Experiment - 6

6. Implementation of Monkey Banana Problem using LISP/PROLOG

Monkey-Banana.pl Program

```
%Monkey-Banana Problem:-

% initial state: Monkey is at door,
%               Monkey is on floor,
%               Box is at window,
%               Monkey doesn't have a banana.
%

% prolog structure: structName(val1, val2, ... )

% state(Monkey location in the room, Monkey onbox/onfloor, box location, has/hasnot banana)


% legal actions

do( state(middle, onbox, middle, hasnot),   % grab banana
   grab,
   state(middle, onbox, middle, has) ).

do( state(L, onfloor, L, Banana),           % climb box
   climb,
   state(L, onbox, L, Banana) ).

do( state(L1, onfloor, L1, Banana),         % push box from L1 to L2
   push(L1, L2),
   state(L2, onfloor, L2, Banana) ).

do( state(L1, onfloor, Box, Banana),        % walk from L1 to L2
   walk(L1, L2),
   state(L2, onfloor, Box, Banana) ).


% canget(State): monkey can get banana in State
```

```
canget(state(_, _, _, has)).              % Monkey already has it, goal state

canget(State1) :-                         % not goal state, do some work to get it
    do(State1, Action, State2),           % do something (grab, climb, push, walk)
    canget(State2).                       % canget from State2

% get plan = list of actions

canget(state(_, _, _, has), []).          % Monkey already has it, goal state

canget(State1, Plan) :-                   % not goal state, do some work to get it
    do(State1, Action, State2),           % do something (grab, climb, push, walk)
    canget(State2, PartialPlan),          % canget from State2
    add(Action, PartialPlan, Plan).       % add action to Plan

add(X,L,[X|L]).

%-------------------OutPut Query------------------------>
% ?- canget(state(atdoor, onfloor, atwindow, hasnot), Plan).
% Plan = [walk(atdoor, atwindow), push(atwindow, middle), climb, grasp]
% Yes

% ?- canget(state(atwindow, onbox, atwindow, hasnot), Plan ).
% No

% ?- canget(state(Monkey, onfloor, atwindow, hasnot), Plan).
% Monkey = atwindow
% Plan = [push(atwindow, middle), climb, grasp]
% Yes
```

Output:

SWI-Prolog (AMD64, Multi-threaded, version 9.0.3)

File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
Warning: c:/users/admin/desktop/mbp.pl:37:
Warning:    Singleton variables: [Action]
% c:/Users/admin/Desktop/MBP.pl compiled 0.00 sec, 9 clauses
?- canget(state(atdoor, onfloor, atwindow, hasnot)).
true
```

SWI-Prolog (AMD64, Multi-threaded, version 9.0.3)

File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
Warning: c:/users/admin/desktop/monkey-banana.pl:37:
Warning:    Singleton variables: [Action]
% c:/Users/admin/Desktop/Monkey-Banana.pl compiled 0.00 sec, 9 clauses
?- canget(state(Monkey, onfloor, atwindow, hasnot), Plan).
Monkey = atwindow,
Plan = [push(atwindow, middle), climb, grab]
```