The Situation Calculus and the Frame Problem

Using Theorem Proving to Generate Plans

Literature

**Classical Planning**

- restricted state-transition system $\Sigma=(S,A,y)$
- planning problem $\mathcal{P}=(\Sigma,s_i,S_g)$

- Why study classical planning?
  - good for illustration purposes
  - algorithms that scale up reasonably well are known
  - extensions to more realistic models known

- What are the main issues?
  - how to represent states and actions
  - how to perform the solution search

---

**Planning as Theorem Proving**

- idea:
  - represent states and actions in first-order predicate logic
  - prove that there is a state $s$
    - that is reachable from the initial state and
    - in which the goal is satisfied.
  - extract plan from proof
Overview

- Propositional Logic
- First-Order Predicate Logic
- Representing Actions
- The Frame Problem
- Solving the Frame Problem

Propositions

- proposition: a declarative sentence (or statement) that can either true or false
- examples:
  - the robot is at location1
  - the crane is holding a container
- atomic propositions (atoms):
  - have no internal structure
  - notation: capital letters, e.g. P, Q, R, ...
Well-Formed Formulas

- an atom is a formula
- if G is a formula, then (¬G) is a formula
- if G and H are formulas, then (G∧H), (G∨H), (G→H), (G↔H) are formulas.
- all formulas are generated by applying the above rules
- logical connectives: ¬, ∧, ∨, →, ↔

Truth Tables

<table>
<thead>
<tr>
<th>G</th>
<th>H</th>
<th>¬G</th>
<th>G∧H</th>
<th>G∨H</th>
<th>G→H</th>
<th>G↔H</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
Interpretations

- Let $G$ be a propositional formula containing atoms $A_1,\ldots,A_n$.
- An interpretation $I$ is an assignment of truth values to these atoms, i.e. $I: \{A_1,\ldots,A_n\} \rightarrow \{true, false\}$
- example:
  - formula $G$: $(P \land Q) \rightarrow (R \leftrightarrow (\neg S))$
  - interpretation $I$: $P \rightarrow false$, $Q \rightarrow true$, $R \rightarrow true$, $S \rightarrow true$
  - $G$ evaluates to true under $I$: $I(G) = true$

Validity and Inconsistency

- A formula is valid if and only if it evaluates to true under all possible interpretations.
- A formula that is not valid is invalid.
- A formula is inconsistent (or unsatisfiable) if and only if it evaluates to false under all possible interpretations.
- A formula that is not inconsistent is consistent (or satisfiable).
- examples:
  - valid: $P \lor \neg P$, $P \land (P \rightarrow Q) \rightarrow Q$
  - satisfiable: $(P \land Q) \rightarrow (R \leftrightarrow (\neg S))$
  - inconsistent: $P \land \neg P$
Propositional Theorem Proving

- Problem: Given a set of propositional formulas $F_1 \ldots F_n$, decide whether
  - their conjunction $F_1 \land \ldots \land F_n$ is valid or satisfiable or inconsistent or
  - a formula $G$ follows from (axioms) $F_1 \land \ldots \land F_n$, denoted $F_1 \land \ldots \land F_n \models G$

- decidable
- NP-complete, but relatively efficient algorithms known (for propositional logic)

Overview

- Propositional Logic
- First-Order Predicate Logic
- Representing Actions
- The Frame Problem
- Solving the Frame Problem
**First-Order Atoms**

- objects are denoted by terms
  - constant terms: symbols denoting specific individuals
    • examples: loc1, loc2, ..., robot1, robot2, ...
  - variable terms: symbols denoting undefined individuals
    • examples: l, l'
  - function terms: expressions denoting individuals
    • examples: 1+3, father(john), father(mother(x))
- first-order propositions (atoms) state a relation between some objects
  • examples: adjacent(l, l'), occupied(l), at(r, l), ...

