Logic: semantics, proof procedures, soundness and completeness

Alan Mackworth

UBC CS 322 - Logic 2
March 1, 2013

P & M Textbook §5.2
Recap: Propositional Definite Clause Logic (PDCL)
- Syntax
- Semantics

• More on PDCL Semantics

• Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    • Pseudocode and example
    • Time-permitting: Soundness
    • Time-permitting: Completeness
Course Overview

Environments
- Deterministic
- Stochastic

Problem Types
- Logic
- Planning
- Constraint Satisfaction

Course Modules
- Representation
- Reasoning
- Technique

Uncertainty
- Decision Theory

Logics
- Arc Consistency
- Search

Variables + Constraints
- Search

STRIPS
- Search
- Variables + Constraints
- As CSP (using arc consistency)

Bayesian Networks
- Variable Elimination

Decision Networks
- Variable Elimination

Markov Processes
- Value Iteration

Notes:
- Static problems, but with richer representation
Propositional definite clause logic (PDCL) is one such Representation and Reasoning System (RRS).
Example: Electrical Circuit

\[\text{light}_l_1, \text{light}_l_2, \text{ok}_l_1, \text{ok}_l_2, \text{ok}_cb1, \text{ok}_cb2, \text{live}_{\text{outside}}.\]

\[\text{live}_{\text{l}1} \leftarrow \text{live}_{\text{w}0}.\]
\[\text{live}_{\text{w}0} \leftarrow \text{live}_{\text{w}1} \land \text{up}_{\text{s}2}.\]
\[\text{live}_{\text{w}0} \leftarrow \text{live}_{\text{w}2} \land \text{down}_{\text{s}2}.\]
\[\text{live}_{\text{w}1} \leftarrow \text{live}_{\text{w}3} \land \text{up}_{\text{s}1}.\]
\[\text{live}_{\text{w}2} \leftarrow \text{live}_{\text{w}3} \land \text{down}_{\text{s}1}.\]
\[\text{live}_{\text{l}2} \leftarrow \text{live}_{\text{w}4}.\]
\[\text{live}_{\text{w}4} \leftarrow \text{live}_{\text{w}3} \land \text{up}_{\text{s}3}.\]
\[\text{live}_{\text{p}1} \leftarrow \text{live}_{\text{w}3}.\]
\[\text{live}_{\text{w}3} \leftarrow \text{live}_{\text{w}5} \land \text{ok}_{\text{cb}1}.\]
\[\text{live}_{\text{p}2} \leftarrow \text{live}_{\text{w}6}.\]
\[\text{live}_{\text{w}6} \leftarrow \text{live}_{\text{w}5} \land \text{ok}_{\text{cb}2}.\]
\[\text{live}_{\text{w}5} \leftarrow \text{live}_{\text{outside}}.\]
\[\text{lit}_{\text{l}1} \leftarrow \text{light}_{\text{l}1} \land \text{live}_{\text{l}1} \land \text{ok}_{\text{l}1}.\]
\[\text{lit}_{\text{l}2} \leftarrow \text{light}_{\text{l}2} \land \text{live}_{\text{l}2} \land \text{ok}_{\text{l}2}.\]
## Propositional Definite Clauses: Syntax

<table>
<thead>
<tr>
<th>Definition (atom)</th>
<th>Examples: ( p_1 \cdot \text{live}_l_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>An <strong>atom</strong> is a symbol starting with a lower case letter</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition (body)</th>
<th>Examples: ( p \cdot \text{ok}_w_1 \land \text{live}_w_0 \cdot p_1 \land p_2 \land p_3 \land p_4 ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>A <strong>body</strong> is an atom or is of the form ( b_1 \land b_2 ) where ( b_1 ) and ( b_2 ) are bodies.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition (definite clause)</th>
<th>Examples: ( p \cdot p_1 \leftarrow p_2 \land p_3 \land p_4 ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>A <strong>definite clause</strong> is an atom or is 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 )’.)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition (KB)</th>
<th>Example: {p_2, p_3, p_4, p_1 \leftarrow p_2 \land p_3 \land p_4, \text{live}_l_1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>A <strong>knowledge base (KB)</strong> is a set of definite clauses</td>
<td></td>
</tr>
</tbody>
</table>
atoms

definite clauses, KB

rules
Lecture Overview

• Recap: Propositional Definite Clause Logic (PDCL)
  - Syntax
  - Semantics
• More on PDCL Semantics
• Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    • Pseudocode and example
    • Time-permitting: Soundness
    • Time-permitting: Completeness
Propositional Definite Clauses: Semantics

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

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

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 \( I \).
- A rule \( h \leftarrow b \) is false in \( I \) if and only if \( b \) is true in \( I \) and \( h \) is false in \( I \).
PDC Semantics: Example

Truth values under different interpretations
F=false, T=true

<table>
<thead>
<tr>
<th>a₁</th>
<th>a₂</th>
<th>a₁ ∧ a₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I₂</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>I₃</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>I₄</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>h</th>
<th>b</th>
<th>¬b</th>
<th>¬b ∨ h</th>
<th>h ← b</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>I₂</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I₃</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>I₄</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

h ← b ("h if b") is only false if b is true and h is false
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?

<table>
<thead>
<tr>
<th></th>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
<th>p ← q</th>
<th>q</th>
<th>r ← 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>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>F</td>
<td>F</td>
<td>T</td>
<td>no</td>
</tr>
<tr>
<td>I₃</td>
<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>I₄</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>yes</td>
</tr>
<tr>
<td>I₅</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</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 = \[
\begin{align*}
  p & \leftarrow q \\
  q & \\
  r & \leftarrow s
\end{align*}
\]

Which of the interpretations below are models of KB?

<table>
<thead>
<tr>
<th>I_1</th>
<th>I_2</th>
<th>I_3</th>
<th>I_4</th>
<th>I_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
<th>p ← q</th>
<th>q</th>
<th>r ← 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>F</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>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>no</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>no</td>
</tr>
</tbody>
</table>


Lecture Overview

• Recap: Propositional Definite Clause Logic (PDCL)
  - Syntax
  - Semantics

More on PDCL Semantics

• Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    • Pseudocode and example
    • Time-permitting: Soundness
    • Time-permitting: Completeness
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.

- We also say that g logically follows from KB, or that KB entails g.
- In other words, $KB \not\models g$ if there is no interpretation in which KB is true and g is false.
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?
- $KB \not\models p$
- $KB \models q$
- $KB \models r$
- $KB \not\models s$
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.

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

If KB is true, then q is true. Thus $KB \models q$.

If KB is true then both q and $p \leftarrow q$ are true, so p is true ("p if q"). Thus $KB \models p$.

There is a model where r is false, likewise for s (but there is no model where s is true and r is false).
Motivation for Proof Procedure

- We want a proof procedure that can find all and only the logical consequences of a knowledge base

- Why?
User’s View of Semantics

1. Choose a task domain: intended interpretation.
2. Associate an atom with each proposition you want to represent.
3. Tell the system clauses that are true in the intended interpretation: axiomatizing the domain.
4. Ask questions about the intended interpretation.
   – If $\text{KB} \models g$, then $g$ must be true in all models, so it is true in the intended interpretation, which is a model.
   – The user can interpret the answer using their intended interpretation of the symbols.
Computer’s view of semantics

• The computer doesn’t have access to the intended interpretation.
  – All it knows is the knowledge base.

• The computer can determine if a formula is a logical consequence of KB.
  – If $\text{KB} \models g$ then $g$ must be true in the intended interpretation.
  – Otherwise, there is a model of KB in which $g$ is false.
    This could be the intended interpretation.
Role of semantics

In user’s mind:
• l2_broken: light l2 is broken
• sw3_up: switch is up
• power: there is power in the building
• unlit_l2: light l2 isn't lit
• lit_l1: light l1 is lit

In computer:
• l2_broken ← sw3_up ∧ power ∧ unlit_l2.
• sw3_up.
• power ← lit_l1.
• unlit_l2.
• lit_l1.

Conclusion: l2_broken
- The computer doesn’t know the meaning of the symbols
- The user can interpret the symbols using their meaning
Lecture Overview

• Recap: Propositional Definite Clause Logic (PDCL)
  - Syntax
  - Semantics
• More on PDCL Semantics

Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    • Pseudocode and example
    • Time-permitting: Soundness
    • Time-permitting: Completeness
Proofs

• A proof is a mechanically derivable demonstration that a formula logically follows from a knowledge base.
• Given a proof procedure P, $\text{KB} \vdash_P g$ means g can be derived from knowledge base KB with the proof procedure.
• Recall $\text{KB} \models g$ means g is true in all models of KB.

