# -\*- coding: utf-8 -\*-

"""6\_Goal\_stack\_planning.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1SUfjushzd3xzxAE2fed6e2oz3jmZeqx5

"""

#!/usr/bin/env python

# coding: utf-8

# In[1]:

import sys

#Some lists required for algorithm

start\_state = [] #required for storing initial state

goal\_state = [] #required for storing final state

current\_state = [] #required for storing current ongoing state

planning\_stack = [] #stack required in goal stack planning (only going to add sub goals)

actual\_plan = [] #plan generated (output)

#actions and predicates

actions = ["stack", "unstack", "pickup", "putdown"]

predicates = ["on", "clear", "arm\_empty", "holding", "on\_table"]

#necessary functions required for algorithm

#Preconditions append functions:-

def preconditions\_stack(X, Y):

planning\_stack.append("holding "+str(X))

planning\_stack.append("clear "+str(Y))

def preconditions\_unstack(X, Y):

planning\_stack.append("on "+str(X)+" "+str(Y))

planning\_stack.append("clear "+str(X))

def preconditions\_pickup(X):

planning\_stack.append("arm\_empty")

planning\_stack.append("on\_table "+str(X))

planning\_stack.append("clear "+str(X))

def preconditions\_putdown(X):

planning\_stack.append("holding "+str(X))

#Corresponding action required to satisfy the predicates

def for\_on(X, Y):

planning\_stack.append("stack "+str(X)+" "+str(Y))

preconditions\_stack(X, Y)

def for\_ontable(X):

planning\_stack.append("putdown "+str(X))

preconditions\_putdown(X)

def for\_clear(X):

#Finding the block on which X is stacked

check = "on "

for predicate in current\_state:

if check in predicate:

temp\_list = predicate.split()

if temp\_list[2] == X:

break

Y = str(temp\_list[1])

#Appending Unstack operatrion

planning\_stack.append("unstack "+str(Y)+" "+str(X))

preconditions\_unstack(Y, X)

def for\_holding(X):

check = "on\_table "+str(X)

if check in current\_state:

planning\_stack.append("pickup "+str(X))

preconditions\_pickup(X)

else:

#Finding the block on which X is stacked

check = "on "

for predicate in current\_state:

if check in predicate:

temp\_list = predicate.split()

if temp\_list[2] == X:

break

Y = str(temp\_list[1])

#Appending Unstack operatrion

planning\_stack.append("unstack "+str(Y)+" "+str(X))

preconditions\_unstack(Y, X)

def for\_armempty():

print("\nArm empty predicate false\n")

sys.exit()

#Effects of action

def effect\_stack(X, Y):

current\_state.remove("holding "+str(X))

current\_state.remove("clear "+str(Y))

current\_state.append("on "+str(X)+" "+str(Y))

current\_state.append("clear "+str(X))

current\_state.append("arm\_empty")

def effect\_unstack(X, Y):

current\_state.remove("on "+str(X)+" "+str(Y))

current\_state.remove("clear "+str(X))

current\_state.remove("arm\_empty")

current\_state.append("holding "+str(X))

current\_state.append("clear "+str(Y))

def effect\_pickup(X):

current\_state.remove("arm\_empty")

current\_state.remove("on\_table "+str(X))

current\_state.remove("clear "+str(X))

current\_state.append("holding "+str(X))

def effect\_putdown(X):

current\_state.remove("holding "+str(X))

current\_state.append("arm\_empty")

current\_state.append("on\_table "+str(X))

current\_state.append("clear "+str(X))

input\_string = input("Enter start state:- ")

start\_state = input\_string.split("^")

input\_string = input("Enter goal state:- ")

goal\_state = input\_string.split("^")

print("\nEntered Start State:- "+str(start\_state))

print("\nEntered Goal State:- "+str(goal\_state)+"\n")

current\_state = start\_state.copy()

for predicate in goal\_state:

planning\_stack.append(predicate)

while len(planning\_stack) > 0:

print("Planning Stack:- "+str(planning\_stack))

print("Current State:- "+str(current\_state)+"\n")

top = planning\_stack.pop()

temp = top.split()

if temp[0] in predicates: #if top of stack is predicate

if top in current\_state: #if predicate is true:

continue #You have already poped it.

