Assignment No: 7

**Title:**

Implement goal stack planning for the following configurations from the blocks world.

**Theory:**

Planning is process of determining various actions that often lead to a solution. Planning is useful for non- decomposable problems where sub goals often interact. Goal Stack Planning (in short GSP) is the one of the simplest planning algorithm that is designed to handle problems having compound goals. And it utilizes STRIP as a formal language for specifying and manipulating the world with which it is working. his approach uses a Stack for plan generation. The stack can contain Sub-goal and actions described using radiates. The Sub-goals can be solved one by one in any order.

Introduction: The planning in Artificial Intelligence is about the decision making tasks performed by the robots or computer programs to achieve a specific goal. The execution of planning is about choosing a sequence of actions with a high likelihood to complete the specific task.

**Components of Planning System**

* The planning consists of following important steps:
* Choose the best rule for applying the next rule based on the best available heuristics.
* Apply the chosen rule for computing the new problem state.
* Detect when a solution has been found.
* Detect dead ends so that they can be abandoned and the system’s effort is directed in more fruitful directions.
* Detect when an almost correct solution has been found.

**Blocks-World planning problem**

The blocks-world problem is known as Sussman Anomaly. Non interleaved planners of the early 1970s were unable to solve this problem, hence it is considered as anomalous. When two subgoals G1 and G2 are given, a non-interleaved planner produces either a plan for G1 concatenated with a plan for G2, or vice-versa.In blocks-world problem, three blocks labeled as 'A', 'B', 'C' are allowed to rest on the flat surface. The given condition is that only one block can be moved at a time to achieve the goal.

**Goal stack planning**

This is one of the most important planning algorithms, which is specifically used by STRIPS.

The stack is used in an algorithm to hold the action and satisfy the goal. A knowledge base is used to hold the current state, actions.

Goal stack is similar to a node in a search tree, where the branches are created if there is a choice of an action.

The important steps of the algorithm are as stated below:

i. Start by pushing the original goal on the stack. Repeat this until the stack becomes empty. If stack top is a compound goal, then push its unsatisfied subgoals on the stack.

ii. If stack top is a single unsatisfied goal then, replace it by an action and push the action’s precondition on the stack to satisfy the condition.

iii. If stack top is an action, pop it from the stack, execute it and change the knowledge base by the effects of the action.

iv. If stack top is a satisfied goal, pop it from the stack. Basic Idea to handle interactive compound goals uses goal stacks, Here the stack contains

**Algorithm:**

Push the Goal state in to the Stack

Push the individual Predicates of the Goal State into the Stack Loop till the

Stack is empty

Pop an element E from the stack

IF E is a Predicate

IF E is True then

Do Nothing

ELSE

Push the relevant action into the Stack

Push the individual predicates of the Precondition of the action into the

Stack

Else IF E is an Action

Apply the action to the current State.

Add the action ‘a’ to the plan

**Program:**

**GoalStackPlanning.java:**

import java.util.ArrayList;

import java.util.Scanner;

import java.util.Stack;

public class GoalStackPlanning {

State curr,goal;

String goal\_state;

ArrayList<String> steps;

Stack<String> st;

int n;

public GoalStackPlanning(int n, String start,String Goal){

this.n=n;

this.goal\_state=new String(Goal);

curr= new State(n,start);

this.goal=new State(n,Goal);

steps=new ArrayList<String>();

st=new Stack<String>();

st.push(goal\_state);

}

public void plan(){

String s[]=goal\_state.split("['^']");

for(int i=s.length-1;i>=0;i--){

st.push(s[i]);

}

while(!st.empty()){

String sub=st.pop();

if(sub.contains("^")){

if(sub.equals(goal\_state) && !curr.satisfy(sub)){

st.push(goal\_state);

}

if(!curr.satisfy(sub)){

String goals[]=sub.split("['^']");

for(int i=goals.length-1;i>=0;i--){

st.push(goals[i]);

}

}

}

else if(sub.contains("on ") && !curr.satisfy(sub)){

String elements[]=sub.split("[() ]+");

st.push("(stack "+elements[2].charAt(0)+" " + elements[3]+")");

st.push("(clear "+elements[2].charAt(0)+")^(clear "+elements[3]+")^(AE)");

}

else if(sub.contains("ontable") && !curr.satisfy(sub)){

String elements[]=sub.split("[() ]+");

st.push("(putdown "+elements[2].charAt(0)+")");

st.push("(hold "+elements[2].charAt(0)+")");

}

else if(sub.contains("clear") && !curr.satisfy(sub)){

String elements[]=sub.split("[() ]+");

if(curr.hold[elements[2].charAt(0)-97]==1){

st.push("(putdown "+elements[2].charAt(0)+")");

st.push("(hold "+elements[2].charAt(0)+")");

}

else{

int temp=curr.checkTop(elements[2].charAt(0));

if(temp!=-1){

st.push("(unstack "+Character.toString((char)(temp+97))+" "+elements[2].charAt(0)+")");

st.push("(on "+Character.toString((char)(temp+97))+" "+elements[2].charAt(0)+")^(clear "+Character.toString((char)(temp+97))+")^(AE)");

