Propositional Definite Clause Logic: Syntax, Semantics, R&R and Proofs

Computer Science cpsc322, Lecture 7

(Textbook Chpt 5.1-5.2)

May, 29, 2012
Lecture Overview

- **Recap**: Logic intro and Propositional Definite Clause (PDCL)
- **Semantics** (interpretation, model, logical consequence)
- **Bottom-up** Proof Procedure for PDCL
  - Soundness and Completeness
- Using PDCL for **R&R in a domain** (Electrical System)
- **Top-Down** Proof Procedure for PDCL
Logics as a R&R system

- formalize a domain

$\text{on}_{-}l_1 \text{ if } \text{live}_{-}w_1 \land \text{on}_{-}\text{sw}_2 \land \text{live}_{-}w_3$

- reason about it

If the agent knows $\text{on}_{-}\text{sw}_2$ and $\text{live}_{-}w_3$, it should be able to infer $\text{on}_{-}e_1$. 
Logics in AI: Similar slide to the one for planning

Propositional Definite Clause Logics

Semantics and Proof Theory

Satisfiability Testing (SAT)

Description Logics

Production Systems

Hardware Verification

Ontologies

Cognitive Architectures

Product Configuration

Semantic Web

Information Extraction

Video Games

Summarization

Tutoring Systems

CPSC 322, Lecture 7
Propositional (Definite Clauses) Logic: Syntax

We start from a restricted form of Prop. Logic:

\[ \neg (p_1 \lor p_2) \iff (p_3 \lor \neg p_3) \]

Only two kinds of statements:
- that a proposition is true
- that a proposition is true if one or more other propositions are true

A definite clause is:
- either an atom
- or an atom \( \iff \) body

\[ p_3 \iff p_2 \land p_2 \]

\[ p_2 \land p_2 \iff p_3 \]
Lecture Overview

• **Recap**: Logic intro and Propositional Definite Clause (PDCL)

• **Semantics** (interpretation, model, logical consequence)

• **Bottom-up** Proof Procedure for PDCL
  • Soundness and Completeness

• Using PDCL for **R&R in a domain** (Electrical System)

• **Top-Down** Proof Procedure for PDCL
Propositional Definite Clauses Semantics: Interpretation

Semantics allows you to relate the symbols in the logic to the domain you're trying to model. An **atom** can be..... 

Definition (interpretation)
An interpretation $I$ assigns a truth value to each atom.

If your domain can be represented by four atoms (propositions):

<table>
<thead>
<tr>
<th></th>
<th>q</th>
<th>p</th>
<th>s</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

So an **interpretation** is just a...*possible world*. 

PDC Semantics: Body

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

**Definition (truth values of statements):** A body $b_1 \land b_2$ is true in $I$ if and only if $b_1$ is true in $I$ and $b_2$ is true in $I$.

<table>
<thead>
<tr>
<th></th>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>$I_2$</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>$I_3$</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>$I_4$</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>$I_5$</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
PDC Semantics: definite clause

Definition (truth values of statements cont’): A rule \( h ← b \) is false in \( I \) if and only if \( b \) is true in \( I \) and \( h \) is false in \( I \).

<table>
<thead>
<tr>
<th></th>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_1 )</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>( I_2 )</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>( I_3 )</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>( I_4 )</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

In other words: "if \( b \) is true I am claiming that \( h \) must be true, otherwise I am not making any claim"
PDC Semantics: Knowledge Base (KB)

A **knowledge base** KB is true in I if and only if every clause in KB is true in I.

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>KB₁</th>
<th>KB₂</th>
<th>KB₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>p</td>
<td>p</td>
</tr>
<tr>
<td>r</td>
<td>q</td>
<td>q</td>
</tr>
<tr>
<td>s ← q ∧ p</td>
<td>s ← q</td>
<td>q ← r ∧ s</td>
</tr>
</tbody>
</table>

Which of the three KB above are True in I₁
A knowledge base $KB$ is true in $I$ if and only if every clause in $KB$ is true in $I$.

<table>
<thead>
<tr>
<th></th>
<th>$p$</th>
<th>$q$</th>
<th>$r$</th>
<th>$s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

$KB_1$

- $p$
- $r$
- $s \leftarrow q \land p$

$KB_2$

- $p$
- $q$
- $s \leftarrow q$

$KB_3$

- $p$
- $q \leftarrow r \land s$

Which of the three $KB$ above are True in $I_1$? $KB_3$
Models

Definition (model)
A model of a set of clauses (a KB) is an interpretation in which all the clauses are true.
Example: Models

KB = \begin{cases} 
  p \leftarrow q. \\
  q. \\
  r \leftarrow s. 
\end{cases}

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

Which interpretations are models?
Definition (logical consequence)
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.
Example: Logical Consequences

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>l₁</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>l₂</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>l₃</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>l₄</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>l₅</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>l₆</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>l₇</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>l₈</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

Which of the following is true?

• $KB \not\models q$, $KB \not\models p$, $KB \not\models s$, $KB \not\models r$

$2^4 = 16$ interpretations in total, only 3 are models

remaining 8 cannot be models because q is false
Lecture Overview

• **Recap**: Logic intro and Propositional Definite Clause (PDCL)
• **Semantics** (interpretation, model, logical consequence)
• **Bottom-up** Proof Procedure for PDCL
  • Soundness and Completeness
• Using PDCL for **R&R in a domain** (Electrical System)
• **Top-Down** Proof Procedure for PDCL
One simple way to prove that $G$ logically follows from a KB

- Collect all the models of the KB
- Verify that $G$ is true in all those models

Any problem with this approach?

- The goal of proof theory is to find proof procedures that allow us to prove that a logical formula follows from a KB avoiding the above

intractable time complexity

you have to check all the $2^n$ interpretations
Soundness and Completeness

• If I tell you I have a **proof procedure for PDCL**
• What do I need to show you in order for you to trust my procedure?
  - $KB \vdash G$ means $G$ can be derived by my proof procedure from $KB$.
  - Recall $KB \nvDash G$ means $G$ is true in all models of $KB$.

**Definition (soundness)**

A proof procedure is **sound** if $KB \vdash G$ implies $KB \nvdash G$.

**Definition (completeness)**

A proof procedure is **complete** if $KB \nvDash G$ implies $KB \vdash G$. 
Bottom-up Ground Proof Procedure

One rule of derivation, a generalized form of *modus ponens*:

If “$h \leftarrow b_1 \land \ldots \land b_m$” is a clause in the knowledge base, and each $b_i$ has been derived, then $h$ can be derived.

You are *forward chaining* on this clause. (This rule also covers the case when $m=0$.)
Bottom-up proof procedure

\[ KB \vdash G \] if \( G \subseteq C \) at the end of this procedure:

\[ C := \{} \]

repeat

select clause \( "h \leftarrow b_1 \land \ldots \land b_m" \) in \( KB \) such that \( b_i \in C \) for all \( i \), and \( h \notin C \);

\[ C := C \cup \{ h \} \]

until no more clauses can be selected.
Bottom-up proof procedure: Example

\[ z \leftarrow f \land e \quad C = \{ \top, r, b_1, a, e, z \} \]

\[ q \leftarrow f \land g \land z \leftarrow \]

\[ e \leftarrow a \land b \leftarrow \]

\[ a \leftarrow \]

\[ b \leftarrow \]

\[ r \leftarrow \]

\[ f \leftarrow \]

\[ r \vdash q \vdash z \leftarrow \]

\[ KB \vdash_{BU} q \]

\[ KB \vdash_{BU} z \]

\[ C := \{} \]

repeat

select clause "\( h \leftarrow b_1 \land \ldots \land b_m \)" in \( KB \) such that \( b_i \in C \) for all \( i \), and \( h \notin C \);

\[ C := C \cup \{ h \} \]

until no more clauses can be selected.
Lecture Overview

• **Recap**: Logic intro and Propositional Definite Clause (PDCL)

• **Semantics** (interpretation, model, logical consequence)

