CS 1571 Introduction to AI
Lecture 22

Planning (cont.)

Uncertainty

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Administration

- **No new homework this week**
- **Homework 9 is due on Monday, November 27, 2006**

- **Final exam:**
  - December 11, 2006
  - 12:00-1:50pm, 5129 Sennott Square

Planning

Planning problem:
- find a sequence of actions that achieves some goal
- An instance of a search problem

Methods for modeling and solving a planning problem:
- State space search
- Situation calculus based on FOL
  - Inference rules
  - Resolution refutation

Planning problems

Properties of (real-world) planning problems:

- The description of the **state of the world is very complex**
- **Many possible actions** to apply in any step
- **Actions are typically local**
  - - they affect only a small portion of a state description
- **Goals** are defined as conditions and **refer only to a small portion of state**
- Plans consists of a **long sequence of actions**

- The state space search and situation calculus frameworks may be too cumbersome and inefficient to represent and solve the planning problems

Situation calculus: problems

Extends first order logic to situations
- **Allows us to model activities and changes in the world**

Problems:
- **Frame problem** refers to:
  – The need to represent a large number of frame axioms
- **Inferential frame problem:**
  – We need to derive properties that remain unchanged

Other problems:
- **Qualification problem** – enumeration of all possibilities under which an action holds
- **Ramification problem** – enumeration of all inferences that follow from some facts

STRIPS planner

Defines a **restricted representation language** as compared to the situation calculus

Advantage: leads to more efficient planning algorithms.
  – State-space search with structured representations of states, actions and goals
  – Action representation avoids the frame problem

STRIPS planning problem:
- much like a standard search (planning) problem;

Search in STRIPS

Objective:

Find a sequence of operators (a plan) from the initial state to the state satisfying the goal

Two approaches to build a plan:

- **Forward state space search (goal progression)**
  - Start from what is known in the initial state and apply operators in the order they are applied
- **Backward state space search (goal regression)**
  - Start from the description of the goal and identify actions that help to reach the goal

State-space search

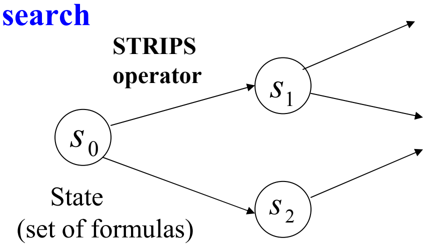
- **Forward and backward state-space planning approaches:**
  - Work with strictly linear sequences of actions
- **Disadvantages:**
  - no **problem decompositions**
    - the goal consists of a set of independent or nearly independent sub-goals
  - Plans cannot be **built from the middle**
  - No **least commitment** in terms of the action ordering

State space vs. plan space search

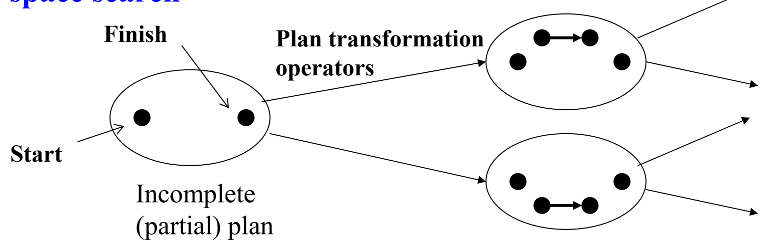
- **Plan:** Defines a sequence of operators to be performed
- **Partial plan:**
  - plan that is not complete
    - Some plan steps are missing
  - some orderings of operators are not finalized
    - Only relative order is given
- **Benefits of plan space search:**
  - Goal decomposition
  - We do not have to commit to a specific action sequence

