CS 510 Lecture 5:
Logic and Planning

Rachel Greenstadt
October 28, 2008
Reminders

• Revised proposals due next week
• Remember to add a brief section discussing progress
• Also, an election next week, if you need to be late to vote, okay (bring sticker)
Midterm Stats

- Mean 73
- Stdev 18
- Solutions posted (soon)
Overview

- Knowledge Bases
- Propositional logic
- Exercise
- Planning Representations
Knowledge Bases

- Knowledge base = set of sentences in a formal language
- Declarative approach to building an agent - TELL it what it needs to know
- Then it ASKs itself what to do, answers follow from knowledge base
- “Building a model of the world” part from Lecture 1
- BDI is one paradigm for this
Architecture of a Knowledge Base

- Inference Engine
- Knowledge Base

Domain-independent algorithms
Domain-specific content
Knowledge-based agent

The agent must be able to:

Represent states, actions, etc
Incorporate new percepts
Update internal representation of the world
Deduce hidden properties of the world
Deduce appropriate actions

```
function KB-AGENT( percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time
    TELL(KB, MAKE-PERCEPT-SENTENCE( percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t← t + 1
    return action
```
Wumpus World PAGE description

Percepts Breeze, Glitter, Smell

Actions Left turn, Right turn,
   Forward, Grab, Release, Shoot

Goals Get gold back to start
   without entering pit or wumpus square

Environment
   Squares adjacent to wumpus are smelly
   Squares adjacent to pit are breezy
   Glitter if and only if gold is in the same square
   Shooting kills the wumpus if you are facing it
   Shooting uses up the only arrow
   Grabbing picks up the gold if in the same square
   Releasing drops the gold in the same square
Logic

• **Logics**: formal languages for representing information
• **Syntax** defines the sentences in the language
• **Semantics** define the “meaning” of sentences
Types of Logic

- Logics are characterized by what they commit to as primitives
- Ontological commitment - What exists?
  - Facts, objects, times, beliefs?

<table>
<thead>
<tr>
<th>Language</th>
<th>Ontological Commitment</th>
<th>Epistemological Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>facts, objects, relations</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Temporal logic</td>
<td>facts, objects, relations, times</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>facts</td>
<td>degree of belief 0…1</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>degree of truth</td>
<td>degree of belief 0…1</td>
</tr>
</tbody>
</table>
Entailment

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

- Knowledge Base KB entails a sentence \( \alpha \) if and only if
  - \( \alpha \) is true in all worlds where KB is true
  - E.g., the KB containing “the Giants won” and “the Reds won” entails “either the Giants won or the Reds won”
Inference

- $KB|\text{-}i\alpha$ : sentence $\alpha$ can be derived from KB by procedure i
- Soundness : i is sound if whenever $KB|\text{-}i\alpha$ it is also true that $KB \models \alpha$
- Completeness: i is complete if whenever $KB \models \alpha$, it is also true that $KB|\text{-}i\alpha$
Propositional Logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas
- The proposition symbols $P_1, P_2$ etc are sentences

If $S$ is a sentence, $\neg S$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \land S_2$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \lor S_2$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \Rightarrow S_2$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \Leftrightarrow S_2$ is a sentence
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol.

E.g. $A \quad B \quad C$

$\begin{array}{ccc}
\text{True} & \text{True} & \text{False} \\
\end{array}$

Rules for evaluating truth with respect to a model $m$:

$\neg S$ is true iff $S$ is false

$S_1 \land S_2$ is true iff $S_1$ is true and $S_2$ is true

$S_1 \lor S_2$ is true iff $S_1$ is true or $S_2$ is true

$S_1 \Rightarrow S_2$ is true iff $S_1$ is false or $S_2$ is true

i.e., $S_1$ is false iff $S_1$ is true and $S_2$ is false

$S_1 \iff S_2$ is true iff $S_1 \Rightarrow S_2$ is true and $S_2 \Rightarrow S_1$ is true
Inference: Enumeration Method

Let $\alpha = A \lor B$ and $KB = (A \lor C) \land (B \lor \neg C)$

Is it the case that $KB \models \alpha$?
Check all possible models—$\alpha$ must be true wherever $KB$ is true

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$A \lor C$</th>
<th>$B \lor \neg C$</th>
<th>$KB$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Enumeration Solution

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$A \lor C$</th>
<th>$B \lor \neg C$</th>
<th>$KB$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<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>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</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>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
Normal Forms

Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms.

**Conjunctive Normal Form (CNF—universal)**

\[
\text{conjunction of \ disjunctions of literals}\frac{\text{clauses}}{\text{terms}}
\]

E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

**Disjunctive Normal Form (DNF—universal)**

\[
\text{disjunction of \ conjunctions of literals}\frac{\text{terms}}{\text{clauses}}
\]

E.g., \((A \land B) \lor (A \land \neg C) \lor (A \land \neg D) \lor (\neg B \land \neg C) \lor (\neg B \land \neg D)\)

**Horn Form (restricted)**

\[
\text{conjunction of \ Horn \ clauses} \quad \text{(clauses with} \leq 1 \text{positive literal)}
\]

E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

Often written as set of implications:

\(B \Rightarrow A \text{ and } (C \land D) \Rightarrow B\)
Inference Rules for Propositional Logic

Resolution (for CNF): complete for propositional logic

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

Modus Ponens (for Horn Form): complete for Horn KBs

\[
\frac{\alpha_1, \ldots, \alpha_n, \quad \alpha_1 \land \cdots \land \alpha_n \Rightarrow \beta}{\beta}
\]

Can be used with forward chaining or backward chaining
Forward chaining

- Idea: fire any rule whose premises are satisfied in the KB, add its conclusion to the KB, until query is found

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Proof of completeness

• FC derives every atomic sentence that is entailed by $KB$

1. FC reaches a fixed point where no new atomic sentences are derived

2. Consider the final state as a model $m$, assigning true/false to symbols

3. Every clause in the original $KB$ is true in $m$

   $a_1 \land \ldots \land a_k \Rightarrow b$

4. Hence $m$ is a model of $KB$
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 $q$

Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal

   1. has already been proved true, or

   2. has already failed
Forward vs. backward chaining

• FC is **data-driven**, automatic, unconscious processing,
  – e.g., object recognition, routine decisions

• May do lots of work that is irrelevant to the goal

• BC is **goal-driven**, appropriate for problem-solving,
  – e.g., Where are my keys? How do I get into a PhD program?

• Complexity of BC 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 Horn clauses
Limits on Propositional Logic

• $B_{1,1}$ iff $(P_{1,2} \lor P_{2,1})$ vs “Squares adjacent to pits are breezy”

• But natural language has drawbacks for representation (ambiguity without context)
First Order Logic

- Objects (like Propositional logic) - people, houses, numbers, theories, colors, wars, centuries

- Relations
  - Unary (properties like red, round, prime, multistoried)
  - n-ary : brother-of, bigger than, inside, is part of, has color, occurred after, owns, comes between
  - Functions (1 to 1) : father of, best friend, third inning of, one more than

- Facts about some or all of the objects
First order logic syntax

Constants     KingJohn, 2, UCB,…
Predicates    Brother, >,…
Functions     Sqrt, LeftLegOf,…
Variables     x, y, a, b,…
Connectives   ∧ ∨ ¬ → ⇔
Equality       =
Quantifiers   ∀ ∃

∃x ∀y Loves(x, y)
“There is a person who loves everyone in the world”

∀y ∃x Loves(x, y)
“Everyone in the world is loved by at least one person”
First order logic syntax

Atomic sentence = predicate(term₁, ..., termₙ)
    or term₁ = term₂

Term = function(term₁, ..., termₙ)
    or constant or variable

E.g., Brother(KingJohn, RichardTheLionheart)
    > (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn))

Complex sentences are made from atomic sentences using connectives

¬S,  S₁ ∧ S₂,  S₁ ∨ S₂,  S₁ → S₂,  S₁ ↔ S₂

E.g. Sibling(KingJohn, Richard) ⇒ Sibling(Richard, KingJohn)
    >(1, 2) ∨ ≤(1, 2)
    >(1, 2) ∧ ¬>(1, 2)
Truth in First Order Logic (Semantics)

Sentences are true with respect to a model and an interpretation.

Model contains objects and relations among them.

Interpretation specifies referents for:
- constant symbols → objects
- predicate symbols → relations
- function symbols → functional relations

An atomic sentence $\text{predicate}(\text{term}_1, \ldots, \text{term}_n)$ is true iff the objects referred to by $\text{term}_1, \ldots, \text{term}_n$ are in the relation referred to by $\text{predicate}$.
Making plans with logic

Initial condition in KB:
\[ At(Agent, [1, 1], S_0) \]
\[ At(Gold, [1, 2], S_0) \]

Query: \( \text{Ask}(KB, \exists s \; \text{Holding}(Gold, s)) \)
i.e., in what situation will I be holding the gold?

Answer: \{s/Result(Grab, Result(Forward, S_0))\}
i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at \( S_0 \) and that \( S_0 \) is the only situation described in the KB
Better logical planning

Represent plans as action sequences \([a_1, a_2, \ldots, a_n]\)

\(\text{PlanResult}(p, s)\) is the result of executing \(p\) in \(s\)

Then the query \(\text{Ask}(KB, \exists p \ \text{Holding}(Gold, \text{PlanResult}(p, S_0)))\)
has the solution \(\{p/[Forward, \text{Grab}]\}\)

Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner
The Game of Clue

It was Professor Plum, in the Library, with the candlestick

• Pros and Cons of the Clue representation
• Can you come up with a better one
BDI Exercise

- Ontological commitments for BDI logic?
- Epistemological commitments?

<table>
<thead>
<tr>
<th>Language</th>
<th>Ontological Commitment</th>
<th>Epistemological Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>facts, objects, relations</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Temporal logic</td>
<td>facts, objects, relations, times</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>facts</td>
<td>degree of belief 0…1</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>degree of truth</td>
<td>degree of belief 0…1</td>
</tr>
</tbody>
</table>
Planning

• Search vs planning
• STRIPS
• Partial Order Planning (POP)
Search vs Planning

Consider the task *get milk, bananas, and a cordless drill*

Standard search algorithms seem to fail miserably:
Planning as Search

\[ PlanResult(p, s) \text{ is the situation resulting from executing } p \text{ in } s \]
\[ PlanResult([], s) = s \]
\[ PlanResult([a|p], s) = PlanResult(p, Result(a, s)) \]

**Initial state** \( At(\text{Home}, S_0) \land \neg Have(\text{Milk}, S_0) \land \ldots \)

**Actions** as Successor State axioms
\[ Have(\text{Milk}, Result(a, s)) \iff [(a = \text{Buy(Milk)} \land At(\text{Supermarket}, s)) \lor (Have(\text{Milk}, s) \land a \neq \ldots)] \]

**Query**
\[ s = PlanResult(p, S_0) \land At(\text{Home}, s) \land Have(\text{Milk}, s) \land \ldots \]

**Solution**
\[ p = [\text{Go(Supermarket)}, \text{Buy(Milk)}, \text{Buy(Bananas)}, \text{Go(HW S)}, \ldots] \]

Principal difficulty: unconstrained branching, hard to apply heuristics
Problems with Search

• Overwhelmed by irrelevant actions
  • Need to know what the result of actions are (buy(x) results in have(x))

• Can’t count on user to supply heuristics
  • Need the agent to be able to decompose problem into subgoals

• All comes down to representation
Planning representations

(STRIPS)

- Use knowledge (logic?) to make planning problem tractable
- States - Conjunction of literals (Poor ^ Unknown)
  - Closed world assumption - any conditions not mentioned are false
- Goals - Partially specified state (Rich ^ Famous) : state s satisfies goal g if s contains all atoms in g (Rich ^ Famous ^ Miserable)
- Action Schema
  - Action name and parameter list $Fly(p,from,to)$
  - Precondition - what must be true before action executed
  - Effect - how state changes when action executed
STRIPS Example

Tidily arranged actions descriptions, restricted language

**ACTION:** *Buy(x)*

**PRECONDITION:** *At(p), Sells(p, x)*

**EFFECT:** *Have(x)*

[Note: this abstracts away many important details!]

Restricted language $\Rightarrow$ efficient algorithm

- Precondition: conjunction of positive literals
- Effect: conjunction of literals

```
At(p)  Sells(p, x)
      
  Buy(x)

      Have(x)
```
State space vs. Plan space

- Standard search: node = concrete world state
- Planning search: node = partial plan
- Defn: **open condition** is a precondition of a step not yet fulfilled
- Operators on partial plans
  - add a link from an existing action to an open condition
  - add a step to fulfill an open condition
  - order one step wrt another

Gradually move from incomplete/vague plans to complete correct plans
Partially ordered plans

A plan is complete iff every precondition is achieved.

A precondition is achieved iff it is the effect of an earlier step and no possibly intervening step undoes it.
Suggested Exercises

- Chapter 7 [7.2, 7.10, 7.12]
- Chapter 8 [8.2, 8.6]
- Chapter 11
Readings next week


Strengths and Weaknesses of BDI