Review of Last Time

- $|= \text{ means “logically follows”}$
- $|-_i \text{ means “can be derived from”}$
- If your inference algorithm derives only things that follow logically from the KB, the inference is sound
- If everything that follows logically from the KB can be derived using your inference algorithm, the inference is complete
Outline

1. Propositional Logic: Syntax and Semantics
2. Reasoning Patterns in Propositional Logic
Syntax: Backus-Naur Form grammar of sentences in propositional logic
Sentence → AtomicSentence | ComplexSentence
AtomicSentence → True | False | Symbol
Symbol → P | Q | R | …
ComplexSentence → ¬ Sentence
| ( Sentence ∧ Sentence )
| ( Sentence ∨ Sentence )
| ( Sentence ⇒ Sentence )
| ( Sentence ⇔ Sentence )

Atomic Sentences

- The indivisible syntactic elements
- Consist of a single propositional symbol eg. P, Q, R that stands for a proposition that can be true or false eg. P=true, Q=false
- We also call an atomic sentence a literal
- 2 special propositional symbols:
  – True (the always true proposition)
  – False (the always false proposition)
Complex Sentences

- Made up of sentences (either complex or atomic)
- 5 common logical connectives:
  - ¬ (not): negates a literal
  - ∧ (and): conjunction eg. P ∧ Q where P and Q are called the conjuncts
  - ∨ (or): disjunction eg. P ∨ Q where P and Q are called the disjuncts
  - ⇒ (implies): eg. P ⇒ Q where P is the premise/antecedent and Q is the conclusion/consequent
  - ⇔ (if and only if): eg. P ⇔ Q is a biconditional

Precedence of Connectives

- In order of precedence, from highest to lowest: ¬, ∧, ∨, ⇒, ⇔
- Eg. ¬P ∨ Q ∧ R ⇒ S is equivalent to ((¬P) ∨ (Q ∧ R)) ⇒ S
- You can rely on the precedence of the connectives or use parentheses to make the order explicit
- Parentheses are necessary if the meaning is ambiguous
Semantics (Are sentences true?)

- Defines the rules for determining if a sentence is true with respect to a particular model
- For example, suppose we have the following model: P=true, Q=false, R=true
- Is \((P \land Q \land R)\) true?

For atomic sentences:
- True is true, False is false
- A symbol has its value specified in the model

For complex sentences (for any sentence \(S\) and model \(m\)):
- \(\neg S\) is true in \(m\) iff \(S\) is false in \(m\)
- \(S_1 \land S_2\) is true in \(m\) iff \(S_1\) is true in \(m\) and \(S_2\) is true in \(m\)
- \(S_1 \lor S_2\) is true in \(m\) iff \(S_1\) is true in \(m\) or \(S_2\) is true in \(m\)
- \(S_1 \Rightarrow S_2\) is true in \(m\) iff \(S_1\) is false in \(m\) or \(S_2\) is true in \(m\)
  i.e., can translate it as \(\neg S_1 \lor S_2\)
- \(S_1 \Leftrightarrow S_2\) is true iff \(S_1 \Rightarrow S_2\) is true in \(m\) and \(S_2 \Rightarrow S_1\) is true in \(m\)
Note on implication

• P ⇒ Q seems weird...doesn’t fit intuitive understanding of “if P then Q”
• Propositional logic does not require causation or relevance between P and Q
• Implication is true whenever the antecedent is false (remember P ⇒ Q can be translated as ¬P ∨ Q)
  – Implication says “if P is true, then I am claiming that Q is true. Otherwise I am making no claim”
  – The only way for this to be false is if P is true but Q is false

Truth Tables for the Connectives

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>Q</th>
<th>P ∧ Q</th>
<th>P ∨ Q</th>
<th>P ⇒ Q</th>
<th>P ⇔ Q</th>
</tr>
</thead>
<tbody>
<tr>
<td></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>true</td>
<td>false</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

With the truth tables, we can compute the truth value of any sentence with a recursive evaluation eg.

