“I know what you’re thinking about,” said Tweedledum; “but it isn’t so, nohow.”

“Contrariwise,” continued Tweedledee, “if it was so, it might be; and if it were so, it would be; but as it isn’t, it ain’t. That’s logic.”

Propositional Logic

Last time

- Propositional Logic
  - Connectives (logic gates)
  - Entailment (inference)

- We used human brains to deduce new facts from those that we already knew.
  - i.e., where it’s safe to go in the Wumpus world.
Today

- A tiny review
- How do we automate inference of propositional logic?
  - Simple method:
    - Model Checking
  - Constructing proofs
  - Resolution
  - Search strategies
    - Forward Chaining
    - PL-Resolution

Five Logical Connectives

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>(\neg P) (not)</th>
<th>(P \land Q) (and)</th>
<th>(P \lor Q) (or)</th>
<th>(P \Rightarrow Q) (implies)</th>
<th>(P \Leftrightarrow Q) (if and only if)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Odd Implications

- “5 is odd” implies “Tokyo is the capital of Japan”
- “5 is even” implies “Tokyo is the capital of Japan”
- “5 is even” implies “Tokyo is the capital of the United States”

Mini Quiz

Are the following sentences:
- Valid?
- Satisfiable?

Consider these sentences (with respect to the vacuum world)
- \((D_A \lor D_B) \land \neg D_B \land D_A\) ?
- \([D_A \lor D_B] \land \neg D_B \Rightarrow D_A\) ?
- \([D_A \lor D_B] \land \neg D_B \Leftarrow D_A\) ?
Inference

- Process by which some sentences are \textit{derived} from others.
  - Aka \textit{reasoning}
  - Record of an inference process called a \textit{proof}
- Can derive $\alpha$ from KB using method $i$:

\[ \text{KB} \vdash_i \alpha \]

Properties of Inference methods

- \textbf{Soundness}

\[ \text{KB} \vdash_i \alpha \quad \text{only when} \quad \text{KB} \models \alpha \]

\textit{i.e., no false conclusions}

- \textbf{Completeness}

\[ \text{whenever} \quad \text{KB} \models \alpha \quad \text{it is true that} \quad \text{KB} \vdash_i \alpha \]

\textit{i.e., no missing conclusions}
Gottfried Wilhelm Leibnitz (1646-1716)

... if we could find characters or signs appropriate for expressing all our thoughts as definitely and as exactly as arithmetic expresses numbers..., we could in all subjects in so far as they are amenable to reasoning accomplish what is done in Arithmetic... For all inquiries... would be performed by the transposition of characters and by a kind of calculus, which would immediately facilitate the discovery of beautiful results...

—*Dissertio de Arte Combinatoria*, 1666

Inference Methods

- Now, all we need is an inference method we can implement!

- We’ll describe several!
  - Model Checking
  - Proof Trees
  - Forward Chaining
  - Backwards Chaining
Model Checking

- A generic inference mechanism
- Enumerate all models (i.e., truth assignments) and check that $\alpha$ is valid in which KB is true
  - I.e., check that $\alpha$ is valid given KB
  - Time complexity: $O(2^n)$
  - Space complexity: $O(n)$
- **Sound:**
  - Yes: directly implements the definition of entailment
- **Complete**
  - Yes: given finite KB and $\alpha$ (because there are only finitely many models to examine)

Model Checking Example

- **KB:**
  
  $$(\text{IsDog}(Fido) \lor \text{IsCat}(Fido)) \land$$
  
  $$(\text{IsCat}(Fido) \iff \text{Meows}(Fido)) \land$$
  
  $$(\neg \text{Meows}(Fido))$$

- **Does $KB \models \text{IsDog}(Fido)$?**

<table>
<thead>
<tr>
<th>IsDog(Fido)</th>
<th>IsCat(Fido)</th>
<th>Meows(Fido)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Model Checking Example

KB:

(IsDog(Fido) ∨ IsCat(Fido)) ∧
(IsCat(Fido) ⇔ Meows(Fido)) ∧
(¬ Meows(Fido))

Does KB |= IsDog(Fido) ?

<table>
<thead>
<tr>
<th>IsDog(Fido)</th>
<th>IsCat(Fido)</th>
<th>Meows(Fido)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Model Checking Example

- **KB:**
  
  \[
  (\text{IsDog}(Fido) \lor \text{IsCat}(Fido)) \land \\
  (\text{IsCat}(Fido) \iff \text{Meows}(Fido)) \land \\
  (\neg \text{Meows}(Fido))
  \]

- **KB implies IsDog(Fido)?**

<table>
<thead>
<tr>
<th>IsDog(Fido)</th>
<th>IsCat(Fido)</th>
<th>Meows(Fido)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

  - For all world models in which KB is true, IsDog(Fido) is true, so YES!
Model Checking: Hindsight

- We started this section by criticizing the belief state approach:
  - Requires enumerating every state

- Model checking enumerates every state!
  - We haven’t yet really “cashed in” on our new fancy representation of incomplete knowledge!

