Artificial Intelligence Programming
Propositional Logic

Chris Brooks

Department of Computer Science
University of San Francisco
So far, we’ve talked about search, which is a means of considering alternative possibilities.

The other side of the coin involves representing these possibilities.

Our choice of representation influences:
- What sorts of environments an agent can deal with.
- The complexity of search
- The sophistication of our agent.
Logic is the classical way to represent an agent’s knowledge.

Allows us to program and describe agents *declaratively*

- In terms of relations between objects.
- Knowledge is represented as facts and relations
- Agents maintain a *knowledge base* that allows them to reason about a problem.

This is sometimes called programming at the *knowledge level*.

- Specify facts known by an agent, along with goals.
- Actions are chosen because they achieve goals.
- Implementation is abstracted away.
R & N use the Wumpus World as an example domain.

Environment: 4x4 grid of rooms.
- Gold in one room, wumpus in another
- Pits in some rooms

Actions: Move forward, turn left, turn right, shoot arrow, grab gold.

Sensors: Perceive stench, perceive breeze, perceive gold, sense wall, hear wumpus death.

Goal: maximize performance, which means finding gold quickly without encountering the wumpus or falling into a pit.
11-3: Wumpus World
A knowledge base is composed of *sentences* that assert facts about the world.

Some definitions:

- **Syntax**: Defines whether a sentence is properly constructed.
  - In arithmetic, $x + 2 = 5$ is syntactically correct, whereas $x + = 3$ is not.
  - In fox-and-chickens, $nfoxesLeft = 3$ is syntactically correct.

- **Semantics**: Defines when a sentence is true or false.
  - The semantics of $x + 2 = 5$ are that this sentence is true in worlds where $x = 3$ and false otherwise.
  - Logical sentences must be true or false; no “degree of truth”.
Model: A model is an assignment of values to each of the variables of interest in our state.

- A model for the fox-and-chickens world would indicate the values taken on by nfoxesLeft, nfoxesRight, nchickensLeft, nchickensRight, and boatPos.
- We’ll often be interested in finding models that make a sentence true or false.
Entailment: Entailment is the idea that one sentence follows logically from another.

- **Written as:** $a \models b$
- **Technically,** this says: for all models where $a$ is true, $b$ is also true.
  
  - $a + 2 = 5 \models a = 3$

Entailment will allow us to perform inference and add new facts to our agent’s knowledge base.
A knowledge base plus a model allow us to perform inference.

- For a given set of sentences, plus some assignment of values to variables, what can we conclude?

Entailment tells us that a sentence can be derived.

Inference tells us how it is derived.

An algorithm that only derives entailed sentences is said to be sound.

- Doesn’t make mistakes or conclude incorrect sentences.

An algorithm that can derive all entailed sentences is complete.

- If a sentence is entailed, a complete algorithm will eventually infer it.
Propositional logic is a very simple logic.

- Nice for examples
- Computationally feasible.
- Limited in representational power.

Terms (R & N call these atomic sentences) consist of a single symbol.

- $3\text{FoxesLeft}, \text{BoatOnRight}$

A complex sentence is a set of terms conjoined with $\lor$, $\lnot$, $\land$, $\Rightarrow$, $\Leftrightarrow$.

- $3\text{foxesLeft} \land 3\text{ChickensRight} \land \text{BoatRight}$
- $3\text{foxesLeft} \land 2\text{ChickensLeft} \land \text{BoatRight} \Rightarrow \text{ChickensEaten}$
11-9: Propositional Logic

6 \( A \land B \) - AND. sentence is true if both A and B are true.

6 \( A \lor B \) OR. Sentence is true if either A or B (or both) are true.

6 \( \neg A \) NOT. Sentence is true if A is false.

6 \( A \Rightarrow B \) Implies. Sentence is true if A is false or B is true.

6 \( A \iff B \) Equivalence. Sentence is true if A and B have the same truth value.
Implication is a particularly useful logical construct.

The *sentence* $A \Rightarrow B$ is true if:
- $A$ is true and $B$ is true.
- $A$ is false.

Example: If it is raining right now, then it is cloudy right now.

$A \Rightarrow B$ is equivalent to $\neg A \lor B$.

Implication will allow us to perform inference.
Logical equivalence: Two sentences are logically equivalent if they are true for the same set of models.

- \( P \land Q \) is logically equivalent to \( \neg(\neg P \lor \neg Q) \)

Validity (tautology): A sentence is valid if it is true for all models.

- \( A \lor \neg A \)

Contradiction: A sentence that is false in all models.

- \( A \land \neg A \)
Satisfiability: A sentence is satisfiable if it is true for some model.

- $3foxesleft \lor 2foxesleft$ is true in some worlds.
- Often our problem will be to find a model that makes a sentence true (or false).
- This will be the solution to our problem.
Logical reasoning proceeds by using existing sentences in an agent’s KB to deduce new sentences.

