Alternative Representations

Propositions and State-Variables

Literature

Classical Representations

- **propositional representation**
  - world state is set of propositions
  - action consists of precondition propositions, propositions to be added and removed

- **STRIPS representation**
  - like propositional representation, but first-order literals instead of propositions

- **state-variable representation**
  - state is tuple of state variables \( \{x_1, \ldots, x_n\} \)
  - action is partial function over states

Classical Planning

- **task:** find solution for planning problem

- **planning problem**
  - **initial state**
    - atoms (relations, objects)
  - **planning domain**
    - operators (name, preconditions, effects)
  - **goal**
  - **solution** (plan)
Overview

- World States
  - Domains and Operators
  - Planning Problems
  - Plans and Solutions
  - Expressiveness

Knowledge Engineering

- What types of objects do we need to represent?
  - example: cranes, robots, containers, …
  - note: objects usually only defined in problem
- What relations hold between these objects?
  - example: at(robot, location), empty(crane), …
  - static vs. fluent relations
# Representing World States

<table>
<thead>
<tr>
<th></th>
<th>STRIPS</th>
<th>propositional</th>
<th>state-variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>set of atoms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>atom</td>
<td>first-order atom</td>
<td>proposition</td>
<td>state-variable expression</td>
</tr>
<tr>
<td>relations</td>
<td>yes</td>
<td>no</td>
<td>functions</td>
</tr>
<tr>
<td>objects/types</td>
<td>yes/maybe</td>
<td>no/no</td>
<td>yes/maybe</td>
</tr>
<tr>
<td>static relations</td>
<td>yes</td>
<td>not necessary</td>
<td>no</td>
</tr>
</tbody>
</table>

---

## DWR Example: STRIPS States

\[
\text{state} = \{\text{attached}(p1,\text{loc1}), \\
\quad \text{attached}(p2,\text{loc1}), \\
\quad \text{in}(c1,p1),\text{in}(c3,p1), \\
\quad \text{top}(c3,p1), \text{on}(c3,c1), \\
\quad \text{on}(c1,\text{pallet}), \text{in}(c2,p2), \\
\quad \text{top}(c2,p2), \text{on}(c2,\text{pallet}), \\
\quad \text{belong}(\text{crane1},\text{loc1}), \\
\quad \text{empty}(\text{crane1}), \\
\quad \text{adjacent}(\text{loc1},\text{loc2}), \\
\quad \text{adjacent}(\text{loc2}, \text{loc1}), \\
\quad \text{at}(r1,\text{loc2}), \text{occupied}(\text{loc2}), \\
\quad \text{unloaded}(r1)\} 
\]
**DWR Example: Propositional States**

- $L = \{\text{onpallet, onrobot, holding, at1, at2}\}$
- $S = \{s_0, \ldots, s_5\}$
  - $s_0 = \{\text{onpallet, at2}\}$
  - $s_1 = \{\text{holding, at2}\}$
  - $s_2 = \{\text{onpallet, at1}\}$
  - $s_3 = \{\text{holding, at1}\}$
  - $s_4 = \{\text{onrobot, at1}\}$
  - $s_5 = \{\text{onrobot, at2}\}$

**State Variables**

- some relations are functions
  - example: at(r1,loc1): relates robot r1 to location loc1 in some state
    - truth value changes from state to state
    - will only be true for exactly one location / in each state
- idea: represent such relations using state-variable functions mapping states into objects
  - example: functional representation:
    $rloc: \text{robots} \times S \rightarrow \text{locations}$
DWR Example: State-Variable State Descriptions

- simplified: no cranes, no piles
- state-variable functions:
  - rloc: robots×S → locations
  - rolad: robots×S→containers ∪ {nil}
  - cpos: containers×S → locations ∪ robots
- sample state-variable state descriptions:
  - \{rloc(r1)=loc1, rolad(r1)=nil, cpos(c1)=loc1, 
    cpos(c2)=loc2, cpos(c3)=loc2\}
  - \{rloc(r1)=loc1, rolad(r1)=c1, cpos(c1)=r1, 
    cpos(c2)=loc2, cpos(c3)=loc2\}

Overview

- World States
- Domains and Operators
- Planning Problems
- Plans and Solutions
- Expressiveness
Knowledge Engineering

- What types of actions are there?
  - example: move robots, load containers, ...
- For each action type, and each relation, what must (not) hold for the action to be applicable?
  - preconditions
- For each action type, and each relation, what relations will (no longer) hold due to the action?
  - effects (must be consistent)
- For each action type, what objects are involved in performing the action?
  - any object mentioned in the preconditions and effects
  - preconditions should mention all objects

