Artificial Intelligence Laboratory

CS 312

Task 9: Planning

Kunal Kumar
170010012

---

1. Pseudo code

Goal stack Planning Algorithm take Input as:

1. Start state : List of Predicates
2. Goal state : List of Predicates

It returns a List of Action as output.

```
def goal_stack_planning(start_state, goal):
   stack = list()       # stack for gsp
   plan = list()        # list of action
   state = start_state
   stack.append(goal)
   stack.extend(goal)
   while stack:
       # taking top sentence
       temp = stack.pop()
       if isinstance(temp, Action):
           plan.append(temp)    # putting in plan
           state.extend(temp.positive_effects)
           # new_state = old_state + action.positive_effects -
action.negative_effects
           state = [s for s in state if s not in
temp.negative_effects]
       elif isinstance(temp, list):
           solved = True
```

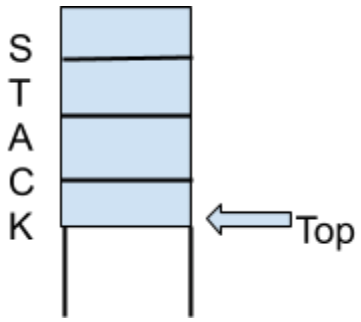
```
            # Checks if a conjunct of predicates is satisfied in
given state
            for predicate in temp:
                if not satisfy(predicate, state):
                    solved = False
            if not solved:
                stack.append(temp)
                stack.extend(temp)
        elif isinstance(temp, Predicate) and not satisfy(temp,
state):
            # choses an action to satisfy that predicate and state
            if temp.type_ == 'on':
                action = Stack(temp.args[0], temp.args[1])
            elif temp.type_ == 'ontable':
                action = Putdown(temp.args[0])
            elif temp.type_ == 'clear':
                if Predicate('hold', temp.args[0]) in state:
                    action = Putdown(temp.args[0])
                else:
                    action = Unstack(find_top(temp, state),
temp.args[0])
            elif temp.type_ == 'hold':
                if Predicate('ontable', temp.args[0]) in state:
                    action = Pick(temp.args[0])
                else:
                    action = Unstack(find_top(temp, state),
temp.args[0])
            elif temp.type_ == 'AE':
                action = Putdown(find_hold(state))
            # put action in stack
            stack.append(action)
            stack.append(action.preconditions)
            stack.extend(action.preconditions)
    return plan
```

2. Input and output for 3 given examples

S. No	Input	Output
1	4 (on b a)^(ontable a)^(ontable c)^(ontable d)^(AE) (on c a)^(on b d)^(ontable a)^(ontable d)	(unstack b a) (putdown b) (stack c a) (stack b d)
2	4 (ontable a)^(ontable b)^(ontable c)^(ontable d) (on a b)^(on b c)^(on c d)	(stack a b) (unstack a b) (putdown a) (stack b c) (unstack b c) (putdown b) (stack c d) (stack a b) (unstack a b) (putdown a) (stack b c) (stack a b)
3	3 (ontable a)^(ontable b)^(ontable c) (on a b)^(on b c)	(stack a b) (unstack a b) (putdown a) (stack b c) (stack a b)

3. Stack visualization for first example

Stack Structure:



S No.	Stack
0 (starting)	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (on c a)
1	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a) (AE)^(clear a)^(clear c) (AE) (clear a) (clear c)
2	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a) (AE)^(clear a)^(clear c) (AE) (clear a)

3	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a) (AE)^(clear a)^(clear c) (AE) (unstack b a) (AE)^(clear b)^(on b a) (AE) (clear b) (on b a)
4	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a) (AE)^(clear a)^(clear c) (AE) (unstack b a) (AE)^(clear b)^(on b a) (AE) (clear b)
5	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a) (AE)^(clear a)^(clear c) (AE) (unstack b a) (AE)^(clear b)^(on b a) (AE)
6	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a) (AE)^(clear a)^(clear c) (AE) (unstack b a) (AE)^(clear b)^(on b a)

7	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a) (AE)^(clear a)^(clear c) (AE) (unstack b a)
8	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a) (AE)^(clear a)^(clear c) (AE)
9	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a) (AE)^(clear a)^(clear c) (putdown b) (hold b) (hold b)
10	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a) (AE)^(clear a)^(clear c) (putdown b) (hold b)
11	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a) (AE)^(clear a)^(clear c) (putdown b)

12	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a) (AE)^(clear a)^(clear c)
13	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d) (stack c a)
14	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (on b d)
15	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (stack b d) (AE)^(clear d)^(clear b) (AE) (clear d) (clear b)
16	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (stack b d) (AE)^(clear d)^(clear b) (AE) (clear d)
17	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (stack b d) (AE)^(clear d)^(clear b) (AE)
18	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (stack b d) (AE)^(clear d)^(clear b)

19	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a) (stack b d)
20	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d) (ontable a)
21	(ontable d)^(ontable a)^(on b d)^(on c a) (ontable d)
22	(ontable d)^(ontable a)^(on b d)^(on c a)

Plan Output:

(unstack b a)
(putdown b)
(stack c a)
(stack b d)