

Meta-Logical Predicates

Prolog includes some meta-logical predicates to make programming simpler and allow the users to have more control on the program executions

These predicates cannot be modeled in first-order logic

`var(X)` X is currently a free variable
`nonvar(X)` X is not a free variable
`ground(X)` X is a term not containing variables
`number(X)` X is a number
`string(X)` X is a string

These predicates never cause error, or instantiate variables, but the state of variables can be inspected safely

They do not have a first-order reading, since the ordering of the goals matters for them:

`?- var(X), X = 3.`

yes

`?- X = 3, var(X).`

no

Predicates which implement standard order

Prolog has a notion of a so-called standard order among all terms

This means that any two terms (being them atoms, structures, variables, numbers, etc.), can be compared for equality, disequality, and precedence

This order is somewhat arbitrary, but it is usually adequate for most applications

<code>== / 2</code>	Identity of terms
<code>\== / 2</code>	Nonidentity of terms
<code>@> / 2, @>= / 2, @< / 2, @=< / 2</code>	Comparison

The order among terms is the following:

1. Variables, oldest first.
2. Floats, in numeric order.
3. Integers, in numeric order.
4. Atoms, in alphabetical order.
5. Structures, ordered first by arity, then by the name of the principal functor, then by the arguments left-to-right.

It is interesting to note that the identity comparison `== / 2` compares variables without binding them

In fact, it does not report two variables being equal unless they are the same

```
?- X == Y.  
no  
?- X = Y, X == Y.  
   Y = X ?  
yes
```

Example

The following chunk of code can maintain an ordered list of terms, possibly including variables, numbers, atoms, etc. `insert(List, Term, NewList)` adds `Term` to `List` (ordered and without repetitions) to obtain `NewList`, also ordered without repetitions.

```
insert([], It, [It]).
insert([H|T], It, [H|T]):- H == It.
insert([H|T], It, [It, H|T]):- H @> It.
insert([H|T], It, [H|NewT]) :-
    H @< It,
    insert(T, It, NewT).
```

Note the use of `== / 2` to check for identity, so that variables can be added without further instantiating them.