Introduction

Sungmoon Joo

Institute for Robotics & Intelligent Machines
School of Interactive Computing
Georgia Institute of Technology

What is Planning?

Given:
- States
- Actions
- Initial State and Goal State
- Constraints

Task 1 (eg. Classical Planning):
Find a sequence of actions that take you from Init. to Goal

Task 2 (eg. Policy, Controls):
Find actions that take you from any state to Goal

Task 3 (eg. MDP, POMDP):
Decide the best action to take now in order to improve your odds of reaching Goal

Task 4 (eg. Motion Planning):
Find a continuous path (in state space) that takes you from Init. to Goal
...
What is Planning?

Sequence of Actions

Init. (0,0) → (3,3) → (6,6) → (7,6) → Goal (7,6)

Init., Goal States
Actions
Constraints

What is Planning?

MDP, POMDP

Init. (0,0) → (3,3) → (6,6) → (7,6) → Goal (7,6)

Init., Goal States
Actions
Constraints
Uncertainties
- action
- observation

Probability distribution
What is Planning?

Motion Planning

Control/Policy

Find actions that take you from any state to Goal
What is Planning?
What is Planning?

Course Overview

Learn Tools/Frameworks:
• Classical Planning
• Motion Planning
• MDP, POMDP
• Controls

Apply to Real-world Problems:
• Project

Grading: 2 Group HWs + Final Project
• Classical Planning (30%)
• Motion Planning (30%)
• Project (40%) – Grad(CS7649) / Undergrad(CS4649)

Schedule
• Sep. 16, Nov. 18/20 – TBD (IROS, Humanoids conference)
• Oct. 14, Nov. 27 – No Class (Fall Recess, Thanksgiving)
• Project Presentation – Dec 4. or Final exam day
* Course website: https://joosm.github.io/RIP2014
Representation of Information: Predicate Logic

Motivation:
- Historically the first (McCarthy, Nilsson, Fikes)
- More efficient than explicitly representing states
- Only gives initial state, operators generate the rest as needed
- Easier to analyze due to simplicity and generality

First Representation: Predicate Logic

Statement → Predicate

(Unary) Predicate: P(x)
Introduce a (functional) symbol(P) for the predicate, and put the subject(x) as an argument to the functional symbol.

N-ary predicate is defined similarly

Examples:
- x is happy → Happy(x): Unary predicate
- The suitcase contains a bomb → Contains(Suitcase, Bomb): Binary predicate
- X is less than Y → LessThan(x,y): Binary predicate
- P(x1,x2,…,xn): n-ary predicate
First Representation: Predicate Logic

Objects (Constants): \( a, 123, \text{house}, \text{mike}, \text{robot}, \text{suitcase} \)

Variables: \( x, y, z \ldots \)

Relations (Predicates): \( \text{LessThan, Contains, Parent, Happy} \)

Connectives: \( \neg, \lor, \land, \Rightarrow \)

Any expression is either true or false: \( \text{LessThan}(1, 2) \)

\( \neg \text{Contains(Suitcase, Bomb)} \)

Truth Table

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( \neg A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
First Representation: Predicate Logic

Objects (Constants): \( a, 123, house, mike, robot, suitcase \)

Variables: \( x, y, z, \ldots \)

Relations (Predicates): \( LessThan, Contains, Parent, Happy \)

Connectives: \( \neg, \lor, \land, \Rightarrow \)

Any expression is either true or false:

\[
\neg Contains(Suitcase, Bomb) \\
Working(Robot) \land Has(Robot, Intelligence) \\
LessThan(1, 2)
\]

Truth Table

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( \neg A )</th>
<th>( A \land B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
First Representation: Predicate Logic

Objects (Constants): $a, 123, house, mike, robot, suitcase$

Variables: \( x, y, z \ldots \)

Relations (Predicates): \( \text{LessThan}, \text{Contains}, \text{Parent}, \text{Happy} \)

Connectives: \( \neg, \vee, \wedge, \Rightarrow \)

Any expression is either true or false:

\[
\text{Planning(Robot)} \Rightarrow \text{Working(Robot)}
\]

Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>$\neg A$</th>
<th>$A \wedge B$</th>
<th>$A \Rightarrow B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
First Representation: Predicate Logic

Objects (Constants): \(a, 123, \text{house}, \text{mike}, \text{robot}, \text{suitcase}\)

Variables: \(x, y, z, \ldots\)

Relations (Predicates): \(\text{LessThan}, \text{Contains}, \text{Parent}, \text{Happy}\)

Connectives: \(\neg, \lor, \land, \Rightarrow\)

Any expression is either true or false:

\[
\text{Planning}(\text{Robot}) \Rightarrow \text{Working}(\text{Robot})
\]

\[
\neg\text{Planning}(\text{Robot}) \lor \text{Working}(\text{Robot})
\]

