# CS 312: Artificial Intelligence Laboratory
# Lab 9 Report

S V Praveen - 170010025

April 20, 2020

## 1 Introduction

The objective of this task is to simulate goal stack planning in the block world domain for the given start state and goal state.

## 2 Pseudo Code

**Algorithm 1** Goal Stack Planning

```
procedure GSP(givenState, givenGoal, actions)
    state ← givenState
    plan ← () {start with empty plan}
    stack ← emptyStack {start with empty stack}
    PushSet(givenGoal, stack)
    while not Empty(stack) do
      x ← Pop(stack)
      if x ∈ actions then
        plan ← (plan · x)
        state ← Progress(x, state)
      else if x is conjunct of goal predicates C then
        solvedFlag ← TRUE
        for each G ∈ C do
          if G is unsatisfied in state then
            solvedFlag ← FALSE
          end if
        end for
        if solvedFlag = FALSE then
          pushSet(C, stack)
        end if
      else if x ∉ givenState then
        a ← chooseAction(x, state)
        if a is None then
          return FAILURE
        end if
        Push(a, stack)
        PushSet(Preconditions(a), stack)
      end if
    end while
    return plan
```

# 3 Input-Output for Given Examples

| Sl No. | Input | Output |
|---|---|---|
| 1. | 4<br>(on b a)^(ontable a)^(ontable c)^(ontable d)^(AE)<br>(on c a)^(on b d)^(ontable a)^(ontable d) | (unstack b a)<br>(putdown b)<br>(stack c a)<br>(stack b d) |
| 2. | 4<br>(ontable a)^(ontable b)^(ontable c)^(ontable d)<br>(on a b)^(on b c)^(on c d) | (stack a b)<br>(unstack a b)<br>(putdown a)<br>(stack b c)<br>(unstack b c)<br>(putdown b)<br>(stack c d)<br>(stack a b)<br>(unstack a b)<br>(putdown a)<br>(stack b c)<br>(stack a b) |
| 3. | 3<br>(ontable a)^(ontable b)^(ontable c)<br>(on a b)^(on b c) | (stack a b)<br>(unstack a b)<br>(putdown a)<br>(stack b c)<br>(stack a b) |

# 4 Example 1: Stack Visualization

Pushed Goal state into stack initially.

1. **Stack:** pop()

   (ontable d)∧(ontable a)∧(on b d)∧(on c a)
   (ontable d)
   (ontable a)
   (on b d)

   Pushed stack ['c', 'a'] and preconditions

2. **Stack:** pop()

   (ontable d)∧(ontable a)∧(on b d)∧(on c a)
   (ontable d)
   (ontable a)
   (on b d)
   (stack c a)
   (clear c)∧(clear a)∧(AE )
   (clear c)
   (clear a)

3. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )
(clear c)

Pushed unstack ['b', 'a'] and preconditions

4. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )
(clear c)
(unstack b a)
(on b a)∧(clear b)∧(AE )
(on b a)
(clear b)

5. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )
(clear c)
(unstack b a)
(on b a)∧(clear b)∧(AE )
(on b a)

6. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )
(clear c)
(unstack b a)
(on b a)∧(clear b)∧(AE )

7. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )
(clear c)
(unstack b a)

8. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )
(clear c)

9. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )

10. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)

11. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )
(clear c)
(clear a)

Pushed putdown ['b'] and preconditions

12. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )
(clear c)
(clear a)
(putdown b)
(hold b)

13. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )
(clear c)
(clear a)
(putdown b)

14. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )
(clear c)
(clear a)

15. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )
(clear c)

16. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)
(clear c)∧(clear a)∧(AE )

17. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)
(stack c a)

18. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(on b d)

19. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)

Pushed stack ['b', 'd'] and preconditions

20. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(stack b d)
(clear b)∧(clear d)∧(AE )
(clear b)
(clear d)

21. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(stack b d)
(clear b)∧(clear d)∧(AE )
(clear b)

22. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(stack b d)
(clear b)∧(clear d)∧(AE )

23. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)
(stack b d)

24. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)
(ontable a)

25. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)
(ontable d)

26. **Stack:** pop()

(ontable d)∧(ontable a)∧(on b d)∧(on c a)

27. **Stack:** pop()

EMPTY