CS 2710 Foundations of AI
Lecture 14

Planning

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

Administration

- **Midterm exams**
  - **Mean: 77.75**
  - **Median: 78.5**
  - FOL, translation of English to FOL and proof of theorems by resolution refutation (Problem 6) the main problem
- **The results of the tic-tac-toe competition**

Tic-tac-toe player competition

- **1 Swapna Somasundaran**
- **2 Amruta Parundare**
- **3 Chang Liu**

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:

- **Classic State space search**
- **Situation calculus based on FOL**
  - Search by proving the theorem
  - Inference rules or Resolution refutation approaches
- **STRIPS – a restricted FOL language**
  - More efficient (goal progression and goal regression)
  - Partial order planners (plan space search

Planning problems

Properties of many (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 referring only to a small portion of state
- Plans consists of a large number of actions

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

Solutions

- **Complex state description and local action effects:**
  - avoid the enumeration and inference of every state component, focus on changes only
- **Many possible actions:**
  - Apply actions that make progress towards the goal
  - Understand what the effect of actions is and reason with the consequences
- **Sequences of actions in the plan can be too long:**
  - Many goals consists of independent or nearly independent sub-goals
  - Allow goal decomposition & divide and conquer strategies

STRIPS planner

- Defines a more restricted FOL-based 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;

STRIPS planner

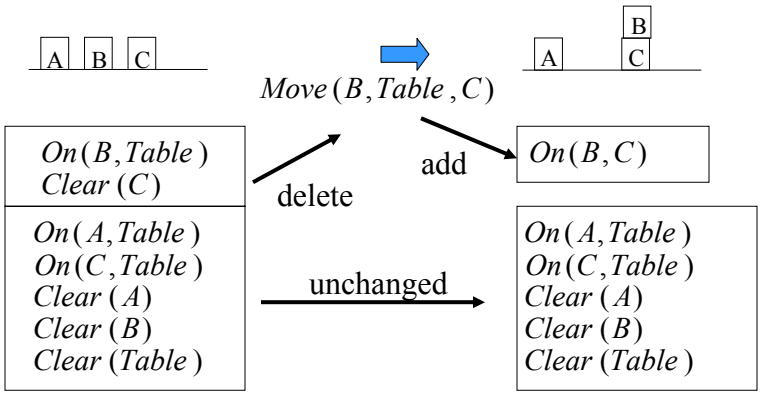
- **States:**
  - conjunction of literals, e.g. *On(A,B), On(B,Table), Clear(A)*
  - represent facts that are true at a specific point in time
- **Actions (operators):**
  - **Action:** *Move (x,y,z)*
  - **Preconditions:** conjunctions of literals with variables

    On(x,y), Clear(x), Clear(z)
  - **Effects.** Two lists:
    - **Add list:** *On(x,z), Clear(y)*
    - **Delete list:** *On(x,y), Clear(z)*
    - Everything else remains untouched (is preserved)

STRIPS planning

Operator: *Move (x,y,z)*

- **Preconditions:** *On(x,y), Clear(x), Clear(z)*
- **Add list:** *On(x,z), Clear(y)*
- **Delete list:** *On(x,y), Clear(z)*



STRIPS planning

Initial state:

- Conjunction of literals that are true

Goals in STRIPS:

- A goal is a partially specified state
- Is defined by a conjunction of ground literals
  - No variables allowed in the description of the goal

Example:

On(A,B) $\wedge$ *On(B,C)*

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

Divide and conquer

- **Divide and conquer strategy**:
  - divide the problem to a set of smaller sub-problems,
  - solve each sub-problem independently
  - combine the results to form the solution

In planning we would like to satisfy a set of goals

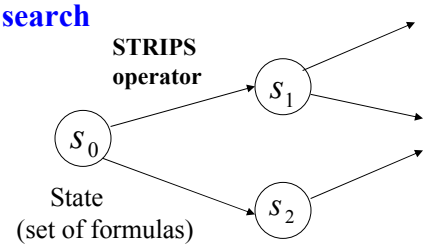
- **Divide and conquer in planning:**
  - Divide the planning goals along individual goals
  - Solve (find a plan for) each of them independently
  - Combine the plan solutions in the resulting plan

State space vs. plan space search

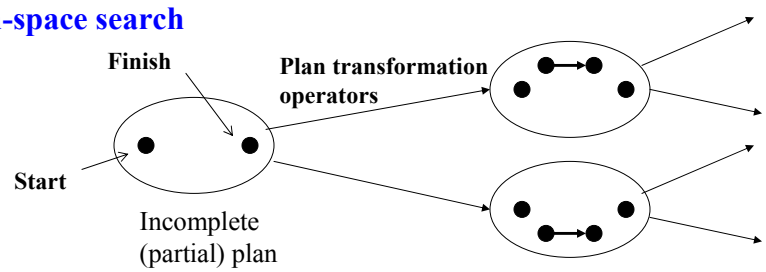
- An alternative to planning algorithms that search states (configurations of world)
- **Plan:** Defines a sequence of operators to be performed
- **Partial plan:**
  - plan that is not complete
    - Some plan steps are missing
  - some orderings of operators are not finalized
    - Only relative order is given
- **Benefits of working with partial plans:**
  - We do not have to build the sequence from the initial state or the goal
  - We do not have to commit to a specific action sequence
  - We can work on sub-goals individually (divide and conquer)

State-space vs. plan-space search

State-space search

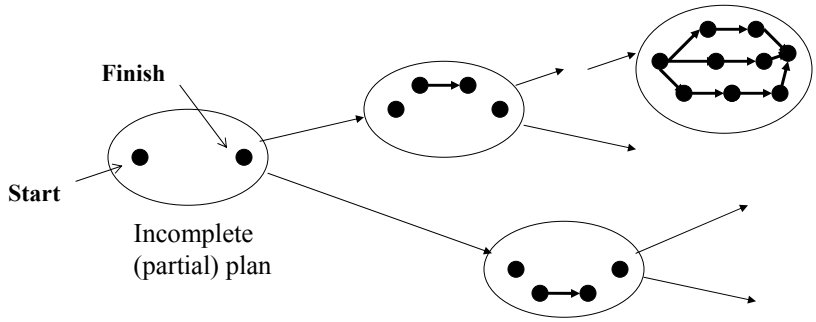


Plan-space search



Partial order planning.

- Search the space of partial plans



- POP: **is sound and complete** for STRIPS

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

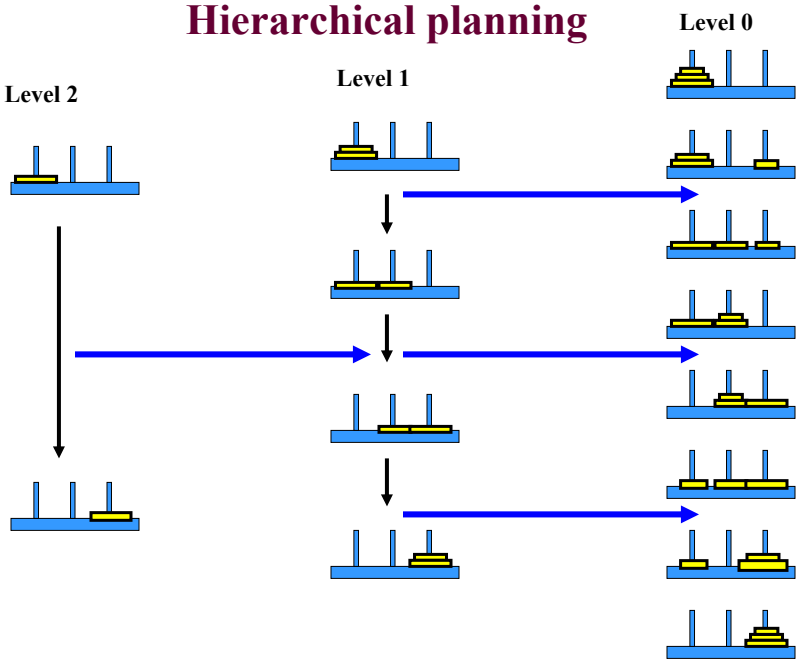
Assume:

the largest disk – criticality level 2

the medium disk – criticality level 1

the smallest disk – criticality level 0

Hierarchical planning



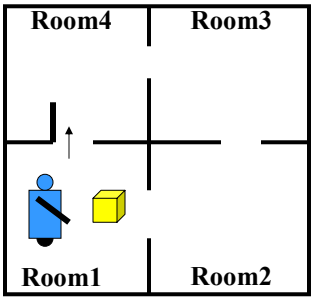
Planning with incomplete information

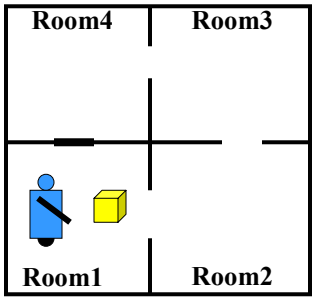
Some conditions relevant for planning can be:

– **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

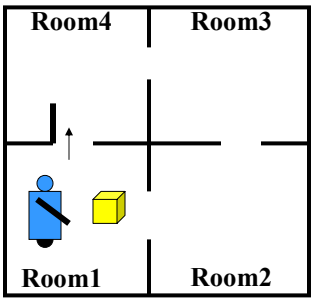


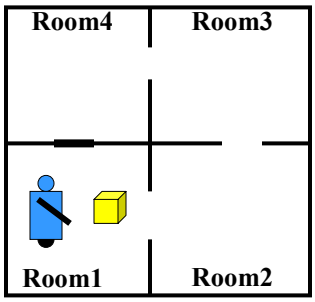


Planning with incomplete information

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

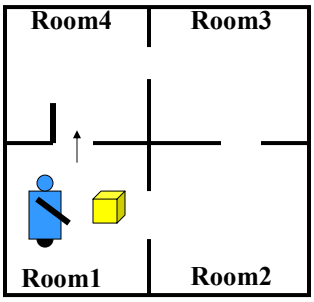
- **Different plans:**
  - **If not closed**: pick the block, go to room 4, drop the block
  - **If closed**: pick the block, go to room2, then room3 then room4 and drop the block

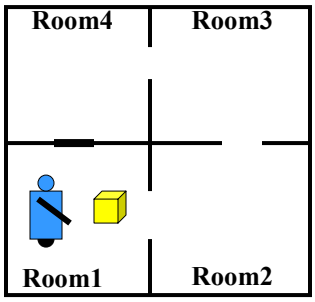




Conditional planners

- 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

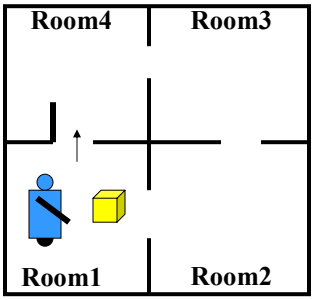
Example:

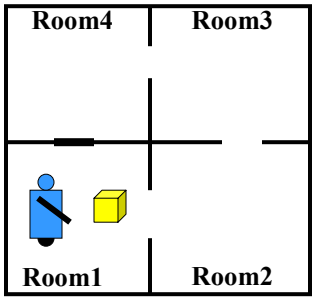
CheckDoor(d): checks the door d

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: