Logic: semantics, proof procedures, soundness and completeness

Alan Mackworth

UBC CS 322 - Logic 2

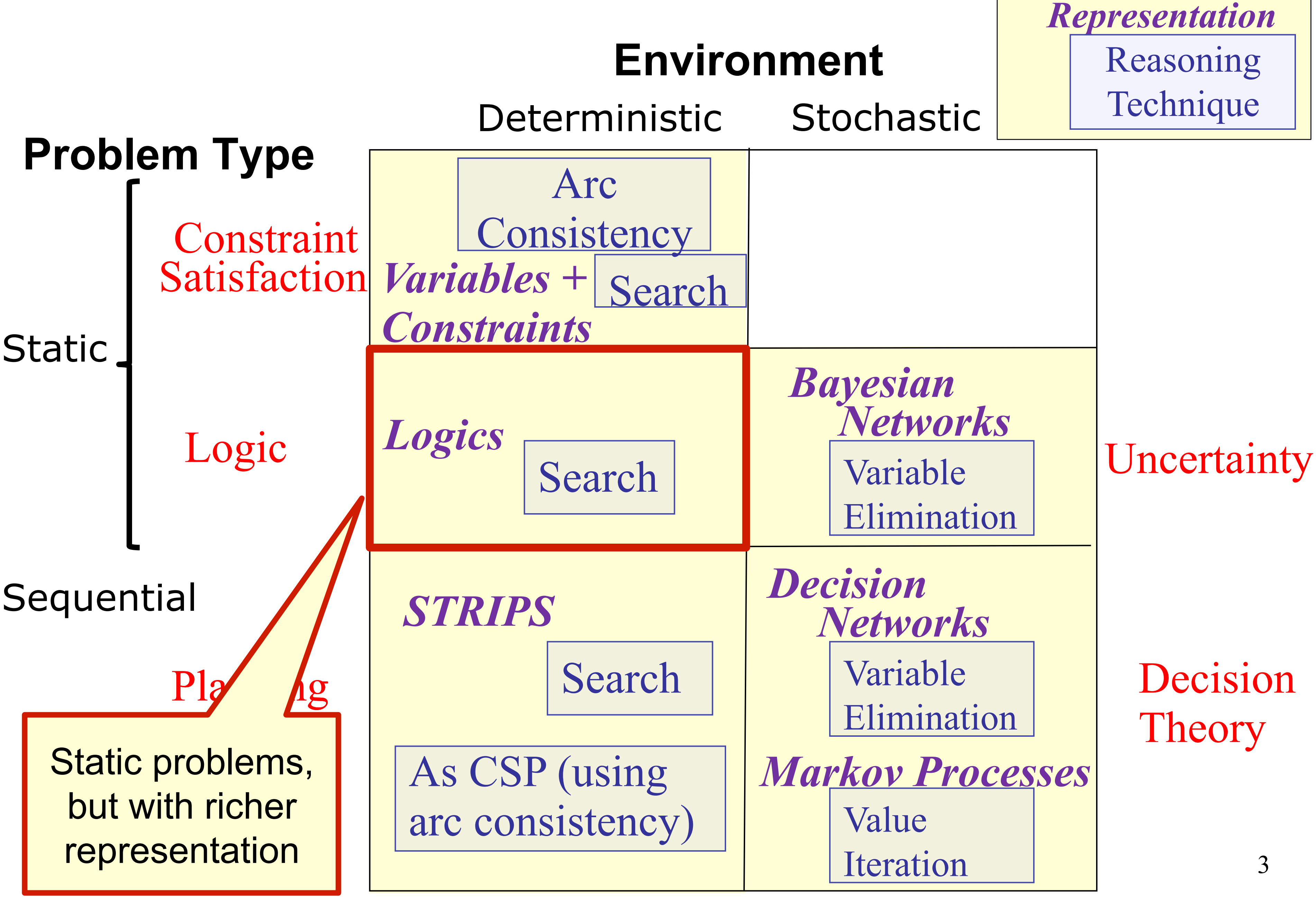
March 1, 2013

P & M Textbook §5.2

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

Course Overview



Representation and Reasoning System (RRS)