• Example: simple proof procedure S
  – Enumerate all interpretations
  – For each interpretation I, check whether all clauses in KB hold
    • If all clauses are true, I is a model
    • $\text{KB} \vdash_S g$ if g holds in all such models
Soundness of a proof procedure

Definition (soundness)
A proof procedure P is sound if KB ├ P g implies KB ⊨ g.

sound: everything it derives follows logically from KB (i.e. is true in every model)

• Soundness of some proof procedure P: need to prove that
  
  If g can be derived by the procedure (KB ├ P g) then g is true in all models of KB (KB ⊨ g)

• Example: simple proof procedure S
  – For each interpretation I, check whether all clauses in KB hold
    • If all clauses are true, I is a model
    • KB ├ S g if g holds in all such models

• The simple proof procedure S is sound:
  
  If KB ├ S g, then it is true in all models, i.e. KB ⊨ g
Completeness of a proof procedure

**Definition (completeness)**
A proof procedure $P$ is **complete** if $KB \models g$ implies $KB \vdash_P g$.

complete: everything that logically follows from $KB$ is derived

- Completeness of some proof procedure $P$: need to prove that
  
  **If** $g$ is true in all models of $KB$ ($KB \models g$)  
  **then** $g$ is derived by the procedure ($KB \vdash_P g$)

- Example: simple proof procedure $S$
  - For each interpretation $I$, check whether all clauses in $KB$ hold
    - If all clauses are true, $I$ is a model
    - $KB \vdash_S g$ if $g$ holds in all such models

- The **simple proof procedure $S$** is complete:
  
  **If** $KB \models g$, i.e. $g$ is true in all models, **then** $KB \vdash_S g$
Another example for a proof procedure

- Unsound proof procedure U:
  - U derives every atom: for any g, KB $\vdash_U g$

- Proof procedure U is complete:
  
  If $\text{KB} \vdash g$, then $\text{KB} \vdash_S g$ (because $\text{KB} \vdash_U g$ for any g)

- Proof procedure U is not sound:
  
  Proof by counterexample: $\text{KB} = \{ a \leftarrow b. \}$

  $\text{KB} \vdash_U a$, but not $\text{KB} \not\vdash a$

  (a is false in some model, e.g. a=false, b=false)
Problem of the simple proof procedure S

• Simple proof procedure: enumerate all interpretations
  – For each interpretation, check whether all clauses in KB hold
    • If all clauses hold, the interpretation is a model
    • KB ⊨ g if g holds in all such models

• What’s the problem with this approach?

  Space complexity  Time complexity  Not sound  Not complete
Problem of the simple proof procedure S

- Enumerate all interpretations
  - For each interpretation, check whether all clauses of the knowledge base hold
  - If all clauses hold, the interpretation is a model

- Very much like the generate-and-test approach for CSPs

- Sound and complete, but there are a lot of interpretations
  - For n propositions, there are $2^n$ interpretations
Lecture Overview

• Recap: Propositional Definite Clause Logic (PDCL)
  - Syntax
  - Semantics

• More on PDCL Semantics

• Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    • Pseudocode and example
    • Time-permitting: Soundness
    • Time-permitting: Completeness
Bottom-up 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.

• This rule also covers the case when $m = 0$. 
Bottom-up proof procedure

\[
\begin{align*}
C & := \{\}; \\
\text{repeat} \\
\quad \text{select} \quad \text{clause } h & \leftarrow b_1 \land \ldots \land b_m \text{ in } \text{KB} \\
\quad & \text{such that } b_i \in C \text{ for all } i, \text{ and } h \notin C; \\
\quad C & := C \cup \{h\} \\
\text{until} \quad \text{no more clauses can be selected.}
\end{align*}
\]

\[
\text{KB} \vdash g \text{ if } g \in C \text{ at the end of this procedure.}
\]
Bottom-up proof procedure: example

C := {};
repeat
    select clause h ← b₁ ∧ … ∧ bₘ in KB
    such that bᵢ ∈ C for all i, and h ∉ C;
    C := C ∪ {h}
until no more clauses can be selected.

a ← b ∧ c
a ← e ∧ f
b ← f ∧ k
c ← e
d ← k
e.
f ← j ∧ e
f ← c
j ← c
Bottom-up proof procedure: example

\[
\begin{align*}
C & := \{\}; \\
\text{repeat} & \\
\hspace{1cm} \textbf{select} & \text{ clause } h \leftarrow b_1 \land \ldots \land b_m \text{ in } KB \\
& \quad \text{such that } b_i \in C \text{ for all } i, \text{ and } h \not\in C; \\
C & := C \cup \{h\} \\
\textbf{until} & \text{ no more clauses can be selected.}
\end{align*}
\]

\[
\begin{align*}
a & \leftarrow b \land c \quad \{\} \\
a & \leftarrow e \land f \quad \{e\} \\
b & \leftarrow f \land k \quad \{c,e\} \\
c & \leftarrow e \quad \{c,e,f\} \\
d & \leftarrow k \quad \{c,e,f,j\} \\
e. & \\
f & \leftarrow j \land e \quad \{a,c,e,f,j\} \\
f & \leftarrow c \\
j & \leftarrow c \quad \text{Done.}
\end{align*}
\]
Lecture Overview

• Recap: Propositional Definite Clause Logic (PDCL)
  - Syntax
  - Semantics

• More on PDCL Semantics

• Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    • Pseudocode and example

Time-permitting: Soundness
• Time-permitting: Completeness
Soundness of bottom-up proof procedure BU

**Definition (soundness)**
A proof procedure $P$ is **sound** if $\text{KB} \vdash_P g$ implies $\text{KB} \not\models g$.