Rules of inference.

- Modus Ponens
  - $A, A \Rightarrow B$, conclude $B$

- And-Elimination
  - $A \land B$, conclude $A$.

- Or-introduction
  - $A$, conclude $A \lor B$
11-14: Logical Reasoning

Rules of inference.

- **Contraposition**: $A \Rightarrow B$ can be rewritten as $\neg B \Rightarrow \neg A$

- **Double negative**: $\neg(\neg A) = A$

- **Distribution**
  - $A \lor (B \land C') = (A \lor B) \land (A \lor C')$
  - $A \land (B \lor C') = (A \land B) \lor (A \land C')$

- **DeMorgan’s theorem**
  - $A \lor B$, rewrite as $\neg(\neg A \land \neg B)$
  - or $A \land B \leftrightarrow \neg(\neg A \lor \neg B)$
We can then use good old breadth-first search to perform inference and determine whether a sentence is entailed by a knowledge base.

Basic idea: Begin with statements in our KB.

Actions are applications of implication.

- For example, say we know 1) \( A \rightarrow B \), 2) \( B \Rightarrow C \), and 3) \( A \).
- One possible action is to apply Modus Ponens to 1 and 3 to conclude \( B \).
- We can then apply Modus Ponens again to conclude \( C \).
Our search can proceed in a breadth-first manner (what are all the possible conclusions from the original KB), depth-first (take one inference, then use it to make further inferences, and so on) or somewhere in-between.

The result of this search is called a *proof*. 
The preceding rules are sound, but not necessarily complete.

Luckily, there is a complete rule for inference (when coupled with a complete search algorithm).

This is called resolution.

The basic resolution rule looks like this:

- $A \lor B$ and $\neg A \lor C$ allows us to conclude $B \lor C$.
- $A$ is either true or not true. If $A$ is true, then $C$ must be true.
- If $A$ is false, then $B$ must be true.

This can be generalized to clauses of any length.
11-18: Conjunctive Normal Form

6 Resolution works with disjunctions.
6 this means that our knowledge base needs to be in this form.
6 Conjunctive Normal Form is a conjunction of clauses that are disjunctions.
6 \((A \lor B \lor C) \land (D \lor E \lor F) \land (G \lor H \lor I) \land \ldots\)
6 Every propositional logic sentence can be converted to CNF.
11-19: Conjunctive Normal Form
Recipe

1. Eliminate equivalence
   \[ A \iff B \text{ becomes } A \Rightarrow B \land B \Rightarrow A \]

2. Eliminate implication
   \[ A \Rightarrow B \text{ becomes } \neg A \lor B \]

3. Move \( \neg \) inwards using double negation and DeMorgan’s
   \[ \neg(\neg A) \text{ becomes } A \]
   \[ \neg(A \land B) \text{ becomes } (\neg A \lor \neg B) \]

4. Distribute nested clauses
   \[ (A \lor (B \land C)) \text{ becomes } (A \lor B) \land (A \lor C) \]
Once your KB is in CNF, you can do resolution by refutation.

Basic idea: we want to show that sentence $A$ is true.

Insert $\neg A$ into the KB and try to derive a contradiction.
11-21: Horn clauses

- Standard resolution theorem proving is exponentially hard.
- However, if we’re willing to restrict ourselves a bit, the problem becomes easy.
- A Horn clause is a disjunction with at most one positive literal.
  - $\neg A \lor \neg B \lor \neg C \lor D$
  - $\neg A \lor \neg B$
- These can be rewritten as implications with one consequent.
  - $A \land B \land C \Rightarrow D$
  - $A \land B \Rightarrow False$
- Horn clauses are the basis of logic programming.
11-22: Forward Chaining

- General idea: start from KB and continually apply Modus Ponens to derive all possible facts.
- This is sometimes called data-driven reasoning.
- Start with domain knowledge and see what that knowledge tells you.
- This is very useful for discovering new facts or rules.
- Less helpful for proving a specific sentence true or false.
  - Search is not directed towards a goal.
- Jess and CLIPS use forward chaining.
Backward chaining starts with the goal and “works backward” to the start.

Example: If we want to show that $A$ is entailed, find a sentence whose consequent is $A$.

Then try to prove that sentence’s antecedents.

This is sometimes called query-driven reasoning.

More effective at proving a particular query, since search is focused on a goal.

Less likely to discover new and unknown information.

Means-ends analysis is a similar sort of reasoning.

Prolog uses backward chaining.
11-24: Strengths of Propositional Logic

6 Declarative - knowledge can be separated from inference.
6 Can handle partial information
6 Can compose more complex sentences out of simpler ones.
6 Sound and complete inference mechanisms (efficient for Horn clauses)
Exponential increase in number of literals

No way to describe relations between objects

No way to quantify over objects.

First-order logic is a mechanism for dealing with these problems.

As always, there will be tradeoffs.

> There’s no free lunch!