State-space vs. plan-space search

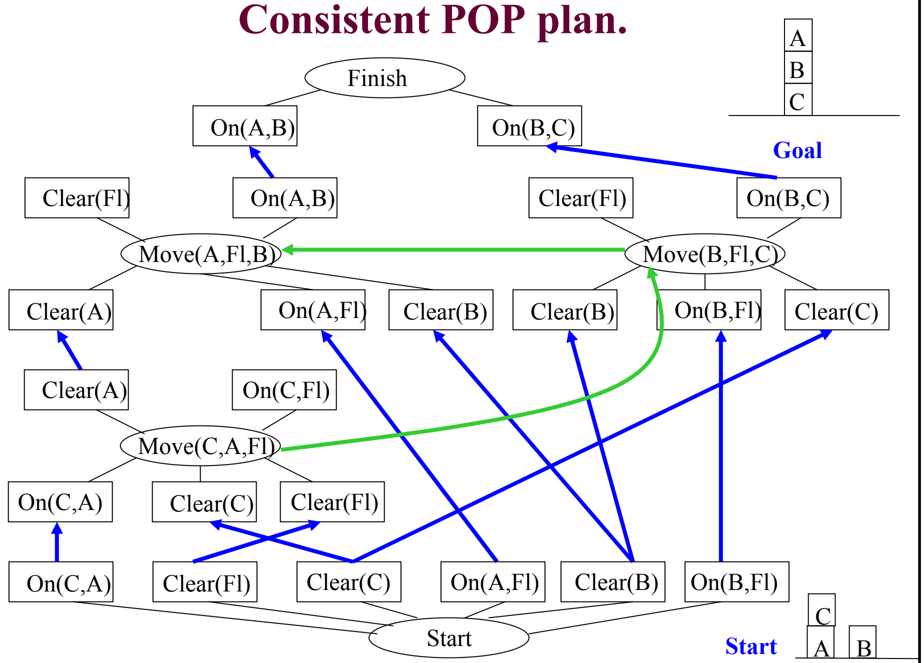
State-space search



Plan-space search

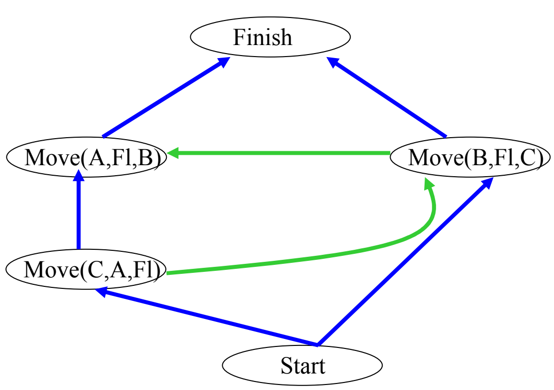


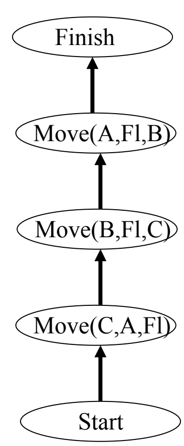
Consistent POP plan.



Partial order planning. Result plan.

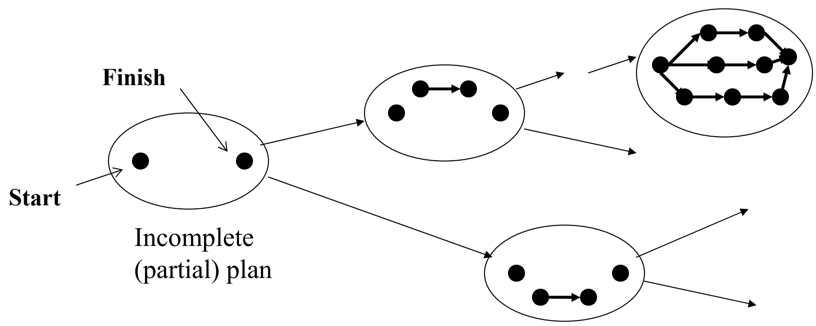
Plan: a topological sort of a graph





Partial order planning.

- **Remember** we search the space of partial plans



- POP: **is sound and complete**

Hierarchical planners

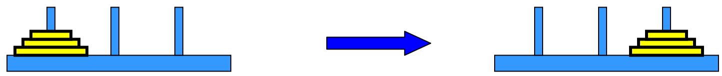
Extension of STRIPS planners.

- Example planner: ABSTRIPS.

