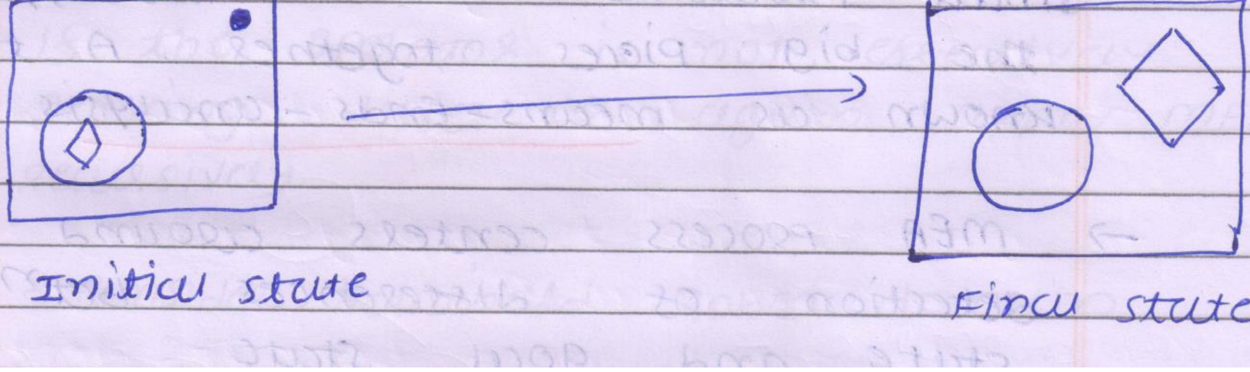
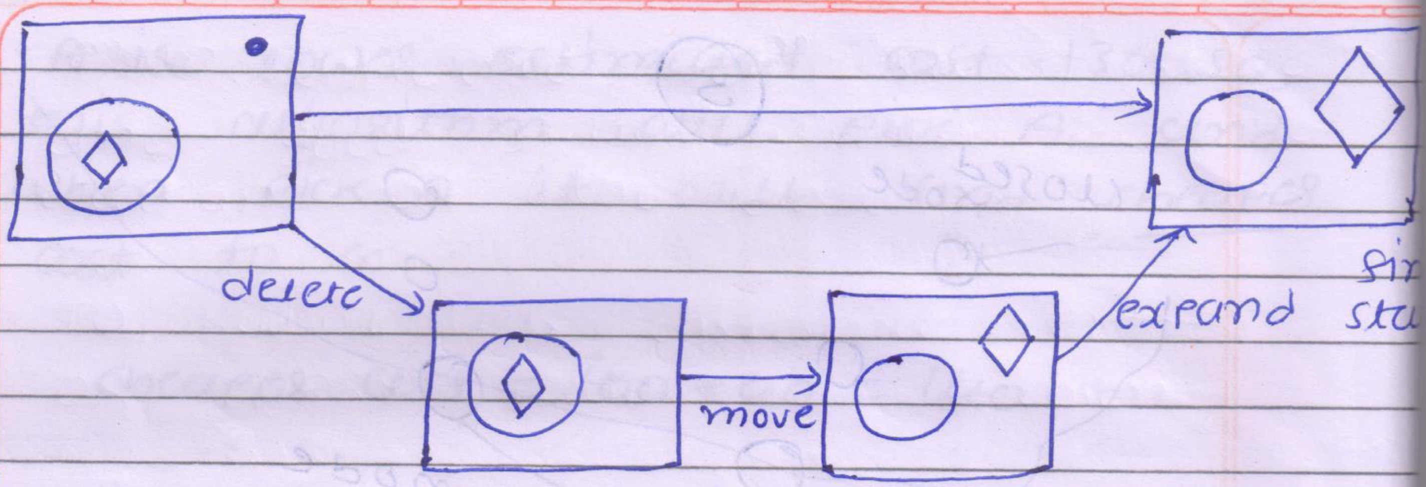
Means - Ends Analysis (Problem solving Technique) [MEA]





- → Backward + Forward search strategy is used
- → Such a mixed strategy would make it possible to solve the major parts of the problem first and then go back and solve the small problems that arise in gluing the big pieces together. A technique known as <u>means-Ends-analysis</u>

- → MEA process centers around the detection of differences betn current state and goal state

