Lecture Overview [7.1-7.4]

- Logical Agents
- Wumpus World
- Propositional Logic
- Inference
- Theorem Proving via model checking

Motivation

- Existing techniques help us solve:
  - Shortest path problems
  - Some classes of optimization problems
- What about problems that require logical reasoning?
  - eg creating a Sherlock Holmes agent
    - “When you have eliminated the impossible, whatever remains, however improbable, must be the truth.”

Logical Agents

- Maintain representation of knowledge of the world
- Use factored state representation
  - States are assignments of values to variables
- Like CSPs can generalize to many different problems
- Can also generalize to different goals
Knowledge-based agents

• Logical agents maintain world knowledge
  • Knowledge base (KB)
  • Knowledge stored in *sentences*
    • Each sentence represents knowledge about the world
      • Sherlock Holmes was a fictional detective

• Add knowledge: TELL
• Query knowledge: ASK

• Agent loop:
  • TELL KB about perceptions
  • ASK actions to perform
  • ASK not necessarily formulated explicitly

Knowledge

• Declarative:
  • TELL an agent what is needed
    • No extra knowledge

• Procedural:
  • Encode knowledge in program code
  • SAS is often procedural
  • Generalized planning is declarative

Wumpus World

• Performance
  • 1000 for getting gold and returning to start
  • -1000 for dying
  • -10 for shooting the arrow
  • -1 for each action
Wumpus World

- Environment
  - 4x4 grid of rooms
  - Agent has heading
  - Agent starts at [1, 1]
  - Gold & wumpus randomly placed
  - Probability 0.2 of a pit

Wumpus World (WW)

- Sensors
  - Can perceive *stench* from location adjoining (vertically/horizontally) a wumpus
  - Can perceive *breeze* from location adjoining a pit
  - Can perceive *glitter* in cell with gold
  - Can perceive *scream* when wumpus dies

Wumpus World

- Actuators
  - Turn right
  - Turn left
  - Forward
  - Shoot
  - Grab
  - Exit
Logic

• Syntax: defines well-formed sentences
• Semantics: what sentences mean
  • \( x + y = 4 \) is true when \( x = 2 \) and \( y = 2 \)
• Model: possible world
  • Includes all assignments of values to \( x/y \)
  • If \( \alpha \) is true in \( m \): \( m \) satisfies \( \alpha \)
  • \( M(\alpha) \) is the set of all models of \( \alpha \)
• What models exist for WW problem?

Entailment

• \( \alpha \) entails \( \beta \) or \( \alpha \models \beta \)
  • \( \beta \) follows logically from \( \alpha \)
  • In every model in which \( \alpha \) is true, \( \beta \) is also true
    • \( M(\alpha) \subseteq M(\beta) \)

Entailment examples

• Reminder \( \alpha \models \beta \): \( \beta \) follows logically from \( \alpha \); \( M(\alpha) \subseteq M(\beta) \)
• \( \alpha = (x = 0), \beta = (xy = 0) \)
• \( \alpha = (\text{AI lectures on Wednesday}), \beta = (\text{No AI lectures on the weekends}) \)
• \( \alpha = (\text{dogs have tails}), \beta = (\text{puppies have tails}) \)
• \( \alpha = (\text{girls like flowers; Rachel is a girl}), \beta = (\text{Rachel likes flowers}) \)
• Everyone in class give their own example
\[ \alpha = \text{KB} \]
\[ \beta = \text{No pit in [2, 2]} \]

**Entailment**
- This shows how entailment can be used to derive conclusions about the world
  - Performing *logical inference*
  - Model checking
    - Generate all possible models
      - Must be a finite number of models
    - Check if hypothesis is true

**Inference**
- \( \text{KB} \vdash \alpha \)
  - \( \alpha \) is derived from KB by inference algorithm \( i \)
  - A *sound* inference algorithm only derives entailed sentences
  - A *complete* inference algorithm can derive any entailed sentence
  - Model checking is sound & complete (when applicable)

**Propositional Logic**
- Simple form of logic
- Can seem limited, but more complex forms of logic can be reduced to propositional logic
Propositional Logic: Symbols

- Not: \( \neg \)
- And: \( \land \)
- Or: \( \lor \)
- Implies: \( \Rightarrow \) or \( \rightarrow \)
- If and only if: \( \Leftrightarrow \)

Prop. Logic Syntax

- Sentence \( \rightarrow \) AtomicSentence \( \mid \) ComplexSentence
- AtomicSentence \( \rightarrow \) True \( \mid \) False \( \mid \) P \( \mid \) Q \( \mid \) R \( \mid \) ...
- Complex Sentence \( \rightarrow \) (Sentence) \( \mid \) [Sentence]
  \( \mid \) \( \neg \) Sentence \( \mid \) Sentence \( \land \) Sentence
  \( \mid \) Sentence \( \lor \) Sentence \( \mid \) Sentence \( \Rightarrow \) Sentence
  \( \mid \) Sentence \( \Leftrightarrow \) Sentence
- Operator precedence: \( \neg \), \( \land \), \( \lor \), \( \Rightarrow \), \( \Leftrightarrow \)

Prop. Logic Semantics

- A model fixes the values of all variables to true or false
- True/False are always True/False
- Variables have their values defined in a model
- \( \neg P \) is true iff P is false in model
- \( P \land Q \) is true iff P and Q are both true in model
- \( P \lor Q \) is true iff P or Q are both true in model
- \( P \Rightarrow Q \) is true iff P is false or P&Q are both true in model
- \( P \Leftrightarrow Q \) is true iff P&Q have the same values in model

Semantics

- \( \Rightarrow \) and \( \Leftrightarrow \) not strictly needed
- \( A \Rightarrow B \) is the same as \( \neg A \lor B \)
- \( A \Leftrightarrow B \) is the same as \( (A \Rightarrow B) \land (B \Rightarrow A) \)
Construct WW KB

- \( P_{x,y} \) is true if there is a pit in \([x, y]\)
- \( W_{x,y} \) is true if there is a wumpus in \([x, y]\)
- \( B_{x,y} \) is true if the agent perceives breeze in \([x, y]\)
- \( S_{x,y} \) is true if the agent perceives stench in \([x, y]\)

WW KB

- There is no pit in \([1, 1]\)
  - \( R_1: \neg P_{1,1} \)
- A square is breezy iff there is a pit in a neighboring square
  - \( R_2: B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \)
  - \( R_3: B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \)
- Percepts:
  - \( R_4: \neg B_{1,1} \)
  - \( R_5: B_{2,1} \)

Prop. Logic: Simple inference

- How many variables? How many models?
- In how many is KB true?
### Selection of possible models

<table>
<thead>
<tr>
<th>B_{11}</th>
<th>B_{21}</th>
<th>P_{11}</th>
<th>P_{12}</th>
<th>P_{21}</th>
<th>P_{22}</th>
<th>P_{31}</th>
<th>R_1</th>
<th>R_2</th>
<th>R_3</th>
<th>R_5</th>
<th>KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

### Simple model checking

- How could we turn this into an algorithm?
- What is the running time?

---

Homework: 7.2