Prolog

Built-ins

- X is Y — the value of X is unified with Y
- X =:= Y — the values of X and Y are equal
- X =\= Y — the values of X and Y are different
- X > Y — the value of X is greater than the one of Y
- X < Y — the value of X is lower than the one of Y
- X >= Y — the value of X is greater than or equal to the one of Y
- X =< Y — the value of X is lower than or equal to the one of Y

- X is Y — the value of X is unified with Y
- X =:= Y — the values of X and Y are equal
- X =\= Y — the values of X and Y are different
- X > Y — the value of X is greater than the one of Y
- X < Y — the value of X is lower than the one of Y
- X >= Y — the value of X is greater than or equal to the one of Y
- X =< Y — the value of X is lower than or equal to the one of Y

- ! — *Cut* predicate, allows us to prune useless paths

An example of the use of !

```
compare(X, Y, lower) :- X < Y.
compare(X, Y, equal) :- X =:= Y.
compare(X, Y, greater) :- X > Y.

?- compare(3, 5, lower).
```

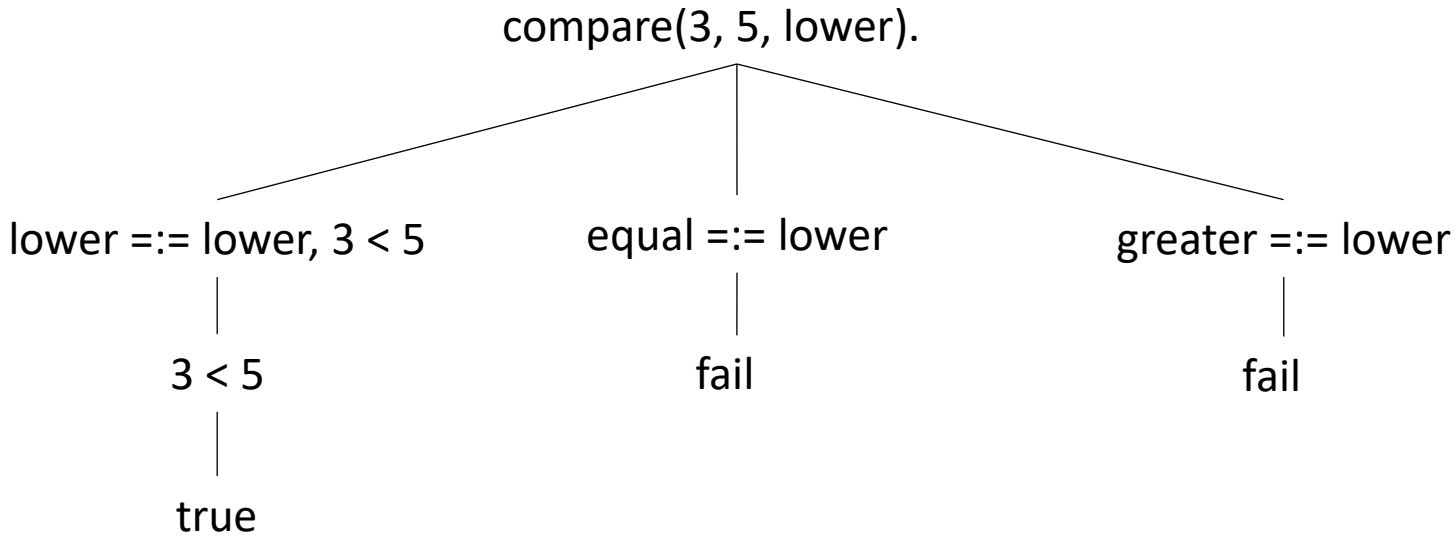
An example of the use of !

compare(X, Y, lower) :- X < Y.

compare(X, Y, equal) :- X =:= Y.

compare(X, Y, greater) :- X > Y.

?- compare(3, 5, lower).

