LabISS | Using Prolog

Using Prolog

Let us recall here the fundamental aspects of Prolog, by means of a set of examples.
The code of these examples can be found in LabIss | Using the QActor (meta)model

Main concepts to remember

- Prolog as an interpreted language
- Facts and Rules are called Terms
- Rules as relations: declarative semantics.
- Rules as procedures: procedural semantics.
- Selective Linear Definite clause resolution (See SLD)
- Constants, Variables, Terms and Unification
- Backtracking and Cut

Facts

```
vertical(    line(point(X, Y), point(X, Z)) ).
horizontal(  line(point(X, Y), point(Z, Y)) ).

pos(1,point(1,5)).
pos(2,point(3,1)).
pos(3,point(3,3)).
pos(4,point(3,5)).
pos(4,point(3,7)).
pos(4,point(7,1)).
pos(4,point(7,5)).
pos(4,point(7,9)).
```

Rules

```
horizontalLine(P1,P2):-
        pos(START,P1), horizontal( line(P1,P2) ),
        pos(POS,P2), POS \== START.

allHLines(P1,HL):-
        findall( secondPoint(P2), horizontalLine(P1,P2), HL).
```

A knowledge base (userKb.pl)

Two complex *Terms* representing a **point** are bundled together as the two arguments of another complex *Term* with the functor **line**.
In effect, we represent a **line** by a complex term which has two arguments which are complex terms themselves and represent points.

We're using Prolog's ability to build complex terms to work our way up a hierarchy of concepts.
The term pos/2 represent a position in a two-dimensional space.

See Introduction to PROLOG
tuProlog Manual

```
solve( consult("sysRules.pl")          )
solve( consult("userKb.pl")   )

solve( unify( p(X,X ), p(1,2) ))
println( currentSolution )          //  no.

solve( unify( p(X,b(X) ), p(1,Y) ) )
println( currentSolution )          //X / 1  Y / b(1)
println( "X=${getCurSol(\"X\")} Y=${getCurSol(\"Y\")}" )

solve( vertical(line( point(16, 4), point(16, 72) )) )
println( currentSolution )               //yes.
solve(  horizontal(line(point(1,1),point(2,Y)))  )
ifSolved{ println( "Y=${getCurSol(\"Y\")}" ) } //Y=1.

solve(  horizontalLine( point(1,5),P )  )
ifSolved{ println( "P=${getCurSol(\"P\")} " ) }     //P=point(3,5)

solve(  allHLines( point(1,5),L )  )
ifSolved{ println( "all lines=${getCurSol(\"L\")} " }
//all lines=[secondPoint(point(3,5)),secondPoint(point(7,5))]
```

Built-in operations

- **solve(G)**: calls the Prolog interpreter for the goal **G**
- **currentsolution**: a variable that gives the solution of the last solve executed by the actor
- **getCurSol(V)**: gets the value of the variable **V** in currentsolution

Examples are given in prologusage.qak.

Logical

```
unify( A, B ) :- A = B.

getCtxNames(CTXNAMES) :-
        findall( NAME, context( NAME, _, _, _ ), CTXNAMES).
```

State (side effects)

```
addRule( Rule ):-       assert( Rule ).
removeRule( Rule ):-retract( Rule ), !.
removeRule( A  ):-      retract( A :- B ),!.
removeRule( _  ).

replaceRule(Rule,NewR):- removeRule(Rule),addRule(NewR).

assign( I,V ) :-  retract( value(I,_) ),!, assert( value( I,V )).
assign( I,V ) :-  assert( value( I,V )).
getVal( I, V ):-  value(I,V), !.
getVal( I, fail ).
inc(I,K,N):- value( I,V ),      N is V + K, assign( I,N ).
dec(I,K,N):- value( I,V ),      N is V - K,     assign( I,N ).
```

The file sysRules.pl

The rules on the left are examples of rules provided by the generated file sysRules.pl.

```
State exampleElab{
        solve( assign(n,3) )
        solve( inc(n,10,N1) )
        solve( getVal( N1,V ) )

        println( "V=${getCurSol(\"V\")}" )    //V=13
}
```

See prologusage.qak.

```
State handleCmd{
  printCurrentMessage
  onMsg ( local_buttonCmd : local_buttonCmd(CMD) ){
     forward  robotcontrol -m robotCmd : robotCmd($payloadArg(0))
  }
 }

onMsg( polar : p( D,A )){
 run resources.radarSupport.spot(payloadArg(0),payloadArg(1)))
}


State radarTest{
   solve ( getData(D,A)   )
  ifSolved run  resources.radarSupport.spot( @D,@A   )
}
```

Shortcut

```
 "payloadArg(N)
gives (as String)  the argument N (0<=N<=arity) of a msg payload
 example:: run ...(payloadArg(0))
 example:: onMsg( m : m(X) ){ println("...$payloadArg(0)") }

 "$" varName= ID  => $VARID
used within a (produced) String
example:: msg(_,$Curmove)

 "#" varName= VARID  => ${getCurSol("VARID").toString()}
used to access a logic variable in a (produced) String
 example::solve(move(M));println( #M )

 "@" varName= VARID  => getCurSol("VARID").toString()
used to access a logic variable
 example::solve(move(M));doMove( @M )
```