THREE LOGICIANS WALK INTO A BAR...

DOES EVERYONE WANT BEER?

I DON'T KNOW.

I DON'T KNOW.

YES!
Knowledge-based agents

Knowledge base (KB) = set of sentences in a formal language

Declarative approach to building an agent (or other system):
- Tell it what it needs to know
- Then it can ask itself what to do - answers should follow from the KB

Distinction between data and program

Fullest realization of this philosophy was in the field of expert systems or knowledge-based systems in the 1970s and 1980s
What is logic?

• **Logic** is a formal system for manipulating facts so that true conclusions may be drawn
  – “The tool for distinguishing between the true and the false” – Averroes (12th cen.)

• **Syntax:** rules for constructing valid sentences
  – E.g., $x + 2 \geq y$ is a valid arithmetic sentence, $\geq x2y +$ is not

• **Semantics:** “meaning” of sentences, or relationship between logical sentences and the real world
  – Specifically, semantics defines truth of sentences
  – E.g., $x + 2 \geq y$ is true in a world where $x = 5$ and $y = 7$
Overview

- Propositional logic
- Inference rules and theorem proving
- First order logic
Propositional logic: Syntax

• **Atomic sentence:**
  – A *proposition symbol* representing a true or false statement

• **Negation:**
  – If $P$ is a sentence, $\neg P$ is a sentence

• **Conjunction:**
  – If $P$ and $Q$ are sentences, $P \land Q$ is a sentence

• **Disjunction:**
  – If $P$ and $Q$ are sentences, $P \lor Q$ is a sentence

• **Implication:**
  – If $P$ and $Q$ are sentences, $P \Rightarrow Q$ is a sentence

• **Biconditional:**
  – If $P$ and $Q$ are sentences, $P \iff Q$ is a sentence

• $\neg, \land, \lor, \Rightarrow, \iff$ are called *logical connectives*
Propositional logic: Semantics

- A **model** specifies the true/false status of each proposition symbol in the knowledge base
  - E.g., $P$ is true, $Q$ is true, $R$ is false
  - With three symbols, there are 8 possible models, and they can be enumerated exhaustively

- Rules for evaluating truth with respect to a model:
  
  $\neg P$ is true iff $P$ is false
  $P \land Q$ is true iff $P$ is true and $Q$ is true
  $P \lor Q$ is true iff $P$ is true or $Q$ is true
  $P \Rightarrow Q$ is true iff $P$ is false or $Q$ is true
  $P \Leftrightarrow Q$ is true iff $P \Rightarrow Q$ is true and $Q \Rightarrow P$ is true
Truth tables

- A **truth table** specifies the truth value of a composite sentence for each possible assignments of truth values to its atoms.

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$\neg P$</th>
<th>$P \land Q$</th>
<th>$P \lor Q$</th>
<th>$P \Rightarrow Q$</th>
<th>$P \Leftrightarrow Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

- The truth value of a more complex sentence can be evaluated *recursively* or *compositionally*.
Logical equivalence

- Two sentences are logically equivalent iff they are true in the same models.

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) & \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) & \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv (\alpha \land (\beta \land \gamma)) & \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv (\alpha \lor (\beta \lor \gamma)) & \text{associativity of } \lor \\
\neg(\neg \alpha) & \equiv \alpha & \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \beta \Rightarrow \neg \alpha) & \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \alpha \lor \beta) & \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) & \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv (\neg \alpha \lor \neg \beta) & \text{de Morgan} \\
\neg(\alpha \lor \beta) & \equiv (\neg \alpha \land \neg \beta) & \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) & \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) & \text{distributivity of } \lor \text{ over } \land
\end{align*}
\]
Validity, satisfiability

A sentence is **valid** if it is true in **all** models,
e.g., *True, A ∨¬A, A ⇒ A, (A ∧ (A ⇒ B)) ⇒ B*

A sentence is **satisfiable** if it is true in **some** model
e.g., *A∨B, C*

A sentence is **unsatisfiable** if it is true in **no** models
e.g., *A∧¬A*
Entailment

- **Entailment** means that a sentence follows from the premises contained in the knowledge base:

\[ KB \models \alpha \]

- Knowledge base \( KB \) entails sentence \( \alpha \) if and only if \( \alpha \) is true in all models where \( KB \) is true
  - E.g., \( x = 0 \) entails \( x \times y = 0 \)
  - Can \( \alpha \) be true when \( KB \) is false?

- \( KB \models \alpha \) iff \((KB \Rightarrow \alpha)\) is **valid**

- \( KB \models \alpha \) iff \((KB \land \neg \alpha)\) is **unsatisfiable**
Inference

- **Logical inference:** a procedure for generating sentences that follow from a knowledge base KB

- An inference procedure is **sound** if whenever it derives a sentence $\alpha$, $KB \models \alpha$
  - A sound inference procedure can derive only true sentences

- An inference procedure is **complete** if whenever $KB \models \alpha$, $\alpha$ can be derived by the procedure
  - A complete inference procedure can derive every entailed sentence
Inference

• How can we check whether a sentence $\alpha$ is entailed by KB?
• How about we enumerate all possible models of the KB (truth assignments of all its symbols), and check that $\alpha$ is true in every model in which KB is true?
  – Is this sound?
  – Is this complete?
• Problem: if KB contains $n$ symbols, the truth table will be of size $2^n$
• Better idea: use inference rules, or sound procedures to generate new sentences or conclusions given the premises in the KB
Inference rules

• Modus Ponens

\[
\alpha \Rightarrow \beta, \alpha \\
\beta
\]

premises

• And-elimination

\[
\alpha \wedge \beta \\
\alpha
\]
Inference rules

• And-introduction

\[
\begin{align*}
\alpha, \beta \\
\hline
\alpha \land \beta
\end{align*}
\]

• Or-introduction

\[
\begin{align*}
\alpha \\
\hline
\alpha \lor \beta
\end{align*}
\]
Inference rules

• Double negative elimination

\[ \neg\neg\alpha \]
\[ \alpha \]

• Unit resolution

\[ \alpha \lor \beta, \neg\beta \]
\[ \alpha \]
Resolution

\[
\frac{\alpha \lor \beta, \neg \beta \lor \gamma}{\alpha \lor \gamma} \quad \text{or} \quad \frac{\alpha \lor \beta, \beta \Rightarrow \gamma}{\alpha \lor \gamma}
\]

• Example:
  \(\alpha\): “The weather is dry”
  \(\beta\): “The weather is rainy”
  \(\gamma\): “I carry an umbrella”
Resolution is complete

\[
\alpha \lor \beta, \neg \beta \lor \gamma
\]

\[
\alpha \lor \gamma
\]

- To prove \( KB \models \alpha \), assume \( KB \land \neg \alpha \) and derive a contradiction
- Rewrite \( KB \land \neg \alpha \) as a conjunction of clauses, or disjunctions of literals
  - *Conjunctive normal form* (CNF)
- Keep applying resolution to clauses that contain *complementary literals* and adding resulting clauses to the list
  - If there are no new clauses to be added, then \( KB \) does not entail \( \alpha \)
  - If two clauses resolve to form an *empty clause*, we have a contradiction and \( KB \models \alpha \)
Complexity of inference

• Propositional inference is co-NP-complete
  – Complement of the SAT problem: $\alpha \models \beta$ if and only if the sentence $\alpha \land \neg \beta$ is unsatisfiable
  – Every known inference algorithm has worst-case exponential running time

• Efficient inference possible for restricted cases
Definite clauses

• A **definite clause** is a disjunction with exactly one positive literal

• Equivalent to \((P_1 \land \ldots \land P_n) \Rightarrow Q\)

• Basis of logic programming (Prolog)

• Efficient (linear-time) complete inference through *forward chaining* and *backward chaining*
Forward chaining

- Idea: find any rule whose premises are satisfied in the $KB$, add its conclusion to the $KB$, and keep going until query is found

\[
P \Rightarrow Q \\
L \land M \Rightarrow P \\
B \land L \Rightarrow M \\
A \land P \Rightarrow L \\
A \land B \Rightarrow L \\
A \\
B
\]
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Backward chaining

Idea: work backwards from the query $q$:
  to prove $q$ by BC,
    check if $q$ is known already, or
  prove by BC all premises of some rule concluding $q$
Backward chaining example

[Diagram of a logic tree with nodes labeled P, Q, M, L, A, and B, illustrating the backward chaining process.]
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Forward vs. backward chaining

- Forward chaining is **data-driven**, automatic processing
  - May do lots of work that is irrelevant to the goal

- Backward chaining is **goal-driven**, appropriate for problem-solving
  - Complexity can be **much less** than linear in size of KB
Summary

• Logical agents apply inference to a knowledge base to derive new information and make decisions

• Basic concepts of logic:
  – syntax: formal structure of sentences
  – semantics: truth of sentences wrt models
  – entailment: necessary truth of one sentence given another
  – inference: deriving sentences from other sentences
  – soundness: derivations produce only entailed sentences
  – completeness: derivations can produce all entailed sentences

• Resolution is complete for propositional logic

• Forward, backward chaining are linear-time, complete for definite clauses