}

}

}

else if(sub.contains("hold") && !curr.satisfy(sub)){

String elements[]=sub.split("[() ]+");

if(curr.ontable[elements[2].charAt(0)-97]==1){

st.push("(pick "+elements[2].charAt(0)+")");

st.push("(ontable "+elements[2].charAt(0)+")^(clear "+elements[2].charAt(0)+")^(AE)");

}

else{

int temp=curr.checkTop(elements[2].charAt(0));

if(temp!=-1){

st.push("(unstack "+Character.toString((char)(temp+97))+" "+elements[2].charAt(0)+")");

st.push("(on "+Character.toString((char)(temp+97))+" "+elements[2].charAt(0)+")^(clear "+Character.toString((char)(temp+97))+")^(AE)");

}

}

}

else if(sub.contains("AE") && !curr.satisfy(sub)){

for(int i=0;i<n;i++){

if(curr.hold[i]==1){

st.push("(putdown "+Character.toString((char)(i+97))+")");

st.push("(hold "+Character.toString((char)(i+97))+")");

}

}

}

else if(sub.contains("pick") || sub.contains("putdown") || sub.contains("stack") || sub.contains("unstack")){

curr.prformAction(sub);

steps.add(sub);

}

}

}

public void printSteps(){

System.out.println();

System.out.println("Steps taken:");

for(String step:steps){

System.out.println(step);

}

}

public static void main(String[] args) {

int n;

String start,goal;

Scanner sc=new Scanner(System.in);

System.out.println("Enter no of blocks");

n=sc.nextInt();

sc.nextLine();

System.out.println("Enter Start state: ");

start=sc.nextLine();

System.out.println("Enter goal state: ");

goal=sc.nextLine();

GoalStackPlanning obj=new GoalStackPlanning(n,start,goal);

obj.plan();

obj.printSteps();

}

}

**State.java :**

public class State {

int on[][];

int hold[];

int clear[];

int ontable[];

Boolean arm\_empty;

public State(int n,String state){

on=new int[n][n];

hold=new int[n];

clear=new int[n];

ontable=new int[n];

arm\_empty=Boolean.TRUE;

setState(state);

}

public void setState(String state){

String subgoals[]=state.split("['^']+");

for(String sub:subgoals){

String elems[]=sub.split("[() ]+");

if(elems[1].equals("on")){

on[elems[2].charAt(0)-97][elems[3].charAt(0)-97]=1;

}

else if(elems[1].equals("ontable")){

ontable[elems[2].charAt(0)-97]=1;

}

else if(elems[1].equals("clear")){

clear[elems[2].charAt(0)-97]=1;

}

else if(elems[1].equals("hold")){

hold[elems[2].charAt(0)-97]=1;

}

else if(sub.contains("AE")){

arm\_empty=Boolean.TRUE;

}

}

}

//satisfy compares current state with goal state ,if same return true

public Boolean satisfy(String goal){

String subgoals[]=goal.split("['^']+");

for(String sub:subgoals){

String elems[]=sub.split("[() ]+");

if(elems[1].equals("on") && on[elems[2].charAt(0)-97][elems[3].charAt(0)-97]==1){

continue;

}

else if(elems[1].equals("ontable") && ontable[elems[2].charAt(0)-97]==1){

continue;

}

else if(elems[1].equals("clear") && clear[elems[2].charAt(0)-97]==1){

continue;

}

else if(elems[1].equals("hold") && hold[elems[2].charAt(0)-97]==1){

continue;

}

else if(elems[1].equals("AE") && arm\_empty==Boolean.TRUE){

continue;

}

else{

return Boolean.FALSE;

}

}

return Boolean.TRUE;

}

public void prformAction(String action){

String elems[]=action.split("[() ]+");

if(elems[1].equals("putdown")){

ontable[elems[2].charAt(0)-97]=1;

clear[elems[2].charAt(0)-97]=1;

hold[elems[2].charAt(0)-97]=0;

arm\_empty=Boolean.TRUE;

}

else if(elems[1].equals("pick")){

ontable[elems[2].charAt(0)-97]=0;

clear[elems[2].charAt(0)-97]=0;

hold[elems[2].charAt(0)-97]=1;

arm\_empty=Boolean.FALSE;

}

else if(elems[1].equals("unstack")){

on[elems[2].charAt(0)-97][elems[3].charAt(0)-97]=0;

clear[elems[2].charAt(0)-97]=0;

clear[elems[3].charAt(0)-97]=1;

hold[elems[2].charAt(0)-97]=1;

arm\_empty=Boolean.FALSE;

}

else if(elems[1].equals("stack")){

on[elems[2].charAt(0)-97][elems[3].charAt(0)-97]=1;

clear[elems[2].charAt(0)-97]=1;

clear[elems[3].charAt(0)-97]=0;

hold[elems[2].charAt(0)-97]=0;

arm\_empty=Boolean.TRUE;

}

}

public int checkTop(char c){

int len=hold.length;

for(int i=0;i<len;i++){

if(on[i][c-97]==1){

return i;

}

}

return -1;

}

}

**Input.txt:**

4

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

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

**Output:**

