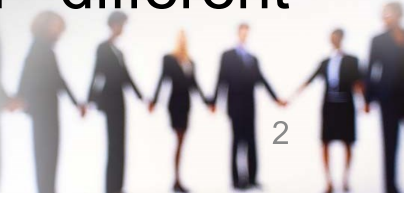
Unit 3

Planning

What is Planning?

- The task of coming up with a sequence of actions that will achieve a goal is called planning.
- Planning Environments
    1. Classical Planning Environments
        - Fully observable, deterministic, finite, static and discrete.
    2. Non classical Planning Environments
        - Partially observable, stochastic with different algorithms and agent designs.



Planning (Contd..)

- Difficulty in problem-solving agent
  - Performs irrelevant actions
    - Have to explore all the states
    - Ex: buy book with 10-digit ISBN→ 10^{10} actions
  - Finding heuristics function
    - Problem-solving agent lacks autonomy because it depends on human to supply heuristic function for each new problem.
  - No problem decomposition
- **All these problems are overcome by planning agent by representing the goal as conjunction of subgoals.**

PLANNING PROBLEM

The planning problem is actually the question how to go to next state or the goal state from the current state. It involves two things 'how' and 'when'.

- The planning problem is defined with:

1. Domain model
2. Initial state
3. Goal state (next state)

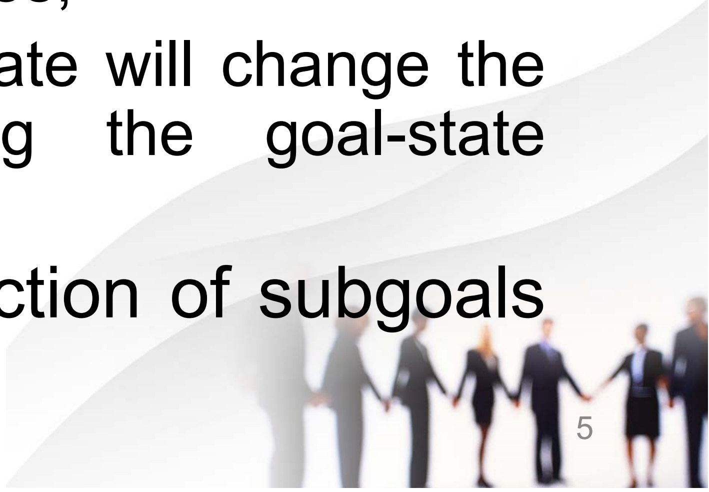
The domain model defines the actions along with the objects. It is necessary to specify the operators too that actually describe the action. Along with this, information about actions and state constraints while acting should also be given. This entirely formulates the domain model.

The initial state is the state where any action is yet to take place (the stage when the exam schedule is put up!).

The final state or the goal state is the state which the plan is intended to achieve.

Planning Problem

- Find **sequence of actions** - achieves a given **goal** from a given **initial world state**.
  - a set of operator descriptions
  - an initial state description, and
  - a goal state description or predicate,
- compute a plan
  - a sequence of operator instances,
  - executing them in the initial state will change the world to a state satisfying the goal-state description.
- Goals - specified as a conjunction of subgoals to be achieved

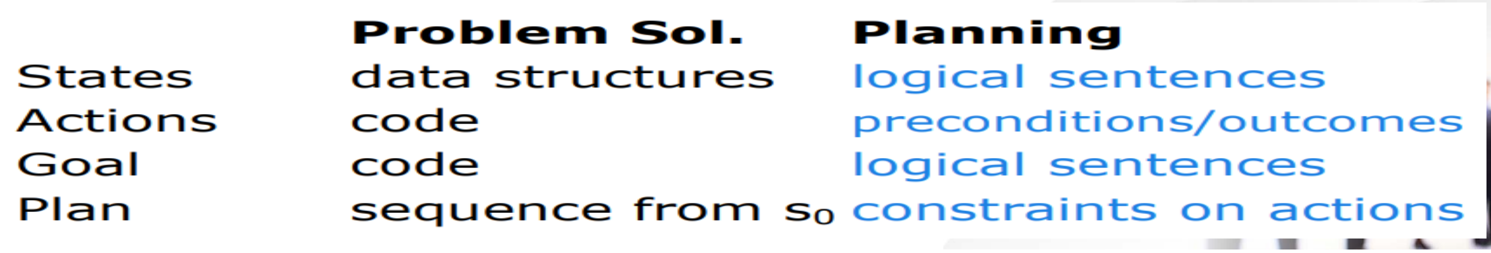


Simple Planning Agent

- An agent interacts with real world via perception and actions
- Perception - sense the world and assess the situation
- Actions - what the agent does in the domain.
- Planning involves reasoning about actions that the agent intends to carry out
- This reasoning involves the representation of the world that the agent has - representation of its actions.
- Hard constraints - objectives *have to* be achieved completely for success
- The objectives - soft constraints, or *preferences*, to be achieved as much as possible