Representing Operators

<table>
<thead>
<tr>
<th></th>
<th>STRIPS</th>
<th>propositional</th>
<th>state-variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>(n(x_1, \ldots, x_k))</td>
<td>(name)</td>
<td>(n(x_1, \ldots, x_k))</td>
</tr>
<tr>
<td>preconditions</td>
<td>first-order literals</td>
<td>propositions</td>
<td>state-variable expressions</td>
</tr>
<tr>
<td>applicability</td>
<td>(\text{precond}^+(a) \subseteq s \land \text{precond}(a) \cap s = {})</td>
<td>(\text{precond}(a) \subseteq s)</td>
<td>(\text{precond}(a) \subseteq s)</td>
</tr>
<tr>
<td>effects</td>
<td>first-order literals</td>
<td>propositional literals</td>
<td>(x_s \leftarrow v)</td>
</tr>
<tr>
<td>(\gamma(s, a))</td>
<td>( (s - \text{effects}(a)) \cup \text{effects}^+(a))</td>
<td>( (s - \text{effects}(a)) \cup \text{effects}^+(a))</td>
<td>({x_i = c \mid x_i \in X} \text{ where } x_i \neq c \in \text{effects}(a) ) or (x_i = c \in s) otherwise</td>
</tr>
</tbody>
</table>
**DWR Example: STRIPS Operators**

- **move(r,l,m)**
  - precond: adjacent(l,m), at(r,l), ¬occupied(m)
  - effects: at(r,m), occupied(m), ¬occupied(l), ¬at(r,l)

- **load(k,l,c,r)**
  - precond: belong(k,l), holding(k,c), at(r,l), unloaded(r)
  - effects: empty(k), ¬holding(k,c), loaded(r,c), ¬unloaded(r)

- **put(k,l,c,d,p)**
  - precond: belong(k,l), attached(p,l), holding(k,c), top(d,p)
  - effects: ¬holding(k,c), empty(k), in(c,p), top(c,p), on(c,d), ¬top(d,p)

**DWR Example: Propositional Actions**

<table>
<thead>
<tr>
<th>a</th>
<th>precond(a)</th>
<th>effects⁻(a)</th>
<th>effects⁺(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>take</td>
<td>{onpallet}</td>
<td>{onpallet}</td>
<td>{holding}</td>
</tr>
<tr>
<td>put</td>
<td>{holding}</td>
<td>{holding}</td>
<td>{onpallet}</td>
</tr>
<tr>
<td>load</td>
<td>{holding,at1}</td>
<td>{holding}</td>
<td>{onrobot}</td>
</tr>
<tr>
<td>unload</td>
<td>{onrobot,at1}</td>
<td>{onrobot}</td>
<td>{holding}</td>
</tr>
<tr>
<td>move1</td>
<td>{at2}</td>
<td>{at2}</td>
<td>{at1}</td>
</tr>
<tr>
<td>move2</td>
<td>{at1}</td>
<td>{at1}</td>
<td>{at2}</td>
</tr>
</tbody>
</table>
DWR Example: State-Variable Operators

- **move(r,l,m)**
  - precond: rloc(r)=l, adjacent(l,m)
  - effects: rloc(r)←m

- **load(r,c,l)**
  - precond: rloc(r)=l, cpos(c)=l, rload(r)=nil
  - effects: cpos(c)←r, rload(r)←c

- **unload(r,c,l)**
  - precond: rloc(r)=l, rload(r)=c
  - effects: rload(r)←nil, cpos(c)←l

Overview

- World States
- Domains and Operators
  - Planning Problems
- Plans and Solutions
- Expressiveness
Representing Planning Problems

<table>
<thead>
<tr>
<th></th>
<th>STRIPS</th>
<th>propositional</th>
<th>state-variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial state</td>
<td>world state in respective representation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>domain</td>
<td>domain (set of operators) in respective representation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>goal</td>
<td>same as preconditions in respective representation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DWR Example: STRIPS Planning Problem

- $\Sigma$: STRIPS planning domain for DWR domain
- $s_i$: any state
  - example: $s_0 = \{\text{attached(pile,loc1), in(cont,pile), top(cont,pile), on(cont,pallet), belong(crane,loc1), empty(crane), adjacent(loc1,loc2), adjacent(loc2,loc1), at(robot,loc2), occupied(loc2), unloaded(robot)}\}$
- $g$: any subset of $L$
  - example: $g = \{\neg\text{unloaded(robot), at(robot,loc2)}\}$, i.e. $S_g = \{s_0\}$
DWR Example: Propositional Planning Problem

- $\Sigma$: propositional planning domain for DWR domain
- $s_i$: any state
  - example: initial state $= s_0 \in S$
- $g$: any subset of $L$
  - example: $g=\{\text{onrobot,at2}\}$, i.e. $S_g=\{s_5\}$

Overview

- World States
- Domains and Operators
- Planning Problems
  - Plans and Solutions
- Expressiveness
A plan is any sequence of actions \( \pi = \langle a_1, \ldots, a_k \rangle \), where \( k \geq 0 \).