Definition (RRS)
A Representation and Reasoning System (RRS) consists of:

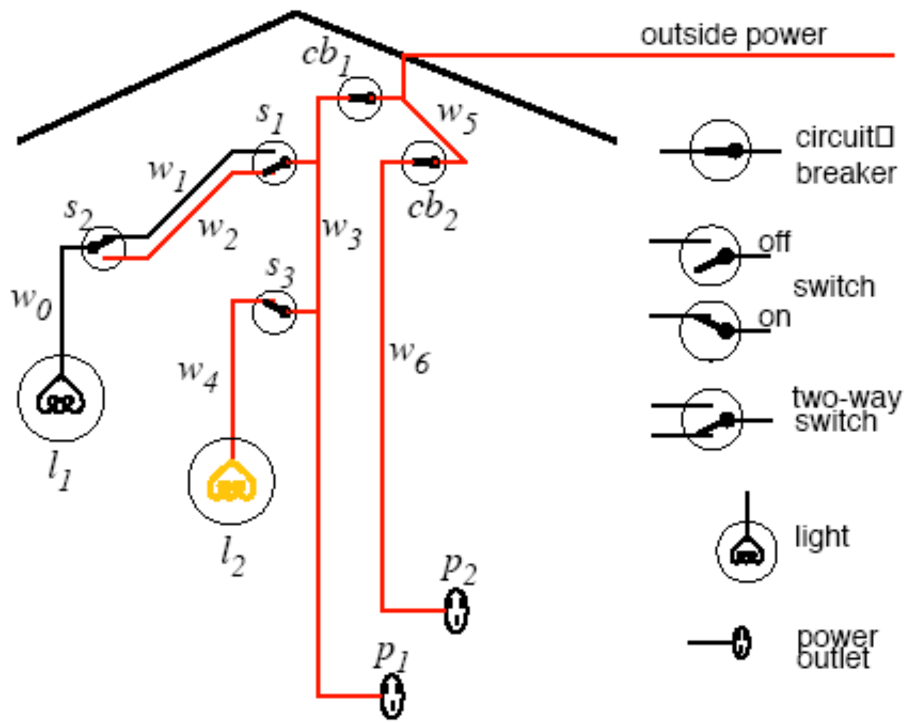
- syntax: specifies the symbols used, and how they can be combined to form legal sentences
- semantics: specifies the meaning of the symbols
- reasoning theory or proof procedure: a (possibly nondeterministic) specification of how an answer can be produced.

Propositional definite clause logic (PDCL) is one such Representation and Reasoning System

Example: Electrical Circuit

$light\_l_1.$
$light\_l_2.$
$ok\_l_1.$
$ok\_l_2.$
$ok\_cb_1.$
$ok\_cb_2.$
$live\_outside.$

$live\_l_1 \leftarrow live\_w_0.$
$live\_w_0 \leftarrow live\_w_1 \wedge up\_s_2.$
$live\_w_0 \leftarrow live\_w_2 \wedge down\_s_2.$
$live\_w_1 \leftarrow live\_w_3 \wedge up\_s_1.$
$live\_w_2 \leftarrow live\_w_3 \wedge down\_s_1.$
$live\_l_2 \leftarrow live\_w_4.$
$live\_w_4 \leftarrow live\_w_3 \wedge up\_s_3.$
$live\_p_1 \leftarrow live\_w_3.$
$live\_w_3 \leftarrow live\_w_5 \wedge ok\_cb_1.$
$live\_p_2 \leftarrow live\_w_6.$
$live\_w_6 \leftarrow live\_w_5 \wedge ok\_cb_2.$
$live\_w_5 \leftarrow live\_outside.$
$lit\_l_1 \leftarrow light\_l_1 \wedge live\_l_1 \wedge ok\_l_1.$
$lit\_l_2 \leftarrow light\_l_2 \wedge live\_l_2 \wedge ok\_l_2.$



Propositional Definite Clauses: Syntax

Definition (atom)

Examples: p_1. $live\_l_1$

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

Definition (body)

Examples: p. $ok\_w_1 \wedge live\_w0$. $p_1 \wedge p_2 \wedge p_3 \wedge p_4$.