Planning vs. Problem solving

- Planning agent is very similar to problem solving agent
  - Constructs plans to achieve goals, then executes them
- Planning is more powerful because of the representations and methods used
- Search - proceeds through *plan space* rather than *state space*
- Sub-goals - planned independently, it reduce the complexity of the planning problem



	Problem Sol.	**Planning**
States	data structures	logical sentences
Actions	code	preconditions/outcomes
Goal	code	logical sentences
Plan	sequence from s_0	constraints on actions

Planning Agents

- **problem-solving agents** are able to plan ahead - to consider the consequences of *sequences* of actions - before acting.
- **knowledge- based agents** can select actions based on explicit, logical representations of the current state and the effects of actions.
  - This allows the agent to succeed in complex, inaccessible environments that are too difficult for a problem-solving agent

– **Problem Solving Agents + Knowledge-based Agents = Planning Agents**

Simple Planning Agent

- A simple planning agent is very similar to problem-solving agents in that it constructs plans that achieve its goals, and then executes them.
- The limitations of the problem- solving approach motivates the design of planning systems.

To solve a planning problem using a state-space search approach we would let the:

- initial state = initial situation
- goal-test predicate = goal state description
- successor function computed from the set of operators
- once a goal is found, solution plan is the sequence of operators in the path from the start node to the goal node

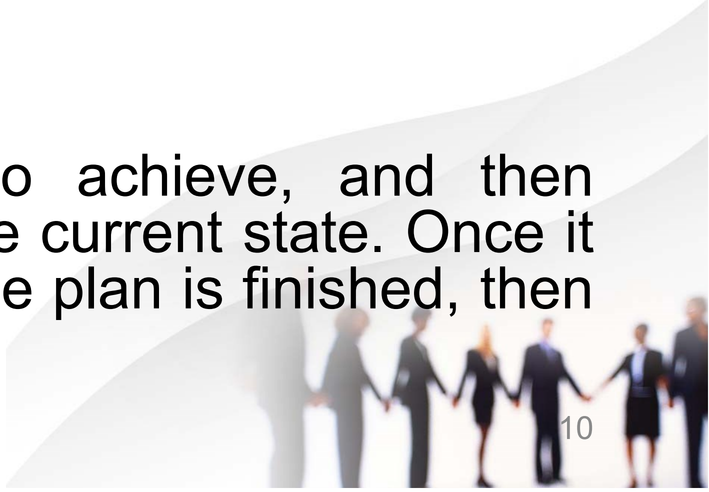
Simple Planning Agent

- Planning can be viewed as a type of problem solving in which the agent uses beliefs about actions and their consequences to search for a solution over the more abstract space of plans, rather than over the space of situations.

Algorithm of a simple planning agent:

1. Generate a goal to achieve
2. Construct a plan to achieve goal from current state
3. Execute plan until finished
4. Begin again with new goal

- The agent first generates a goal to achieve, and then constructs a plan to achieve it from the current state. Once it has a plan, it keeps executing it until the plan is finished, then begins again with a new goal.



Key Ideas Behind Planning

- Planning emphasizes what is in operator and goal representations.