Proving without Model Checking

- Standard patterns of inference that can be applied to derive chains of conclusions.
- Pattern:

\[
\begin{array}{c}
\text{premise} \\
\hline
\text{conclusion}
\end{array}
\]

- If KB contains premise, can add conclusion
- Provides a different (than enumeration) way of deciding entailment: by proof
  - A proof is a sequence of applications of sound inference rules.
Why Proof?

- Big win if the desired entailment can be derived without even caring about the truth values of most of the propositions!
- In our vacuum cleaner world, we might have 3 propositions: $D_a, D_b, R_a$.
  - If we know $(D_a \lor D_b)$ and $\neg D_b$ then we should be able to conclude $D_a$ without having to consider whether $R$ is in $a$ or not!
  - We can! Proof using resolution (you’ll see!)
  - More pronounced savings with propositions about the door is open, the lights are on, the weather, etc.

Some Inference Rules

- Modus Ponens (MP) \[ \alpha \Rightarrow \beta, \alpha \quad \beta \]
- And-elimination (AE) \[ \alpha \land \beta \quad \alpha \]
- Biconditional-elim (BE) \[ \alpha \leftrightarrow \beta \quad (\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha) \quad \alpha \leftrightarrow \beta \]
- Contrapositive (CP) \[ (\alpha \Rightarrow \beta) \quad (\neg \beta \Rightarrow \neg \alpha) \]
  - Many more sound rules... (verify through truth-tables)
DeMorgan’s

- What do you have now?
  - Not of Ands: NA \( \neg(A \land B) \)
  - Not of Ors: NO \( \neg(A \lor B) \)
  - Or of Nots: ON \( \neg(A \lor \neg B) \)
  - And of Nots: AN \( \neg A \land \neg B \)

- Reverse the letters, swap the O’s for A’s.
  - NO \( \equiv \) AN
  - NA \( \equiv \) ON
  - AN \( \equiv \) NO
  - ON \( \equiv \) NA

Proof Example: Pit World

\[
\begin{array}{|c|}
\hline
Q: P_{1,2} \\
\hline
\end{array}
\]

\[
\begin{align*}
S_1: & \neg P_{1,1} \\
S_2: & B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \\
S_3: & B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \\
S_4: & \neg B_{1,1} \\
S_5: & B_{2,1} \\
\end{align*}
\]

\[
\begin{align*}
S_6: & (B_{1,1} \implies (P_{1,2} \lor P_{2,1})) \\
& \land ((P_{1,2} \lor P_{2,1}) \implies B_{1,1}) \\
\end{align*}
\]

[AE, \( S_6 \)]

\[
S_7: (P_{1,2} \lor P_{2,1}) \implies B_{1,1}
\]

[contrapositive, \( S_7 \)]

\[
S_6: \neg B_{1,1} \implies \neg (P_{1,2} \lor P_{2,1})
\]

[MP, \( S_4, S_6 \)]

\[
S_5: \neg (P_{1,2} \lor P_{2,1})
\]

[de Morgan’s, \( S_9 \)]

\[
S_{10}: \neg P_{1,2} \land \neg P_{2,1}
\]

[AE \( S_{10} \)]

\[
S_{10}: \neg P_{1,2}
\]
Proof as Search

- **State**: Set of sentences
  - Initial state: KB
- **Successors**: all possible applications of sound inference rules
- **Can use any search method**
  - But do they all make sense?
  - Can be dramatically more efficient than model checking
- **Inference in PL is NP-complete**
  - In the worst-case searching for proofs is no more efficient than enumeration

Propositional Logic Proof Search Tree
Monotonicity

- The set of entailed sentences can only increase as sentences are added to KB.
- For any sentences $\alpha$ and $\beta$

\[
\text{if } \text{KB} \models \alpha \text{ then } \text{KB} \land \beta \models \alpha
\]

- Implications
  - Conclusions from sound inference rules are never defeated by further inference.
  - Search method never needs to backtrack: no "branching."

Propositional Logic Proof Search Tree

- Anything we can do here...
- So, what we'd like to do is to apply ops early on that answer our questions, and then simply stop!

We can do here!
Soundness and Completeness

- Are proof procedures sound?
  - If inference rules are valid.
  - How would you prove validity?

- Are proof procedures complete?
  - If inference rules can derive all entailments.
  - How can we be sure we’ve got all the rules we need?

