CAP 5602
Summer, 2011

Lesson 6: More Built-in Predicates

The topics

1. predicates for data objects
2. constructing and decomposing structures
3. bagof, setof, findall

1. Predicates for data objects

Here we discuss the predicates var(X), nonvar(X), atom(X), integer(X), float(X), number(X), atomic(X), compound(X).

The goal var(X) is satisfied, if X is non-instantiated variable.

The query

?- var(X).

produces

true, while

?- X = 2, var(X).

gives

false

because X was instantiated (assigned) the value 2.

The goal nonvar(X) succeeds if X is not a variable, or X was instantiated.

The goal atom(X) succeeds if X is an atom, as shown in the queries below.

?- nonvar(X).
false.
?- atom([]).
true.
?- atom(date(may,29,2011)).
false.
?- atom(X).
false.

The predicates integer(X) and float(X) are satisfied if X stands for an integer, respectively a float. Stands for an integer means that either X is an integer, or is a variable that is currently instantiated to an integer.

?- float(3.2e-5).
true.
?- integer(-234567).
true.
?- float(24).
false.

The goal number(X) is satisfied if X stands for a number.
?- number(2.4).
true.
?- number(-24.6E+10).
true.
?- number(alpha). false.
?- X = 57, number(X).
X = 57.
In the last query, SWI prolog printed the value of X, without saying true.

The goal atomic(X) is satisfied if X is an atom or a number.
?- atomic(?-).
true.
?- atomic(=).
true.
?- X = 8.27, atomic(X).
X = 8.27.
?- atomic(address(23, 'caca street', slumville)).
false.
The first 2 queries tells us that the strings ?- and = are atoms, as suspected.

Finally, compound(X) is satisfied if X stands for a structure.
?- compound(date(may, 30, 2011)).
true.
?- X= father(bill,chris), compound(X).
X = father(bill, chris).

Example

Let us write the prolog predicate integer_list(L) that is satisfied when all items of L are integers. We use tail recursion. The empty list is an integer

list. A non-empty list is an integer list if the head is an integer and the tail is an integer list.

```
% integer_list(L) is satisfied if all items of L are integers
integer_list([]). % the base case
integer_list([Head | Tail]):- integer(Head), integer_list(Tail).
```

2. Constructing and decomposing structures

The univ predicate, Object =.. List, is satisfied if Object is a data object and L is a list that has the functor of Object as head and the arguments of Object as tail. If Object is atomic, the list contains only the Object. The queries below show how univ works.

```
?- f(a,g(b),c) =.. L.
L = [f, a, g(b), c].
?- O =.. [f,X,a,b].
O = f(X, a, b).
?- a =.. L.
L = [a].
?- O =.. [23].
O = 23.
```

The predicate functor(Object,F,N) is satisfied if F is the functor of Object and N is the arity of F. the 3 queries below show how functor works.

```
?- functor(f(a,b,X),F,N).
F = f,
N = 3.
?- functor(a,F,N).
F = a,
N = 0.
?- functor(2,F,N).
F = 2,
N = 0.
```

The predicate arg(N,Object,A) is satisfied if A is the N-th argument of Object. It is assumed that the arguments are labeled 1, 2, ... The queries below show how arg works.

```
?- arg(3,f(a,U,V),A).
V = A.
```

?- arg(3,f(a,b,c),A).
A = c.
?- arg(3,f(a,b),A).
false.
?- arg(N,f(a,b,c),b).
N = 2 ;
false.
?- arg(N,a,A).
false.

If Object has fewer than N arguments or stands for an atomic object, arg(N,Object,A) returns false.

3. bagof,setof, findall

We discuss 3 built-in predicates bagof, setof, and findall that generate all objects that satisfy a given given goal.

bagof

The predicate bagof(X,P,L) callects all objects X that satisfy the predicate P in the the list L.

Example

Let's look at prog17, listed below.

```
% the likes predicate
likes(john,apples).
likes(mary,pineapples).
likes(susan,cherries).
likes(mark,oranges).
likes(bill, mangoes).
likes(ann,oranges).
likes(peter,X):- fruit(X), sweet(X). % a rule
fruit(melons).
sweet(melons).
fruit(apricots).
sweet(apricots).
fruit(avocados).
fruit(mangoes).
sweet(mangoes).
```

The query

?- bagof(X,likes(X,mangoes),L).

produces

L = [bill, peter].

because the only X's that satisfy the goal likes(X,mangoes) are bill and peter.

Of course, bagOf(X,P,L) doesnot make sense if X does not occur in P. Look at what SWI prolog does for the query ?- bagof(Y,likes(X,mangoes),L).

X = bill,
L = [_G4449] ;
X = peter,
L = [_G4441].

The goal likes(X,mangoes) is satisfied, so L is not empty, but Y is not instantiated because it has no relation to X.

If we use the query ?- bagof(X,likes(X,F),L) we get the values of F and L for the first match. We can type ; return to get the second one, and so on.

F = apples,
L = [john] ;
F = apricots,
L = [peter] ;
F = cherries,
L = [susan] ;
F = mangoes,
L = [bill, peter] ;
F = melons,
L = [peter] ;
F = oranges,
L = [mark, ann] ;
F = pineapples,
L = [mary].

If we do not care about the value of F and want the list of all X's such that likes(X,F) is satisfied for some F, we use the query

?-bagof(X, F$\hat{}$likes(X,F),L).

We get the answer

L = [john, mary, susan, mark, bill, ann, peter, peter, peter].

The value of X is listed for each match, so peter appears 3 times for F = melons, apricots, mangoes.

setof(X,P,L) works like bagof(X,P,L) with the difference that L is ordered and the duplicates are removed.

Let us see what happens if we use the query ?- setof(X,FÎlikes(X,F),L) on the prog17.

We get L = [ann, bill, john, mark, mary, peter, susan]. The ordering is done accoring to the buil-in predicate @¡.

If we do not put any restrictions on F, the query ?-setof(X,likes(X,F),L) works like bagof(X,likes(X,F),L) except that L is ordered and has no duplicates.

F = apples,
L = [john] ;
F = apricots,
L = [peter] ;
F = cherries,
L = [susan] ;
F = mangoes,
L = [bill, peter] ;
F = melons,
L = [peter] ;
F = oranges,
L = [ann, mark] ;
F = pineapples,
L = [mary].

The collected objects can be structures as well as atoms. In SWI prolog, the query ?- setof(X/F, likes(X,F),L) produces

L = [ann/oranges, bill/mangoes, john/apples, mark/oranges, mary/pineapples, peter/apricots, peter/mangoes, peter/melons, ... / ...].

findall(X,P,L) also produces the list of all objects that satisfy P. The difference is that **all** objects X are collected regardless of the values of the other variables of P. The query ?- findall(X,likes(X,F),L) produces L = [john, mary, susan, mark, bill, ann, peter, peter, peter].