Idea:

- Assign a **criticality level** to each conjunct in preconditions list of the operator
- Planning process refines the plan gradually based on criticality threshold, starting from the highest criticality value:
  - Develop the plan ignoring preconditions of criticality less than the criticality threshold value (assume that preconditions for lower criticality levels are true)
  - Lower the threshold value by one and repeat previous step

Towers of Hanoi



Start position

Goal position

Hierarchical planning

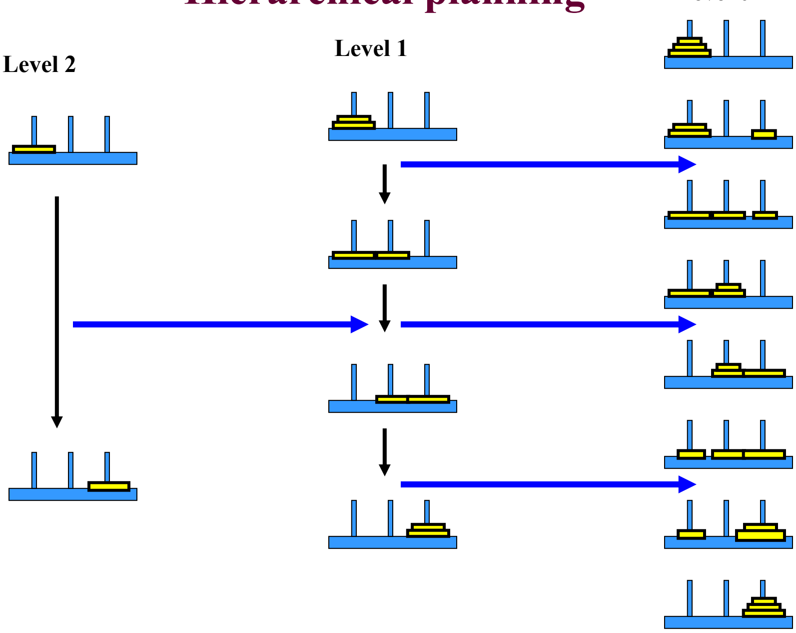
Assume:

the largest disk – criticality level 2

the medium disk – criticality level 1

the smallest disk – criticality level 0

Hierarchical planning



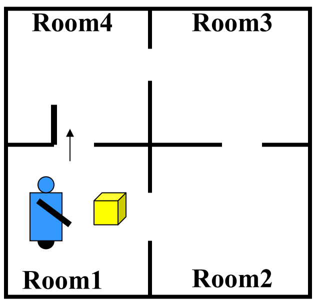
Planning with incomplete information

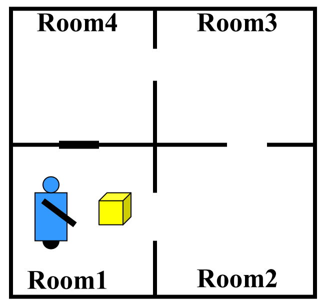
Some conditions relevant for planning can be:

– **true, false or unknown**

Example:

- Robot and the block is in Room 1
- **Goal:** get the block to Room 4
- **Problem:** The door between Room1 and 4 can be closed

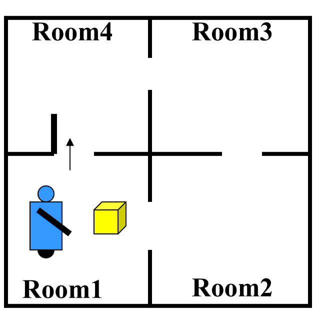


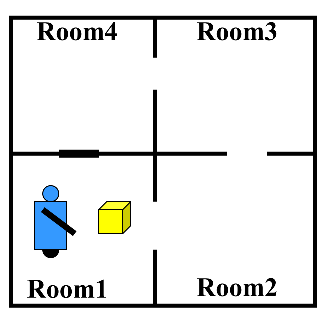


Planning with incomplete information

Initially we do not know whether the door is opened or closed:

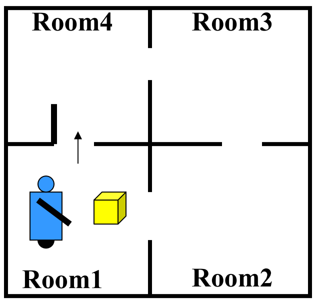
- **Different plans:**
  – **If not closed**: pick the block, go to room 4, drop the block
  – **If closed**: pick the block, go to room2, then room3 then room4 and drop the block

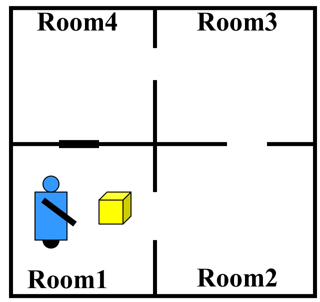




Conditional planners

- Are capable to create conditional plans that cover all possible situations (contingencies) – also called **contingency planners**
- Plan choices are applied when the missing information becomes available
- Missing information can be sought actively through actions
  - **Sensing actions**





