Propositional Logic Intro, Syntax

CPSC 322 Lecture 18

R&R systems we'll cover in this course

		Environment	
Problem		**Deterministic**	**Stochastic**
Static	Constraint Satisfaction	*Variables + Constraints* Search Arc Consistency Local Search	
	Query	*Logics* Search	*Bayesian (Belief) Networks* Variable Elimination
Sequential	Planning	*STRIPS* Search	*Decision Networks* Variable Elimination

Representation
Reasoning Technique

Learning Goals for today's class

You can:

- Verify whether a logical statement belongs to the language of full propositional logics.
- Verify whether a logical statement belongs to the language of propositional definite clauses.

Lecture Overview

- **Logic Intro**
- Propositional Definite Clause Logic: Syntax

What you already know about logic...

From programming: Some logical operators

```
If ((amount > 0) && (amount < 1000)) || !(age < 30)
...
```

You know what they mean in a “procedural” way

Logic is the language of Mathematics, used to define formal structures (e.g., sets, graphs) and to prove statements about them **(as you did in CPSC 121)**

We are going to look at Logic as a **Representation and Reasoning System** that can be used to **formalize a domain (e.g., an electrical system, an organization)** and to **reason about it**

Logic: A general framework for representation & reasoning

- Consider **how to represent an environment** about which we have only partial (but certain) information
- What do we need to represent?
  - **Facts** about the environment
  - **Rules** of how the environment works

Logics

- **We begin with propositional logics**, since this is the starting point for more complex logics
- **Natural** to express **knowledge** about the world
  - What is true (boolean variables)
  - How it works (logical formulas)
- Well understood formal properties
- Boolean nature can be exploited for efficiency

Why Logics?

"Natural" way to express **knowledge**
(more natural than a "flat" set of variables & constraints)

"Every 322 student will pass the midterm"

Midterm(m1) Course(c1)
NameOf(c1,cpsc322) CourseOfMidterm(m1,c1)
∀s Student(s) ^ Registered(s,cpsc322) → pass(m1,s)

- It is easy to incrementally add knowledge
- It is easy to check and debug knowledge
- Provide language for asking complex queries
- Well understood formal properties

Complex Query

“Will Sue pass all her midterms?”

$$\forall c,m\ \text{Registered}(\text{sue},c) \wedge \text{CourseOfMidterm}(m,c) \Rightarrow ?\ \text{pass}(m,\text{sue})$$

Propositional Logic

- The primitive elements are **propositions**: Boolean variables that can be {*true, false*}
- The goal is to illustrate the basic ideas as a starting point for more complex logics (e.g., **first-order logic**)
- Boolean nature can be exploited for efficiency.

Propositional logic: Complete Language

The **proposition** symbols p_1, p_2 … etc are sentences

- If S is a sentence, $\neg$**S** is a sentence (negation)
- If S_1 and S_2 are sentences, $\mathbf{S_1} \wedge \mathbf{S_2}$ is a sentence (conjunction)
- If S_1 and S_2 are sentences, $\mathbf{S_1} \vee \mathbf{S_2}$ is a sentence (disjunction)
- If S_1 and S_2 are sentences, $\mathbf{S_1} \Rightarrow \mathbf{S_2}$ is a sentence (implication)
- If S_1 and S_2 are sentences, $\mathbf{S_1} \Leftrightarrow \mathbf{S_2}$ is a sentence (biconditional)

Sample formula:

((p1 v p2) ^ p3) $\Leftrightarrow$ ((p2 $\Rightarrow$ $\neg$p4) v p5)

Propositional Logics in practice

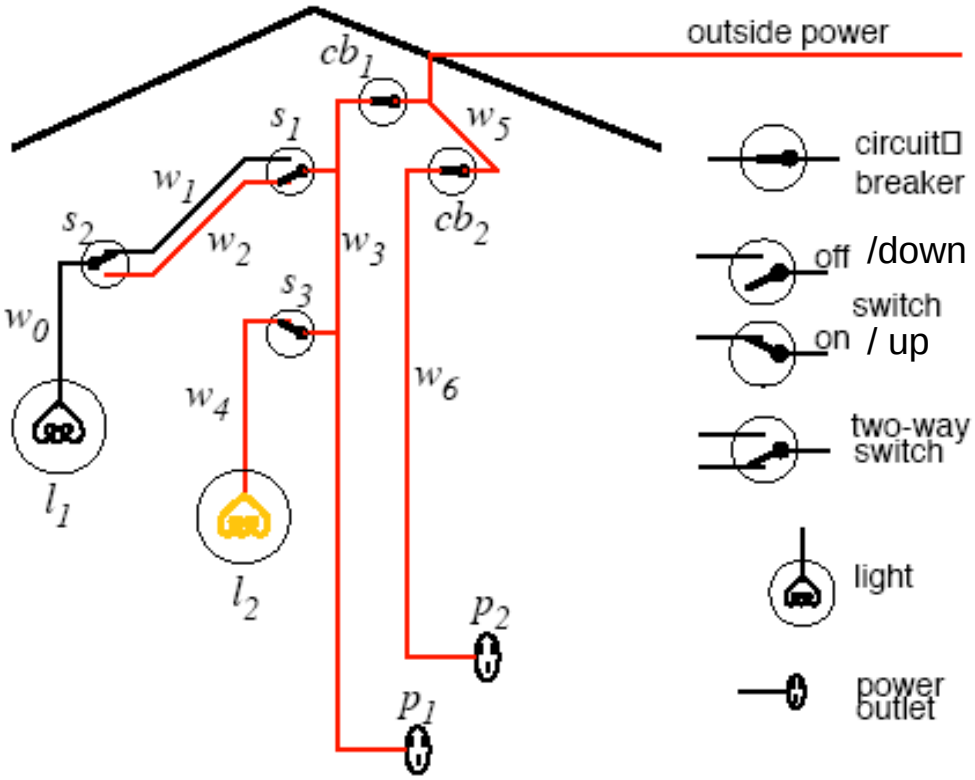
- Agent is told (perceives) some facts about the world (a set of true propositions)
- Agent is told (already knows / learns) how the world works (a set of logical formulas)
- Agent can answer yes/no questions about whether other facts **must be true**

