K. J. Somaiya Institute of Engineering and Information Technology, Sion, Mumbai

Accredited 'A' Grade by NAAC with 3.21 CGPA
3 Programs Accredited by National Board of Accreditation
Permanently Affiliated to University of Mumbai,
Best College Award by University of Mumbai (Urban Region),
ISTE (MH), and CSI (Mumbai)
UGC Recognized Institute under Section 2(f) and 12(B) of the UGC Act, 1956

DEPARTMENT OF INFORMATION TECHNOLOGY

Academic Year (2020-2021) Odd Semester VII
Course: Artificial Intelligence
Experiment No. 07

Aim: Design of a Planning System Using STRIPS**.**

Objectives: To impart a basic understanding of some of the more advanced topics of AI such as planning.

Outcomes: Attain the capability to represent various real-life problem domains using logic-based techniques and use this to perform inference or planning.

Theory:

Language of Planning Problem:

Introduction to STRIPS (The Stanford Research Institute Problem):

The Stanford Research Institute Problem Solver (STRIPS) is an automated planning technique that works by executing a domain and problem to find a goal. With STRIPS, we first describe the world. We do this by providing objects, actions, preconditions, and effects. These are all the types of things we can do in the game world.

Once the world is described, you then provide a problem set. A problem consists of an initial state and a goal condition. STRIPS can then search all possible states, starting from the initial one, executing various actions, until it reaches the goal.

A common language for writing STRIPS domain and problem sets is the Planning Domain Definition Language (PDDL). PDDL lets us write most of the code with English words so that it can be clearly read and (hopefully) well understood. It's a relatively easy approach to writing simple AI planning problems.

Problem Statement: Design a planning agent for a Blocks World problem. Assume a suitable initial state and final state for the problem. Designing the Agent Idea is to give an agent:

Designing the Agent

Idea is to give an agent:

• Representation of goal/intention to achieve

• Representation of actions it can perform; and

• Representation of the environment; Then have the agent generate a plan to achieve the goal.

The plan is generated entirely by the planning system, without human intervention. Assume start &amp; goal states as below:

a. STRIPS: A planning system – Has rules with precondition deletion list and addition list

The sequence of actions:

b. Grab C

c. Pickup C

d. Place on table C

e. Grab B

f. Pickup B

g. Stack B on C

h. Grab A

i. Pickup A

j. Stack A on B Rules:

k. R1: pickup(x)

1. Precondition &amp; Deletion List: hand empty, on (x, table), clear(x)

2. Add List: holding(x)

l. R2: putdown(x)

1. Precondition &amp; Deletion List: holding(x)

2. Add List: hand empty, on (x, table), clear(x)

m. R3: stack (x, y)

1. Precondition &amp; Deletion List: holding(x), clear(y)

2. Add List: on (x, y), clear(x)

n. R4: unstack (x, y)

1. Precondition &amp; Deletion List: on (x, y), clear(x)

2. Add List: holding(x), clear(y)

Plan for the assumed blocks world problem for the given problem, start $\rightarrow$ Goal can be achieved by the following sequence:

1. Unstack (C, A)
2. Putdown(C)
3. Pickup(B)
4. Stack (B, C)
5. Pickup(A)
6. Stack (A, B)

Source Code :

Domain: Blocks World 3

Code:

```
(define (domain blocksworld)
  (:requirements :strips)
  (:action move
     :parameters (?b ?x ?y)
```