Sensing actions

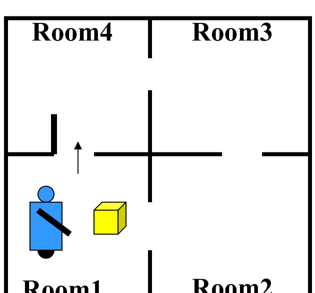
Example:

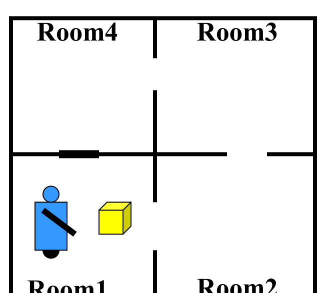
CheckDoor(d): checks the door d

Preconditions: Door(d,x,y) – one way door between x and y

& At(Robot,x)

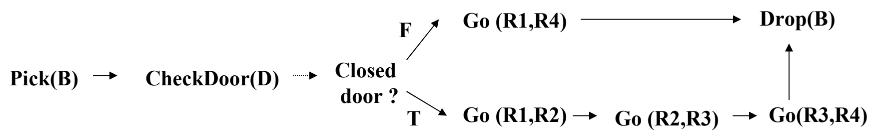
Effect: (Closed(d) v ¬Closed(d)) - one will become true

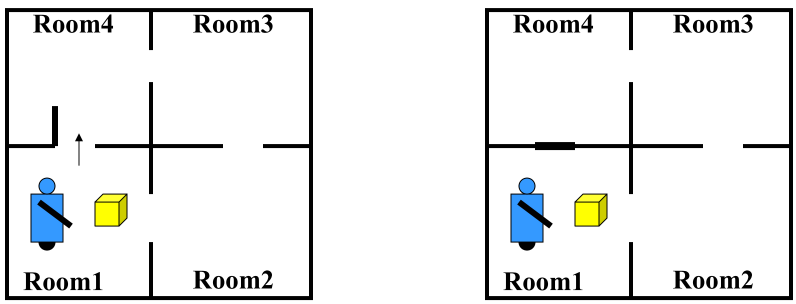




Conditional plans

Sensing actions and conditions incorporated within the plan:





Representing and reasoning with uncertainty

KB systems. Medical example.

We want to build a KB system for the **diagnosis of pneumonia**.

Problem description:

- **Disease:** pneumonia
- **Patient symptoms (findings, lab tests)**:
  - Fever, Cough, Paleness, WBC (white blood cells) count, Chest pain, etc.

Representation of a patient case:

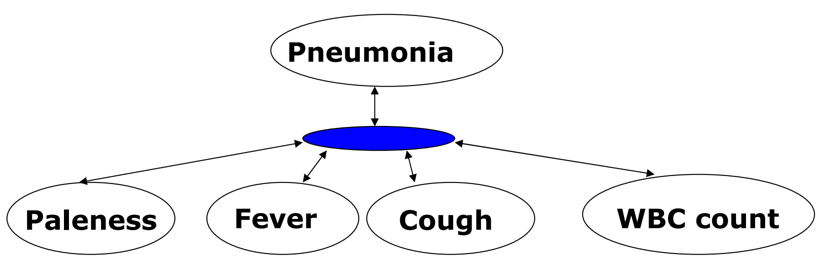
- Statements that hold (are true) for the patient.

  E.g: Fever =*True*
  Cough =*False*
  WBCcount=*High*

Diagnostic task: we want to decide whether the patient suffers from the pneumonia or not given the symptoms

Uncertainty

To make diagnostic inference possible we need to represent knowledge (axioms) that relate symptoms and diagnosis



Problem: disease/symptoms relations are not deterministic

- **They are uncertain (or stochastic) and** vary from patient to patient

Uncertainty

Two types of uncertainty:

- **Disease ⟶ Symptoms uncertainty**
  - A patient suffering from pneumonia may not have fever all the times, may or may not have a cough, white blood cell test can be in a normal range.