• **Bottom-up** Proof Procedure for PDCL
  - **Soundness** and Completeness

• Using PDCL for **R&R in a domain** (Electrical System)

• **Top-Down** Proof Procedure for PDCL
Soundness of bottom-up proof procedure

Generic Soundness of proof procedure:
If \( G \) can be proved by the procedure \( (KB \vdash G) \) then \( G \) is logically entailed by the KB \( (KB \models G) \)

For Bottom-Up proof
if \( G \leq C \) at the end of procedure then \( G \) is logically entailed by the KB

So BU is sound, if all the atoms in \( C \) are logically entailed by the KB
Suppose this is not the case.

1. Let $h$ be the first atom added to $C$ that is not entailed by $KB$ (i.e., that's not true in every model of $KB$).

2. Suppose $h$ isn't true in model $M$ of $KB$.

3. Since $h$ was added to $C$, there must be a clause in $KB$ of form: $\neg h \lor b_1 \lor ... \lor b_m$

4. Each $b_i$ is true in $M$ (because of 1.). $h$ is false in $M$. So...... the clause is false in $M$

5. Therefore $M$ is not a model

6. Contradiction! thus no such $h$ exists.
Learning Goals for today’s class – part 1

You can:

• Verify whether an **interpretation** is a **model** of a PDCL KB.
• Verify when a conjunction of atoms is a **logical consequence** of a knowledge base.
• Define/read/write/trace/debug the **bottom-up** proof procedure.
• Prove that BU proof procedure is **sound**
Lecture Overview

• **Recap**: Logic intro and Propositional Definite Clause (PDCL)

• **Semantics** (interpretation, model, logical consequence)

• **Bottom-up** Proof Procedure for PDCL
  • Soundness and **Completeness**

• Using PDCL for **R&R in a domain** (Electrical System)

• **Top-Down** Proof Procedure for PDCL

30 mins BREAK
Completeness of Bottom Up

Generic Completeness of proof procedure:
If $G$ is logically entailed by the KB ($\text{KB} \models G$),
then $G$ can be proved by the procedure ($\text{KB} \vdash G$)

$G \subseteq C$

Sketch of our proof:
1. Suppose $\text{KB} \not\models G$. Then $G$ is true in all models of $KB$.
2. Thus $G$ is true in any particular model of KB
3. We will define a model so that if $G$ is true in that model, $G$ is proved by the bottom up algorithm. $G \subseteq C$
4. Thus $\text{KB} \vdash G$. 
Let's work on step 3

3. We will define a model so that if G is true in that model, G is proved by the bottom up algorithm.

3.1 We will define an interpretation $\mathcal{I}$ so that if G is true in $\mathcal{I}$, G is proved by the bottom up algorithm.

3.2 We will then show that \ldots \ldots is a model
Let’s work on step 3.1

3.1 Define interpretation $I$ so that if $G$ is true in $I$, then $G \subseteq C$.

Let $I$ be the interpretation in which every element of $C$ is \texttt{true} and every other atom is \texttt{false}.

\[
\begin{align*}
  a &\leftarrow e \land g. \\
  b &\leftarrow f \land g. \\
  c &\leftarrow e. \\
  f &\leftarrow c \land e. \\
  e. \\
  d.
\end{align*}
\]

\[
\begin{array}{c}
\text{F F T T T T T F} \\
\{ a, b, c, d, e, f, g \}
\end{array}
\]
Let's work on step 3.2

Claim: \( I \) is a model of \( KB \) (we'll call it the \textit{minimal model}).

Proof: Assume that \( I \) is not a model of \( KB \).