There are three key ideas behind planning:

- to "open up" the representations of state, goals, and operators so that a reasoner can more intelligently select actions when they are needed
- the planner is free to add actions to the plan wherever they are needed, rather than in an incremental sequence starting at the initial state
- most parts of the world are independent of most other parts which makes it feasible to take a conjunctive goal and solve it with a divide-and-conquer strategy

Planning Languages

- Languages must represent..
  - States
  - Goals
  - Actions
- Languages must be
  - Expressive for ease of representation
  - Flexible for manipulation by algorithms

State Representation

- A state is represented with a conjunction of positive literals
- Using
  - Logical Propositions: *Poor* ∧ *Unknown*
  - FOL literals: *At(Plane1,OMA)* ∧ *At(Plan2,JFK)*
- FOL literals must be ground & function-free
  - Not allowed: *At(x,y)* or *At(Father(Fred),Sydney)*
- Closed World Assumption
  - What is not stated are assumed false

Goal Representation

- Goal is a partially specified state
- A proposition satisfies a goal if it contains all the atoms of the goal and possibly others..
  - Example: Rich $\wedge$ Famous $\wedge$ Miserable satisfies the goal Rich $\wedge$ Famous

Action Representation

At(WHI,LNK),Plane(WHI),
Airport(LNK), Airport(OHA)

$\boxed{\text{Fly(WHI,LNK,OHA)}}$

At(WHI,OHA), ¬ At(WHI,LNK)

- Action Schema
  - Action name
  - Preconditions
  - Effects
- Example

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 list and DELETE list

Languages for Planning Problems

- STRIPS
  - Stanford Research Institute Problem Solver
  - Historically important
- ADL
  - Action Description Languages
- PDDL
  - Planning Domain Definition Language

Planning languages

- A language is the one that should be expressive.
- Three Languages
  - I. Stanford Research Institute Problem Solver (STRIPS)
  - 2. Action Description Language (ADL)
  - 3. Planning Domain Description Language (PDDL)

1) Stanford Research Institute Problem Solver (STRIPS) :

Makes use of the first order predicates.

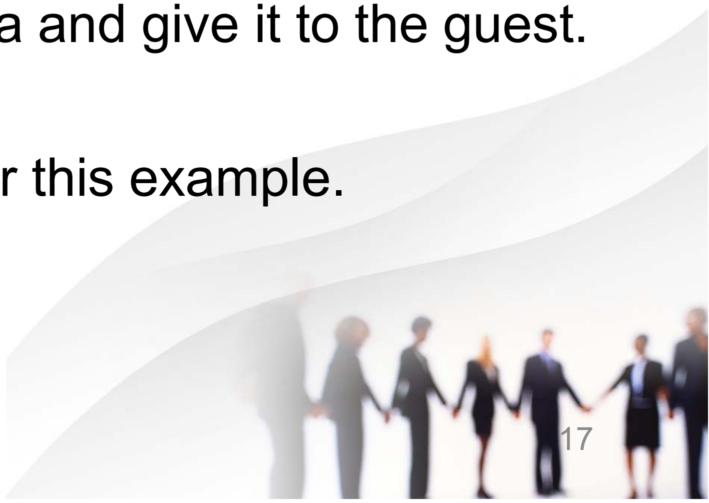
STRIPS allows function-free literals.

Example of a robot: The example involves a robot, a cup tea, guest and two rooms. We want the robot to get the tea and give it to the guest.

The planning with STRIPS is done as follows:

Let us begin with the STRIPS representation for this example.

Initial and final states are depicted in figure.