- **Symptoms ⟶ Disease uncertainty**
  - High fever is typical for many diseases (e.g. bacterial diseases) and does not point specifically to pneumonia
  - Fever, cough, paleness, high WBC count combined do not always point to pneumonia

Uncertainty

Why are relations uncertain?

- **Observability**
  - It is impossible to observe all relevant components of the world
  - Observable components behave stochastically even if the underlying world is deterministic
- **Efficiency, capacity limits**
  - It is often impossible to enumerate and model all components of the world and their relations
  - abstractions can become stochastic

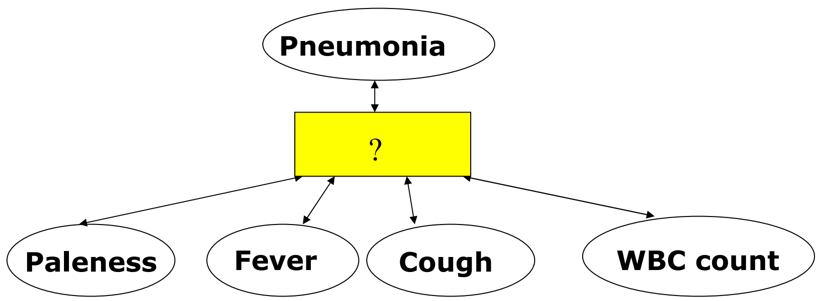
Humans can reason with uncertainty !!!

- Can computer systems do the same?

Modeling the uncertainty.

Key challenges:

- How to represent the relations in the presence of uncertainty?
- How to manipulate such knowledge to make inferences?
  - **Humans can reason with uncertainty.**



Methods for representing uncertainty

Extensions of the propositional and first-order logic

– Use, uncertain, imprecise statements (relations)

Example: Propositional logic with certainty factors

Very popular in 70-80s in knowledge-based systems (MYCIN)

- **Facts (propositional statements)** are assigned a **certainty value** reflecting the belief in that the statement is satisfied:

$$CF(Pneumonia = True) = 0.7$$

- **Knowledge:** typically in terms of **modular rules**

If	1. The patient has cough, and 2. The patient has a high WBC count, and 3. The patient has fever
Then	**with certainty 0.7** the patient has pneumonia

Certainty factors

Problem 1:

- Chaining of multiple inference rules (propagation of uncertainty)

Solution:

- **Rules** incorporate tests on the **certainty values**

$$(A \ \text{ in } [0.5,1]) \wedge (B \ \text{ in } [0.7,1]) \rightarrow C \ \text{ with } \ \mathrm{CF} = 0.8$$

Problem 2:

- Combinations of rules **with the same conclusion**

$$(A \ \text{ in } [0.5,1]) \wedge (B \ \text{ in } [0.7,1]) \rightarrow C \ \text{ with } \ \mathrm{CF} = 0.8$$
$$(E \ \text{ in } [0.8,1]) \wedge (D \ \text{ in } [0.9,1]) \rightarrow C \ \text{ with } \ \mathrm{CF} = 0.9$$

- What is the resulting *CF(C)* ?

Certainty factors

- **Combination of multiple rules**

$$(A \ \text{ in } [0.5,1]) \wedge (B \ \text{ in } [0.7,1]) \rightarrow C \ \text{ with } \ \mathrm{CF} = 0.8$$
$$(E \ \text{ in } [0.8,1]) \wedge (D \ \text{ in } [0.9,1]) \rightarrow C \ \text{ with } \ \mathrm{CF} = 0.9$$

- **Three possible solutions**

$$\left.\begin{aligned} CF(C) &= \max[0.9;0.8] = 0.9 \\ CF(C) &= 0.9 * 0.8 = 0.72 \\ CF(C) &= 0.9 + 0.8 - 0.9 * 0.8 = 0.98 \end{aligned}\right\} \ ?$$

Problems:

- Which solution to choose?
- All three methods break down after a sequence of inference rules

Methods for representing uncertainty

Probability theory

- A well defined theory for modeling and reasoning in the presence of uncertainty
- A natural choice to replace certainty factors

Facts (propositional statements)

- Are represented via **random variables** with two or more values

  Example: *Pneumonia* is a random variable

  values: *True* and *False*
- Each value can be achieved **with some probability:**

$$P(Pneumonia = True) = 0.001$$

$$P(WBCcount = high) = 0.005$$