Planning

Representation of actions, situations, events

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

Problems:
• Logic (FOL) as we had it referred to a static world. How to represent the change in the FOL?
• How to represent actions we can use to change the world?

Planning problem:
• find a sequence of actions that achieves some goal in this complex world
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:
• Situation calculus (extends FOL)
• State space search (STRIPS - restricted FOL)
• Plan-based search (for STRIPS)
• GRAPHPLAN – for propositional languages

Situation calculus

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

• Situation calculus
  – based on first-order logic,
  – a situation variable models new states of the world
  – action objects model activities
  – uses inference methods developed for FOL to do the reasoning
Situation calculus

- Logic for reasoning about changes in the state of the world
- **The world is described by:**
  - Sequences of situations of the current state
  - Changes from one situation to another are caused by actions
- **The situation calculus allows us to:**
  - Describe the initial state and a goal state
  - Build the KB that describes the effect of actions (operators)
  - Prove that the KB and the initial state lead to a goal state
  - extracts a plan as side-effect of the proof

---

Situation calculus

**The language is based on the First-order logic plus:**
- **Special variables:** $s, a$ – objects of type situation and action
- **Action functions:** return actions.
  - E.g. $Move(A, TABLE, B)$ represents a move action
  - $Move(x,y,z)$ represents an action schema
- **Two special function symbols of type situation**
  - $s_0$ – initial situation
  - $DO(a,s)$ – denotes the situation obtained after performing an action $a$ in situation $s$
- **Situation-dependent functions and relations** (also called fluents)
  - **Relation:** $On(x,y,s)$ – object $x$ is on object $y$ in situation $s$;
  - **Function:** $Above(x,s)$ – object that is above $x$ in situation $s$. 

---
Situation calculus. Blocks world example.

Initial state

- On(A, Table, s₀)
- On(B, Table, s₀)
- On(C, Table, s₀)
- Clear(A, s₀)
- Clear(B, s₀)
- Clear(C, s₀)
- Clear(Table, s₀)

Goal

Find a state (situation) s, such that

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

Note: It is not necessary that the goal describes all relations

Clear(A, s)
Blocks world example.

Assume a simpler goal $On(A, B, s)$

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

Initial state

$On(A, Table, s_0)$

$On(B, Table, s_0)$

$On(C, Table, s_0)$

Clear($A, s_0$)

Clear($B, s_0$)

Clear($C, s_0$)

Clear($Table, s_0$)

Goal $On(A, B, s)$

3 possible goal configurations

Knowledge base: Axioms.

Knowledge base needed to support the reasoning:

- Must represent changes in the world due to actions.

Two types of axioms:

- **Effect axioms**
  - changes in situations that result from actions

- **Frame axioms**
  - things preserved from the previous situation
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 a blocks world.

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

Initial state

- On(A, Table, s₀)
- On(B, Table, s₀)
- On(C, Table, s₀)
- Clear(A, s₀)
- Clear(B, s₀)
- Clear(C, s₀)
- Clear(Table, s₀)

Goal

- On(A, B, s)
- On(B, C, s)
- On(C, Table, s)
Planning in the blocks world.

Initial state ($s_0$) $s_1$

$s_0 =$
- $On(A, Table, s_0)$ Clear ($A, s_0$) Clear ($Table, s_0$)
- $On(B, Table, s_0)$ Clear ($B, s_0$)
- $On(C, Table, s_0)$ Clear ($C, s_0$)

**Action:** MOVE ($B, Table, C$)

$s_1 = DO(MOVE (B, Table, C), s_0)$

- $On(A, Table, s_1)$ Clear ($A, s_1$) Clear ($Table, s_1$)
- $On(B, C, s_1)$ Clear ($B, s_1$)
- $On(C, Table, s_1)$ Clear ($C, s_1$)

Planning in the blocks world.

Initial state ($s_0$) $s_1$ $s_2$

$s_1 = DO(MOVE (B, Table, C), s_0)$

- $On(A, Table, s_1)$ Clear ($A, s_1$) Clear ($Table, s_1$)
- $On(B, C, s_1)$ Clear ($B, s_1$)
- $On(C, Table, s_1)$ Clear ($C, s_1$)

**Action:** MOVE ($A, Table, B$)

$s_2 = DO(MOVE (A, Table, B), s_1)$

$= DO(MOVE (A, Table, B), DO(MOVE (B, Table, C), s_0))$

- $On(A, B, s_2)$ $On(A, Table, s_2)$ Clear ($B, s_2$)
- $On(B, C, s_2)$ $On(B, Table, s_2)$ Clear ($C, s_2$)
- $On(C, Table, s_2)$ Clear ($A, s_2$) Clear ($Table, s_2$)
Planning in situation calculus.

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

- Problems with situation calculus:
  - Large search space
  - Large number of axioms to be defined for one action
  - Proof may not lead to the best (shortest) plan.

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

Frame problem refers to:
- The need to represent a large number of frame axioms

Solution: combine positive and negative effects in one rule

\[ On(u, v, DO(MOVE(x, y, z), s)) \Leftrightarrow \neg((u = x) \land (v = y)) \land On(u, v, s) \lor \]
\[ \lor (((u = x) \land (v = z)) \land On(x, y, s) \land Clear(x, s) \land Clear(z, s)) \]

Inferential frame problem:
- We still 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

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 framework

- Defines a restricted version of the FOL 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:** \(On(x,y), Clear(z)\)

---

**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:
\[ On(A,B) \land On(B,C) \]
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 (can be repeated)

---

A | B | C
---|---|---
On(\( B, Table \))
Clear(\( C \))
On(\( A, Table \))
On(\( C, Table \))
Clear(\( A \))
Clear(\( B \))
Clear(\( Table \))

Move(\( B, Table, C \))

\[ \rightarrow \]

A | B | C
---|---|---
On(\( B, C \))

delete
add

unchanged

On(\( A, Table \))
On(\( C, Table \))
Clear(\( A \))
Clear(\( B \))
Clear(\( Table \))
**Forward search (goal progression)**

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

**Search tree:**

- Initial state: A B C
- Move (A, Table, B)
- Move (B, Table, C)
- Move (A, Table, C)
- Move (A, Table, B)

**Goal (G)**

- A
- B
- C

**New goal (G’)**

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

**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

- Move (A, Table, B)
- New goal (G’)

**Goal (G)**

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

**Mapped from G**

- Precondition
- Add
Backward search (goal regression)

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

Search tree:

![Search tree diagram]

- Initial state: A B C
- Move (B, Table, C) → Move (A, Table, B) → goal
  - Move (A, B, Table)