Suppose the model is P=false, Q=false, R=true

¬P ∧ (Q ∨ R) = true ∧ (false ∨ true) = true ∧ true = true
The Wumpus World KB (only dealing with knowledge about pits)

For each $i, j$:
Let $P_{i,j}$ be true if there is a pit in $[i, j]$
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$

The KB contains the following sentences:
1. There is no pit in $[1,1]$:
   $R_1: \neg P_{1,1}$
2. A square is breezy iff there is a pit in a neighboring square: (not all sentences are shown)
   $R_2: B_{1,1} \iff P_{1,2} \lor P_{2,1}$
   $R_3: B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

3. We add the percepts for the first two squares ($[1,1]$ and $[2,1]$) visited in the Wumpus World example:
   $R_4: \neg B_{1,1}$
   $R_5: B_{2,1}$

The KB is now a conjunction of sentences $R_1 \land R_2 \land R_3 \land R_4 \land R_5$ because all of these sentences are asserted to be true.
Inference

- How do we decide if KB |= α?
- Enumerate the models, check that α is true in every model in which KB is true

<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>R_{1}</th>
<th>R_{2}</th>
<th>R_{3}</th>
<th>R_{4}</th>
<th>R_{5}</th>
<th>KB</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>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</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>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</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>true</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>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

Inference

- Suppose we want to know if KB |= ¬P_{1,2}?
- In the 3 models in which KB is true, ¬P_{1,2} is also true

<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>R_{1}</th>
<th>R_{2}</th>
<th>R_{3}</th>
<th>R_{4}</th>
<th>R_{5}</th>
<th>KB</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>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</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>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</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>true</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>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

16
Complexity

• If the KB and $\alpha$ contain $n$ symbols in total, what is the time complexity of the truth table enumeration algorithm?

• Space complexity is $O(n)$ because the actual algorithm uses DFS

The really depressing news

• Every known inference algorithm for propositional logic has a **worse-case** complexity that is **exponential** in the size of the input

  You can’t handle the truth!

• But some algorithms are more efficient in **practice**
Logical equivalence

• Intuitively: two sentences $\alpha$ and $\beta$ are logically equivalent (ie. $\alpha \equiv \beta$) if they are true in the same set of models

• Formally: $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

• Can prove this with truth tables

Standard Logic Equivalences

$(\alpha \land \beta) \equiv (\beta \land \alpha)$ commutativity of $\land$
$(\alpha \lor \beta) \equiv (\beta \lor \alpha)$ commutativity of $\lor$
$((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma))$ associativity of $\land$
$((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma))$ associativity of $\lor$
$\neg(\neg \alpha) \equiv \alpha$ double-negation elimination
$(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha)$ contraposition
$(\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta)$ implication elimination
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))$ biconditional elimination
$\neg(\neg \alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta)$ de Morgan
$\neg(\neg \alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta)$ de Morgan
$(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma))$ distributivity of $\land$ over $\lor$
$(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma))$ distributivity of $\lor$ over $\land$

In the above, $\alpha$, $\beta$, and $\gamma$ are arbitrary sentences of propositional logic
Validity

• A sentence is valid if it is true in all models
• eg. P ∨ ¬P is valid
• Valid sentences = Tautologies
• Tautologies are vacuous

Deduction theorem
For any sentences α and β, α |= β iff the sentence (α ⇒ β) is valid

Satisfiability

• A sentence is satisfiable if it is true in some model.
• A sentence is unsatisfiable if it is true in no models
• Determining the satisfiability of sentences in propositional logic was the first problem proved to be NP-complete
• Satisfiability is connected to validity:
  α is valid iff ¬α is unsatisfiable
• Satisfiability is connected to entailment:
  α |= β iff the sentence (α ∧ ¬β) is unsatisfiable
  (proof by contradiction)
Reasoning Patterns in Propositional Logic

Proof methods

How do we prove that $\alpha$ can be entailed from the KB?
1. Model checking eg. check that $\alpha$ is true in all models in which KB is true
2. Inference rules
Inference Rules

1. Modus Ponens
   \[ \frac{\alpha \implies \beta, \quad \alpha}{\beta} \]

2. And-Elimination
   \[ \frac{\alpha \land \beta}{\alpha} \]

These are both sound inference rules. You don’t need to enumerate models now

Other Inference Rules

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

All of the logical equivalences can be turned into inference rules eg.

\[ \frac{\alpha \iff \beta}{(\alpha \implies \beta) \land (\beta \implies \alpha)} \]
Example

Given the following KB, can we prove \( \neg R \)?