Example of a robot

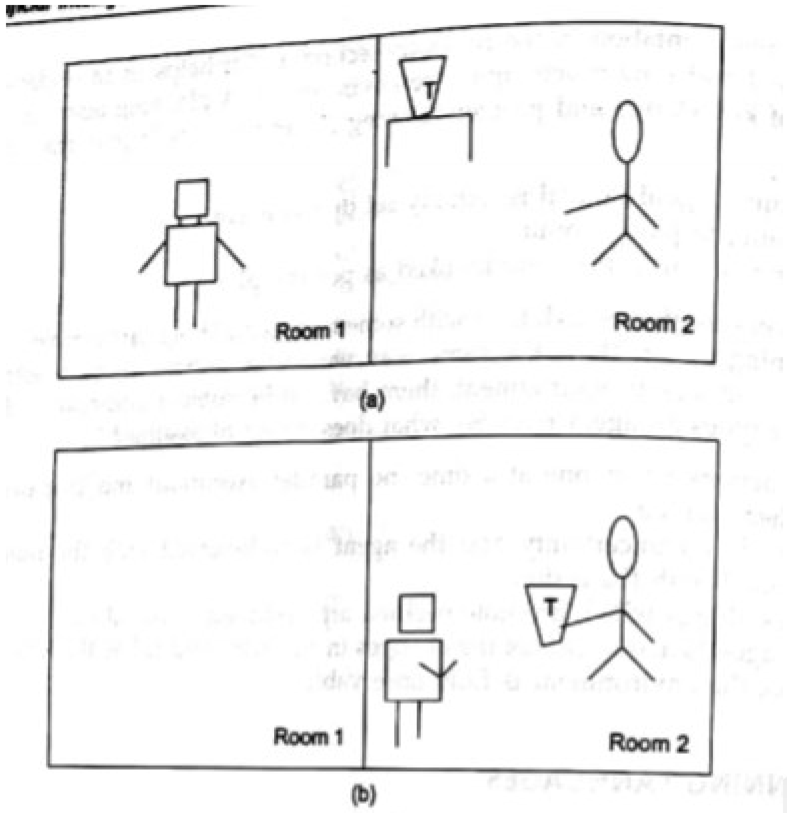


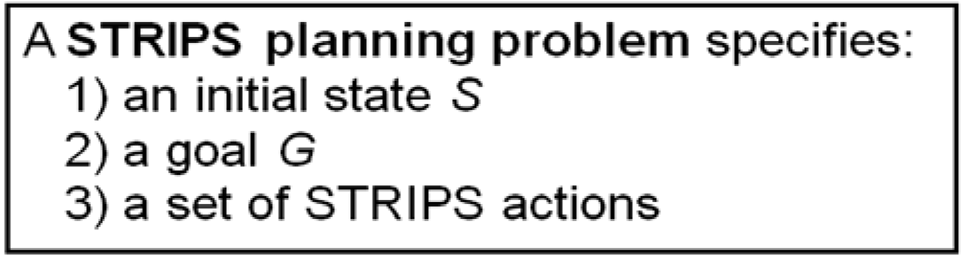
Figure 9.2 (a) Initial stage, (b) Goal state.

Block World

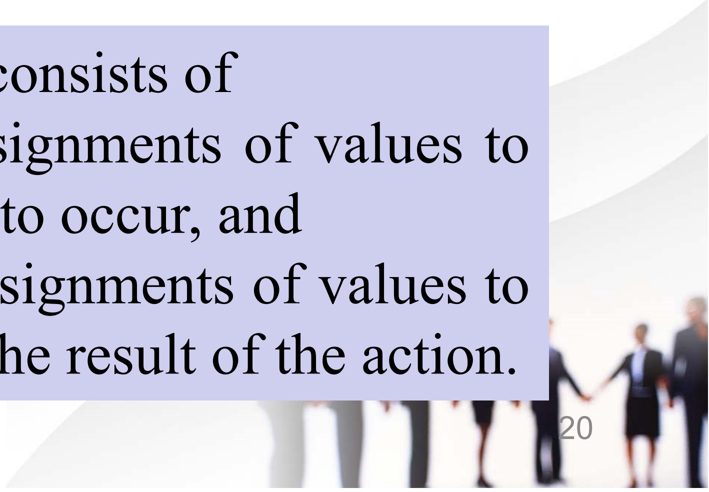
- There are ‘N’ number of Blocks resting on table with specified sequence.
- Goal is to arrange in desired sequence.
- Available moves
  - Put block on table
  - Put a block on another block top
- State is represented using sequence of blocks in current pos.

STRIPS

- STRIPS stands for "STanford Research Institute Problem Solver," was the planner used in Shakey, one of the first robots built using AI technology ,which is an action-centric representation ,for each action , specifies the effect of an action.


A **STRIPS planning problem** specifies:
1) an initial state *S*
2) a goal *G*
3) a set of STRIPS actions

The **STRIPS representation** for an action consists of
- the **precondition**, which is a set of assignments of values to features that must be true for the action to occur, and
- the **effect**, which is a set of resulting assignments of values to those primitive features that change as the result of the action.