- Then there must exist some clause \( h \leftarrow b_1 \land \ldots \land b_m \) in \( KB \) (having zero or more \( b_i \)'s) which is false in \( I \).
- The only way this can occur is if \( b_1 \ldots b_m \) are true in \( I \) (i.e., are in \( C \)) and \( h \) is false in \( I \) (i.e., is not in \( C \)).
- But if each \( b_i \) belonged to \( C \), Bottom Up would have added \( h \) to \( C \) as well.
- So, there can be no clause in the KB that is false in interpretation \( I \) (which implies the claim :-)

Completeness of Bottom Up
(proof summary)

If $KB \not\models G$ then $KB \not\vdash G$

- Suppose $KB \not\models G$.
- Then $G$ is true in all the models.
- Thus $G$ is true in the minimal model.
- Thus $G \subseteq C$.
- Thus $G$ is proved by......
- Thus $KB \vdash_{bu} G$.

\[ KB \models G \quad \text{bu} \]

\[ KB \vdash_{bu} G \]

\[ \text{Soundness} \]

\[ \text{Completeness} \]

Relation between soundness & completeness.
Soundness & completeness of proof procedures

• A proof procedure $X$ is sound …

$$KB \vdash_X G \Rightarrow KB \models G$$

• A proof procedure $X$ is complete….

$$KB \models G \Rightarrow KB \vdash_X G$$

• BottomUp for PDCL is $\leftarrow$ sound & complete

• We proved this in general even for domains represented by thousands of propositions and corresponding KB with millions of definite clauses!
An exercise for you \( \text{BU} \) \( \mathcal{C} = \{ d, e, c, f \} \)

Let’s consider these two alternative proof procedures for PDCL:

A. \( \mathcal{C}_A = \{ \text{All clauses in \( \mathcal{KB} \) with empty bodies} \} = \{ e, d \} \)

B. \( \mathcal{C}_B = \{ \text{All atoms in the knowledge base} \} = \{ e, d, f, c, g, a \} \)

Both A and B are sound and complete.

Both A and B are neither sound nor complete.

A is sound only and B is complete only.

A is complete only and B is sound only.
Can you think of a proof procedure for PDCL....

A: \( C_A = \{ \text{all clauses with empty bodies} \} \)

\( KB \vdash_A G \text{ } G \subseteq C_A \subseteq C_{BU} \)

B: \( C_B = \{ \text{all atoms of } KB \} \)

\( KB \vdash_B G \text{ } G \subseteq C_B \text{ } C_B \subseteq C_B \)

- That is sound but not complete?

\( KB \vdash_A G \Rightarrow KB \vdash_G \)

\( \neg \Rightarrow G \subseteq C_A \subseteq C_{BU} \Rightarrow KB \vdash_{BU} G \Rightarrow KB \vdash_{BU} G \Rightarrow KB \vdash_G \)

- That is complete but not sound?

\( KB \vdash G \Rightarrow KB \vdash_{BU} G \Rightarrow G \subseteq C_{BU} \Rightarrow G \subseteq C_B \Rightarrow KB \vdash_B G \)
Lecture Overview

- **Recap**: Logic intro and Propositional Definite Clause (PDCL)
- **Semantics** (interpretation, model, logical consequence)
- **Bottom-up** Proof Procedure for PDCL
  - Soundness and Completeness
- **Using PDCL for R&R in a domain** (Electrical System)
- **Top-Down** Proof Procedure for PDCL
Electrical Environment
Let's define relevant propositions

- For each wire $w$
- For each circuit breaker $cb$
- For each switch $s$
- For each light $l$
- For each outlet $p$

How many interpretations?

- 7
- 2
- $3 \times 2$
- 2

$\approx 5 \times 10^5$
Let’s now tell system knowledge about how the domain works

\[
\begin{align*}
\text{live}_l_1 & \leftarrow \text{live}_w_0 \\
\text{live}_w_0 & \leftarrow \text{up}_s_2 \land \text{live}_w_2 \\
\text{live}_w_0 & \leftarrow \text{down}_s_2 \land \text{live}_w_2 \\
\text{live}_w_1 & \leftarrow \text{up}_s_1 \land \text{live}_w_3
\end{align*}
\]
More on how the domain works….

\[
\begin{align*}
\text{live}_{w_2} & \leftarrow \text{live}_{w_3} \land \text{down}_{s_1}. \\
\text{live}_{l_2} & \leftarrow \text{live}_{w_4}. \\
\text{live}_{w_4} & \leftarrow \text{live}_{w_3} \land \text{up}_{s_3}. \\
\text{live}_{p_1} & \leftarrow \text{live}_{w_3}.
\end{align*}
\]
More on how the domain works….