- → Use operator that can reduce the difference betn Initial and goal state

- → Perhaps the operator cannot be applied to current state

  e.g In above example we cannot use move, and expand first

  → first use delete operator then we can use other operator move then expand

→ So, we setup a subproblem of getting to a state in which it can be applied. The kind of Backward chaining in which operators are selected and then subgoals are set up to establish the preconditions of operators is called operation subgoaling

→ But may be operator does not produce exactly the goal state we want, Then we have a second subproblem of getting from the state it does produce to the goal

→ if the operator is not effectively reducing difference, again apply MEA recursively

Ex:- consider household robot domain

- available operators as shown in fig

- suppose that robot in this domain were given a problem of moving desk with two things on it from one room to another

- objects on top must also be removed

- main difference betn start state and goal state would be the location of the desk

→ To reduce this difference either PUSH or CARRY could be choosen

→ If carry is choosen, its preconditi[on] must be met
  - location of robot
  - size of desk

→ location of robot can handle by walk, but there are no operato[rs] that can change size of object

→ so path leads to dead-end

→ we attempt to apply PUSH
  four codn
  - robot must be at desk
  - desk must be clear
  - desk is large
  - robot's arm empty

Assumtion: 2 small objects are resting o[n] desk

Step 1

To move robot to place1

start | WALK |←—— difference reduced ——→| goal

A (place1) B Z

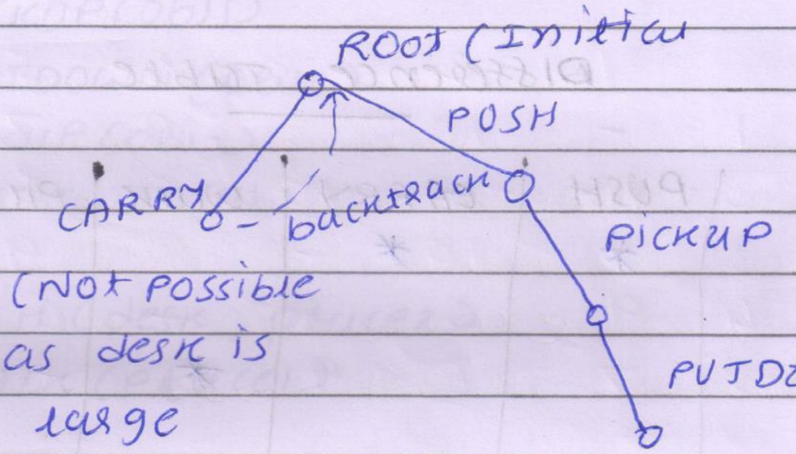
Step2 | WALK | ↑ |<— PUSH —| ↑ |

A B Z

PUSH identified as carry cannot be applied

B: Remove object from desk

Place object on desk again



~~Other~~ Robot's operators

Operator	Preconditions	Results
PUSH(obj, loc)	at(robot, obj) ∧ large(obj) ∧ clear(obj) ∧ armempty	at(obj, loc) ∧ at(robot, loc
carry(obj, loc)	at(robot, obj) ∧ small(obj)	at(obj, loc) ∧ at(robot, loc)
WALK (loc)	none	at(robot, loc)

PICKUP(obj)	at(robot, obj)	holding(obj)
PUTDOWN(obj)	holding(obj)	¬ holding(obj)
PLACE(obj1, obj2)	at(robot, obj2) ∧ holding(obj1)	on(obj1, obj2)

Difference Table

	PUSH	CARRY	WALK	PICKUP	PUTDOWN	PLACE
move object	*	*				
move Robot			*			
clear object				*		
get object on object						*
get arm empty					*	*
Be holding object				*		

— Difference table that describes when each of operators is appropriate

SO, whole sequence of operators

1) WALK

2) CARRY (NOT POSSIBLE, SO backtrack go for PUSH)

~~1) WALK~~

3) PICKUP(obj1)

4) PUTDOWN(obj1)

5) PICKUP(obj2)

6) PUTDOWN (obj2)

7) PUSH(desk, place2)

8) WALK(place1)

9) carry (obj1, place2)

10) WALK (place1)

11) carry (obj2, place2)

12) PICKUP (obj1)

13) place (obj1, desk)

14) PICKUP (obj2)

15) place (obj2, desk)