---

**DWR Example State**

![DWR Example State Diagram]

- crane
- container
- pile (p1 and q1)
- robot
- location
- container
- pallet
- pile (p2 and q2, both empty)
Objects in the DWR Domain

- **locations** \{loc1, loc2, \ldots\}:
  - storage area, dock, docked ship, or parking or passing area
- **robots** \{robot1, robot2, \ldots\}:
  - container carrier carts for one container
  - can move between adjacent locations
- **cranes** \{crane1, crane2, \ldots\}:
  - belongs to a single location
  - can move containers between robots and piles at same location
- **piles** \{pile1, pile2, \ldots\}:
  - attached to a single location
  - pallet at the bottom, possibly with containers stacked on top of it
- **containers** \{cont1, cont2, \ldots\}:
  - stacked in some pile on some pallet, loaded onto robot, or held by crane
- **pallet**:
  - at the bottom of a pile

Topology in the DWR Domain

- **adjacent** \(l, l'\): location \(l\) is adjacent to location \(l'\)
- **attached** \((p, l)\): pile \(p\) is attached to location \(l\)
- **belong** \((k, l)\):
  - crane \(k\) belongs to location \(l\)

- topology does not change over time!
Relations in the DWR Domain (1)

- **occupied(l)**: location \(l\) is currently occupied by a robot
- **at(r,l)**: robot \(r\) is currently at location \(l\)
- **loaded(r,c)**: robot \(r\) is currently loaded with container \(c\)
- **unloaded(r)**: robot \(r\) is currently not loaded with a container

Relations in the DWR Domain (2)

- **holding(k,c)**: crane \(k\) is currently holding container \(c\)
- **empty(k)**: crane \(k\) is currently not holding a container
- **in(c,p)**: container \(c\) is currently in pile \(p\)
- **on(c,c')**: container \(c\) is currently on container/pallet \(c'\)
- **top(c,p)**: container/pallet \(c\) is currently at the top of pile \(p\)
Well-Formed Formulas

- an atom (relation over terms) is a formula
- if $G$ and $H$ are formulas, then $(\neg G)$, $(G \land H)$, $(G \lor H)$, $(G \rightarrow H)$, $(G \leftrightarrow H)$ are formulas

- if $F$ is a formula and $x$ is a variable then $(\exists x \ F(x))$ and $(\forall x \ F(x))$ are formulas

- all formulas are generated by applying the above rules

Formulas: DWR Examples

- adjacency is symmetric:
  \[ \forall l, l' \ \text{adjacent}(l, l') \leftrightarrow \text{adjacent}(l', l) \]

- objects (robots) can only be in one place:
  \[ \forall r, l, l' \ \text{at}(r, l) \land \text{at}(r, l') \rightarrow l = l' \]

- cranes are empty or they hold a container:
  \[ \forall k \ \text{empty}(k) \lor \exists c \ \text{holding}(k, c) \]
Semantics of First-Order Logic

- an interpretation \( I \) over a domain \( D \) maps:
  - each constant \( c \) to an element in the domain: \( I(c) \in D \)
  - each \( n \)-place function symbol \( f \) to a mapping: \( I(f) : D^n \rightarrow D \)
  - each \( n \)-place relation symbol \( R \) to a mapping:
    \[ I(R) : D^n \rightarrow \{ \text{true}, \text{false} \} \]
- truth tables for connectives (\( \neg, \land, \lor, \rightarrow, \leftrightarrow \)) as for propositional logic
- \( I(\exists x \; F(x)) = \text{true} \) if and only if for at least one object \( c \in D: I(F(c)) = \text{true} \).
- \( I(\forall x \; F(x)) = \text{true} \) if and only if for every object \( c \in D: I(F(c)) = \text{true} \).

Theorem Proving in First-Order Logic

- \( F \) is valid: \( F \) is \text{true} under all interpretations
- \( F \) is inconsistent: \( F \) is \text{false} under all interpretations
- theorem proving problem (as before):
  - \( F_1 \land \ldots \land F_n \) is valid / satisfiable / inconsistent or
  - \( F_1 \land \ldots \land F_n \models G \)
- semi-decidable
- resolution constitutes significant progress in mid-60s
Substitutions