\[
\begin{align*}
\text{live}_w_3 & \leftarrow \text{live}_w_5 \land \text{ok}_cb_1. \\
\text{live}_p_2 & \leftarrow \text{live}_w_6. \\
\text{live}_w_6 & \leftarrow \text{live}_w_5 \land \text{ok}_cb_2. \\
\text{live}_w_5 & \leftarrow \text{live}_{\text{outside}}.
\end{align*}
\]
What else we may know about this domain?

- That some simple propositions are true

  *live_outside.*
What else we may know about this domain?

• That some additional simple propositions are true:

\[ \text{down}_s_1, \text{up}_s_2, \text{up}_s_3, \text{ok}_c_b_1, \text{ok}_c_b_2, \text{live}_\text{outside}. \]
All our knowledge.....

\[ \begin{align*}
\text{down}_s_1. \\
\text{up}_s_2. \\
\text{up}_s_3. \\
\text{ok}_cb_1. \\
\text{ok}_cb_2. \\
\text{live}_\text{outside} \\
\text{live}_l_1 & \leftarrow \text{live}_w_0 \\
\text{live}_w_0 & \leftarrow \text{live}_w_1 \land \text{up}_s_2. \\
\text{live}_w_0 & \leftarrow \text{live}_w_2 \land \text{down}_s_2. \\
\text{live}_w_1 & \leftarrow \text{live}_w_3 \land \text{up}_s_1. \\
\text{live}_w_2 & \leftarrow \text{live}_w_3 \land \text{down}_s_1. \\
\text{live}_l_2 & \leftarrow \text{live}_w_4. \\
\text{live}_w_4 & \leftarrow \text{live}_w_3 \land \text{up}_s_3. \\
\text{live}_p_1 & \leftarrow \text{live}_w_3. \\
\text{live}_w_3 & \leftarrow \text{live}_w_5 \land \text{ok}_cb_1. \\
\text{live}_p_2 & \leftarrow \text{live}_w_6. \\
\text{live}_w_6 & \leftarrow \text{live}_w_5 \land \text{ok}_cb_2. \\
\text{live}_w_5 & \leftarrow \text{live}_\text{outside}. 
\end{align*} \]
What Semantics is telling us

• Our KB (all we know about this domain) is going to be true only in a subset of all possible $2^N$ interpretations.

• What is **logically entailed** by our KB are all the propositions that are true in all those models.

• This is what we should be able to derive given a sound and complete proof procedure.
If we apply the bottom-up (BU) proof procedure:

down_s_1.
up_s_2.
up_s_3.
ok_cb_1.
ok_cb_2.
live_outside

dlive_l_1 ← live_w_0
live_w_0 ← live_w_1 ∧ up_s_2.
live_w_0 ← live_w_2 ∧ down_s_2.
live_w_1 ← live_w_3 ∧ up_s_1.
live_w_2 ← live_w_3 ∧ down_s_1.
live_l_2 ← live_w_4.
live_w_4 ← live_w_3 ∧ up_s_3.
live_p_1 ← live_w_3.
live_w_3 ← live_w_5 ∧ ok_cb_1.
live_p_2 ← live_w_6.
live_w_6 ← live_w_5 ∧ ok_cb_2.
live_w_5 ← live_outside.

BU generates C

all the atoms added to C are in green

live_l_2 ⊨ live_l_1

live_l_2 < C ⇒ KB ⊢ live_l_2 ⇒ KB = live_l_2

which is not the case for live_l_1

CPSC 322, Lecture 7
Lecture Overview

- **Recap**: Logic intro and Propositional Definite Clause (PDCL)
- **Semantics** (interpretation, model, logical consequence)
- **Bottom-up** Proof Procedure for PDCL
  - Soundness and Completeness
- Using PDCL for **R&R in a domain** (Electrical System)
- **Top-Down** Proof Procedure for PDCL
Bottom-up vs. Top-down

**Bottom-up**

KB $\rightarrow$ C

G is proved if $G \subseteq C$

When does BU look at the query? g

- In every loop iteration
- Never
- At the end
- At the beginning
Bottom-up vs. Top-down

• **Key Idea of top-down**: search backward from a query $g$ to determine if it can be derived from $KB$.

