UNIT-4

[redacted]

PLANNING (FULLY)

→ Planning: → the task of coming up with a sequence of actions that will achieve a goal is called planning.

→ Planning environment: (P.E)

$$\begin{tikzcd} & \text{Planning environment (P.E)} \arrow[dl] \arrow[dr] & \\ \text{classical P.E} \arrow[d] & & \text{Non-Classical P.E} \arrow[d] \\ \text{fully observable} & & \text{partially} \end{tikzcd}$$

→ Difficulty in Prob. Solving agent:
- → perform irrelevant actions → No prob decomposition
- → finding heuristic solution.

→ Planning Prob
- → Domain Model (Defines Action)
- → initial state (actions yet to take place)
- → final state (plan to achieve)

→ find the sequence of actions, achieve the goal from a gn initial world state

→ Simple Planning Agent:
- → An agent interacts with the real world with perceptions & Actions.

perceptions ↓ sense the world & assess the situation

Actions → wat agent does to the Domain

[Planning Agents:= Prob Solve + Knowle. Based.]

Problem-Solving Agent.
↓
Sequence of actions before acting

Knowledge. Based Agent
↓
logical representation of current State 2 effects of actions.

→ **Key ideas:**

→ planning emphasizes what is in operator 2 goal representation.

(i) to "open up" the representations of State, Goals 2 operators. So that intelligently select actions, when they are needed.

(ii) planner is free to add plan whenever needed than in an incremental Sequence from initial State

(iii) Most of the Parts are independent to other parts which makes more feasible to make goal 2 to solve (Divide 2 Conquer).

→ **Planning Lang:-**

Languages must represent
→ states, goals, actions

Languages must be
→ Expressive for ease of representation
→ flexible for Manipulation by Algo.

→ State Representation: → represented by conjuction of +ve literals.

using → Logical Propositions: Poor ∧ unknown.
→ FOL literals Must be ground & func. free.
↳ At (plane1, OMA) ∧ At (plan 2, JFK)

→ Closed world Assumption (wat is not stated is false).

→ goal representation: ⇒ partially specified state
→ proposition satisfies a goal if it contains all atoms of the goal & possibly others

→ Action representation ⇒ Action schema
(Action name, preconditions, Effects)

Eg: Action (FLY (p, from, to),
PRECOND At (p, from) ∧ plane(p) ∧ Airport(from) ∧ Airport (to)
EFFECT : ¬ At (p, from) ∧ At (p, to))

→ Sometimes effects are split into ADD & DELETE LIST.

Languages for planning problems.

STRIPS
↓
Standford Research institute prob. Solver.
↓
→ Makes use of
→ 1st order predicates
→ Allows func. free literals.

Eg: robot.

ADL
↓
Action Description Lang.

PDDL
↓
Planning Domain Definition Lang

Block world:
→ N no. of blocks resting on table with specified sequence.
→ goal to arange in desired sequence
Moves → put block on top of another and on table.
→ State is represented using sequence of blocks in correct position

STRIPS:

→ planner used in Shakey (1st robots in AI)
→ Action - centric representation (each action to effect)

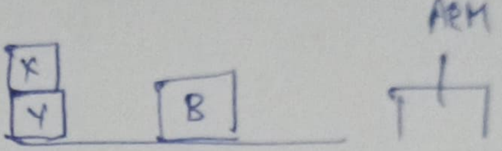
Strips planning prob:
Specifies
→ An initial state S
→ A goal state G
→ A set of strips actions

STRIPS representation:

→ precondition: set of assignment of values to features that must be true for actions to occur

→ Effect: set of resulting assignment of values to change the result of action.

BLOCK WORLD PROBLEM:



Action List:

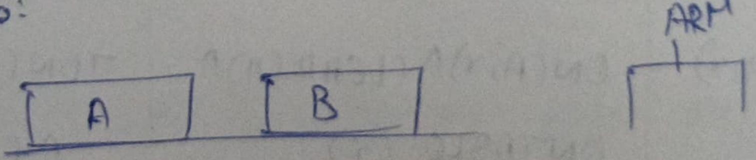
S.NO	Action	Precond.	Effects
1	pickup(x)	ARM Empty on(x, Table) clear.	Holding(x)

2.	Putdown(x)	Holding(x)	Arm empty
			on(x, table)
			Clear(x)

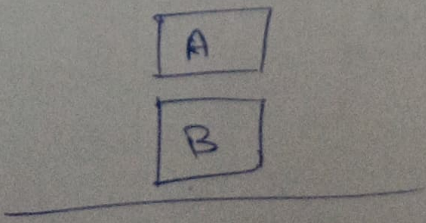
3.	stack(x,y)	Holding(x)
		clear(y)
			on(x,y)
			clear(x)
			Arm empty.