else:

#Already poped above

#push corresponding action that will satisfy that predicate onto stack and push preconditions of that action

if temp[0] == "on":

for\_on(temp[1], temp[2])

elif temp[0] == "on\_table":

for\_ontable(temp[1])

elif temp[0] == "clear":

for\_clear(temp[1])

elif temp[0] == "holding":

for\_holding(temp[1])

elif temp[0] == "arm\_empty":

for\_armempty()

if temp[0] in actions: #if top of stack is action

#Already poped above

#perform the action i.e add and delete it's effects from current state

if temp[0] == "stack":

effect\_stack(temp[1], temp[2])

elif temp[0] == "unstack":

effect\_unstack(temp[1], temp[2])

elif temp[0] == "pickup":

effect\_pickup(temp[1])

elif temp[0] == "putdown":

effect\_putdown(temp[1])

#add that action to the actual plan

actual\_plan.append(top)

print("Final Current State:- "+str(current\_state))

print("\nPlan Generated:- \n")

for step in actual\_plan:

print(step)

**Output:**

Enter start state:- on B A^on\_table A^clear B^arm\_empty

Enter goal state:- on A B^on\_table B^clear A^arm\_empty

Entered Start State:- ['on B A', 'on\_table A', 'clear B', 'arm\_empty']

Entered Goal State:- ['on A B', 'on\_table B', 'clear A', 'arm\_empty']

Planning Stack:- ['on A B', 'on\_table B', 'clear A', 'arm\_empty']

Current State:- ['on B A', 'on\_table A', 'clear B', 'arm\_empty']

Planning Stack:- ['on A B', 'on\_table B', 'clear A']

Current State:- ['on B A', 'on\_table A', 'clear B', 'arm\_empty']

Planning Stack:- ['on A B', 'on\_table B', 'unstack B A', 'on B A', 'clear B']

Current State:- ['on B A', 'on\_table A', 'clear B', 'arm\_empty']

Planning Stack:- ['on A B', 'on\_table B', 'unstack B A', 'on B A']

Current State:- ['on B A', 'on\_table A', 'clear B', 'arm\_empty']

Planning Stack:- ['on A B', 'on\_table B', 'unstack B A']

Current State:- ['on B A', 'on\_table A', 'clear B', 'arm\_empty']

Planning Stack:- ['on A B', 'on\_table B']

Current State:- ['on\_table A', 'holding B', 'clear A']

Planning Stack:- ['on A B', 'putdown B', 'holding B']

Current State:- ['on\_table A', 'holding B', 'clear A']

Planning Stack:- ['on A B', 'putdown B']

Current State:- ['on\_table A', 'holding B', 'clear A']

Planning Stack:- ['on A B']

Current State:- ['on\_table A', 'clear A', 'arm\_empty', 'on\_table B', 'clear B']

Planning Stack:- ['stack A B', 'holding A', 'clear B']

Current State:- ['on\_table A', 'clear A', 'arm\_empty', 'on\_table B', 'clear B']

Planning Stack:- ['stack A B', 'holding A']

Current State:- ['on\_table A', 'clear A', 'arm\_empty', 'on\_table B', 'clear B']

Planning Stack:- ['stack A B', 'pickup A', 'arm\_empty', 'on\_table A', 'clear A']

Current State:- ['on\_table A', 'clear A', 'arm\_empty', 'on\_table B', 'clear B']

Planning Stack:- ['stack A B', 'pickup A', 'arm\_empty', 'on\_table A']

Current State:- ['on\_table A', 'clear A', 'arm\_empty', 'on\_table B', 'clear B']

Planning Stack:- ['stack A B', 'pickup A', 'arm\_empty']

Current State:- ['on\_table A', 'clear A', 'arm\_empty', 'on\_table B', 'clear B']

Planning Stack:- ['stack A B', 'pickup A']

Current State:- ['on\_table A', 'clear A', 'arm\_empty', 'on\_table B', 'clear B']

Planning Stack:- ['stack A B']

Current State:- ['on\_table B', 'clear B', 'holding A']

Final Current State:- ['on\_table B', 'on A B', 'clear A', 'arm\_empty']

Plan Generated:-

unstack B A

putdown B

pickup A

stack A B