**Bottom-up**

$KB \rightarrow C$

g is proved if $G \subseteq C$

**Top-down**

Query $G$

KB $\rightarrow$ answer

When does BU look at the query $G$?
• At the end

TD performs a backward search starting at $G$
Top-down Ground Proof Procedure

**Key Idea:** search backward from a query $G$ to determine if it can be derived from $KB$. 

**Bottom Up**

$KB \xrightarrow{Bu} \{ C \}$

Look at the query only at the end

$G \subseteq C$

$G$ is proved if

**Top Down**

Query $G$

Answer
Top-down Proof Procedure: Basic elements

Notation: An answer clause is of the form:

\[ \text{yes} \leftarrow a_1 \land a_2 \land \ldots \land a_m \]

Express query as an answer clause (e.g., query \( a_1 \land a_2 \land \ldots \land a_m \))

\[ \text{yes} \leftarrow a_2 \land \ldots \land a_m \]

Rule of inference (called SLD Resolution)

Given an answer clause of the form:

\[ \text{yes} \leftarrow a_1 \land a_2 \land \ldots \land a_m \]

and the clause:

\[ a_i \leftarrow b_1 \land b_2 \land \ldots \land b_p \]

You can generate the answer clause

\[ \text{yes} \leftarrow a_1 \land \ldots \land a_{i-1} \land b_1 \land b_2 \land \ldots \land b_p \land a_{i+1} \land \ldots \land a_m \]
Rule of inference: Examples

Rule of inference (called SLD Resolution)
Given an answer clause of the form:

\[ \text{yes} \leftarrow a_1 \land a_2 \land \ldots \land a_m \]

and the clause:

\[ a_i \leftarrow b_1 \land b_2 \land \ldots \land b_p \]

You can generate the answer clause:

\[ \text{yes} \leftarrow a_1 \land \ldots \land a_{i-1} \land b_1 \land b_2 \land \ldots \land b_p \land a_{i+1} \land \ldots \land a_m \]

\[ \text{KB} \]

\[ \text{yes} \leftarrow b \land c. \]
\[ b \leftarrow k \land f. \quad \Rightarrow \quad \text{yes} \leftarrow k \land f \land c \]

\[ \text{yes} \leftarrow e \land f. \]
\[ e. \leftarrow \quad \Rightarrow \quad \text{yes} \leftarrow f \]
(successful) Derivations

• An **answer** is an answer clause with $m = 0$. That is, it is the answer clause $yes \leftarrow$.

• A (successful) **derivation** of query "$q_1 \land \ldots \land q_k$" from $KB$ is a sequence of answer clauses $\gamma_0, \gamma_1, \ldots, \gamma_n$ such that
  • $\gamma_0$ is the answer clause $yes \leftarrow q_1 \land \ldots \land q_k$
  • $\gamma_i$ is obtained by **resolving** $\gamma_{i-1}$ with a clause in $KB$, and
  • $\gamma_n$ is an answer. $yes \leftarrow$.

• An **unsuccessful derivation**.....

$yes \leftarrow a \land b$
Example: derivations

\[
\begin{align*}
  &a \leftarrow e \land f. \\
  &c \leftarrow e. \\
  &f \leftarrow j \land e. \\
  &a \leftarrow b \land c. \\
  &d \leftarrow k. \\
  &f \leftarrow c. \\
  &b \leftarrow k \land f. \\
  &e. \\
  &j \leftarrow c.
\end{align*}
\]

KB

Query: \( a \) (two ways)

1. \( \text{yes} \leftarrow a. \)
   - \( u \leftarrow b \land c. \)
   - \( u \leftarrow k \land c. \)