- replace a variable in an atom by a term
- example:
  - substitution: $\sigma = \{x \leftarrow 4, \ y \leftarrow f(5)\}$
  - atom $A$: $\text{greater}(x, \ y)$
  - $\sigma(F) = \text{greater}(4, \ f(5))$

simple inference rule:
- if $\sigma = \{x \leftarrow c\}$ and $(\forall x \ F(x)) \models F(c)$
- example: $(\forall x \ \text{mortal}(x)) \models \text{mortal}(\text{Confucius})$

Unification

- Let $A(t_1, \ldots, t_n)$ and $A(t'_1, \ldots, t'_n)$ be atoms.
- A substitution $\sigma$ is a unifier for $A(t_1, \ldots, t_n)$ and $A(t'_1, \ldots, t'_n)$ if and only if:
  $\sigma(A(t_1, \ldots, t_n)) = \sigma(A(t'_1, \ldots, t'_n))$
- examples:
  - $P(x, \ 2)$ and $P(3, \ y)$ – unifier: $\{x \leftarrow 3, \ y \leftarrow 2\}$
  - $P(x, \ f(x))$ and $P(y, \ f(y))$ – unifiers: $\{x \leftarrow 3, \ y \leftarrow 3\}, \{x \leftarrow y\}$
  - $P(x, \ 2)$ and $P(x, \ 3)$ – no unifier exists
Overview

- Propositional Logic
- First-Order Predicate Logic
- Representing States and Actions
- The Frame Problem
- Solving the Frame Problem

Representing States

- represent domain objects as constants
  - examples: loc1, loc2, ..., robot1, robot2, ...
- represent relations as predicates
  - examples: adjacent(l,l'), occupied(l), at(r,l), ...
- problem: truth value of some relations changes from state to state
  - examples: occupied(loc1), at(robot1,loc1)
Situations and Fluents

- solution: make state explicit in representation through situation term
  - add situation parameter to changing relations:
    - \text{occupied}(\text{loc1},s): \text{location1 is occupied in situation s}
    - \text{at}(\text{robot1},\text{loc1},s): \text{robot1 is at location1 in situation s}
  - or introduce predicate holds(f,s):
    - \text{holds}(...): \text{holds in situation s}

- fluent: a term or formula containing a situation term

The Blocks World: Initial Situation

- $\Sigma_{si} =$
  - $\text{on}(\text{C},\text{Table},si) \land$
  - $\text{on}(\text{B},\text{C},si) \land$
  - $\text{on}(\text{A},\text{B},si) \land$
  - $\text{on}(\text{D},\text{Table},si) \land$
  - $\text{clear}(\text{A},si) \land$
  - $\text{clear}(\text{D},si) \land$
  - $\text{clear}($ Table $,si)$
**Actions**

- actions are non-tangible objects in the domain denoted by function terms
  - example: \texttt{move(robot1,loc1,loc2)}: move robot1 from location \texttt{loc1} to location \texttt{loc2}
- definition of an action through
  - a set of formulas defining applicability conditions
  - a set of formulas defining changes in the state brought about by the action

**Blocks World: Applicability**

- \( \Delta_a = \)
  \[ \forall x,y,z,s: \text{applicable(move}(x,y,z),s) \leftrightarrow \]
  \[ \text{clear}(x,s) \land \]
  \[ \text{clear}(z,s) \land \]
  \[ \text{on}(x,y,s) \land \]
  \[ x \neq \text{Table} \land \]
  \[ x \neq z \land \]
  \[ y \neq z \land \]
  \[ y \neq z \]
**Blocks World: move Action**

- single action $\text{move}(x,y,z)$: moving block $x$ from $y$ (where it currently is) onto $z$

**Applicability of Actions**

- for each action specify applicability axioms of the form:
  $\forall \text{params}, s: \text{applicable} (\text{action} (\text{params}), s) \leftrightarrow \text{preconds} (\text{params}, s)$
- where:
  - “applicable” is a new predicate relating actions to states
  - $\text{params}$ is a set of variables denoting objects
  - $\text{action} (\text{params})$ is a function term denoting an action over some objects
  - $\text{preconds} (\text{params})$ is a formula that is true iff $\text{action} (\text{params})$ can be performed in $s$
Effects of Actions

- for each action specify effect axioms of the form:
  \[ \forall params, s: \text{applicable}(\text{action}(params), s) \rightarrow \text{effects}(params, \text{result}(\text{action}(params), s)) \]