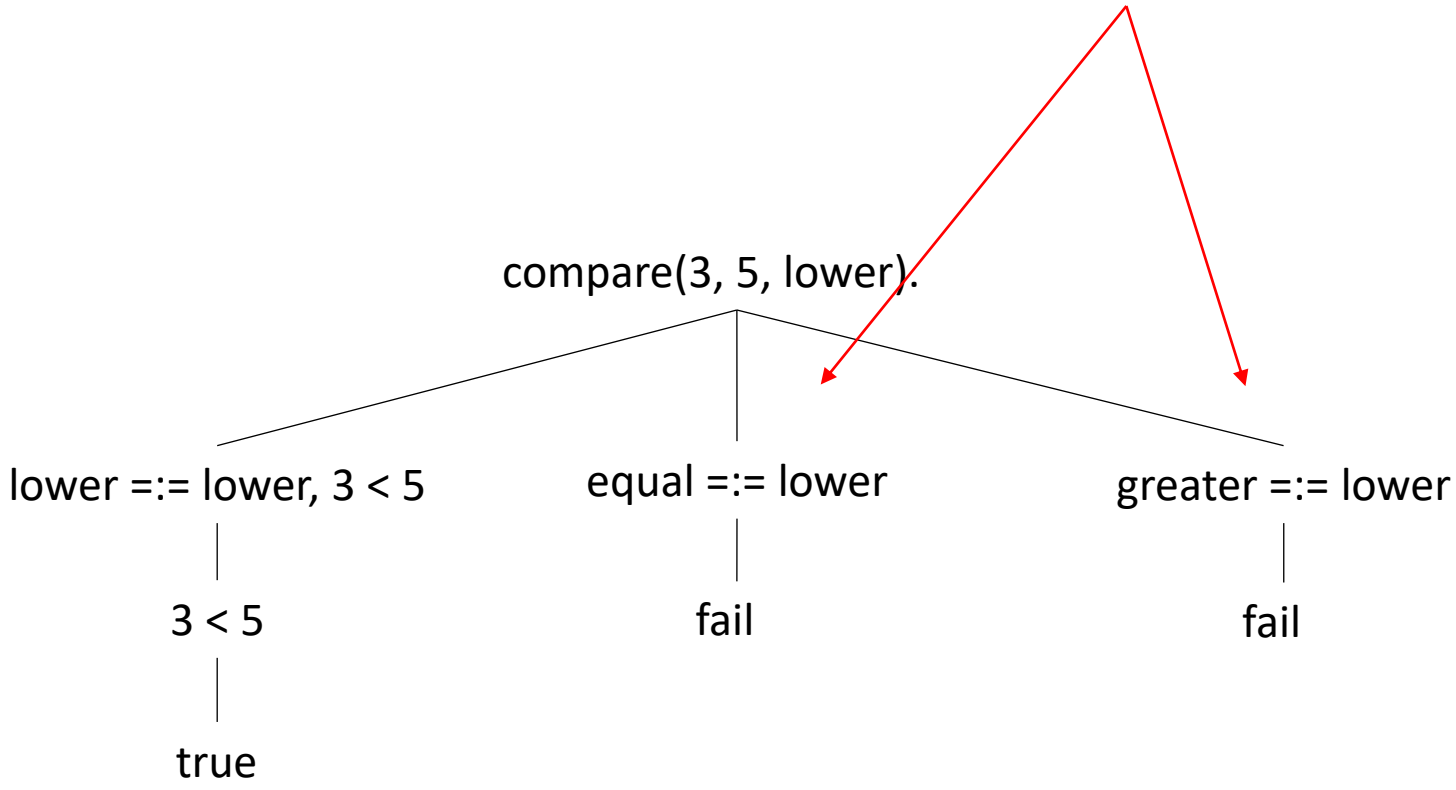


An example of the use of !

```
compare(X, Y, lower) :- X < Y.
compare(X, Y, equal) :- X =:= Y.
compare(X, Y, greater) :- X > Y.
```

```
?- compare(3, 5, lower).
```

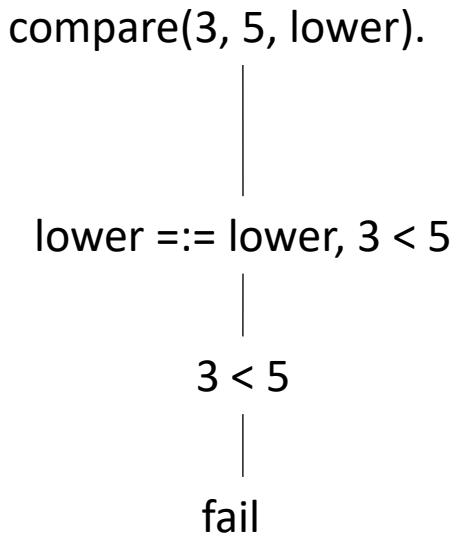
Prolog can return multiple results, leaving open choice points will lead to this derivation, where useless branches are tested too.



An example of the use of !

```
compare(X, Y, lower) :- X < Y , !.
compare(X, Y, equal) :- X =:= Y, !.
compare(X, Y, greater) :- X > Y.
```

```
?- compare(3, 5, lower).
```



Prolog

Exercises

1) Compute the absolute value of a number

1) Compute the absolute value of a number

```
abs(X,X) :- X >= 0.
abs(X,Y) :- X < 0, Y is -X.
```

2) Compute the factorial of a number

2) Compute the factorial of a number

Intuitively:

fatt(0) = 1

fatt(n) = n * fatt(n-1) (per n>0)

2) Compute the factorial of a number

In Prolog:

```
fatt(0,1).
fatt(N,X) :- N>0, N1 is N-1, fatt(N1, X1), X is N*X1.
```

2) Compute the factorial of a number

A different Prolog version, using tail recursion:

```
fatt2(N,X) :- fatt2(N,1,X).
fatt2(0,ACC,ACC).
fatt2(M,ACC,X) :- ACC1 is M*ACC, M1 is M-1,
        fatt2(M1,ACC1,X).
```

3) Compute the greatest common divisor

3) Compute the greatest common divisor

Intuitively:

GCD(x,0) = x

GCD(x,y) = GCD(y, x mod y) (for y>0)

3) Compute the greatest common divisor

In Prolog:

```
gcd(X,0,X).
gcd(X,Y,Z) :- Y>0, X1 is X mod Y, gcd(Y,X1,Z).
```

4) Find the last element of a list

4) Find the last element of a list

```
last([X], X).
last([_|Z], X) :- last(Z,X).
```

4) Find the last element of a list

An alternative, using the built-in function *reverse:*

```
last(L, X) :- reverse(L, [X|_]).
```

5) Check if a list is a sublist of another list

5) Check if a list is a sublist of another list

```
sublist([], _).
sublist([X|L1], [X|L2]) :- sublist(L1, L2).
sublist([X|L1], [_|L2]) :- sublist([X|L1], L2).
```

Note that in this solution we are checking whether L1 is a sublist of L2.