*sound*: everything it derives follows logically from $\text{KB}$ (i.e. is true in every model)

```plaintext
C := {}; repeat
    select clause $h \leftarrow b_1 \land \ldots \land b_m$ in $\text{KB}$ such that $b_i \in C$ for all $i$, and $h \notin C$;
    $C := C \cup \{h\}$
until no more clauses can be selected.
```

For **soundness of bottom-up proof procedure BU**:

- **If** $g \in C$ at the end of BU procedure,
- **then** $g$ is true in all models of $\text{KB}$ ($\text{KB} \not\models g$)
Soundness of bottom-up proof procedure BU

\[
\begin{align*}
C & := \{\}; \\
\text{repeat} & \\
& \quad \textbf{select} \text{ clause } h \leftarrow b_1 \land \ldots \land b_m \text{ in KB} \\
& \quad \quad \text{such that } b_i \in C \text{ for all } i, \text{ and } h \notin C; \\
& \quad C := C \cup \{h\} \\
\text{until} & \text{ no more clauses can be selected.}
\end{align*}
\]

For soundness of bottom-up proof procedure BU: prove

\[\text{If } g \in C \text{ at the end of BU procedure, then } g \text{ is true in all models of KB (KB } \models g)\]

By contradiction: Suppose there is a \( g \) such that \( \text{KB } \vdash g \) but not \( \text{KB } \models g \).

- Let \( h \) be first atom added to \( C \) that’s not true in every model of \( \text{KB} \).
  - In particular, suppose \( I \) is a model of \( \text{KB} \) in which \( h \) isn’t true.
  - There must be a clause in \( \text{KB} \) of form \( h \leftarrow b_1 \land \ldots \land b_m \)
  - Each \( b_i \) is true in \( I \). \( h \) is false in \( I \). So this clause is false in \( I \).
  - Thus, \( I \) is not a model of \( \text{KB} \). Contradiction: thus no such \( g \) exists.
Lecture Overview

- Recap: Propositional Definite Clause Logic (PDCL)
  - Syntax
  - Semantics

- More on PDCL Semantics

- Proof procedures
  - Soundness, Completeness, example
  - Bottom-up proof procedure
    - Pseudocode and example
    - Time-permitting: Soundness
  Time-permitting: Completeness
Minimal Model

- Observe that the C generated at the end of the bottom-up algorithm is a fixed point
  - Further applications of our rule of derivation will not change C!

**Definition (minimal model)**
The minimal model MM is the interpretation in which every element of BU’s fixed point C is true and every other atom is false.

- Lemma: MM is a model of KB.
  - Proof by contradiction. Assume that MM is not a model of KB.
    - Then there must exist some clause of the form \( h \leftarrow b_1 \land \ldots \land b_m \) in KB (with \( m \geq 0 \)) which is false in MM.
    - This can only occur when \( h \) is false (and not in C) and each \( b_i \) is true in MM.
    - Since each \( b_i \) belonged to C, we would have added \( h \) to C as well.
    - But MM is a fixed point, so nothing else gets added. Contradiction!
Completeness of bottom-up procedure

**Definition (completeness)**
A proof procedure is **complete** if $KB ⊨ g$ implies $KB ⊢ g$.

Complete: everything that logically follows from $KB$ is derived

For completeness of BU, we need to prove:

**If** $g$ is true in all models of $KB$ ($KB \models g$)

**then** $g$ is derived by the BU procedure ($KB \vdash_{BU} g$)

Direct proof based on Lemma about minimal model:

- Suppose $KB \not\models g$. Then $g$ is true in all models of $KB$.
- Thus $g$ is true in the minimal model.
- Thus $g$ is generated by the bottom up algorithm.
- Thus $KB \vdash_{BU} g$. 
• PDCL syntax & semantics
  – Verify whether a logical statement belongs to the language of propositional definite clauses
  – Verify whether an interpretation is a model of a PDCL KB.
  – Verify when a conjunction of atoms is a logical consequence of a knowledge bases

• Bottom-up proof procedure
  • Define/read/write/trace/debug the Bottom Up (BU) proof procedure
  • Prove that the BU proof procedure is sound and complete