- where:
  - “result” is a new function that denotes the state that is the result of applying \( \text{action}(params) \) in \( s \)
  - \( \text{effects}(params, \text{result}(\text{action}(params), s)) \) is a formula that is true in the state denoted by \( \text{result}(\text{action}(params), s) \)

Blocks World: Effect Axioms

- \( \Delta_e = \)
  \[ \forall x, y, z, s: \text{applicable}(\text{move}(x, y, z), s) \rightarrow \]
  \[ \text{on}(x, z, \text{result}(\text{move}(x, y, z), s)) \wedge \]
  \[ \forall x, y, z, s: \text{applicable}(\text{move}(x, y, z), s) \rightarrow \]
  \[ \text{clear}(y, \text{result}(\text{move}(x, y, z), s)) \]
Blocks World: Derivable Facts

result(move(A,B,D),si):

- $\Sigma_{si} \land \Delta_{a} \land \Delta_{e} \models \text{on}(A,D,\text{result}(\text{move}(A,B,D),si))$
- $\Sigma_{si} \land \Delta_{a} \land \Delta_{e} \models \text{clear}(B,\text{result}(\text{move}(A,B,D),si))$

Overview

- Propositional Logic
- First-Order Predicate Logic
- Representing States and Actions
- The Frame Problem
- Solving the Frame Problem
The Situation Calculus and the Frame Problem

Blocks World: Non-Derivable Fact

result(move(A,B,D),si):

\[ \Sigma_{si} \land \Delta_a \land \Delta_e \models on(B,C,\text{result}(move(A,B,D),si)) \]

• not derivable:

The Non-Effects of Actions

• effect axioms describe what changes when an action is applied, but not what does not change

• example: move robot
  • does not change the colour of the robot
  • does not change the size of the robot
  • does not change the political system in the UK
  • does not change the laws of physics
Frame Axioms

- for each action and each fluent specify a frame axiom of the form:
  \( \forall \text{params,vars,s}: \text{fluent}(\text{vars},s) \land \text{params} \neq \text{vars} \rightarrow \text{fluent}(\text{vars},\text{result}(\text{action}(%s),s)) \)

- where:
  - \( \text{fluent}(\text{vars},s) \) is a relation that is not affected by the application of the action
  - \( \text{params} \neq \text{vars} \) is a conjunction of inequalities that must hold for the action to not affect the fluent

Blocks World: Frame Axioms

- \( \Delta_f = \)
  \[
  \forall v,w,x,y,z,s: \ on(v,w,s) \land v \neq x \rightarrow \\
  \on(v,w,\text{result}(\text{move}(x,y,z),s)) \land \\
  \forall v,w,x,y,z,s: \ clear(v,s) \land v \neq z \rightarrow \\
  \clear(v,\text{result}(\text{move}(x,y,z),s))
  \]
Blocks World: Derivable Fact with Frame Axioms

result(move(A,B,D),si):

- now derivable:
  \[ \Sigma_{si} \land \Delta_a \land \Delta_e \land \Delta_f \models \text{on}(B,C,\text{result}(\text{move}(A,B,D),si)) \]

Coloured Blocks World

- like blocks world, but blocks have colour (new fluent) and can be painted (new action)
- new information about si:
  - \( \forall x: \text{colour}(x, \text{Blue}, si) \)
- new effect axiom:
  - \( \forall x,y,s: \text{colour}(x,y,\text{result}(\text{paint}(x,y),s)) \)
- new frame axioms:
  - \( \forall v,w,x,y,z,s: \text{colour}(v,w,s) \rightarrow \text{colour}(v,w,\text{result}((x,y,z),s)) \)
  - \( \forall v,w,x,y,s: \text{colour}(v,w,s) \land v \neq x \rightarrow \text{colour}(v,w,\text{result}(\text{paint}(x,y),s)) \)
  - \( \forall v,w,x,y,s: \text{on}(v,w,s) \rightarrow \text{on}(v,w,\text{result}(\text{paint}(x,y),s)) \)
  - \( \forall v,w,x,y,s: \text{clear}(v,w,s) \rightarrow \text{clear}(v,w,\text{result}(\text{paint}(x,y),s)) \)
The Frame Problem

