Shiram Tiwari
5117060
BE comps.

Experiment 04

o Aim:- write a Program to Implement Monkey & Banana Problem.

o Theory:-

A monkey enters the room via a door. In the room, near the window, is a box. In the middle of the room hangs a banana from the ceiling. The Monkey wants to group the banana and can do so after climbing on the box in the middle of the room.

o States:-

For each state we need to record:

- The position of the monkey (door, window, middle).
- The position of the box.
- If the monkey is on the box.
- If the monkey has the banana.

The Initial state (door, window, no, no)
The set of goal state is (*, *, *, yes)

o Moves:-

walk(P): from (M, B, no, H) to (P, B, no, H)
PUSH(P): from (M, M, no, H) to (P, P, no, H)
climb : from (M, M, no, H) to (M, M, yes, H)
grasp : from (middle, B, yes, no) to (middle, B, yes, yes)

Step by Step sol^n.

(a) Initial State description:

At (Monkey, A) ∧ At (Banana, B) ∧ At (Box, C) ∧ Height (Monkey, low) ∧ Height (Box, low) ∧ Height (Banana, High) ∧ Push (Box) ∧ Climbup (Box)

(b) 6 action schemas

1. Goto from one place to another: Action (Go(x,y))
   precondition: At (Monkey, x)
   Effect: At (Monkey, y) ∧ ¬ At (Monkey, x)

2. Push an object from one place to another.
   Action L push (b, x,y)
   Precondition : At (Monkey, x) ∧ Can Push (b)
   Effect : At (b, y) ∧ At (Monkey, y) ∧ ¬ At (Monkey, x) ∧ At (b, x)

3. Climb up onto an object
   Action: Climbup (b)
   Precondition: At (b, x) ∧ At (Monkey, x) ∧ Climbup (b)
   Effect: On (Monkey, b) ∧ ¬ Height (Monkey, High)

4. Climb down from an object
   Action: Climb down (b);
   Precondition: On (Monkey, b) ∧ Height (Monkey, High)
   Effect → ¬On (Monkey, b) ~~∧~~ Height (Monkey; High) ∧ Height (Monkey, low).

5) Grasp on object

Action:- Grasp: (Object, Position, height)
Precondition: Height (Monkey, h) ∧ Height (b, h) ∧ At (Monkey, x)
Effect : Has (Monkey, G)

6) Ungrasp an object

Action: Ungrasp (b);
Precondition Have (Monkey, b);
Effect : Has (Monkey, object).

Conclusion:

Thus, we have successfuly implemented the monkey banana problem and understood the steps.

Roll No: 5117060

Banana Monkey Problem

Code:

```
move(state(middle,onbox,middle,hasnot),
    grasp,
    state(middle,onbox,middle,has)).
move(state(P,onfloor,P,H),
    climb,
    state(P,onbox,P,H)).
move(state(P1,onfloor,P1,H),
    push(P1,P2),
    state(P2,onfloor,P2,H)).
move(state(P1,onfloor,B,H),
    walk(P1,P2),
    state(P2,onfloor,B,H)).
canget(state(_,_,_,has)).
canget(State1)   :-
    move(State1,_,State2),
    canget(State2).
```

Output:

```
| ?- change_directory('C:/Users/asus/Desktop/Notes BE 2020-21/AI').


yes
| ?- [prolog]
.
compiling C:/Users/asus/Desktop/Notes BE 2020-21/AI/prolog.pl for
byte code...
C:/Users/asus/Desktop/Notes BE 2020-21/AI/prolog.pl compiled, 15
lines read - 2185 bytes written, 22 ms


(16 ms) yes
```

```
| ?- canget(state(atdoor,onfloor,atwindow,hasnot)).


true ?


(16 ms) yes

| ?- trace.

The debugger will first creep -- showing everything (trace)


yes

{trace}

| ?- canget(state(atdoor,onfloor,atwindow,hasnot)).

      1    1  Call: canget(state(atdoor,onfloor,atwindow,hasnot)) ?

      2    2  Call:
move(state(atdoor,onfloor,atwindow,hasnot),_52,_92) ?

      2    2  Exit:
move(state(atdoor,onfloor,atwindow,hasnot),walk(atdoor,_80),state(_8
0,onfloor,atwindow,hasnot)) ?

      3    2  Call: canget(state(_80,onfloor,atwindow,hasnot)) ?

      4    3  Call:
move(state(_80,onfloor,atwindow,hasnot),_110,_150) ?

      4    3  Exit:
move(state(atwindow,onfloor,atwindow,hasnot),climb,state(atwindow,on
box,atwindow,hasnot)) ?

      5    3  Call: canget(state(atwindow,onbox,atwindow,hasnot)) ?

      6    4  Call:
move(state(atwindow,onbox,atwindow,hasnot),_165,_205) ?

      6    4  Fail:
move(state(atwindow,onbox,atwindow,hasnot),_165,_193) ?

      5    3  Fail: canget(state(atwindow,onbox,atwindow,hasnot)) ?

      4    3  Redo:
move(state(atwindow,onfloor,atwindow,hasnot),climb,state(atwindow,on
box,atwindow,hasnot)) ?

      4    3  Exit:
move(state(atwindow,onfloor,atwindow,hasnot),push(atwindow,_138),sta
te(_138,onfloor,_138,hasnot)) ?

      5    3  Call: canget(state(_138,onfloor,_138,hasnot)) ?

      6    4  Call: move(state(_138,onfloor,_138,hasnot),_168,_208)
?
```

```
      6    4  Exit:
move(state(_138,onfloor,_138,hasnot),climb,state(_138,onbox,_138,has
not)) ?

      7    4  Call: canget(state(_138,onbox,_138,hasnot)) ?

      8    5  Call: move(state(_138,onbox,_138,hasnot),_223,_263) ?

      8    5  Exit:
move(state(middle,onbox,middle,hasnot),grasp,state(middle,onbox,midd
le,has)) ?

      9    5  Call: canget(state(middle,onbox,middle,has)) ?

      9    5  Exit: canget(state(middle,onbox,middle,has)) ?

      7    4  Exit: canget(state(middle,onbox,middle,hasnot)) ?

      5    3  Exit: canget(state(middle,onfloor,middle,hasnot)) ?

      3    2  Exit: canget(state(atwindow,onfloor,atwindow,hasnot))
?

      1    1  Exit: canget(state(atdoor,onfloor,atwindow,hasnot)) ?


true ?
```