(Unit) Resolution

\[
\frac{\alpha \lor \beta, \neg \beta}{\alpha}
\]

- “logically equivalent” to modus ponens
- \((\neg b \Rightarrow a) \land \neg b\)
- Often expressed as requiring \textit{litersals}
  \[
  \frac{(c \lor a_1 \lor a_2 \lor \ldots \lor a_j) \land \neg c}{a_1 \lor a_2 \lor \ldots \lor a_j}
  \]
Unit Resolution Example

- In our vacuum cleaner world, we might have 3 propositions: $D_a$, $D_b$, $R_a$.
  - If we know $(D_a \lor D_b)$ and $\neg D_b$ then we should be able to conclude $D_a$ without having to consider whether $R$ is in $a$ or not!
  - (Unit) resolution is an inference rule that allows this conclusion!

Full Resolution Rule

$$
\frac{(c \lor a_1 \lor a_2 \lor \ldots \lor a_j) \land (\neg c \lor b_1 \lor b_2 \lor \ldots \lor b_k)}{a_1 \lor a_2 \lor \ldots \lor a_j \lor b_1 \lor b_2 \lor \ldots \lor b_k}
$$

- Resulting clause should contain only one copy of each literal
  - i.e., combine any $a$’s and $b$’s that are the same literal
Soundness of Resolution

\[(c \lor a_1 \lor a_2 \lor \ldots \lor a_j) \land (\neg c \lor b_1 \lor b_2 \lor \ldots \lor b_k)\]

\[a_1 \lor a_2 \lor \ldots \lor a_j \lor b_1 \lor b_2 \lor \ldots \lor b_k\]

- Suppose \(c\) is false. Then:
  - \(a_1 \lor a_2 \lor \ldots \lor a_j\) must be true.

- Suppose \(c\) is true. Then:
  - \(b_1 \lor b_2 \lor \ldots \lor b_k\) must be true.

- \(c\) must be either true or false. Thus:
  - Either the \(a\)'s must be true OR the \(b\)'s must be true.

Conjunctive Normal Form

- Resolution rule applies directly only to disjunctions of literals
- Every PL sentence is logically equivalent to a conjunction of disjunction of literals (a CNF sentence)
- \(k\)-CNF has exactly \(k\) literals per clause
  \[(l_{1,1} \lor \ldots \lor l_{1,k}) \land (l_{2,1} \lor \ldots \lor l_{2,k}) \land \ldots \land (l_{n,1} \lor \ldots \lor l_{n,k})\]
- Every PL sentence can be transformed into a 3-CNF sentence
Conversion to CNF

Example sentence $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$

1) Eliminate $\iff$, replacing $\alpha \iff \beta$ by $(\alpha \implies \beta) \land (\beta \implies \alpha)$

\[ (B_{1,1} \implies (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \implies B_{1,1}) \]

2) Eliminate $\implies$, replacing $\alpha \implies \beta$ by $\neg \alpha \lor \beta$

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg (P_{1,2} \lor P_{2,1}) \lor B_{1,1}) \]

3) Ensure $\neg$ applies only to literals by moving inwards

\[ (\neg (\neg \beta) \equiv \beta; \quad (\neg (\alpha \land \beta) \equiv \neg \alpha \lor \neg \beta; \quad (\neg (\alpha \lor \beta) \equiv \neg \alpha \land \neg \beta) \]

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \lor \neg P_{2,1}) \lor B_{1,1}) \]

4) Distribute $\lor$ over $\land$ wherever possible

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1}) \]

Simple Full Resolution Example

- Some rooms are dirty:
  - S1: $(D_a \lor D_b)$
- Vacuuming cleans them:
  - S2: $(V_a \implies \neg D_a)$ converted to $(\neg V_a \lor \neg D_a)$
- Resolving S1 and S2 gives us:
  - $(\neg V_a \lor D_b)$
Refutation

\[ \alpha \nvdash \alpha \lor \beta \]

- *Cannot* derive \( \alpha \lor \beta \) from \( \alpha \) using resolution rule.
- However we *can* use resolution to prove the conclusion.
- **Refutation proof:**
  - Assume negation of goal sentence, derive a contradiction (false).
  - If successful, goal sentence must be entailed.

