CS 312: Artificial Intelligence Laboratory
Lab 9 Report

S V Praveen - 170010025

April 21, 2020

1 Introduction

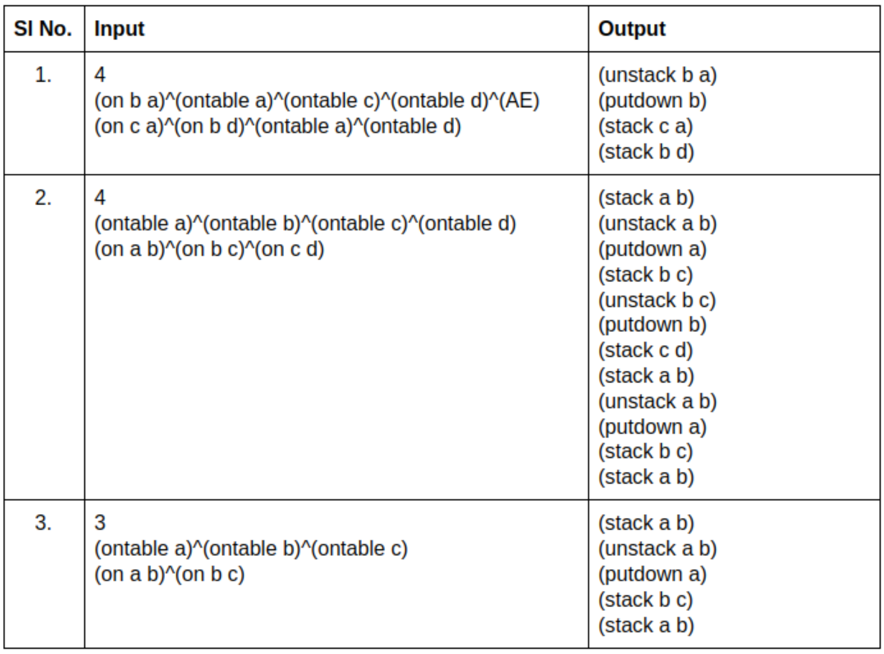
The objective of this task is to simulate goal stack planning in the block world domain for the given start state and goal state.

2 Pseudo Code

Algorithm 1 Goal Stack Planning

```
procedure GSP(givenState, givenGoal, actions)
    $state \leftarrow givenState$
    $plan \leftarrow ()$ {start with empty plan}
    $stack \leftarrow emptyStack$ {start with empty stack}
    $PushSet(givenGoal, stack)$
    while not Empty(stack) do
        $x \leftarrow Pop(stack)$
        if $x \in actions$ then
            $plan \leftarrow (plan \cdot x)$
            $state \leftarrow Progress(x, state)$
        else if $x$ is conjunct of goal predicates $C$ then
            $solvedFlag \leftarrow TRUE$
            for each $G \in C$ do
                if $G$ is unsatisfied in $state$ then
                    $solvedFlag \leftarrow FALSE$
                end if
            end for
            if $solvedFlag = FALSE$ then
                $pushSet(C, stack)$
            end if
        else if $x \notin givenState$ then
            $a \leftarrow chooseAction(x, state)$
            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 (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)

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)
(AE )∧(clear a)∧(clear c)
(AE )
(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)
(AE )∧(clear a)∧(clear c)
(AE )
```

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)
(AE )∧(clear a)∧(clear c)
(AE )
(unstack b a)
(AE )∧(clear b)∧(on b a)
(AE )
(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)
(AE )∧(clear a)∧(clear c)
(AE )
(unstack b a)
(AE )∧(clear b)∧(on b a)
(AE )
```

6. **Stack:** pop()

```
(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. **Stack:** pop()

```
(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. **Stack:** pop()

```
(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. **Stack:** pop()

```
(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)
```

Pushed putdown ['b'] and preconditions

10. **Stack:** pop()

```
(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. **Stack:** pop()

```
(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. **Stack:** pop()

```
(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. **Stack:** pop()

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

14. **Stack:** pop()

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

15. **Stack:** pop()

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

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

16. **Stack:** pop()

```
(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. **Stack:** pop()

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

18. **Stack:** pop()

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

19. **Stack:** pop()

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

20. **Stack:** pop()

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

21. **Stack:** pop()

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

22. **Stack:** pop()

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

23. **Stack:** pop()

```
EMPTY
```