- problem: need to represent a long list of facts that are not changed by an action

- the frame problem:
  - construct a formal framework
  - for reasoning about actions and change
  - in which the non-effects of actions do not have to be enumerated explicitly

Overview

- Propositional Logic
- First-Order Predicate Logic
- Representing States and Actions
- The Frame Problem
  - Solving the Frame Problem
Approaches to the Frame Problem

- use a different style of representation in first-order logic (same formalism)
- use a different logical formalism, e.g. non-monotonic logic
- write a procedure that generates the right conclusions and forget about the frame problem

Criteria for a Solution

- representational parsimony: representation of the effects of actions should be compact
- expressive flexibility: representation suitable for domains with more complex features
- elaboration tolerance: effort required to add new information is proportional to the complexity of that information
### The Universal Frame Axiom

- frame axiom for all actions, fluents, and situations:
  \[ \forall a,f,s: \text{holds}(f,s) \land \neg \text{affects}(a,f,s) \rightarrow \text{holds}(f,\text{result}(a,s)) \]
- where “affects” is a new predicate that relates actions, fluents, and situations
- \( \neg \text{affects}(a,f,s) \) is true if and only if the action \( a \) does not change the value of the fluent \( f \) in situation \( s \)

### Coloured Blocks World Example Revisited

- coloured blocks world new frame axioms:
  - \( \forall v,w,x,y,z,s: x \neq v \rightarrow \neg \text{affects}(\text{move}(x,y,z), \text{on}(v,w), s) \)
  - \( \forall v,w,x,y,s: \neg \text{affects}(\text{paint}(x,y), \text{on}(v,w), s) \)
  - \( \forall v,x,y,z,s: y \neq v \land z \neq v \rightarrow \neg \text{affects}(\text{move}(x,y,z), \text{clear}(v), s) \)
  - \( \forall v,x,y,s: \neg \text{affects}(\text{paint}(x,y), \text{clear}(v), s) \)
  - \( \forall v,w,x,y,z,s: \neg \text{affects}(\text{move}(x,y,z), \text{colour}(v,w), s) \)
  - \( \forall v,w,x,y,s: x \neq v \rightarrow \neg \text{affects}(\text{paint}(x,y), \text{colour}(v,w), s) \)

- more compact, but not fewer frame axioms
Explanation Closure Axioms

- idea: infer the action from the affected fluent:
  - $\forall a,v,w,s: \text{affects}(a, \text{on}(v,w), s) \rightarrow \exists x,y: a=\text{move}(v,x,y)$
  - $\forall a,v,s: \text{affects}(a, \text{clear}(v), s) \rightarrow (\exists x,z: a=\text{move}(x,v,z)) \lor (\exists x,y: a=\text{move}(x,y,v))$
  - $\forall a,v,w,s: \text{affects}(a, \text{colour}(v,w), s) \rightarrow \exists x: a=\text{paint}(v,x)$
- allows to draw all the desired conclusions
- reduces the number of required frame axioms
- also allows to the draw the conclusion:
  - $\forall a,v,w,x,y,s: a\neq\text{move}(v,x,y) \rightarrow \neg\text{affects}(a, \text{on}(v,w), s)$

The Limits of Classical Logic

- monotonic consequence relation:
  $\Delta \vDash \phi$ implies $\Delta \land \delta \vDash \phi$
- problem:
  - need to infer when a fluent is not affected by an action
  - want to be able to add actions that affect existing fluents
- monotonicity: if $\neg\text{affects}(a, f, s)$ holds in a theory it must also hold in any extension
Using Non-Monotonic Logics

- non-monotonic logics rely on default reasoning:
  - jumping to conclusions in the absence of information to the contrary
  - conclusions are assumed to be true by default
  - additional information may invalidate them
- application to frame problem:
  - explanation closure axioms are default knowledge
  - effect axioms are certain knowledge

Overview

- Propositional Logic
- First-Order Predicate Logic
- Representing States and Actions
- The Frame Problem
- Solving the Frame Problem