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 Representations

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

• **STRIPS representation**
  - named after STRIPS planner
  - like propositional representation, but first-order literals instead of propositions
  - most popular for restricted state-transitions systems

• **state-variable representation**
  - state is tuple of state variables \( \{x_1, \ldots, x_n\} \)
  - action is partial function over states
  - useful where state is characterized by attributes over finite domains

• equally expressive: planning domain in one representation can also be represented in the others
Classical Planning

• task: find solution for planning problem

• planning problem
  • initial state
    • atoms (relations, objects)
  • planning domain
    • operators (name, preconditions, effects)
  • goal

• solution (plan)

Classical Planning

• task: find solution for planning problem

• planning problem
  • initial state
    • state is a set of atoms (relations, objects)
    • difference between representations: what constitutes an atom
  • 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(crate), …
  • static vs. fluent relations

Knowledge Engineering

• What types of objects do we need to represent?
  • example: cranes, robots, containers, …
  • note: objects usually only defined in problem
    • type hierarchy usually not found in planning domain (in PDDL) but ontology is very important for KE

• What relations hold between these objects?
  • example: at(robot, location), empty(crate), …
    • define skeleton for readability (optional in PDDL)
  • static vs. fluent relations
Representing World States

- states are sets of atoms in all cases; difference lies in what is an atom
DWR Example: STRIPS States

• predicate symbols: relations for DWR domain

• constant symbols: for objects in the domain \{loc1, loc2, r1, crane1, p1, p2, c1, c2, c3, pallet\}

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

• $L = \{\text{onpallet}, \text{onrobot}, \text{holding}, \text{at1}, \text{at2}\}$

• meaning: container is on the ground, container on the robot, crane is holding the container, robot is at location1, robot is at location2

• $S = \{s_0, \ldots, s_5\}$

• as shown in graph

• $s_0 = \{\text{onpallet}, \text{at1}\}$

• $s_1 = \{\text{holding}, \text{at1}\}$

• $s_2 = \{\text{onpallet}, \text{at1}\}$

• $s_3 = \{\text{holding}, \text{at1}\}$

• $s_4 = \{\text{onrobot}, \text{at1}\}$

• $s_5 = \{\text{onrobot}, \text{at2}\}$
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 l in each state

• idea: represent such relations using state-variable functions mapping states into objects
  • example: functional representation:
    rloc:robots×S→locations

• STRIPS state containing at(r1,loc1) and at(r1,loc2) usually inconsistent

• idea: represent such relations using state-variable functions mapping states into objects

• advantage: reduces possibilities for inconsistent states, smaller state space

• example: functional representation:
  rloc:robots×S→locations
  • in general: maps objects and state into object
  • rloc is state-variable symbol that denotes state-variable function
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\}

DWR Example: State-Variable State Descriptions

- simplified: no cranes, no piles
  - robots can load and unload containers autonomously
- state-variable functions:
  - rloc: robots×S → locations
  - location of a robot in a state
  - rolad: robots×S→containers ∪ {nil}
  - what a robot has loaded in a state; nil for nothing loaded
  - cpos: containers×S → locations ∪ robots
  - where a container is in a state; at a location or on some robot
- 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

- preconditions and effects essentially sets of atoms again (where atoms are different per representation)
  - propositional representation allows only for positive preconditions
  - state-variable representation only allows for equality in preconditions, no inequality
  - effects: positive and negative in all cases
DWR Example: STRIPS Operators

- \textit{move}(r,l,m)
  - \textit{precond}: adjacent(l,m), at(r,l), \neg \text{occupied}(m)
  - \textit{effects}: at(r,m), occupied(m), \neg \text{occupied}(l), \neg at(r,l)

- \textit{load}(k,l,c,r)
  - \textit{precond}: belong(k,l), holding(k,c), at(r,l), unloaded(r)
  - \textit{effects}: empty(k), \neg holding(k,c), loaded(r,c), \neg unloaded(r)

- \textit{put}(k,l,c,d,p)
  - \textit{precond}: belong(k,l), attached(p,l), holding(k,c), top(d,p)
  - \textit{effects}: \neg holding(k,c), empty(k), in(c,p), top(c,p), on(c,d), \neg top(d,p)

- similar: unload and take operators
- action: just substitute variables with values consistently
DWR Example: Propositional Actions

- $a$ : $\text{precond}(a)$, $\text{effects}^{-}(a)$, $\text{effects}^{+}(a)$
  - $a$ is action name
- $\text{take} : \{\text{onpallet}\}, \{\text{onpallet}\}, \{\text{holding}\}$
- $\text{put} : \{\text{holding}\}, \{\text{holding}\}, \{\text{onpallet}\}$
- $\text{load} : \{\text{holding, at1}\}, \{\text{holding}\}, \{\text{onrobot}\}$
- $\text{unload} : \{\text{onrobot, at1}\}, \{\text{onrobot}\}, \{\text{holding}\}$
- $\text{move1} : \{\text{at2}\}, \{\text{at2}\}, \{\text{at1}\}$
- $\text{move2} : \{\text{at1}\}, \{\text{at1}\}, \{\text{at2}\}$

