

LP1 Assignment AI&R 3

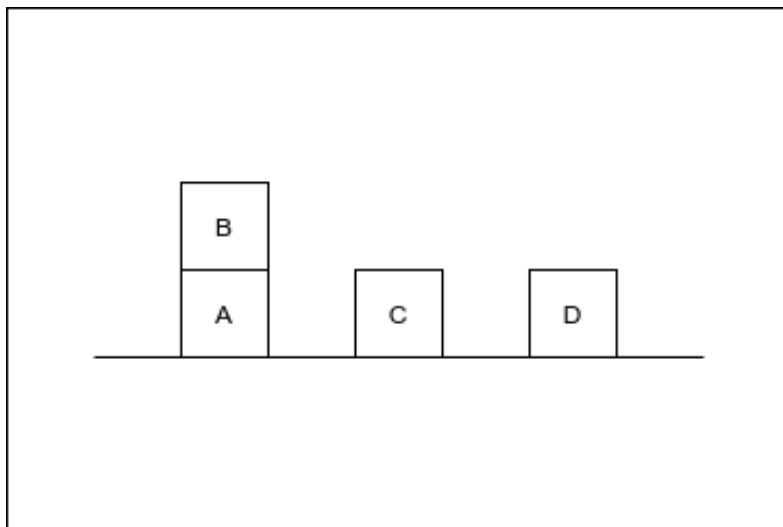
Date: 5th November, 2020

Title: Goal Stack Planning

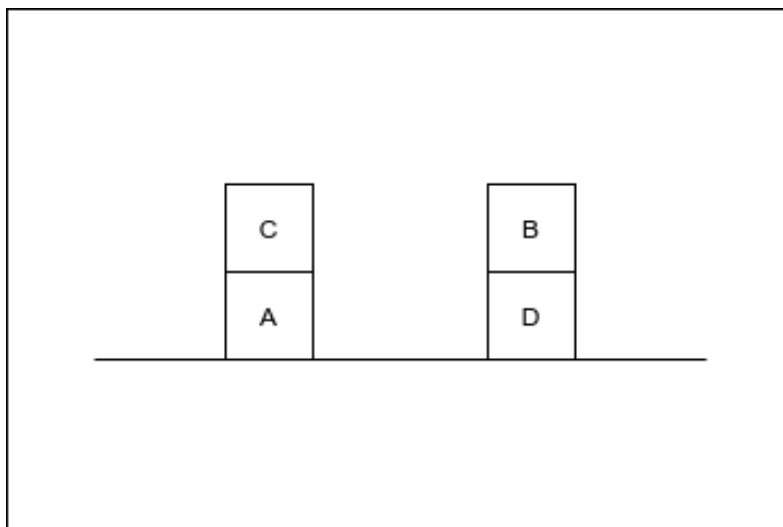
Problem Definition:

Implement goal stack planning for the following configuration from the blocks world

Initial State



Goal State



Learning Objectives:

- To learn and implement goal stack planning

Learning Outcomes:

- I will be able to learn and implement Goal Stack Planning

S/W & H/W Packages:

- Operating System: 64-bit Windows 10 OS
- Programming Language: Python 3
- Jupyter Notebook Environment: Google Colaboratory

Related Mathematics:

$$S = \{s; e; X; Y; Fme; Ff; DD; NDD\}$$

s = initial state

- $ON(B,A) \wedge ONTABLE(A) \wedge ONTABLE(C) \wedge ONTABLE(D)$

e = goal state

- $ON(C,A) \wedge ON(B,D) \wedge ONTABLE(A) \wedge ONTABLE(D)$

X = {X1}

- $X1 = s$

Y = {Y1}

- $Y1 = e$

$F_{me} = \{f_0\}$

- f_0 = function to perform Goal Stack Planning

$F_f = \{f_1, f_2, f_3, f_4, f_5\}$ where

- f_1 = function to display final path
- f_2 = function to replace unsatisfied goal with an action
- f_3 = function to check if object is predicate
- f_4 = function to check if object is action
- f_5 = function to get status of the arm

DD = List of Predicates of Initial State

NDD = No non deterministic data

Concepts Related to Theory:

Block World Problem

- There is a flat surface on which blocks can be placed
- There are a number of square blocks, all the same size
- They can be stacked one upon another
- There is a robot arm that can manipulate the blocks

