Knowledge Representation

CS 486/686: Introduction to Artificial Intelligence
Outline

• Knowledge-based agents
• Logics in general
• Propositional Logic & Reasoning
• First Order Logic
Introduction

• So far we have taken the following approach
  - Figure out exactly what the problem is (problem definition)
  - Design or pick an algorithm to solve the problem (search algorithm)
  - Execute the program
Knowledge-Based Agents

• An alternative approach
  - Identify the knowledge needed to solve the problem
  - Write down this knowledge in some language
  - Use logical consequences to solve the problem
Knowledge-Based Agents

• Ideally
  - We tell the agent what it needs to know
  - The agent infers what to do and how to do it

• Agent has two parts
  - **Knowledge base**: Set of facts expressed in a formal standard language
  - **Inference engine**: Rules for deducing new facts
An Example: Wumpus World

- **Goal**: Get gold back to start without falling into a pit or getting eaten by the wumpus

- **Environment**
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square

- **Sensors**: Stench, Breeze, Glitter, Bump, Scream

- **Actuators**: Left turn, Right turn, Forward, Grab, Release, Shoot
# Wumpus World

<table>
<thead>
<tr>
<th>1,4</th>
<th>2,4</th>
<th>3,4</th>
<th>4,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,3</td>
<td>2,3</td>
<td>3,3</td>
<td>4,3</td>
</tr>
<tr>
<td>1,2</td>
<td>2,2</td>
<td>3,2</td>
<td>4,2</td>
</tr>
<tr>
<td>1,1</td>
<td>2,1</td>
<td>3,1</td>
<td>4,1</td>
</tr>
</tbody>
</table>

(a)

### Legend:
- **A** = Agent
- **B** = Breeze
- **G** = Glitter, Gold
- **OK** = Safe square
- **P** = Pit
- **S** = Stench
- **V** = Visited
- **W** = Wumpus

<table>
<thead>
<tr>
<th>1,4</th>
<th>2,4</th>
<th>3,4</th>
<th>4,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,3</td>
<td>2,3</td>
<td>3,3</td>
<td>4,3</td>
</tr>
<tr>
<td>1,2</td>
<td>2,2</td>
<td><strong>P?</strong></td>
<td>4,2</td>
</tr>
<tr>
<td>1,1</td>
<td><strong>V</strong></td>
<td><strong>B</strong></td>
<td><strong>P?</strong></td>
</tr>
</tbody>
</table>

(b)

<table>
<thead>
<tr>
<th>1,4</th>
<th>2,4</th>
<th>3,4</th>
<th>4,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,3</td>
<td><strong>W!</strong></td>
<td>2,3</td>
<td>4,3</td>
</tr>
<tr>
<td>1,2</td>
<td><strong>A</strong></td>
<td>2,2</td>
<td>4,2</td>
</tr>
<tr>
<td>1,1</td>
<td><strong>V</strong></td>
<td>3,1</td>
<td><strong>P!</strong></td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>1,4</th>
<th>2,4</th>
<th>3,4</th>
<th>4,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,3</td>
<td><strong>W!</strong></td>
<td><strong>A</strong></td>
<td><strong>S</strong></td>
</tr>
<tr>
<td>1,2</td>
<td><strong>S</strong></td>
<td>2,2</td>
<td>4,2</td>
</tr>
<tr>
<td>1,1</td>
<td><strong>V</strong></td>
<td>3,1</td>
<td><strong>P!</strong></td>
</tr>
</tbody>
</table>

(b)
Outline

• Knowledge-based agents

• Logics in general

• Propositional Logic & Reasoning

• First Order Logic
What Is A Logic?

• Logic
  - A formal language for representing information so that conclusions can be drawn

• Logics have 2 components
  - Syntax: defines the sentences of the language
  - Semantics: defines the meaning of the sentences
Entailment

• Entailment means that “one thing follows from another”
  - \( \text{KB} \models \alpha \)

• Knowledge base (KB) entails sentence \( \alpha \) if and only if \( \alpha \) is true in all possible worlds where KB is true

• Example:
  - KB: I finished the AI assignment. I am happy
  - \( \alpha \): I finished the AI assignment and I am happy.
Models

• A model is a formal “possible world” where a sentence can be evaluated
  - m is a model of sentence α if α is true in m
• M(α) is the set of all models of α
• KB |=α if and only of M(KB)⊆M(α)

KB: I finished the AI homework and I did not sleep last night
⟨: I finished the AI homework
Inference

