Reminders:

- Project 2 due on Thursday, March 22. Tournament will be March 27.
- Research paper outline with references due March 22.

Questions?
Chapter 7 – Logical Agents
- Propositional Logic
- Propositional Theorem Proving
Simple WW KB

- Review: Define propositional symbols:
  - $P_{x,y}$ 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 agent perceives a breeze in $[x,y]$.
  - $S_{x,y}$ is true if agent perceives a stench is $[x,y]$.
Simple WW KB

- (Typos from last lecture corrected.) Write sentences based on initial knowledge.
  - $R_1: \neg P_{1,1}$ ([1,1] has no pit)
  - $R_2: B_{1,1} \iff P_{1,2} \lor P_{2,1}$ (meaning of Breeze percept)
  - $R_3: B_{2,1} \iff P_{1,1} \lor P_{2,2} \lor P_{3,1}$

- Write sentences based on initial perceives.
  - $R_4: \neg B_{1,1}$ (no breeze in [1,1])
  - $R_5: B_{2,1}$ (breeze in [2,1])
**Simple Inference for WW KB**

- Can use truth table for model checking KB

<table>
<thead>
<tr>
<th>$B_{1,1}$</th>
<th>$B_{2,1}$</th>
<th>$P_{1,1}$</th>
<th>$P_{1,2}$</th>
<th>$P_{2,1}$</th>
<th>$P_{2,2}$</th>
<th>$P_{3,1}$</th>
<th>$KB$</th>
<th>$\alpha_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
Simple Inference for WW KB

- Truth of KB is truth of $R_1 \land R_2 \land R_3 \land R_4 \land R_5$

- For each possible combination of propositions, (i.e. models) determine if KB is true. E.g. for first line of table:
  
  $R_1$ is _______, $R_2$ is _______,
  
  $R_3$ is _______, $R_4$ is _______,
  
  $R_5$ is _______, so KB = _______.

- Similarly, for middle lines, all sentences are true, so KB is true for those models.
TT-Entails?

Receives: KB, knowledge base; α, a sentence
Returns: true if KB ⊨ α, false if not

1. symbols = list of all propositional symbols in KB and α
2. Return TT-Check-All (KB, α, symbols, { })
TT-Check-All

Receives: KB, \( \alpha \), symbols, model
Returns: boolean

1. If Empty? (symbols) then
   1.1 If PL-True? (KB, model) then
      1.1.1 Return PL-True? (\( \alpha \), model)
   1.2 Else Return true // always when KB is false

2. Else
   2.1 P = First (symbols)
   2.2 rest = Rest (symbols)
   2.3 Return TT-Check-All (KB, \( \alpha \), rest, model \( \cup \) \{ P = true \}) AND TT-Check-All (KB, \( \alpha \), rest, model \( \cup \) \{ P, false \})
Model Checking

- Model checking in this way is **sound**, it implements the definition of entailment directly, and it is **complete**, it works for any KB and $\alpha$, and always terminates since there are a finite number of models to examine.

- Unfortunately, for $n$ symbols in KB and $\alpha$, there are $2^n$ models, so time complexity is $O(2^n)$. *Every known inference algorithm for general propositional logic has worst-case complexity that is exponential in size of the input.*
Entailment can be done by *theorem proving* – applying rules of inference directly to the sentences in KB.

*Logical equivalence*: sentences $\alpha$ and $\beta$ are logically equivalent if they are true in the same models. Written $\alpha \equiv \beta$. An alternate definition is: any two sentences $\alpha$ and $\beta$ are equivalent if each of them entails the other:

$$\alpha \equiv \beta \text{ if and only if } \alpha \models \beta \text{ and } \beta \models \alpha$$
Logical Equivalence

- Some standard equivalences:

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

- A sentence is **valid** if it is true in **all** models, i.e., it is logically equivalent to proposition *True*. E.g. $P \lor \neg P$ is valid. Also known as a **tautology** – it is necessarily true.

- Use validity in the **deduction theorem**: 
  
  For any sentences $\alpha$ and $\beta$, $\alpha \models \beta$ if and only if the sentence $(\alpha \Rightarrow \beta)$ is valid.

- Allows checking for $\alpha \models \beta$ by checking that $(\alpha \Rightarrow \beta)$ is true in every model.
A sentence is **satisfiable** if it is true in, or satisfied by, some model. (Note determining this is an NP-complete problem.) E.g., WW KB is satisfiable, since there are 3 models in which it is true. CSPs are satisfiability problems.