KB:

\[
P \Rightarrow \neg (Q \lor R)
\]

P

Proof:

\( \neg (Q \lor R) \) by Modus Ponens

\( \neg Q \land \neg R \) by De Morgan’s Law

\( \neg R \) by And-Elimination

Proofs

- A sequence of applications of inference rules is called a proof
- Instead of enumerating models, we can search for proofs
- Proofs ignore irrelevant propositions
- 2 methods:
  - Go forward from initial KB, applying inference rules to get to the goal sentence
  - Go backward from goal sentence to get to the KB
Monotonicity

• Proofs only work because of monotonicity
• Monotonicity: the set of entailed sentences can only increase as information is added to the knowledge base
• For any sentences $\alpha$ and $\beta$, if $\text{KB} \models \alpha$ then $\text{KB} \land \beta \models \alpha$

Resolution

• An inference rule that is sound and complete
• Forms the basis for a family of complete inference procedures
• Here, complete means refutation completeness: resolution can refute or confirm the truth of any sentence with respect to the KB
Resolution

• Here’s how resolution works ($\neg l_2$ and $l_2$ are called complementary literals):

\[
\begin{align*}
  l_1 \lor l_2, & \quad \neg l_2 \lor l_3 \\
  \hline
  l_1 \lor l_3
\end{align*}
\]

• Note that you need to remove multiple copies of literals (called factoring) ie.

\[
\begin{align*}
  l_1 \lor l_2, & \quad \neg l_2 \lor l_1 \\
  \hline
  l_1
\end{align*}
\]

• If $l_i$ and $m_j$ are complementary literals, the full resolution rule looks like:

\[
\begin{align*}
  l_1 \lor \cdots \lor l_k, & \quad m_1 \lor \cdots \lor m_n \\
  \hline
  l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n
\end{align*}
\]

Conjunctive Normal Form

• Resolution only applies to sentences of the form $l_1 \lor l_2 \lor \cdots \lor l_k$

• This is called a disjunction of literals

• It turns out that every sentence of propositional logic is logically equivalent to a conjunction of disjunction of literals

• Called Conjunctive Normal Form or CNF

eg. $(l_1 \lor l_2 \lor l_3 \lor l_4) \land (l_5 \lor l_6 \lor l_7 \lor l_8) \land \ldots$

• k-CNF sentences have exactly k literals per clause

eg. A 3-CNF sentence would be $(l_1 \lor l_2 \lor l_3) \land (l_4 \lor l_5 \lor l_6) \land (l_7 \lor l_8 \lor l_9)$
Recipe for Converting to CNF

1. Eliminate $\leftrightarrow$, replacing $\alpha \leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$
2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg \alpha \lor \beta$
3. Move $\neg$ inwards using:
   - $\neg(\neg \alpha) \equiv \alpha$ (double-negation elimination)
   - $\neg(\alpha \land \beta) \equiv \neg \alpha \lor \neg \beta$ (De Morgan’s Law)
   - $\neg(\alpha \lor \beta) \equiv \neg \alpha \land \neg \beta$ (De Morgan’s Law)
4. Apply distributive law $(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma))$

A resolution algorithm

To prove $\text{KB} \models \alpha$, we show that $(\text{KB} \land \neg \alpha)$ is unsatisfiable
(Remember that $\alpha \models \beta$ iff the sentence $(\alpha \land \neg \beta)$ is unsatisfiable)

The algorithm:
1. Convert $(\text{KB} \land \neg \alpha)$ to CNF
2. Apply resolution rule to resulting clauses. Each pair with complementary literals is resolved to produce a new clause which is added to the KB
3. Keep going until
   - There are no new clauses that can be added (meaning $\text{KB} \not\models \alpha$)
   - Two clauses resolve to yield the empty clause (meaning $\text{KB} \models \alpha$)

The empty clause is equivalent to false because a disjunction is true only if one of its disjuncts is true
Resolution Pseudocode

function PL-RESOLUTION(KB, α) returns true or false
    clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
    new ← {} 
    loop do
        for each Cᵢ, Cⱼ in clauses do
            resolvents ← PL-RESOLVE(Cᵢ, Cⱼ)
            if resolvents contains the empty clause then return true
            new ← new ∪ resolvents
            if new ⊆ clauses then return false
        clauses ← clauses ∪ new
    end loop
end function

Things you should know

• Understand the syntax and semantics of propositional logic
• Know how to do a proof in propositional logic using inference rules
• Know how resolution works
• Know how to convert arbitrary sentences to CNF
In-class Exercise

KB

Person $\Rightarrow$ Mortal
Socrates $\Rightarrow$ Person

Can we show that :

$KB \models (\text{Socrates} \Rightarrow \text{Mortal})$?

In-class Exercise

| If it is October, there will not be a football game at OSU |
| If it is October and it is Saturday, I will be in Corvallis |
| If it doesn’t rain or if there is a football game, I will ride my bike to OSU |
| Today is Saturday and it is October |
| If I am in Corvallis, it will not rain |

Can you prove that I will ride my bike to OSU?