• Given a KB, we want to be able to draw conclusions from it

• **Inference procedure:** $\text{KB} \vdash \alpha$
  
  - Sentence $\alpha$ can be derived from KB by inference algorithm

• Desired properties:
  
  - **Soundness:** the procedure only infers true statements
    
    - If KB $\vdash \alpha$ then KB $\models \alpha$
  
  - **Completeness:** the procedure can generate all true statements
    
    - IF KB $\models \alpha$ then it is true that KB $\vdash \alpha$
Outline

- Knowledge-based agents
- Logics in general
- Propositional Logic & Reasoning
- First Order Logic
Propositional Logic

• **Atomic Symbols**: P, Q, R,...
  - Each symbol stands for a proposition that can be either True or False

• **Logical Connectives**
  - ¬ (negation)
  - ∨ (or)
  - ∧ (and)
  - ⇒ (implies)
  - ⇔ (if and only if, equivalence)
Inference: Propositional Logic

• Using truth tables is
  - **Sound**: direct definition of entailment
  - **Complete**: works for any KB and α and always terminates

• But...
  - Really inefficient
  - If there are n symbols, then there are $2^n$ models
Inference Rules

• Given a KB we want to derive conclusions
  - Proof: sequence of inference rule applications

\[ \text{Modus Ponens} \]
\[
\alpha, \alpha \Rightarrow \beta \\
\frac{}{\beta}
\]

\[ \text{Resolution} \]
\[
\alpha \lor \beta, \neg \beta \lor \gamma \\
\frac{}{\alpha \lor \gamma}
\]

\[ \text{And Elimination} \]
\[
\alpha \land \beta \\
\frac{}{\alpha}
\]

\[ \text{Unit Resolution} \]
\[
\alpha \lor \beta, \neg \beta \\
\frac{}{\alpha}
\]
• Resolution is a **sound and complete** inference rule

- Any complete search algorithm, applying only the resolution rule, can derive any conclusion entailed by any knowledge base in propositional logic.

**Caveat:** Given that \( \langle \) is true, we can not automatically generate \( \langle V \rangle \text{ is true}. \) However, we can find the answer to the question "Is \( \langle V \rangle \text{ true}".
Conjunctive Normal Form

• Resolution is applied to clauses of the form $\alpha \lor \beta \lor \ldots \lor \gamma$

• Any clause in propositional logic is logically equivalent to a clause in CNF
  - conjunction of disjunctions
  - eg. $(P \lor \neg Q \lor R) \land (\neg Q \lor A \lor B) \land \ldots$
Converting to CNF

1. Eliminate $\iff$, replacing $P \iff Q$ with $(P \implies Q) \land (Q \implies P)$

2. Eliminate $\implies$, replacing $P \implies Q$ with $\neg P \lor Q$

3. Move "\neg" inwards, using $\neg(\neg P) = P$, $\neg(P \land Q) = \neg P \lor \neg Q$ and $\neg(P \lor Q) = \neg P \land \neg Q$

4. Distribute $\lor$ over $\land$ where possible
Resolution Algorithm

• Recall: To show $KB|=\alpha$, we show that $(KB\land\neg\alpha)$ is unsatisfiable

• **Resolution Algorithm:**
  - Convert $(KB\land\neg\alpha)$ to CNF
  - For every pair of clauses that contain complementary literals
    - Apply resolution to produce a new clause
    - Add new clause to set of clauses
  - Continue until
    - No new clauses are being added ($KB$ does not entail $\alpha$) or
    - Two clauses resolve to produce empty clause ($KB|=\alpha$)
Complexity of Inference

• Inference for propositional logic is NP-complete
• If all clauses are Horn clauses, then inference is linear in size of KB!
  - Horn clause: Disjunction of literals where at most one literal is positive
    - \( \neg P \lor Q \lor \neg R \) is a Horn clause
    - \( P \lor Q \lor R \) is not a Horn clause
  - Every Horn clauses establishes exactly one new fact
    - \( \neg P \lor Q \lor \neg R \iff (P \land R) \Rightarrow Q \)
    - We add all new facts in \( n \) passes
Forward Chaining

• When a new sentence $\alpha$ is added to the KB
  - Look for all sentences that share literals with $\alpha$
  - Perform resolution
  - Add new sentence to KB and continue

• Forward chaining is
  - Data-driven
  - Eager: new facts are inferred as soon as possible
Backward Chaining

- When a query q is asked of the KB
  - If q is in the KB, return True
  - Otherwise, use resolution for q with other sentences in the KB and continue from result

- Backward chaining is
  - Goal driven: Centers reasoning around query being asked
  - Lazy: new facts are inferred only when needed
Forward vs Backward

• Which is better? That depends!

• Backward Chaining:
  - Does not grow the KB as much
  - Focused on proof so is generally more efficient
  - Does nothing until a question is asked
  - Typically used in proofs by contradiction
Forward vs Backward

• Forward Chaining
  - Extends the KB and improves understanding of the world
  - Typically used in tasks where the focus is on providing a model of the world
Outline

• Knowledge-based agents
• Logics in general
• Propositional Logic & Reasoning
• First Order Logic
First Order Logic

• New elements
  - Predicates
    - Define objects, properties, relationships
  - Quantifiers
    - ∀ (for all), ∃ (there exists) are used in statements that apply to a class of objects

• Example: ∀x On(x, Table) ⇒ Fruit(x)
Sentences

- **Terms**
  - Constants, variables, function($term_1, ..., term_n$)

- **Atomic Sentences**
  - Predicate($term_1, term_2$), $term_1 = term_2$

- **Complex Sentences**
  - Combine atomic sentences with connectives
  - $\text{Likes}(\text{Alice}, \text{IceCream}) \land \text{Likes}(\text{Bob}, \text{IceCream})$
Semantics
Inference and FOL

- We know how to do inference in Propositional Logic: find $\alpha$ such that $KB \models \alpha$
  - Is it possible to use these techniques for FOL?
  - Have to handle quantifiers, predicates, functions, ...
Universal Instantiation

• Given sentence $\forall x \ P(x) \land Q(x) \Rightarrow R(x)$ then we want to infer $P(\text{John}) \land Q(\text{John}) \Rightarrow R(\text{John})$ and $P(\text{Anne}) \land P(\text{Anne}) \Rightarrow R(\text{Anne})$ and ...

Universal Instantiation (UI)

$\forall v \alpha \Rightarrow \text{SUBST}([v/g]\alpha)$

- $\forall$ is a variable
- $\alpha$ is a sentence
- $\text{SUBST}([v/g]\alpha)$
- Substitute $g$ for all occurrences of $v$ in $\alpha$
- $g$ is a ground term*
Existential Instantiation

• For any sentence $\alpha$, variable $v$ and constant symbol $K$ that does not appear anywhere in the KB

$$\exists v \alpha$$
$$\text{SUBST}(\{x/K\}, \alpha)$$

**Example**

$$\exists x \text{Crown}(x) \text{ yields}$$

$$\text{Crown}(C_1) \quad (C_1 \text{ is a new constant})$$
Reduction to Propositional Inference

• Suppose the KB contained the following
  – ∀x Cat(x) ∧ Orange(x) ⇒ Cute(x)
  – Orange(Kitty)
  – Cat(Kitty)
  – Sister(Kitty, Katy)

• Instantiating the universal sentence in all possible ways we have a new KB:
  – Cat(Kitty) ∧ Orange(Kitty) ⇒ Cute(Kitty)
  – Cat(Katy) ∧ Orange(Katy) ⇒ Cute(Katy)
  – Cat(Kitty)
  – Sister(Kitty, Katy)

• The new KB is in propositional form. The symbols are
  – Cat(Kitty), Cat(Katy), Orange(Kitty), Cute(Katy), Sister(Kitty,Katy), …
Reduction Continued

- Every FOL KB can be propositionalized
  - Transformed into propositional logic

- This preserves entailment
  - A ground sentence is entailed by the new KB if and only if it was entailed in the original KB

- Thus we can apply resolution (sound and complete) and return the result?
Reduction Continued

• **Problem**: Works if $\alpha$ is entailed by the KB but it loops forever if $\alpha$ is not entailed

• **Theorem**: (Turing 1936, Church 1936) Entailment in FOL is semi-decidable.
  - Algorithms exist that say yes to every entailed sentence
  - No algorithm exists that says no to every unentailed sentence
• **Theorem**: (Turing 1936, Church 1936) Entailment in FOL is semi-decidable.

• **Proof Intuition**
  - Can write infinitely expandable statements.
  - Even with IDS, never know when to stop expanding and give up.
Can we reason with FOL?

- Problem is with universal instantiation
  - Generates many irrelevant sentences due to substitutions
- Workaround: Unification (See R&N 9.2)
Conclusion

• Syntax, semantics, entailment and inference
• Propositional logic and FOL
• Understand how forward-chaining, backward-chaining and resolution work