Propositional Logic: Syntax and Semantics

CPSC 322 Lecture 18

February 22, 2006
Textbook §4.0 – 4.2

Logic: A more general framework for reasoning

- Let's now think about how to represent a world about which we have only partial (but certain) information
- Our tool: propositional logic
- General problem:
  - tell the computer how the world works
  - tell the computer some facts about the world
  - ask a yes/no question about whether other facts must be true

Why Propositions?

We'll be looking at problems that could still be represented using CSPs. Why use propositional logic?

- Specifying logical formulae is often more natural than filling in tables (i.e., arbitrary constraints)
- It is easier to check and debug formulae than tables
- We can exploit the Boolean nature for efficient reasoning
- We need a language for asking queries that may be more complicated than asking for the value of one variable
- It is easy to incrementally add formulae
- It can be extended to infinitely many variables (using logical quantification)
- This is a starting point for more complex logics (e.g., first-order logic) that go beyond CSPs.

Representation and Reasoning System

A Representation and Reasoning System (RRS) is made up of:

- syntax: specifies the symbols used, and how they can be combined to form legal sentences
- semantics: specifies the meaning of the symbols
- reasoning theory or proof procedure: a (possibly nondeterministic) specification of how an answer can be produced.

Using an RRS

1. Begin with a task domain.
2. Distinguish those things you want to talk about (the ontology).
3. Choose symbols in the computer to denote propositions
4. Tell the system knowledge about the domain.
5. Ask the system whether new statements about the domain are true or false.

Propositional Definite Clauses

- Propositional Definite Clauses: our first representation and reasoning system.
- Two kinds of statements:
  - that a proposition is true
  - that a proposition is true if one or more other propositions are true
- To define this RSS, we'll need to specify:
  - syntax
  - semantics
  - proof procedure

Propositional Definite Clauses: Syntax

- An atom is a symbol starting with a lower case letter
- A body is an atom or is of the form $b_1 \wedge b_2$ where b_1 and b_2 are bodies.
- A definite clause is an atom or is a rule of the form $h \leftarrow b$ where h is an atom and b is a body.
  - read this as "h if b"
- A knowledge base is a set of definite clauses

Syntax: Example

The following are syntactically correct statements in our language:

- ai_is_fun
- $ai\_is\_fun \leftarrow get\_good\_grade$
- $ai\_is\_fun \leftarrow get\_good\_grade \wedge not\_too\_much\_work$
- $ai\_is\_fun \leftarrow get\_good\_grade \wedge not\_too\_much\_work \wedge prof\_can\_operate\_laptop$

The following statements are syntactically incorrect:

- $ai\_is\_fun \vee ai\_is\_boring$
- $ai\_is\_fun \wedge relaxing\_term \leftarrow get\_good\_grade \wedge not\_too\_much\_work$

Do any of these statements *mean* anything? Syntax doesn't answer this question.

Propositional Definite Clauses: Semantics

Semantics allows you to relate the symbols in the logic to the domain you're trying to model.

- An interpretation I assigns a truth value to each atom.

We can use the interpretation to determine the truth value of clauses and knowledge bases:

- A body $b_1 \wedge b_2$ is true in I if b_1 is true in I and b_2 is true in I.
- A rule $h \leftarrow b$ is false in I if b is true in I and h is false in I. The rule is true otherwise.
- A knowledge base KB is true in I if and only if every clause in KB is true in I.

Models and Logical Consequence

- A model of a set of clauses is an interpretation in which all the clauses are *true*.
- If KB is a set of clauses and g is a conjunction of atoms, g is a logical consequence of KB, written $KB \models g$, if g is *true* in every model of KB.
  - we also say that g logically follows from KB, or that KB entails g.
- In other words, $KB \models g$ if there is no interpretation in which KB is *true* and g is *false*.