   \( \text{K cannot be eliminated so will Fail} \)

Query: \( b \) (\( k, f \) different order)

\( \text{yes} \leftarrow b. \)
Learning Goals for today’s class – part 2

You can:

• Prove that BU proof procedure is complete

• Model a relatively simple domain with propositional definite clause logic (PDCL)

• Trace query derivation using SLD resolution rule of inference
Course Big Picture

Environment

Deterministic

Stochastic

Problem

Constraint Satisfaction

Inference

Search

vars + constraints

Arc Consistency

SLS

Belief Nets

Var. Elimination

Decision Nets

Markov Processes

Value Iteration

Static

Inference

Search

STRIPS

CSP

Reasoning Technique

Representation

Planning

Sequential
Next Class

- Finish Logics (Datalog) (12.3)
- MAJOR TRANSITION

STOCHASTIC ENVIRONMENTS

- Probability Theory and Conditional Probability (6.1)
(Propositional) Logic: Key ideas

Given a domain that can be represented with \( n \) propositions you have \( 2^n \) interpretations (possible worlds).

If you do not know anything you can be in any of those interpretations.

If you know that some logical formulas are true (your interpretations). You know that you can be only in those interpretations in which the KB is true (i.e., models of KB).

It would be nice to know what else is true in all those models what is logically entailed.
PDCL syntax / semantics / proofs

Domain can be represented by three propositions: \( p, q, r \)

\[ KB = \begin{cases} 
q \leftarrow \\
r \leftarrow \\
p \leftarrow q \land r.
\end{cases} \]

Interpretations?

<table>
<thead>
<tr>
<th></th>
<th>r</th>
<th>q</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Models?

What is logically entailed?

Prove

\[ G = (q \land p) \]

\( r, q, p \) \( G \subseteq C \)

\( C = \{ q, r, p \} \)

\( KB \uparrow_{BU} G \)

CPSC 322, Lecture 7
PDCL syntax / semantics / proofs

Interpretations

<table>
<thead>
<tr>
<th>r</th>
<th>q</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Models

What is logically entailed?

Prove \( G = (q \land p) \)

\( C = \{ q \} \)

\( G \not\subseteq C \)

KB \not\vdash G
To Define a Logic We Need

• **Syntax**: specifies the symbols used, and how they can be combined to form legal sentences
  • **Knowledge base** is a set of sentences in the language
• **Semantics**: specifies the meaning of symbols and sentences
• **Reasoning theory** or **proof procedure**: a specification of how an answer can be produced.
  • **Sound**: only generates correct answers with respect to the semantics
  • **Complete**: Guaranteed to find an answer if it exists
Propositional Definite Clauses: Syntax

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

**Examples:** $p_1$; $\text{live}_I_1$

**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.

**Examples:** $p_1 \land p_2$; $\text{ok}_w_1 \land \text{live}_w_0$

**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 $b$".)

**Examples:** $p_1 \leftarrow p_2$; $\text{live}_w_0 \leftarrow \text{live}_w_1 \land \text{up}_s_2$

**Definition (KB)**
A **knowledge base (KB)** is a set of definite clauses.
Definition (model)
A **model** of a knowledge base KB is an interpretation in which every clause in KB is true.

\[
\begin{align*}
    p & \leftarrow q \\
    q \\
    r & \leftarrow s
\end{align*}
\]

**KB =**

<table>
<thead>
<tr>
<th></th>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>I₂</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I₃</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I₄</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>I₅</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Which of the interpretations below are models of KB?

- I₁: yes
- I₂: yes
- I₃: yes
- I₄: yes
- I₅: yes
Definition (model)
A **model** of a knowledge base KB is an interpretation in which every clause in KB is true.

\[
\text{KB} = \begin{cases} 
  p \leftarrow q \\
  q \\
  r \leftarrow s 
\end{cases}
\]

Which of the interpretations below are models of KB?

