time(Current),
stamp_date_time(Current,Date,local),
date_time_value(time, Date, Timestamp).

time_elapsed(time(H1,M1,S1),
time(H2,M2,S2),
Seconds) :- Seconds is 3600 * (H2 - H1) + 60 * (M2 - M1) + (S2 - S1).
```

Example for factorial:

```
?- current_time(T1), factorial(50000, X),
    current_time(T2), time_elapsed(T1,T2,T).
?- current_time(T1), factAcc(50000, X),
    current_time(T2), time_elapsed(T1,T2,T).
```

Where factorial=the predicate without accumulators, and factAcc=the predicate with accumulators.

4 Open lists and Difference lists

a) Example:

6)

```
?- L=[1,2,3|X],X=[t,g,h].    /* the ''hole'' from the list L is fullfilled
                                 in a single step with X */
L = [1, 2, 3, t, g, h],
X=[t,g,h]

?- L2=[s,d,a|T],T=[head|T2].    /* the ''hole'' is fullfilled with
                                    the open list  */
L2 = [s, d, a, head|T2],
T=[head|T2]
```

```
7)
appendopenlists(L1,G,L2):-G=L2.

 ?- X=[a,b,c|G], appendopenlists(X,G,[d,e,f]).
X = [a, b, c, d, e, f]
```

b) We represent the difference lists as the difference between the open list and its “hole”.

Example: $[1, 2, 3|Hole] - Hole$

8) Introduce the rule:

```
appendOpen(DL-Hole,L2):-Hole=L2.
```

Try for:

```
 ?- X=[m,n,p|Hole]-Hole,appendOpen(X,[1,2,3]).
```

Write a rule such that you obtain $X = [m, n, p, 1, 2, 3]$.

9)The predicate appendDiff2/3:

```
appendDiff2(DL1-Hole1,DL2-Hole2,DL1-Hole2):-Hole1=DL2.

?- X=[a,f|G]-G, appendDiff2(X,[p,l|G2]-G2,Answer).

?- X=[a,f|G]-G, appendDiff2(X,[p,l|G2]-G2,Answer-[]).
```

10) Test for appendDiff3.

```
appendDiff3(DL1-Hole1,Hole1-Hole2,DL1-Hole2):-Hole1=DL2.

?- X=[a,f|G]-G, appendDiff3(X,[p,l|G2]-G2,Answer).

 ?- X=[a,f|G]-G, appendDiff33(X,[p,l|G2]-G2,Answer-[]).
```

5 Homework:

Finish all the exercises from Homework 3 and implement queues with open lists.

Deadline: next laboratory.