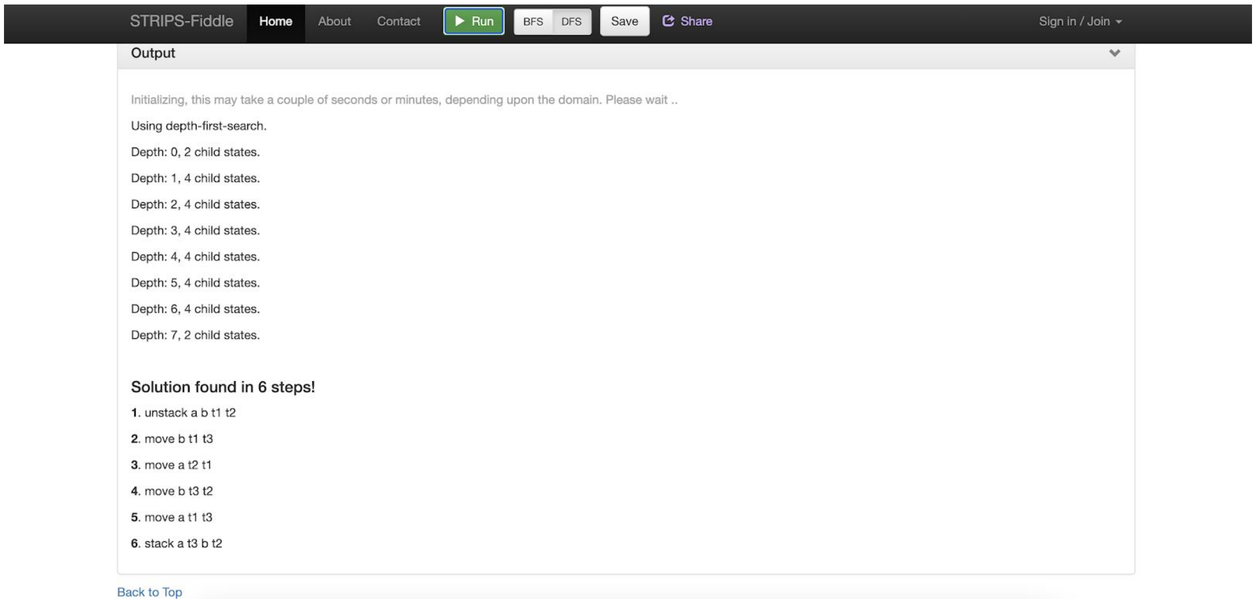
```
    :precondition (and (block ?b) (table ?x) (table ?y) (on ?b ?x) (clear ?b) (clear ?y))
    :effect (not (on ?b ?x) (on ?b ?y) (clear ?x) not (clear ?y))
    )
  (:action stack
    :parameters (?a ?x ?b ?y)
    :precondition (and (block ?a) (block ?b) (table ?x) (table ?y) (clear ?a) (clear ?b) (on ?a ?x)
(on ?b ?y))
    :effect (and (on ?a ?b) not (on ?a ?x) not (clear ?b) (clear ?x))
    )
  (:action unstack
    :parameters (?a ?b ?x ?y)
    :precondition (and (block ?a) (block ?b) (table ?x) (table ?y) (on ?b ?x) (on ?a ?b) (clear ?a)
(clear ?y))
    :effect (and (on ?a ?y) not (on ?a ?b) (clear ?b) (clear ?a) not (clear ?y))
    )
)
```

Problem : Stack Blocks Stacked a b From Table 1 to Stacked a b Table 2, One Pile Per Table

Code :

```
(define (problem stack-blocks-stacked-ab-from-table1-to-stacked-ab-table2-onepilepertable)
   (:domain blocksworld)
  (:init (and (block a) (block b) (table t1) (table t2) (table t3)
        (on b t1) (on a b) (clear a) (clear t2) (clear t3)))
  (:goal (and (on a b) (on b t2)))
)
```

Output :



```
Initializing, this may take a couple of seconds or minutes, depending upon the domain. Please wait ..
Using depth-first-search.
Depth: 0, 2 child states.
Depth: 1, 4 child states.
Depth: 2, 4 child states.
Depth: 3, 4 child states.
Depth: 4, 4 child states.
Depth: 5, 4 child states.
Depth: 6, 4 child states.
Depth: 7, 2 child states.

Solution found in 6 steps!
1. unstack a b t1 t2
2. move b t1 t3
3. move a t2 t1
4. move b t3 t2
5. move a t1 t3
6. stack a t3 b t2
```

Conclusion:

From this experiment, we understood the concept of design of a planning system using STRIPS. We also understood the capability to represent various real-life problem domains using logic-based techniques and use this to perform inference or planning.