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

• simplified domain: no piles, no cranes – only three operators:

• move\((r,l,m)\)
  • move robot \(r\) from location \(l\) to adjacent location \(m\)
  • precond: \(\text{rloc}(r)=l\), \(\text{adjacent}(l,m)\)
  • effects: \(\text{rloc}(r)\leftarrow m\)

• load\((r,c,l)\)
  • robot \(r\) loads container \(c\) at location \(l\)
  • precond: \(\text{rloc}(r)=l\), \(\text{cpos}(c)=l\), \(\text{rload}(r)=\text{nil}\)
  • effects: \(\text{cpos}(c)\leftarrow r\), \(\text{rload}(r)\leftarrow c\)

• unload\((r,c,l)\)
  • robot \(r\) unloads container \(c\) at location \(l\)
  • precond: \(\text{rloc}(r)=l\), \(\text{rload}(r)=c\)
  • effects: \(\text{rload}(r)\leftarrow \text{nil}\), \(\text{cpos}(c)\leftarrow l\)
Overview

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

- essentially the same for all representations

<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

• Σ: STRIPS planning domain for DWR domain
  • see previous slides

• s_i: any state
  • example: s_0 = {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)}

• note: s_0 is not necessarily initial state

• g: any subset of L
  • example: g = {¬unloaded(robot), at(robot,loc2)}, i.e. S_g = {s_5}

• other relations will hold, but they are not mentioned in the goal = partial specification of a state
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: \( g = \{ \text{onrobot}, \text{at2} \} \), i.e. \( S_g = \{ s_5 \} \)

- see previous slides
- example: initial state = \( s_0 \in S \)
- note: \( s_0 \) is not necessarily initial state
- example: \( g = \{ \text{onrobot}, \text{at2} \} \), 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 \( P = (\Sigma, s_i, g) \) be a planning problem. A plan \( \pi \) is a solution for \( P \) if \( \gamma(s, \pi) \) satisfies \( g \).

**Classical Plans**

-Note: classical definitions apply to all representations

A plan is any sequence of actions \( \pi = \langle a_1, \ldots, a_k \rangle \), where \( k \geq 0 \).

- \( k = 0 \) means no actions in the empty plan
- The length of plan \( \pi \) is \( |\pi| = k \), the number of actions.
- If \( \pi_1 = \langle a_1, \ldots, a_k \rangle \) and \( \pi_2 = \langle a'_1, \ldots, a'_j \rangle \) are plans, then their concatenation is the plan \( \pi_1 \cdot \pi_2 = \langle a_1, \ldots, a_k, a'_1, \ldots, a'_j \rangle \).
- 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

- Plan corresponds to a path through the state space
Overview

- World States
- Domains and Operators
- Planning Problems
- Plans and Solutions
- Expressiveness
Grounding a STRIPS Planning Problem

- Let \( P = (O, s, g) \) be the statement of a STRIPS planning problem and \( C \) the set of all the constant symbols that are mentioned in \( s \). 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 \( P' = (\text{ground}(O), s, g) \) is a statement of a STRIPS planning problem and \( P' \) has the same solutions as \( P \).

- the number of operators will increase exponentially here

Then \( P' = (\text{ground}(O), s, g) \) is a statement of a STRIPS planning problem and \( P' \) has the same solutions as \( P \).

- the problems are equivalent (except for exponential increase in size)
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: 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
    • $\text{effects}(o) = \text{effects}^+(a) \cup \{\neg p \mid p \in \text{effects}^-(a)\}$.

• adds negation sign to negative effects
• result is a statement of a ground STRIPS planning problem
Translation: Ground STRIPS to Propositional Representation

• Let $P=(O,s,g)$ be a ground statement of a classical planning problem.
  • In the operators $O$, in the initial state $s$, and in the goal $g$ replace every atom $P(v_1,\ldots,v_n)$ with a propositional atom $Pv_1,\ldots,v_n$.
  • 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 $(\text{precond}(o), \text{effects}^-(a), \text{effects}^+(a))$.

• problem: operators may contain negated preconditions
• In the operators $O$, in the initial state $s$, and in the goal $g$ replace every atom $P(v_1,\ldots,v_n)$ with a propositional atom $Pv_1,\ldots,v_n$.
• idea: introduce new proposition symbols that represent the negations of existing propositions
• 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 $(\text{precond}(o), \text{effects}^-(a), \text{effects}^+(a))$.

• result is a statement of a propositional planning problem
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 $p(t_1,...,t_n)$ with a state-variable expression $p(t_1,...,t_n)=1$ or $p(t_1,...,t_n)\leftarrow 1$ in the operators’ effects, and
- replace every negative literal $\neg p(t_1,...,t_n)$ with a state-variable expression $p(t_1,...,t_n)=0$ or $p(t_1,...,t_n)\leftarrow 0$ in the operators’ effects.

result is a statement of a state-variable planning problem
Translation: State-Variable to STRIPS Representation

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

- result is a statement of a STRIPS planning problem
Overview

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

Overview

World States

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