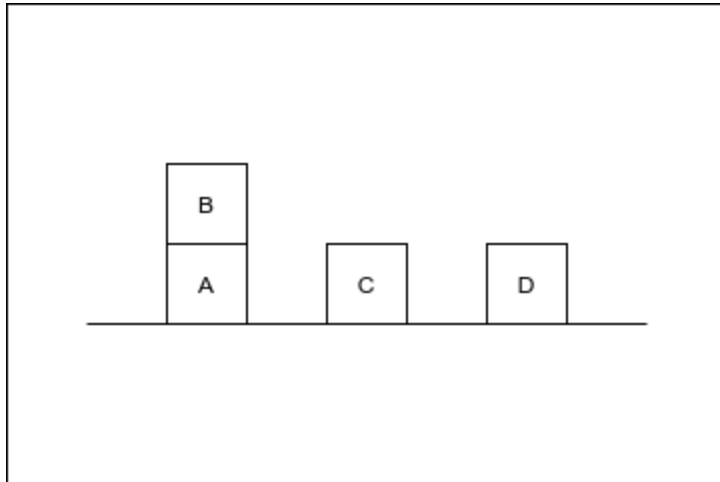
Goal Stack Planning

- We work backwards from the goal, looking for an operator which has one or more of the goal literals as one of its effects and then trying to satisfy the preconditions of the operator.
- The preconditions of the operator become subgoals that must be satisfied. We keep doing this until we reach the initial state.

- Goal stack planning uses a stack to hold goals and actions to satisfy the goals, and a knowledge base to hold the current state, action schemas and domain axioms

Initial State

$ON(B,A) \wedge ONTABLE(A) \wedge ONTABLE(C) \wedge ONTABLE(D) \wedge CLEAR(B) \wedge CLEAR(C) \wedge CLEAR(D)$



Perception

Predicates

PREDICATE	MEANING
$ON(A,B)$	Block A is on B
$ONTABLE(A)$	A is on table
$CLEAR(A)$	Nothing is on top of A
$HOLDING(A)$	Arm is holding A
$ARMEMPTY$	Arm is holding nothing

First Order Logic

LOGICAL STATEMENT	MEANING
$[\exists X : HOLDING(X)] \rightarrow \sim ARMEMPTY$	If arm is holding anything, then it is not empty
$\forall X : ONTABLE(X) \rightarrow \sim \exists Y : ON(X,Y)$	If a block is on table, then it is not on another block
$\forall X : [\sim \exists Y : ON(Y,X)] \rightarrow CLEAR(X)$	Any block with no block on it is clear

Cognition

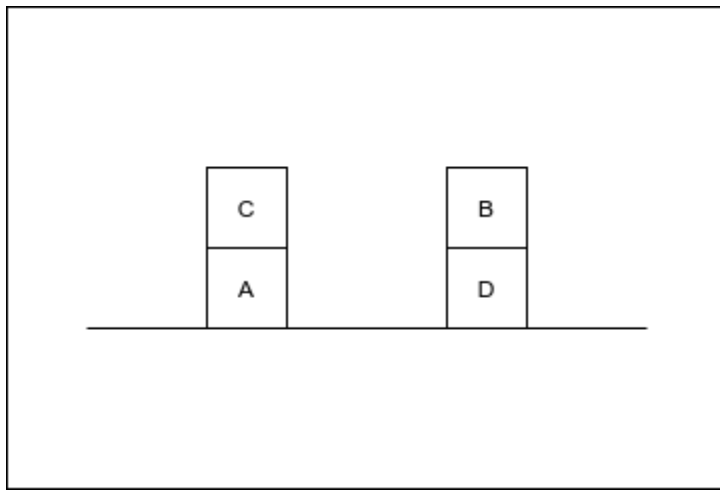
Goal Stack Planning Algorithm

1. Push the goal state on the stack.
2. Repeat until the stack is empty:
 1. If stack top is a compound goal
 1. push its unsatisfied subgoals on the stack.
 2. If stack top is a single unsatisfied goal
 1. replace it by an action that makes it satisfied
 2. push the action's precondition on the stack.
 3. If stack top is an action
 1. check for unsatisfied prerequisites
 2. if all prerequisites are satisfied
 1. pop action from the stack
 2. execute it
 3. change the knowledge base by the action's effects.
 3. else
 1. push unsatisfied preconditions on the stack
 4. If stack top is a satisfied goal
 1. pop it from the stack.