Resolution Refutation

- Show that \( (KB \land \neg \alpha) \) is unsatisfiable.
- Convert \( (KB \land \neg \alpha) \) to CNF
- Apply resolution rule to resulting clauses.
  - Each pair that contains complementary literals is resolved to produce a new clause
  - Add to the set if it is not already present
- Continues until one of two things happen
  - There are no new clauses that can be added, in which case KB does not entail \( \alpha \)
  - An application of the resolution rule derives the empty clause, in which case KB does entail \( \alpha \)
PL Resolution

function PL-resolution (KB, α) returns true or false
inputs KB, the knowledge base, a sentence in PL
α, the query, a sentence in PL

clauses ← CNF representation of (KB ∧ ¬α)
new ← {}

loop do
  for each Ci, Cj in clauses do
    resolvents ← PL-resolve (Ci, Cj)
    if resolvents contains {} then return true
    new ← new ∪ resolvents
    if new ⊆ clauses then return false
  clauses ← clauses ∪ new

Example

Given P, prove (P ∨ Q):
- S1: P ∨ False (right?)
- S2: ¬(P ∨ Q) ⊃
  - S2a: ¬P
  - S2b: ¬Q

Resolve S1 and S2a to get
- S3: False

- If we can ever derive the empty clause, we’ve derived that “false” is entailed (as a true statement) by the KB combined with the negated sentence to prove.
- Since False can’t be true, there is a contradiction in the KB with the negated sentence.
- Since we assume the KB was correct, the contradiction must have been introduced by the negated sentence.
Example

Try to prove: \( \neg P_{1,2} \)

- \( S_1: \neg P_{1,2} \lor B_{1,1} \)
- \( S_2: \neg B_{1,1} \lor P_{1,2} \lor P_{2,1} \)
- \( S_3: \neg P_{1,2} \lor B_{1,1} \)
- \( S_4: \neg B_{1,1} \)

Empty clause

Your Turn

Prove Modus Ponens:

\[ [P \land (P \Rightarrow Q)] \Rightarrow Q \]

- Step 1: We'll prove by refutation, so negate the hypothesis.

- Step 2: Convert to CNF.

- Step 3. Find a sequence of resolutions
  - We expect to find a contradiction.

Solution:

Negate it: \( \neg [P \land (P \Rightarrow Q)] \Rightarrow Q \)

Convert to CNF:

\[ \neg (P \land (P \Rightarrow Q)) \Rightarrow Q \]

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

\[ (\neg P \land (\neg P \lor Q)) \lor \neg Q \]

\[ ((P \land (P \lor Q)) \land \neg Q) \]

\[ P \land (P \lor Q) \land \neg Q \]

\[ S_1: P \]

\[ S_2: \neg P \land Q \]

\[ S_3: \neg Q \]

Resolve \( S_1 \) and \( S_2 \) to get

\[ S_4: Q \]

Resolve \( S_3 \) and \( S_4 \) to get

\[ S_5: \text{FALSE} \]

Negation is unsatisfiable, so non-negated must be valid.
Special Cases

- All sentences *conjunctions* of prop symbols
  - No disjunction or negation allowed
    - E.g., \( A \land B \land C \land D \land \ldots \)
  - KB is a database
  - Answer queries by lookup

- All sentences of the form
  \[ P_1 \land P_2 \land \ldots \land P_n \Rightarrow Q \]
  - Horn sentences
    - At most one positive literal in CNF form. (Why is this the same as the form above?)
  - \( P_i \) a premise, \( Q \) the head
  - Forward chaining answers all queries in time linear in KB

Forward Chaining

- Maintain for each rule a *count* of unsatisfied premises
- Initialize *agenda* to the known facts
- Loop until empty *agenda*
  - \( p \leftarrow \text{Pop}(agenda) \)
  - if \( p \) not already marked as *inferred*
    - mark \( p \) as *inferred*
    - decrement *count* for rules with \( p \) as a premise
    - when *count* = 0, put rule’s head on *agenda*
Forward Chaining Example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]

Back-Chaining

KB:
\[ \neg A \lor \neg B \]
\[ B \lor \neg C \]
\[ \neg A \lor C \]

Is KB satisfiable?
Hard Problems

- At the heart of inference is **satisfiability**.
  - We’ve been doing constraint satisfaction!

- Why is n-queens easy?
  - Somehow related to the density of solutions

- Consider random 3-CNF sentences. e.g.

  \[
  (\neg D \lor \neg B \lor C) \land (B \lor \neg A \lor \neg C) \land
  (\neg C \lor \neg B \lor E) \land (E \lor \neg D \lor B) \land
  (B \lor E \lor \neg C)
  \]

  - Hard problems seem to cluster near **clauses/symbols = 4.3** (critical point)

---

Hard satisfiability problems

[Graph showing the distribution of satisfiability problems as a function of clause/symbol ratio.]
Hard satisfiability problems

- Median runtime for 100 satisfiable random 3-CNF sentences, \( n = 50 \)

Next Time

- First Order Logic
  - Enrich our language to understand objects and relations between those objects
    - Wumpus rules will be easier & less tedious to express

- See you this evening!