Using Logics to make inferences…

1) Begin with a **task domain**.
2) Distinguish those **things** you want to talk about
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

Electrical Environment

1) Begin with a **task domain**
2) Distinguish those **things** you want to talk about
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



Lecture Overview

- Logic Intro
- **Propositional Definite Clause Logic: Syntax**

Propositional Definite Clauses

Our first logical representation and reasoning system. (very simple!)

- Only two kinds of statements:
  - that **a proposition is true** p_2
  - that **a proposition is true if one or more other propositions are true** $p_2 \leftarrow p_3 \wedge p_1$
- Why still useful?
  - Adequate in many domains (with some adjustments)
  - Reasoning steps easy to follow by humans
  - Inference linear in size of your set of statements
  - Similar formalisms used in cognitive architectures

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 $b_1 \wedge b_2$ 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.'')

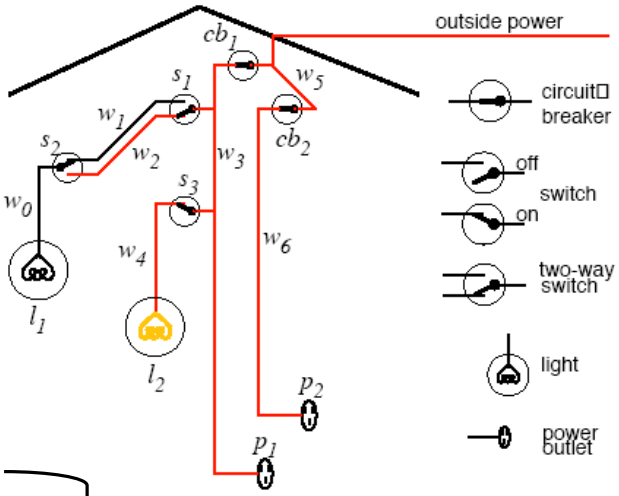
Definition (KB)

A **knowledge base** is a set of definite clauses

definite clauses, KB

atoms

light_l1.
light_l2.
ok_l1.
ok_l2.
ok_cb1.
ok_cb2.
live_outside.



rules

live_l1 ←live_w0.
live_w0 ←live_w1 ∧up_s2.
live_w0 ←live_w2 ∧down_s2.
live_w1 ←live_w3 ∧up_s1.
live_w2 ←live_w3 ∧down_s1.
live_l2 ←live_w4.
live_w4 ←live_w3 ∧up_s3.
live_p1 ←live_w3.
live_w3 ←live_w5 ∧ok_cb1.
live_p2 ←live_w6.
live_w6 ←live_w5 ∧ok_cb2.
live_w5 ←live_outside.
lit_l1 ←light_l1 ∧live_l1 ∧ok_l1.
lit_l2 ←light_l2 ∧live_l2 ∧ok_l2.

PDC Syntax: more examples

Definition (definite clause)

A **definite clause** is

- an atom or
- a rule of the form $h \leftarrow b$ where h is an atom ('head') and b is a body. (Read this as 'h if b.')

a) *ai_is_fun*

b) *ai_is_fun* $\vee$ *ai_is_boring*

c) *ai_is_fun* $\leftarrow$ *learn_useful_techniques*

d) *ai_is_fun* $\leftarrow$ *learn_useful_techniques* $\wedge$ *notTooMuch_work*

e) *ai_is_fun* $\leftarrow$ *learn_useful_techniques* $\wedge \neg$ *TooMuch_work*

f) *ai_is_fun* $\leftarrow$ *f(time_spent, material_learned)*

g) *srtsyj* $\leftarrow$ *errt* $\wedge$ *gffdgdgd*

A. Legal B. Not Legal

PDC Syntax: more examples

Legal PDC clause

Not a legal PDC clause

a) *ai_is_fun*

b) *ai_is_fun* $\vee$ *ai_is_boring*

c) *ai_is_fun* $\leftarrow$ *learn_useful_techniques*

d) *ai_is_fun* $\leftarrow$ *learn_useful_techniques* $\wedge$ *notTooMuch_work*

e) *ai_is_fun* $\leftarrow$ *learn_useful_techniques* $\wedge \neg$ *TooMuch_work*

f) *ai_is_fun* $\leftarrow$ *f(time_spent, material_learned)*

g) *srtsyj* $\leftarrow$ *errt* $\wedge$ *gffdgdgd*

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

Next class

- Definite clauses Semantics and Proofs (textbook 5.1.2, 5.3.2)