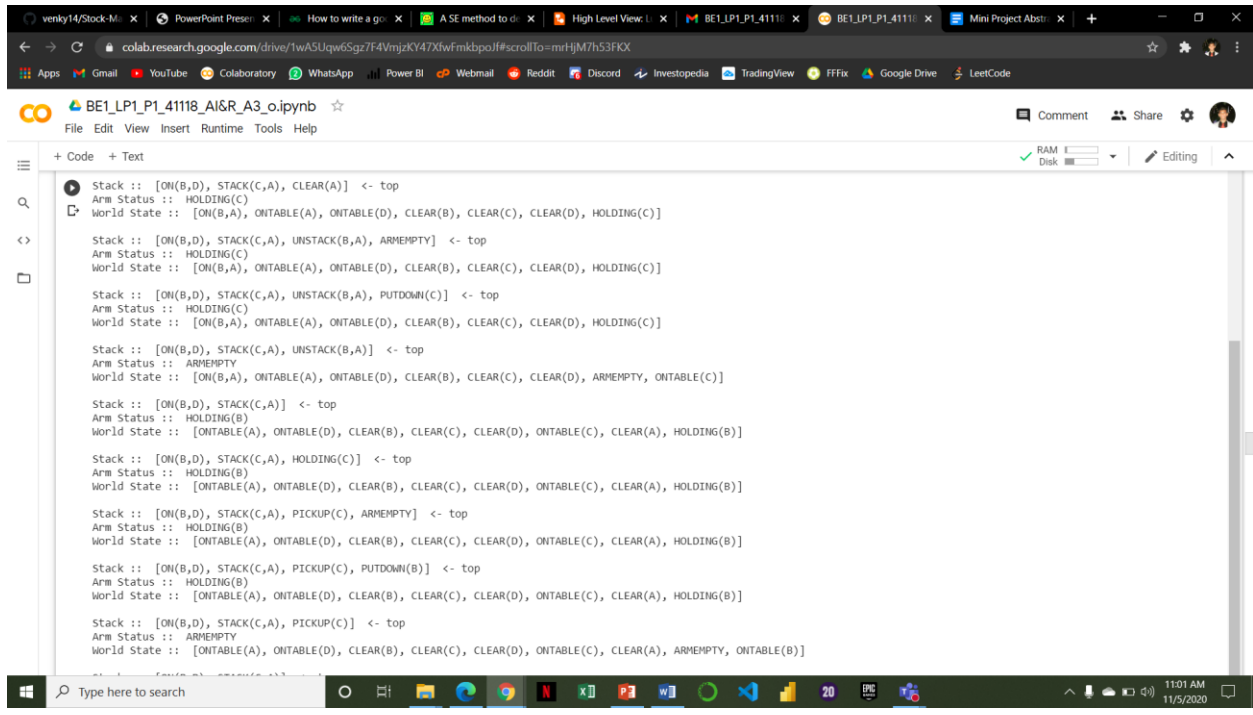
Action

OPERATORS	PRECONDITION	DELETE	ADD
STACK(X,Y)	CLEAR(Y) \wedge HOLDING(X)	CLEAR(Y) HOLDING(X)	ARMEMPTY ON(X,Y)
UNSTACK(X,Y)	ARMEMPTY \wedge ON(X,Y) \wedge CLEAR(X)	ARMEMPTY \wedge ON(X,Y)	HOLDING(X) \wedge CLEAR(Y)
PICKUP(X)	CLEAR(X) \wedge ONTABLE(X) \wedge ARMEMPTY	ONTABLE(X) \wedge ARMEMPTY	HOLDING(X)
PUTDOWN(X)	HOLDING(X)	HOLDING(X)	ONTABLE(X) \wedge ARMEMPTY

Goal State



Output:

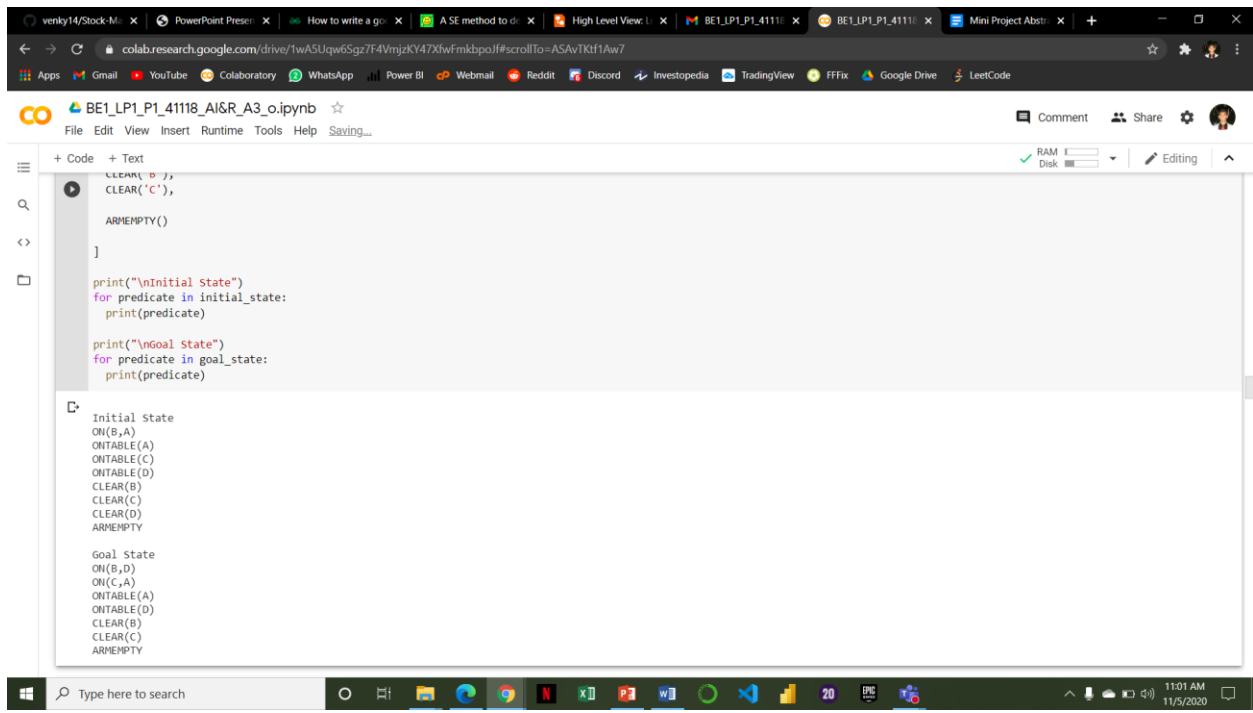


The screenshot shows a Google Colab notebook with the title "BE1_LP1_P1_41118_AI&R_A3_o.ipynb". The code is written in Prolog-like syntax and shows a sequence of state transitions. The initial state is defined as:

```
Stack :: [ON(B,D), STACK(C,A), CLEAR(A)] <- top
Arm Status :: HOLDING(C)
World State :: [ON(B,A), ONTABLE(A), ONTABLE(D), CLEAR(B), CLEAR(C), CLEAR(D), HOLDING(C)]
```

The code then shows a series of actions (UNSTACK, PUTDOWN, PICKUP) and their corresponding state updates. The final state is:

```
Stack :: [ON(B,D), STACK(C,A), PICKUP(C)] <- top
Arm Status :: ARMEMPTY
World State :: [ONTABLE(A), ONTABLE(D), CLEAR(B), CLEAR(C), CLEAR(D), ONTABLE(C), CLEAR(A), ARMEMPTY, ONTABLE(B)]
```



The screenshot shows a Google Colab notebook with the title "BE1_LP1_P1_41118_AI&R_A3_o.ipynb". The code is written in Python and shows the initial and goal states of a problem. The initial state is defined as:

```
Initial State
ON(B,A)
ONTABLE(A)
ONTABLE(C)
ONTABLE(D)
CLEAR(B)
CLEAR(C)
CLEAR(D)
ARMEMPTY
```

The goal state is defined as:

```
Goal State
ON(B,D)
ON(C,A)
ONTABLE(A)
ONTABLE(D)
CLEAR(B)
CLEAR(C)
ARMEMPTY
```

```
World State :: [ONTABLE(A), ONTABLE(D), CLEAR(B), CLEAR(C), CLEAR(D), ONTABLE(B), ARMEMPTY, ON(C,A)]

Stack :: [STACK(B,D)] <- top
Arm Status :: ARMEMPTY
World State :: [ONTABLE(A), ONTABLE(D), CLEAR(B), CLEAR(C), CLEAR(D), ONTABLE(B), ARMEMPTY, ON(C,A)]

Stack :: [STACK(B,D)] <- top
Arm Status :: HOLDING(B)
World State :: [ONTABLE(A), ONTABLE(D), CLEAR(B), CLEAR(C), CLEAR(D), ON(C,A), HOLDING(B)]

Stack :: [] <- top
Arm Status :: ARMEMPTY
World State :: [ONTABLE(A), ONTABLE(D), CLEAR(B), CLEAR(C), ON(C,A), ARMEMPTY, ON(B,D)]

[9] print('Goal State : ', goal_state)
print('World State : ', world_state)
print('Are the two states equal? : ', (set(goal_state)==set(world_state)))

Goal State : [ON(B,D), ON(C,A), ONTABLE(A), ONTABLE(D), CLEAR(B), CLEAR(C), ARMEMPTY]
World State : [ONTABLE(A), ONTABLE(D), CLEAR(B), CLEAR(C), ON(C,A), ARMEMPTY, ON(B,D)]
Are the two states equal? : True

[10] #Steps from Initial State to Goal State
steps

[PICKUP(C),
PUTDOWN(C),
UNSTACK(B,A),
PUTDOWN(B),
PICKUP(C),
STACK(C,A),
PICKUP(B),
STACK(B,D)]
```

Notebook Link:

<https://colab.research.google.com/drive/1wA5Uqw6Sgz7F4VmizKY47XfwFmkbpoJf?usp=sharing>

Conclusion:

I have successfully designed and implemented Goal Stack Planning for Block World Problem