4	unstack (x,y)	on(x,y) Clear (x) Arm empty	Holding(x) Clear(Y

Start state:



goal state



Start State:

on(A, table)

on(B, table.

Goal State:

on(A, B)

Solution

on(A, B)

$\boxed{\text{Stack(A,B)}}$

$\boxed{\text{Pickup(A)}}$

Precond:

Arm Empty

On(A, table)

Clear(A)

STRIPS STYLE OPERATORS:

A ON(A, B, S0) ^

B ONTABLE (B, S0) ^ CLEAR (A, S0)

Operator	Pre cond.	Effects.
Move (A, X, Y)	ON(A, X) ^ CLEAR(A) ^ ONTABLE(Y)	¬ ON(A, X) ^ ON(A, Y) ¬CLEAR(Y) ^ CLEAR(
MoveToTable(A, X)	ON(A, X) ^ CLEAR(A) ^ ONTABLE (X)	¬ON(A, X) ^ ONTABLE(A) ^ CLEAR(X)

Means- Ends Analysis: (Mixture of Backward & forward search tech)

→ Search strategies either forward or Backward.

→ Mixed Strategy: Solve major probe then solve the smaller then combine them.

→ process centers around finding the diff bet current state & goal state.

→ Solve recursively.

MEA Algo:

Step1: Compare current to goal, if no diff then return success & exit.

Step2: Select the Most significant difference & reduce it

→ select operator O else failure.

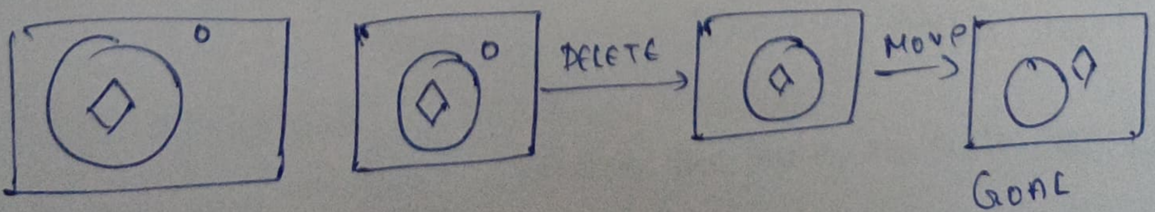
→ operator O to CURRENT (O-START), (O-RESULT)

→ if (FIRST-PART ← ---- MEA(current, O-START)

AND

(LAST-PART ← --- MEA (O-RESULT, GOAL)

Return the result combining FIRST-PART, O & LAST-PART

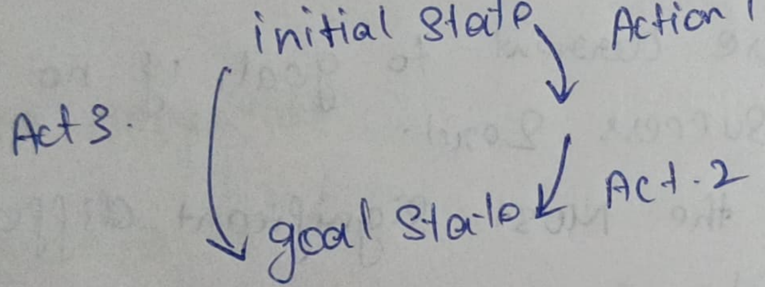


NON - LINEAR PLANNING: (NLP)

→ A plan that consist of sub probs, which are solved simultaneously k.a NLP

→ In case of goal stack planning there are some probs.

→ To achieve any goal, it could have an impact on the one that has been achieved.



→ There is a concept of constraint posting that comes with non-linear planning.

→ Posting states the plan can be built by.

* Addition of op / suggested op
* Ordering them
* Binding the variables to the operators.

Conditional planning:

3 Kinds of environments:

→ fully observable. (Agent also knows the current state)

→ Partially observable (Agent knows only a certain amt abt the actual state)

→ unknown (nothing known)

→ Agents used conditional steps to check the State of envir. to Decide wat to Do next.

→ Plan info stores in Lib:

Action (Left) ? clean ∪ Right

Syntax: if then plan_A else plan-B

Reactive planning:

→ planning under uncertainity.

→ makes use of if-then rules.

→ concept that able to handle unknown situation.

→ Rule selection → priority & holding condition that max.

present in execution(Active) & holding (pre-Action priority others inactive)

→ B-tree structure.

→ Algo selects the rule, no rule can be selected

→ Dependent on Algo implementation for rule selection.

Rote learning Slide (44) ✓

Reinforce learning (82) ✓

feed forward neural N/w.

Ensemble Learning. (102)

ML $\rightarrow$ $\textcircled{36}$ types (49) ✓