Propositional Logic Intro, Syntax

CPSC 322 – Logic 1

Textbook §5.0 – 5.2
Lecture Overview

1. Recap

2. Logic Intro

3. Propositional Definite Clause Logic: Syntax
Forward Planning

**Idea:** search in the state-space graph.

- The nodes represent the states
- The arcs correspond to the actions: The arcs from a state $s$ represent all of the actions that are legal in state $s$.
- A plan is a path from the state representing the initial state to a state that satisfies the goal.
We don’t have to worry about searching forwards if we set up a planning problem as a CSP.

To do this, we need to “unroll” the plan for a fixed number of steps.

- this is called the horizon

To do this with a horizon of $k$:

- construct a variable for each feature at each time step from 0 to $k$
- construct a boolean variable for each action at each time step from 0 to $k - 1$. 
The constraints shown represent the preconditions of actions and the effects of actions.

Propositional Logic Intro, Syntax
Lecture Overview

1 Recap

2 Logic Intro

3 Propositional Definite Clause Logic: Syntax
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 constructing arbitrary constraints
- It is **easier to check and debug** formulae than constraints
- 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
- Logic 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 do go beyond CSPs.
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
Lecture Overview

1. Recap
2. Logic Intro
3. Propositional Definite Clause Logic: Syntax
Propositional Definite Clauses: Syntax

Definition (atom)
An atom is a symbol starting with a lower case letter.
Propositional Definite Clauses: Syntax

**Definition (atom)**

An **atom** is a symbol starting with a lower case letter.

**Definition (body)**

A **body** is an atom or is of the form $b_1 \land b_2$ where $b_1$ and $b_2$ are bodies.
### Definition (atom)

An **atom** is a symbol starting with a lower case letter.

### Definition (body)

A **body** is an atom or is of the form \( b_1 \land b_2 \) where \( b_1 \) and \( b_2 \) are bodies.

### Definition (definite clause)

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 \( b \)”.)
### Definition (atom)
An **atom** is a symbol starting with a lower case letter.

### Definition (body)
A **body** is an atom or is of the form $b_1 \land b_2$ where $b_1$ and $b_2$ are bodies.

### Definition (definite clause)
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 $b$.”)  

### Definition (knowledge base)
A **knowledge base** is a set of definite clauses.
Syntax: Example

The following are syntactically correct statements in our language:

- $ai\_is\_fun$
Syntax: Example

The following are syntactically correct statements in our language:

- $ai\_is\_fun$
- $ai\_is\_fun \leftarrow get\_good\_grade$
Syntax: Example

The following are syntactically correct statements in our language:

- $ai\_is\_fun$
- $ai\_is\_fun \leftarrow get\_good\_grade$
- $ai\_is\_fun \leftarrow get\_good\_grade \land not\_too\_much\_work$
Syntax: Example

The following are syntactically correct statements in our language:

- ai_is_fun
- ai_is_fun ← get_good_grade
- ai_is_fun ← get_good_grade ∧ not_too_much_work
- ai_is_fun ← get_good_grade ∧ not_too_much_work ∧ prof_can_operate_laptop

The following statements are syntactically incorrect:

- ai_is_fun ∨ ai_is_boring
- ai_is_fun ∧ relaxing
term ← get_good_grade ∧ not_too_much_work

Do any of these statements mean anything? Syntax doesn’t answer this question.
Syntax: Example

The following are syntactically correct statements in our language:

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

The following statements are syntactically incorrect:

- $ai\_is\_fun \lor ai\_is\_boring$
Syntax: Example

The following are syntactically correct statements in our language:

- ai_is_fun
- ai_is_fun ← get_good_grade
- ai_is_fun ← get_good_grade \land not-too-much_work
- ai_is_fun ← get_good_grade \land not-too-much_work \land prof_can_operate_laptop

The following statements are syntactically incorrect:

- ai_is_fun \lor ai_is_boring
- ai_is_fun \land relaxing_term ← get_good_grade \land not-too-much_work
Syntax: Example

The following are syntactically correct statements in our language:

- *ai_is_fun*
- *ai_is_fun ← get_good_grade*
- *ai_is_fun ← get_good_grade ∧ not_too_much_work*
- *ai_is_fun ← get_good_grade ∧ not_too_much_work ∧ prof_can_operate_laptop*

The following statements are syntactically incorrect:

- *ai_is_fun ∨ ai_is_boring*
- *ai_is_fun ∧ relaxing_term ← get_good_grade ∧ not_too_much_work*

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