Planning

Planning problem:
• find a sequence of actions that achieves some goal
• an instance of a search problem
• the state description is typically very complex and relies on a logic-based representation

Methods for modeling and solving a planning problem:
• State space search
• Situation calculus based on FOL
• STRIPS – state-space search algorithm
• Partial-order planning algorithms
State-space search

- **Forward and backward state-space planning approaches:**
  - Work with strictly linear sequences of actions

- **Disadvantages:**
  - They cannot take advantage of the problem decompositions in which the goal we want to reach consists of a set of independent or nearly independent sub-goals
  - Action sequences cannot be built from the middle
  - No mechanism to represent least commitment in terms of the action ordering

---

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

- Is it always safe to use divide and conquer?
  - No. There can be interacting goals.
Sussman’s anomaly.

- An example from the blocks world in which the divide and conquer fails due to interacting goals

\[
\begin{array}{c}
C \\
A \\
B
\end{array} \quad \rightarrow \quad \begin{array}{c}
A \\
B \\
C
\end{array}
\]

Initial state \quad Goal

\[
On(A, B) \\
On(B, C)
\]

Sussman’s anomaly

1. Assume we want to satisfy \( On(A, B) \) first

\[
\begin{array}{c}
C \\
A \\
B
\end{array} \quad \rightarrow \quad \begin{array}{c}
A \\
B \\
C
\end{array}
\]

Initial state

But now we cannot satisfy \( On(B, C) \) without undoing \( On(A, B) \)
Sussman’s anomaly

1. Assume we want to satisfy $\text{On}(A, B)$ first.

\[
\begin{array}{c}
\text{C} \\
\text{A} \\
\text{B}
\end{array} \quad \rightarrow \quad \begin{array}{c}
\text{A} \\
\text{B} \\
\text{C}
\end{array}
\]

Initial state

But now we cannot satisfy $\text{On}(B, C)$ without undoing $\text{On}(A, B)$

2. Assume we want to satisfy $\text{On}(B, C)$ first.

\[
\begin{array}{c}
\text{C} \\
\text{A} \\
\text{B}
\end{array} \quad \rightarrow \quad \begin{array}{c}
\text{B} \\
\text{C} \\
\text{A}
\end{array}
\]

Initial state

But now we cannot satisfy $\text{On}(A, B)$ without undoing $\text{On}(B, C)$

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
- STRIPS operator
- \( S_0 \)
- State (set of formulas)
- \( S_1 \)
- \( S_2 \)
- \( S_3 \)

Plan-space search
- Finish
- Start
- Incomplete (partial) plan
- Plan transformation operators
- Move(A, x, B)
- Move(C, A, D)
- Move(A, H, B)
- Move(C, y, D)
- Move(A, H, B)
- Move(C, A, D)

Plan transformation operators

Examples of:
- Add an operator to a plan so that it satisfies some open condition

- Add link (+ instantiate)

- Order (reorder) operators
Partial-order planners (POP)

- also called **Non-linear planners**
- Use STRIPS operators

Graphical representation of an operator `Move(x,y,z)`

![Graphical representation of an operator](image)

**Delete list is not shown !!!**

Illustration of a POP on the Sussman’s anomaly case

---

Partial order planning. Start and finish.

![Partial order planning diagram](image)
Partial order planning. Start and finish.

**Open conditions**: conditions yet to be satisfied

Partial order planning. Add operator.

We want to satisfy **an open condition**

Always select an operator that helps to satisfy one of the open conditions
Partial order planning. Add link.

Start

On(C,A) Clear(Fl) On(A,Fl) Clear(B) On(B,Fl) Clear(C)

Goal

Finish

On(A,B) On(B,C)

On(A,B) Clear(y)

Move(A,y,B)

Clear(A) On(A,y) Clear(B)

On(C,A) Clear(Fl) On(A,Fl) Clear(B) On(B,Fl) Clear(C)

Add link

Satisfies an open condition
Partial order planning. Add link.

Finish

On(A,B) On(B,C)

Clear(Fl) On(A,B)

Move(A,Fl,B)

Clear(A) On(A,Fl) Clear(B)

On(C,A) Clear(Fl) Clear(C) On(A,Fl) Clear(B) On(B,Fl)

Start

Satisfies an open condition

instantiates y/Fl

Goal

Partial order planning. Add operator.

Finish

On(A,B) On(B,C)

Clear(Fl) On(A,B) Clear(y) On(B,Fl)

Move(A,Fl,B)

Clear(A) On(A,Fl) Clear(B) Clear(B) On(B,y) Clear(C)

On(C,A) Clear(Fl) Clear(C) On(A,Fl) Clear(B) On(B,Fl)

Start
Partial order planning. Add links.

Start

On(A,B) Clear(Fl)

On(A,Fl) Clear(A) Clear(B)

On(B,Fl) Clear(B) Clear(C)

On(B,C) Clear(Fl)

Move(A,Fl,B) Move(B,Fl,C)

Clear(A) Clear(C)

Clear(Fl)

On(A,B) On(B,C)

Finish

Goal

Delete Clear(B)

A is stacked on B

Partial order planning. Interactions.

Start

On(A,B) Clear(Fl)

On(A,Fl) Clear(A) Clear(B)

On(B,Fl) Clear(B) Clear(C)

On(B,C) Clear(Fl)

Move(A,Fl,B) Move(B,Fl,C)

Clear(A) Clear(C)

Clear(Fl)

On(A,B) On(B,C)

Finish

Goal

Delete Clear(B)

A is stacked on B
Partial order planning. Order operators.

Start  

On(A,B)  Clear(Fl)  On(C,A)  Clear(B)

Move(A,Fl,B)  On(A,F1)  Clear(B)

Clear(A)  On(B,F1)  Clear(C)

On(A,B)  Clear(Fl)  On(B,C)  Clear(A)

Move(B,F1,C)  On(B,F1)  Clear(F1)

Clear(A)  On(B,Fl)  Clear(B)

A  B  C

Goal

CS 1571 Intro to AI  M. Hauskrecht

Partial order planning. Add operator

Start  

On(A,B)  Clear(Fl)  On(A,F1)

Move(A,Fl,B)  On(u,A)  Clear(u)  Clear(v)

On(C,A)  Clear(F1)  Clear(C)

On(A,F1)  Clear(B)  On(A,B)  Clear(Fl)

Move(B,F1,C)  On(B,F1)  Clear(C)

Clear(u)  On(B,Fl)  Clear(B)

A  B  C

Goal

CS 1571 Intro to AI  M. Hauskrecht
Partial order planning. Add links.

Start

On(C,A) Clear(Fl)
Move(C,A,Fl)
On(C,A) Clear(Fl) Clear(C)
On(C,Fl) Clear(A)

On(A,B) Clear(Fl)
Move(A,Fl,B)
On(A,B) Clear(A) Clear(B)
On(A,Fl) Clear(A) Clear(B)

Clear(Fl)
Move(A,Fl,B)
On(A,B) Clear(Fl)
On(B,C)
Clear(Fl)

A
B
C

Goal

Finish

Partial order planning. Threats.

Start

On(A,B) Clear(Fl)
Move(A,Fl,B)
On(A,B) Clear(A) Clear(B)
On(A,Fl) Clear(A) Clear(B)

Clear(Fl)
Move(A,Fl,B)
On(A,B) Clear(Fl)
On(B,C)
Clear(Fl)

A
B
C

Goal

Finish

B moved on top of C

Clear(C)

Start

On(C,A) Clear(Fl)
Move(C,A,Fl)
On(C,A) Clear(Fl) Clear(C)
On(C,Fl) Clear(A)

On(A,B) Clear(Fl)
Move(A,Fl,B)
On(A,B) Clear(A) Clear(B)
On(A,Fl) Clear(A) Clear(B)

Clear(Fl)
Move(A,Fl,B)
On(A,B) Clear(Fl)
On(B,C)
Clear(Fl)

A
B
C

Goal

Finish

Start

On(A,B) Clear(Fl)
Move(A,Fl,B)
On(A,B) Clear(A) Clear(B)
On(A,Fl) Clear(A) Clear(B)

Clear(Fl)
Move(A,Fl,B)
On(A,B) Clear(Fl)
On(B,C)
Clear(Fl)

A
B
C

Goal

Finish

Start

On(A,B) Clear(Fl)
Move(A,Fl,B)
On(A,B) Clear(A) Clear(B)
On(A,Fl) Clear(A) Clear(B)

Clear(Fl)
Move(A,Fl,B)
On(A,B) Clear(Fl)
On(B,C)
Clear(Fl)

A
B
C

Goal

Finish

Start
Partial order planning. Order operators.

Move(A,F1,B) comes before Move(A,F1,B)

POP planning. Directions.
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

Hierarchical planning

Assume:
the largest disk – criticality level 2
the medium disk – criticality level 1
the smallest disk – criticality level 0
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 Room 1 and 4 can be closed

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 room 2, then room 3 then room 4 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

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:

Pick(B) → CheckDoor(D) → Closed door?

- F: Go (R1,R4) → Drop(B)
- T: Go (R1,R2) → Go (R2,R3) → Go(R3,R4)

Diagram:

```
Room1  Room2  Room3  Room4
```

CS 1571 Intro to AI
M. Hauskrecht