Validity and satisfiability are related:

- $\alpha$ is valid iff $\neg \alpha$ is unsatisfiable
- $\alpha$ is satisfiable iff $\neg \alpha$ is not valid
- $\alpha \models \beta$ if and only if sentence $(\alpha \land \neg \beta)$ is unsatisfiable
Satisfiability

- Leads to *proof by contradiction*. Assume $\beta$ to be false and show this leads to a contradiction with known axioms $\alpha$, which is exactly the meaning of saying sentence $(\alpha \land \neg \beta)$ is unsatisfiable.
Inference Rules

- Inference rules are used to derive a proof.
- Modus Ponens: \( \alpha \Rightarrow \beta, \alpha \)
  \[ \frac{}{\beta} \]
- And-Elimination: \( \alpha \land \beta \)
  \[ \frac{\alpha}{\alpha} \]
- Logical inferences, e.g., biconditional elimination:
  \[
  \frac{
  \alpha \iff \beta \\
  (\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)
  }{
  \alpha \iff \beta
  }
  \]
Recall: Simple WW KB

- Write sentences based on initial knowledge.
  - \( R_1 : \neg P_{1,1} \) ([1,1] has no pit)
  - \( R_2 : B_{1,1} \iff P_{1,2} \lor P_{2,1} \) (meaning of Breeze percept)
  - \( R_3 : B_{2,1} \iff P_{1,1} \lor P_{2,2} \lor P_{3,1} \)

- Write sentences based on initial percepts.
  - \( R_4 : \neg B_{1,1} \) (no breeze in [1,1])
  - \( R_5 : B_{2,1} \) (breeze in [2,1])
Inference in WW

- Generally, look for a way to get one of the single literal sentences as the left side of an implication with the goal sentence on the right. E.g. to prove [1,2] does not have a pit, try to get $R_4 : \neg B_{1,1}$ on the left and $\neg P_{1,2}$ on the right.

- Start by applying biconditional elimination to $R_2$
  
  - $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

- Apply And-Elimination
Inference in WW

- $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
- $R_7 : ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
- Apply contra-positive rule
Inference in WW

- \( R_6 : (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \)
- \( R_7 : ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \)
- \( R_8 : (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \lor P_{2,1})) \)

- Apply Modus Ponens with \( R_4 \)
Inference in WW

- \( R_6 : (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \)
- \( R_7 : ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \)
- \( R_8 : (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \lor P_{2,1})) \)
- \( R_9 : \neg (P_{1,2} \lor P_{2,1}) \)
- Apply DeMorgan's rule
Inference in WW

- $R_6: (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
- $R_7: ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
- $R_8: (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \lor P_{2,1}))$
- $R_9: \neg(P_{1,2} \lor P_{2,1})$
- $R_{10}: \neg P_{1,2} \land \neg P_{2,1}$

- Neither [1,2] nor [2,1] contain a pit by And-Elimination.
Inference as Search Problem

- Can apply any of Chapter 3 search algorithms to find a sequence of steps that constitutes a proof. Define a proof problem:
  - Initial state: initial KB
  - Actions: all the inference rules applied to all the sentences that match the top half of inference rule
  - Result: add sentence in the bottom half of inference rule
  - Goal: a state that contains the sentence we are trying to prove.
Inference as Search Problem

- Often finding a proof is more efficient than model checking, since it can ignore irrelevant propositions, no matter how many there are.
- For example, the previous proof does not mention $B_{2,1}$, $P_{1,1}$, $P_{2,2}$, or $P_{3,1}$.
- Logic systems have the property of **monotonicity**, which says adding information to KB can only increase the set of entailed sentences: if $\text{KB} \models \alpha$ then $\text{KB} \land \beta \models \alpha$. 
Proof by Resolution

- The inference rules covered so far are sound, but not complete if the available inference rules are inadequate.

- The *resolution* inference rule yields a complete inference algorithm when coupled with any complete search algorithm.
Agent has arrived at [1,2] and perceived a stench but no breeze.