<table>
<thead>
<tr>
<th></th>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
<th>Model of KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>yes</td>
</tr>
<tr>
<td>I₂</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>no</td>
</tr>
<tr>
<td>I₃</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>yes</td>
</tr>
<tr>
<td>I₄</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>yes</td>
</tr>
<tr>
<td>I₅</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>yes</td>
</tr>
</tbody>
</table>
**PDC Semantics: Example for models**

**Definition (model)**
A **model** of a knowledge base KB is an interpretation in which every clause in KB is true.

$$ KB = \begin{cases} 
  p \leftarrow q \\
  q \\
  r \leftarrow s 
\end{cases} $$

Which of the interpretations below are models of KB?
All interpretations where KB is true: \( I_1, I_3, \) and \( I_4 \)

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( r )</th>
<th>( s )</th>
<th>( p \leftarrow q )</th>
<th>( q )</th>
<th>( r \leftarrow s )</th>
<th>Model of KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>yes</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>no</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>yes</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>yes</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>no</td>
</tr>
</tbody>
</table>
Definition (model)
A model of a knowledge base KB is an interpretation in which every clause in KB is true.

Definition (logical consequence)
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

\[
KB = \begin{cases} 
    p \leftarrow q \\
    q \\
    r \leftarrow s 
\end{cases}
\]

Which of the following are true?

- \( KB \not\models p \)
- \( KB \not\models q \)
- \( KB \not\models r \)
- \( KB \not\models s \)
**PDCL Semantics: Logical Consequence**

**Definition (logical consequence)**

If KB is a set of clauses and g is a conjunction of atoms, g is a **logical consequence** of KB, written \( KB \vDash g \), if g is true in every model of KB.

\[
KB = \begin{cases} 
p \leftarrow q \\
q \\
r \leftarrow s 
\end{cases}
\]

Which of the following are true?

- \( KB \vDash p \)
- \( KB \vDash q \)

If KB is true, then q is true. Thus \( KB \vDash q \).

If KB is true then both q and \( p \leftarrow q \) are true, so p is true (“p if q”). Thus \( KB \vDash p \).

There is a model where r is false, likewise for s.
Example: Logical Consequences

<table>
<thead>
<tr>
<th>I</th>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>4</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>5</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>7</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>8</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>9</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>10</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>11</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>12</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

\[ KB = \begin{cases} 
  p \leftrightarrow q. \\
  q. \\
  r \leftrightarrow s. 
\end{cases} \]

Which of the following is true?

- \( KB \not\models q \) \( \quad \) T
- \( KB \not\models p \) \( \quad \) T
- \( KB \not\models s \) \( \quad \) F
- \( KB \not\models r \) \( \quad \) F
An exercise for you
Let’s consider these two alternative proof procedures for PDCL

A.  \( C_A = \{\text{All clauses in KB with empty bodies}\} \)

B.  \( C_B = \{\text{All atoms in the knowledge base}\} \)

\[ KB \\
\begin{align*}
  a & \leftarrow e \land g. \\
  b & \leftarrow f \land g. \\
  c & \leftarrow e. \\
  f & \leftarrow c \\
  e. \\
  d.
\end{align*} \]

Both A and B are sound and complete

Both A and B are neither sound nor complete

A is sound only and B is complete only

A is complete only and B is sound only
An exercise for you
Let’s consider these two alternative proof procedures for PDCL

A. \( C_A = \{ \text{All clauses in KB with empty bodies} \} \)

B. \( C_B = \{ \text{All atoms in the knowledge base} \} \)

A is sound only and B is complete only
Bottom-up vs. Top-down

**Bottom-up**

KB $\rightarrow$ C

$g$ is proved if $g \in C$

When does BU look at the query? $g$

- In every loop iteration
- Never
- At the end
- At the beginning
Bottom-up vs. Top-down

- **Key Idea of top-down**: search backward from a query $g$ to determine if it can be derived from $KB$. $KB \xrightarrow{\text{backward}} C$ 
  
  $g$ is proved if $g \in C$

**Bottom-up**

**Top-down**

- When does BU look at the query $g$?
  - Never
  - It derives the same $C$ regardless of the query

- TD performs a backward search starting at $g$