Truth Table

Negation+OR/Disjunction

<table>
<thead>
<tr>
<th>(A)</th>
<th>(B)</th>
<th>(\neg A)</th>
<th>(A \land B)</th>
<th>(A \Rightarrow B)</th>
<th>(\neg A \lor B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Predicate Logic: Proofs

Represent a World using Axioms (i.e. predicates) \( \rightarrow \) Prove (by deduction)

- **Axioms:**
  
  \[
  \text{Contains(Suitcase, Laptop)}, \text{Contains(Laptop, Slides)} \\
  \text{Contains(Slides, Movies)} \\
  \forall x, y, z (\text{Contains}(x, y) \land \text{Contains}(y, z) \Rightarrow \text{Contains}(x, z))
  \]

- **Prove:** \text{Contains(Suitcase, Movies)}

1) \text{Contains(Suitcase, Laptop)} \land \text{Contains(Laptop, Slides)}

2) \text{Contains(Suitcase, Laptop)} \land \text{Contains(Laptop, Slides)}

\[\Rightarrow \text{Contains(Suitcase, Slides)}\]
Truth Table

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$\neg A$</th>
<th>$A \land B$</th>
<th>$A \Rightarrow B$</th>
<th>$\neg A \lor B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Predicate Logic: Proofs

- **Axioms**: $\text{Contains(Suitcase, Laptop)}$, $\text{Contains(Laptop, Slides)}$  
  $\text{Contains(Slides, Movies)}$

$$\forall x, y, z (\text{Contains}(x, y) \land \text{Contains}(y, z) \Rightarrow \text{Contains}(x, z))$$

- **Prove**: $\text{Contains(Suitcase, Movies)}$

1) $\text{Contains(Suitcase, Laptop)} \land \text{Contains(Laptop, Slides)}$
2) $\text{Contains(Suitcase, Laptop)} \land \text{Contains(Laptop, Slides)}$
   $\Rightarrow \text{Contains(Suitcase, Slides)}$
3) $\text{Contains(Suitcase, Slides)}$
4) $\text{Contains(Suitcase, Slides)} \land \text{Contains(Slides, Movies)}$
   $\Rightarrow \text{Contains(Suitcase, Movies)}$
5) $\text{Contains(Suitcase, Movies)}$

Up to now, the world is static...
Situation Calculus*

To represent and reason about dynamical worlds
To represent ‘change’, ‘state’ is introduced.

- **Fluents** = Add state (time?) variable to relations:
  
  Contains(Suitcase, Laptop, S0)
  Working(Robot, S7)

- **Actions** are reified functions of constants. (They can be treated as constants themselves)
  
  Put(Laptop, Suitcase)  Open(Car)  Lock(Car)

- The **do function**:  
  
  \[do(\alpha, \sigma_0) \rightarrow \sigma_1\]
  
  \[\alpha = \text{action}\]
  
  \[\sigma = \text{state}\]

*Modern version is different from the original for clarity. Calculus = study about ‘change’.

---

Situation Calculus

**Room(rm)**

- **Effect Axioms (Positive + Negative)**
  
  \[In(Robot, Hall, s) \land Open(rm, s) \Rightarrow In(Robot, rm, do(enter(rm), s))\]
  \[In(Robot, Hall, s) \land Open(rm, s) \Rightarrow \neg In(Robot, Hall, do(enter(rm), s))\]

\[S1 = do(enter(Class), S0)\]

\[In(Robot, Hall, S0)\]
\[Open(Class, S0)\]
\[Open(Office, S0)\]

\[In(Robot, Class, S1)\]
\[\neg In(Robot, Hall, S1)\]

S. Joo (sungmoon.joo@cc.gatech.edu)  8/19/2014  26
Situation Calculus

- **Effect Axioms (Positive + Negative)**
  
  \[ \text{In}(\text{Robot}, \text{Hall}, s) \land \text{Open}(\text{rm}, s) = \text{In}(\text{Robot}, \text{rm}, \text{do(\text{enter}(\text{rm}), s)}) \]
  
  \[ \text{In}(\text{Robot}, \text{Hall}, s) \land \text{Open}(\text{rm}, s) = \neg \text{In}(\text{Robot}, \text{Hall}, \text{do(\text{enter}(\text{rm}), s)}) \]

- **States**
  
  \[ S_0 = \text{do(\text{enter}(\text{Class}), S_0)} \]
  
  \[ S_1 = \text{do(\text{enter}(\text{Class}), S_0)} \]
  
  \[ S_2 = \text{do(\text{enter}(\text{Office}), S_0)} \]
  
  \[ \text{In}(\text{Robot}, \text{Class}, S_1) \]
  
  \[ \neg \text{In}(\text{Robot}, \text{Hall}, S_1) \]
  
  \[ \text{In}(\text{Robot}, \text{Office}, S_2) \]
  
  \[ \neg \text{In}(\text{Robot}, \text{Hall}, S_2) \]