The extended state transition function for plans is defined as follows:

- \( \gamma(s, \pi) = s \) if \( k = 0 \) (\( \pi \) is empty)
- \( \gamma(s, \pi) = \gamma(\gamma(s, a_1), \langle a_2, \ldots, a_k \rangle) \) if \( k > 0 \) and \( a_1 \) applicable in \( s \)
- \( \gamma(s, \pi) = \text{undefined} \) otherwise

Let \( \mathcal{P} = (\Sigma, s_i, g) \) be a planning problem. A plan \( \pi \) is a solution for \( \mathcal{P} \) if \( \gamma(s_i, \pi) \) satisfies \( g \).
Grounding a STRIPS Planning Problem

- Let $P=(O,s_i,g)$ be the statement of a STRIPS planning problem and $C$ the set of all the constant symbols that are mentioned in $s_i$. Let $\text{ground}(O)$ be the set of all possible instantiations of operators in $O$ with constant symbols from $C$ consistently replacing variables in preconditions and effects.
- Then $P'=(\text{ground}(O),s_i,g)$ is a statement of a STRIPS planning problem and $P'$ has the same solutions as $P$.

Translation: Propositional Representation to Ground STRIPS

- Let $P=(A,s_i,g)$ be a statement of a propositional planning problem. In the actions $A$:
  - replace every action $(\text{precond}(a), \text{effects}^-(a), \text{effects}^+(a))$ with an operator $o$ with
    - some unique name($o$),
    - $\text{precond}(o) = \text{precond}(a)$, and
    - $\text{effects}(o) = \text{effects}^+(a) \cup \{\neg p \mid p \in \text{effects}^-(a)\}$. 
Translation: Ground STRIPS to Propositional Representation

- Let \( P=(O,s_i,g) \) be a ground statement of a classical planning problem.
  - In the operators \( O \), in the initial state \( s_i \), and in the goal \( g \) replace every atom \( P(v_1,\ldots,v_n) \) with a propositional atom \( P_v \).
  - In every operator \( o \):
    - for all \( \neg p \) in \( \text{precond}(o) \), replace \( \neg p \) with \( p' \).
    - if \( p \) in \( \text{effects}(o) \), add \( \neg p' \) to \( \text{effects}(o) \).
    - if \( \neg p \) in \( \text{effects}(o) \), add \( p' \) to \( \text{effects}(o) \).
  - In the goal replace \( \neg p \) with \( p' \).
  - For every operator \( o \) create an action \( (\text{precond}(o), \text{effects}^-(a), \text{effects}^+(a)) \).

Translation: STRIPS to State-Variable Representation

- Let \( P=(O,s_i,g) \) be a statement of a classical planning problem. In the operators \( O \), in the initial state \( s_i \), and in the goal \( g \):
  - replace every positive literal \( p(t_1,\ldots,t_n) \) with a state-variable expression \( p(t_1,\ldots,t_n)=1 \) or \( p(t_1,\ldots,t_n)\leftarrow 1 \) in the operators’ effects, and
  - replace every negative literal \( \neg p(t_1,\ldots,t_n) \) with a state-variable expression \( p(t_1,\ldots,t_n)=0 \) or \( p(t_1,\ldots,t_n)\leftarrow 0 \) in the operators’ effects.
Translation: State-Variable to STRIPS Representation

Let \( P=(O,s_i,g) \) be a statement of a state-variable planning problem. In the operators’ preconditions, in the initial state \( s_i \), and in the goal \( g \):

- replace every state-variable expression \( p(t_1,\ldots,t_n)=v \) with an atom \( p(t_1,\ldots,t_n,v) \), and

- in the operators’ effects:
  - replace every state-variable assignment \( p(t_1,\ldots,t_n)\leftarrow v \) with a pair of literals \( p(t_1,\ldots,t_n,v), \neg p(t_1,\ldots,t_n,w) \), and add \( p(t_1,\ldots,t_n,w) \) to the respective operators preconditions.

Overview

- World States
- Domains and Operators
- Planning Problems
- Plans and Solutions
- Expressiveness