Representation of actions, situations, events

Propositional and first order logic assume a static world
• Once something is true it cannot become false

But, the world is dynamic:
• What is true now may not be true tomorrow
• Changes in the world may be triggered by our activities

Problems:
• How to represent the change in the FOL?
• How to represent actions we can use to change the world?
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

Situation calculus

Provides a framework for representing change, actions and for reasoning about them

• Situation calculus
  – based on the first-order logic
• How does it represent time?
• Uses a situation variable that models new states of the world
• Example: On(x,y,s)
• How does it represent the change due to actions?
• effect and frame axioms
• What inference method it uses?
• Inference rules, Resolution refutation
Situation calculus. Blocks world example.

Initial state

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
</table>

Goal

| A
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
</tbody>
</table>

Find a state (situation) s, such that

- On(A,B, s)
- On(B,C, s)
- On(C,Table, s)

Blocks world example. Effect axioms.

Effect axioms:

Moving x from y to z.  \( MOVE \ (x, y, z) \)

Effect of move changes on \( On \) relations

\[ On(x, y, s) \land Clear(x, s) \land Clear(z, s) \rightarrow On(x, z, DO(MOVE(x, y, z), s)) \]

\[ On(x, y, s) \land Clear(x, s) \land Clear(z, s) \rightarrow \neg On(x, y, DO(MOVE(x, y, z), s)) \]

Effect of move changes on \( Clear \) relations

\[ On(x, y, s) \land Clear(x, s) \land Clear(z, s) \rightarrow Clear(y, DO(MOVE(x, y, z), s)) \]

\[ On(x, y, s) \land Clear(x, s) \land Clear(z, s) \land (z \neq Table) \rightarrow \neg Clear(z, DO(MOVE(x, y, z), s)) \]
Blocks world example. Frame axioms.

- **Frame axioms.**
  - Represent things that remain unchanged after an action.

**On relations:**

\[
On(u, v, s) \land (u \neq x) \land (v \neq y) \rightarrow On(u, v, DO(MOVE(x, y, z), s))
\]

**Clear relations:**

\[
Clear(u, s) \land (u \neq z) \rightarrow Clear(u, DO(MOVE(x, y, z), s))
\]

Planning in situation calculus

**Planning problem:**

- find a sequence of actions that lead to a goal

Planning in situation calculus is converted to the theorem proving problem

**Goal state:**

\[
\exists s \ On(A, B, s) \land On(B, C, s) \land On(C, Table, s)
\]

- Possible inference approaches:
  - **Inference rule approach**
  - **Conversion to SAT**

- **Plan** (solution) is a byproduct of theorem proving.

- **Example:** blocks world
Planning in the blocks world.

Initial state (s0)  

\[ s_0 = \]

\begin{align*}
On(A, Table, s_0) & \quad \text{Clear} (A, s_0) \quad \text{Clear} (Table, s_0) \\
On(B, Table, s_0) & \quad \text{Clear} (B, s_0) \\
On(C, Table, s_0) & \quad \text{Clear} (C, s_0)
\end{align*}

Action: MOVE (B, Table, C)

\[ s_1 = DO(MOVE (B, Table, C), s_0) \]

\begin{align*}
On(A, Table, s_1) & \quad \text{Clear} (A, s_1) \quad \text{Clear} (Table, s_1) \\
On(B, C, s_1) & \quad \text{Clear} (B, s_1) \\
\neg On(B, Table, s_1) & \quad \neg \text{Clear} (C, s_1) \\
On(C, Table, s_1) & \quad \neg \text{Clear} (C, s_1)
\end{align*}

Planning in the blocks world.

Initial state (s0)  

\[ s_1 = DO(MOVE (B, Table, C), s_0) \]

\begin{align*}
On(A, Table, s_1) & \quad \text{Clear} (A, s_1) \quad \text{Clear} (Table, s_1) \\
On(B, C, s_1) & \quad \text{Clear} (B, s_1) \\
\neg On(B, Table, s_1) & \quad \neg \text{Clear} (C, s_1) \\
On(C, Table, s_1) & \quad \neg \text{Clear} (C, s_1)
\end{align*}

Action: MOVE (A, Table, B)

\[ s_2 = DO(MOVE (A, Table, B), s_1) \]

\[ = DO(MOVE (A, Table, B), DO(MOVE (B, Table, C), s_0)) \]

\begin{align*}
On(A, B, s_2) & \quad \neg On(A, Table, s_2) \quad \neg \text{Clear} (B, s_2) \\
On(B, C, s_2) & \quad \neg On(B, Table, s_2) \quad \neg \text{Clear} (C, s_2) \\
On(C, Table, s_2) & \quad \text{Clear} (A, s_2) \quad \text{Clear} (Table, s_2)
\end{align*}
Planning in situation calculus.

**Planning problem:**
- Find a sequence of actions that lead to a goal
- Is a special type of a search problem
- Planning in situation calculus is converted to theorem proving.

**Problems:**
- Large search space
- Large number of axioms to be defined for one action
- All ‘unchanged’ properties/relations must be explicitly moved to the next situation
- Proof may not lead to the best (shortest) plan.

### Problems and 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 **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 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:** \(\text{On}(x,y), \text{Clear}(x), \text{Clear}(z)\)
- **Add list:** \(\text{On}(x,z), \text{Clear}(y)\)
- **Delete list:** \(\text{On}(x,y), \text{Clear}(z)\)

---

**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:

\[\text{On}(A,B) \land \text{On}(B,C)\]
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

Forward search (goal progression)

• Idea: Given a state $s$
  – Unify the preconditions of some operator $a$ with $s$
  – Add and delete sentences from the add and delete list of an operator $a$ from $s$ to get a new state
Forward search (goal progression)

- Use operators to generate new states to search
- Check new states whether they satisfy the goal

Search tree:

Initial state

Heuristics?
Backward search (goal regression)

Idea: Given a goal G

- Unify the add list of some operator \( a \) with a subset of \( G \)
- If the delete list of \( a \) does not remove elements of \( G \), then the goal regresses to a new goal \( G' \) that is obtained from \( G \) by:
  - deleting add list of \( a \)
  - adding preconditions of \( a \)

\[
\begin{array}{c|c|c}
\text{Goal (G)} & \text{New goal (G')} \\
\hline
\text{On(A,Table)} & \text{On}(A,Table) \\
\text{Clear (B)} & \text{Clear (B)} \\
\text{Clear (A)} & \text{Clear (A)} \\
\text{On}(B,C) & \text{On}(B,C) \\
\text{On}(C,Table) & \text{On}(C,Table) \\
\end{array}
\]

Backward search (goal regression)

- Use operators to generate new goals
- Check whether the initial state satisfies the goal

Search tree:
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

![Diagram](image)

Initial state

Goal

$On(A, B)$

$On(B, C)$

Sussman’s anomaly

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

![Diagram](image)

Initial state

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

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

But now we cannot satisfy $On(B, C)$ without undoing $On(A, B)$

2. Assume we want to satisfy $On(B, C)$ first.

But now we cannot satisfy $On(A, B)$ without undoing $On(B, C)$