STRIPS planning

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

Administration

• Problem set 6 is out
  – due on Tuesday, October 28, 2003
• Midterm:
  – at the end of the lecture
Planning

Planning problem:
• find a sequence of actions that lead to a goal
• is a special type of a search problem

Specifics of a planning problem:
• Very complex states
• Large number of actions
• Every action effects only a “small” subset of relations in the state
• Goal conditions are defined over a “small” set of relations

Ways to deal planning problems:
• Open state, action and goal representations to allow selection, reasoning. Expose the structure.
  – Use FOL or its restricted subset to do the reasoning.
• Drop the need to construct solutions sequentially from the initial state.
  – Apply divide and conquer strategies to sub-goals.

Challenges:
• Build a representation language for modeling action and change
• Design of special search algorithms for a given representation
Planning systems design.

Two planning systems designs:

- **Situation calculus**
  - based on first-order logic,
  - a situation variable models new states of the world
  - use inference methods developed for FOL to do the reasoning

- **STRIPS – like planners**
  - STRIPS – Stanford research institute problem solver
  - Restricted language as compared to the situation calculus
  - Allows for more efficient planning algorithms

---

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 goal state
  - Build the KB that describes the effect of actions (operators)
  - Prove that the KB allows us to derive (prove) the goal state
    - and thereby allow us to extract a plan

---
Situation calculus

The language is based on First order logic plus:

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

Situation calculus. Blocks world example.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial state</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{On}(A,\text{Table}, s_\emptyset)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{On}(B,\text{Table}, s_\emptyset)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{On}(C,\text{Table}, s_\emptyset)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{Clear}(A, s_\emptyset)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{Clear}(B, s_\emptyset)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{Clear}(C, s_\emptyset)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{Clear}(\text{Table}, s_\emptyset)$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{On}(A,B, s)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{On}(B,C, s)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{On}(C,\text{Table}, s)$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Find a state (situation) $s$, such that
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
- On(A, B, s)
- On(B, C, s)
- On(C, Table, s)

Note: It is not necessary that the goal describes all relations, e.g., Clear(A, s)

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

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)

3 possible goal configurations

Goal On(A, B, s)
Knowledge about the world. Axioms.

Knowledge base we need to built 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:**
  
  \[ \text{On}(u, v, s) \land (u \neq x) \land (v \neq y) \rightarrow \text{On}(u, v, \text{DO}(\text{MOVE}(x, y, z), s)) \]

  **Clear relations:**
  
  \[ \text{Clear}(u, s) \land (u \neq z) \rightarrow \text{Clear}(u, \text{DO}(\text{MOVE}(x, y, z), s)) \]

Planning in situation calculus.

**Planning problem:**

- find a sequence of actions that lead to a goal

**Planning in situation calculus is converted to theorem proving.**

**Goal state:**

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

- 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.

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

Planning in the blocks world.

Initial state (s_0)

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

Action: MOVE (B, Table, C)

\[
\begin{align*}
&s_1 = DO(MOVE(B, Table, C), s_0) \\
& On(A, Table, s_1) & Clear(A, s_1) \\
& On(B, C, s_1) & Clear(B, s_1) \\
& -On(B, Table, s_1) & Clear(C, s_1) \\
& On(C, Table, s_1) & Clear(Table, s_1) \\
\end{align*}
\]
Planning in the blocks world.

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

$s_1 = DO(MOVE(B,Table,C),s_0)$
$On(A,Table,s_1)$
$On(B,C,s_1)$
$\neg On(B,Table,s_1)$
$On(C,Table,s_1)$

$Clear(A,s_1)$
$Clear(B,s_1)$
$\neg Clear(C,s_1)$

$Clear(Table,s_1)$

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

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

$On(A,B,s_2)$
$On(B,C,s_2)$
$On(C,Table,s_2)$

$\neg On(A,Table,s_2)$
$\neg On(B,Table,s_2)$
$\neg On(C,Table,s_2)$

$\neg Clear(B,s_2)$
$\neg Clear(C,s_2)$

$Clear(A,s_2)$
$Clear(B,s_2)$
$Clear(Table,s_2)$

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
- Proof may not lead to the best (shortest) plan.
STRIPS representation.

- More restricted representation language as compared to the situation calculus
- **States:**
  - represent facts that are true at a specific point in time conjunction of literals, e.g. \(\text{On}(A,B), \text{On}(B,\text{Table}), \text{Clear}(A)\)
- **Actions (represented by operators):**

  **Operator:** \(\text{Move} (x,y,z)\)
  - **Preconditions:** \(\text{On}(x,y), \text{Clear}(x), \text{Clear}(z)\)
  - **Effect lists:**
    - **Add list:** \(\text{On}(x,z), \text{Clear}(y)\)
    - **Delete list:** \(\text{On}(x,y), \text{Clear}(z)\)
      (Everything else is unaffected)

- **Goals:** conjunctions of literals, e.g. \(\text{On}(A,B), \text{On}(B,C)\)
STRIPS representation. Benefits.

Benefits:
- States, actions and goals have structure
- Action representation:
  - Leads to more intuitive and compact description of actions
    (no need to write many axioms !!!)
  - Avoids the frame problem
- Restrictions lead to more efficient planning algorithms.

STRIPS planning:
- find a sequence of operators from the initial state to the goal
- Search problem definition in STRIPS looks much like the standard search problem definition

STRIPS planning.

STRIPS planning problem:
- Find a sequence of actions that lead to a goal
- States and goals are defined by a conjunctions of literals

Two basic search methods:
- Forward search (goal progression)
  - From the initial state try to reach the goal
- Backward search (goal regression)
  - Start from the goal and try to project it to the initial state

More complex planning method:
- Partial-order planning (POP)
  - Search the space of partially build plans
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

- **Search tree:**
  - Initial state $A B C$
  - Move operators to generate new states to explore
  - Check new states whether they satisfy the goal
Backward search (goal regression)

**Idea:** Given a goal on a goal list $G$,
- find an operator that satisfies it (it is on its add list)
- Add its preconditions to the goal list $G$

**Example:**

```
\begin{array}{c}
\text{A} & \text{B} & \text{C} \\
\end{array}
\quad \xrightarrow{\text{Move} (x,y,z)} \\
\begin{array}{c}
\text{A} & \text{B} & \text{C} \\
\end{array}
```

New goal ($G'$)

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

Goal ($G$)

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

precondition add

```
\begin{array}{c}
\text{A} & \text{B} & \text{C} \\
\end{array}
```

Backward search (goal regression)

**More detailed description:** 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 $a$
Backward search (goal regression)

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

Search tree: