Data structures in PROLOG

Lecture 2

Summary

- Structuring data
- Natural numbers
- Lists
- Esercises

Terms

The set TERM of *terms* is inductively defined as:

1. Every constant symbol is a term (lower case initial);
2. Every variable symbol is a term (upper case initial);
3. If $t_1 \ldots t_n$ are terms and f is an n-ary, $f(t_1, \ldots, t_n)$ is a term (called *functional term*, lower case initial f).

Examples: X, c, $f(X, g(YY, c))$,...

Atoms and clauses are defined as before (remember predicate names lower case initial).

Unification in PROLOG: examples

$p(f(X,Y),a,g(b,W))$ unifies with $p(Z,X,g(b,Y))$.

$p(f(X,Y),a,g(b,W))$ does not unify with $p(Z,f(a),g(b,Y))$.

$p(f(X,Y),a,g(b,W))$ does not unify with $p(X,a,g(b,Y))$.

A program for the class timetable

```
course(ai,timetab(tu,10,12),
       teacher(nardi,d),room(diag,b2)).
course(ai,timetab(we,10,12),
       teacher(nardi,d),room(diag,b2)).
course(ai,timetab(fr,10,12),
       teacher(nardi,d),room(diag,b2)).
...
```

A program for the class timetable

```
teaches(Tea,Course) :- course(Course,Timetab,Tea,Room).
length(Course,Len) :-
    course(Course,timetab(Day,Start,End),Tea,Room),
    plus(Start,Len,End).
hasClass(Tea,Day) :-
    course(Course,timetab(Day,Start,End),Tea,Room).
busy(Room,Day,Time) :-
    course(Course,timetab(Day,Start,End),Tea,Room),
    Start =< Time, Time =< End.
```

Natural numbers

```
natural_number(0).
natural_number(s(X)) :- natural_number(X).

plus1(0,X,X) :- natural_number(X).
plus1(s(X),Y,s(Z)):- plus1(X,Y,Z).

lesseq1(0,X) :- natural_number(X).
lesseq1(s(X),s(Y)) :- lesseq1(X,Y).
```

Lists

Remember that a list of atoms is defined as follows:

- nil is a list;
- if a is an atom and L is a list $cons(a, L)$ is a list

In PROLOG `[a | X]` is the same as $cons(a, X)$

- `[a,b,c,d]` is a 4 element list;
- `[a | X]` is a list whose first element is `a` and the rest of the list is denoted by the variable `X`;
- `[Y | X]` is a list whose first element is denoted by the variable `Y` and the rest of the list is denoted by the variable `X`.

Lists

```
/* member1(X,L) is true when X is an element of L */

member1(X,[X|Xs]).
member1(X,[Y|Ys]) :- member1(X,Ys).

/* append1(X,Y,Z) is true when Z is the
   concatenation of  X and Y */

    append1([],Ys,Ys).
    append1([X|Xs],Ys,[X|Zs]) :- append1(Xs,Ys,Zs).
```

Other programs using lists

```
/* prefix(L1,L) is true when L1 is a prefix of L */

    prefix([],_Ys).
    prefix([X|Xs],[X|Ys]) :- prefix(Xs,Ys).

/* reverse(L1,L2) is true when L2 is the
reverse of L1 (same elements in reversed order */

    reverse1([],[]).
    reverse1([X|Xs],Zs) :- reverse1(Xs,Ys),
                           append1(Ys,[X],Zs).
```

Sorting lists

```
sort1(Xs,Ys) :- permutation(Xs,Ys), ordered(Ys).

permutation(Xs,[Z|Zs]) :- select(Z,Xs,Ys),
                          permutation(Ys,Zs).
permutation([],[]).

ordered([]).
ordered([_X]).
ordered([X,Y|Ys]) :- X =< Y, ordered([Y|Ys]).

select(X,[X|Xs],Xs).
select(X,[Y|Ys],[Y|Zs]):-select(X,Ys,Zs).
```

Programs using lists and numbers

```
len([],0).
len([_X|Xs],s(N)) :- len(Xs,N).

len([],0).
len([_X|Xs],N) :- len(Xs,N1), N is N1 + 1.
```

Home exercises

1. build the search tree for:
   `?- member(c,[a,c,b]).`
   `?- plus1(Y,X,s(s(s(s(s(0)))))).` and
   `?- reverse([a,b,c],X).`
2. Write the PROLOG programs `times`, `power`, `factorial`, `minimum` using the definitions given for natural numbers.
3. Write the PROLOG programs `suffix`,`subset`,`intersection` using lists to represent sets.
4. Write a PROLOG program for a depth-first visit of possibly cyclic graphs, represented through the relation `arc(X,Y)`
5. Write a PROLOG program implementing insertion sort on lists.