Add facts to KB:

- $R_{11} : \neg B_{1,2}$
- $R_{12} : B_{1,2} \iff P_{1,1} \lor P_{2,2} \lor P_{1,3}$

Derive: $R_{13} : \neg P_{2,2}$ and $R_{14} : \neg P_{1,3}$
• Derive using biconditional elimination and Modus Ponens to get:
  $$R_{15} : P_{1,1} \lor P_{2,2} \lor P_{3,1}$$

• Apply the **resolution** rule: the literal $$\neg P_{2,2}$$ in $$R_{13}$$ resolves with the literal $$P_{2,2}$$ in $$R_{15}$$ to give **resolvent**
  $$R_{16} : P_{1,1} \lor P_{3,1}$$
Similarly, \( \neg P_{1,1} \) from \( R_1 \) resolves with \( P_{1,1} \) in \( R_{16} \) to give:

- \( R_{17} : P_{3,1} \) (There is a pit in [3,1])

- Last two steps use the *unit resolution* rule:

\[
\frac{l_1 \lor \ldots \lor l_k, \ m}{l_1 \lor \ldots \lor l_{i-1} \lor l_i \lor l_{i+1} \lor \ldots \lor l_k}
\]

where each \( l \) is a literal and \( l_i \) and \( m \) are *complimentary literals*.
Resolution

- Unit resolution rule takes a **clause** – a disjunction of literals – and a literal (**unit clause**) and produces a new clause.

- Generalize to the full **resolution rule**:

\[
\begin{array}{c}
\begin{array}{c}
l_1 \lor \cdots \lor l_k, \\
m_1 \lor \cdots \lor m_k
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
l_{i-1} \lor l_i \lor l_{i+1} \lor \cdots \lor l_k \\
m_{j-1} \lor m_j \lor m_{j+1} \lor \cdots \lor m_k
\end{array}
\end{array}
\]

where \( l_i \) and \( m_j \) are complementary literals.
Resolution

- A resolution-based theorem prover is both sound and complete. That is, for any sentences $\alpha$ and $\beta$ in propositional logic, it can decide whether $\alpha \models \beta$.

- Algorithm is relatively straightforward, but sentences are required to be in *conjunctive normal form* (CNF). CNF sentences are a *conjunction of clauses* / disjunctions. Every sentence in propositional logic is equivalent to a CNF sentence. Prove $(KB \land \neg \alpha)$ is unsatisfiable.
PL-Resolution

Receives: KB; \( \alpha \) – query sentence
Returns: true if KB \( \models \alpha \); false otherwise

1. clauses = set of CNF clauses of KB \( \land \neg \alpha \)
2. new = {} 
3. Loop
   3.1 For each pair \( C_i, C_j \) in clauses do
      3.1.1 resolvents = PL-Resolve \((C_i, C_j)\)
      3.1.2 If resolvents contains empty clause then return true
      3.1.3 new = new U resolvents
   3.2 If new \( \subseteq \) clauses then return false
   3.3 clauses = clauses U new
Using resolution to prove "no pit in \([1,2]\)". Clauses of WW KB in CNF and negated goal literal \((P_{1,2})\).

Resolution leads to empty clause, so goal literal is true.
Definite Clauses

- Restricted form of sentences can enable more restricted and efficient inference algorithms.

- **Definite clause** is a disjunction of literals of which **exactly one is positive**. E.g. \((\neg P_{1,1} \lor \neg \text{Breeze} \lor B_{1,1})\)

- **Horn clause** is a disjunction of literals of which **at most one is positive**. Clauses without any positive literals are called **goal clauses**.
Definite Clauses

- KB's containing only definite clauses are interesting for three reasons:
  - Every definite clause can be written as an implication. E.g. \((\neg P_{1,1} \lor \neg \text{Breeze} \lor B_{1,1})\) can be written as \((P_{1,1} \lor \neg \text{Breeze}) \Rightarrow B_{1,1}\). In Horn form, the premise is called the body and the conclusion is called the head. A sentence consisting of single positive literal, such as \(B_{1,1}\), is called a fact.
Definite Clauses

- Inference with Horn clauses can be done through *forward-chaining* or *backward-chaining* algorithms. These algorithms are intuitive and the basis of logic programming.

- Deciding entailment with Horn clauses can be done in time that is *linear* in the size of the KB.
In forward-chaining, start with facts and apply Modus Ponens to KB, adding new clauses until goal clause is generated, or no new clauses. This is *data-driven reasoning*.
Backward-Chaining

- Backward chaining starts with the goal and looks for clauses with the goal as the consequence. This is repeated until known facts are reached. Essentially this is the AND-OR-Graph-Search algorithm from Chapter 4.
- Backward chaining is an example of goal-driven reasoning.
- Both chaining methods have efficient algorithms that run in linear time. Often backwards-chaining is much less.