Block World Problem

- Action List:

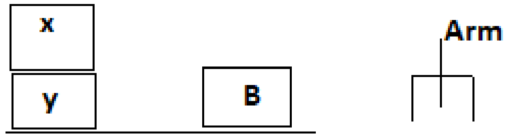
SNo	Action	Precondition	Effect

1	Pickup(x)	Arm Empty
		On(x, Table)
		Clear(x)
			Holding(x)

2	Putdown (x)	Holding(x)	Arm Empty
			On(x, Table)
			Clear(x)

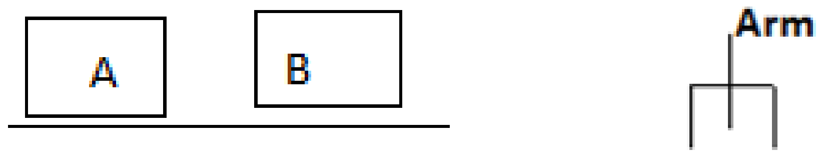
3	Stack(x,y)	Holding(x)
		Clear(y)
			On(x,y)
			Clear(x)
			Arm Empty

4	Unstack(x,y)	On(x,y)
		Clear(x)
		Arm Empty
			Holding(x)
			Clear(y)

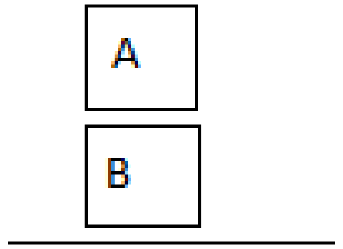


Block World Problem

- Start State:



- Goal State:



Block World Problem

- Start State:
  - On(A, table)
  - On(B, table)
- Goal State:
  - On(A,B)

Solution:

$$\begin{tikzcd} \text{On(A,B)} \arrow[d] \\ \boxed{\text{Stack(A,B)}} \end{tikzcd}$$

Preconditions:

$$\begin{tikzcd} ✗\ \text{Holding(A)} \arrow[r] & \boxed{\text{Pickup(A)}} \end{tikzcd}$$

✓ Clear(B)

Preconditions:

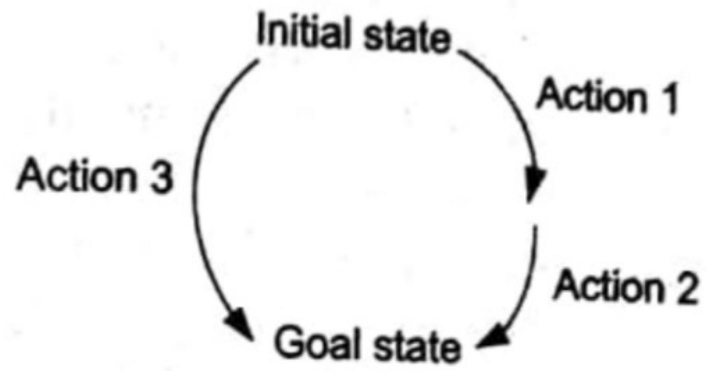
Arm Empty ✓

On(A, Table) ✓

Clear(A) ✓

NON-LINEAR PLANNING

- A plan that consists of sub-problems, which are solved simultaneously is said non-linear plan.
- In case of the goal stack planning there are some problems. To achieve any goal, it could have an impact on the one that has been achieved.



Non-linear planning.

NON-LINEAR PLANNING (Contd..)

There is a concept of constraint posting that comes with non-linear planning. The constraint posting states that the plan can be built by

- 1. Addition of operators or suggesting operators
- 2. Ordering them
- 3. Binding the variables to the operators

Conditional Planning

- Conditional planning has to work regardless of the outcome of an action.
- The outcome of actions cannot be determined so the environment is said to be nondeterministic.
- It's a way to deal with uncertainty by checking what is actually happening in the environment at predetermined points in the plan. (Conditional Steps)

Example:

Check whether SFO airport (San Francisco International Airport) is operational. If so, fly there; otherwise, fly to some other place.

Conditional Planning in Fully Observable Environments

- Agent used conditional steps to check the state of the environment to decide what to do next.
- Plan information stores in a library Ex: Action(Left)→ Clean v Right

Syntax: If then plan_A else plan_B

Reactive Planning

- Reactive planning is planning under uncertainty.
- Makes use of the if-then rules.
- The reactive planners are based on the concept that they should be able to handle an unknown situation too. So, the reaction rules are used that help them in doing so.
- A rule selection is based on the priority and a holding condition that maximises the priority.
- The rule which is at present in execution is said to be active whereas the and the ones with holding priority (we can call them possible competitors) are pre-active others are inactive.
- A B-tree structure is used in reactive planning, where the things are algorithm selects the rule. Sometimes, no rule can be selected. In such a case, dependent on the algorithm implementation for rule selection.