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)

Examples: p. $p_1 \leftarrow p_2 \wedge p_3 \wedge p_4$.
$live\_w_0 \leftarrow live\_w_1 \wedge up\_s_2$

A **definite clause** 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*'.)

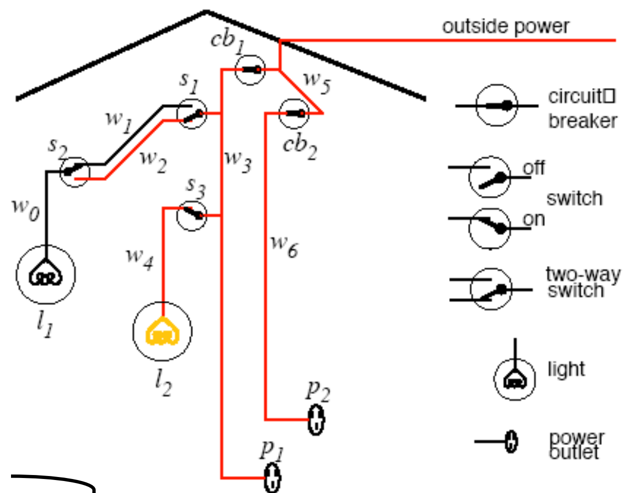
Definition (KB)

Example: $\{p_2.\ p_3.\ p_4.\ p_1 \leftarrow p_2 \wedge p_3 \wedge p_4.\ live\_l_1\}$

A **knowledge base (KB)** is a set of definite clauses

$light\_l_1.$
$light\_l_2.$
$ok\_l_1.$
$ok\_l_2.$
$ok\_cb_1.$
$ok\_cb_2.$
$live\_outside.$

atoms



definite clauses, KB

$live\_l_1 \leftarrow live\_w_0.$
$live\_w_0 \leftarrow live\_w_1 \wedge up\_s_2.$
$live\_w_0 \leftarrow live\_w_2 \wedge down\_s_2.$
$live\_w_1 \leftarrow live\_w_3 \wedge up\_s_1.$
$live\_w_2 \leftarrow live\_w_3 \wedge down\_s_1.$
$live\_l_2 \leftarrow live\_w_4.$
$live\_w_4 \leftarrow live\_w_3 \wedge up\_s_3.$
$live\_p_1 \leftarrow live\_w_3.$
$live\_w_3 \leftarrow live\_w_5 \wedge ok\_cb_1.$
$live\_p_2 \leftarrow live\_w_6.$
$live\_w_6 \leftarrow live\_w_5 \wedge ok\_cb_2.$
$live\_w_5 \leftarrow live\_outside.$
$lit\_l_1 \leftarrow light\_l_1 \wedge live\_l_1 \wedge ok\_l_1.$
$lit\_l_2 \leftarrow light\_l_2 \wedge live\_l_2 \wedge ok\_l_2.$

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

	a_1	a_2	$a_1 \wedge a_2$
I_1	F	F	F
I_2	F	T	F
I_3	T	F	F
I_4	T	T	T

	h	b	$\neg$b	$\neg b \vee h$	$h \leftarrow b$
I_1	F	F	T	T	T
I_2	F	T	F	F	F
I_3	T	F	T	T	T
I_4	T	T	F	T	T

$h \leftarrow b$ ("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?

	p	q	r	s	$p \leftarrow q$	q	$r \leftarrow s$	Model of KB	
I_1	T	T	T	T	T	T	T	yes	no
I_2	F	F	F	F	T	F	T	yes	no
I_3	T	T	F	F	T	T	T	yes	no
I_4	F	T	T	F	F	T	T	yes	no
I_5	T	T	F	T	T	T	F	yes	no

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?

	p	q	r	s	$p \leftarrow q$	q	$r \leftarrow s$	Model of KB
I_1	T	T	T	T	T	T	T	yes
I_2	F	F	F	F	T	F	T	no
I_3	T	T	F	F	T	T	T	yes
I_4	F	T	T	F	F	T	T	no
I_5	T	T	F	T	T	T	F	no

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

PDCL Semantics: Logical Consequence

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 $\models$ g if there is no interpretation in which KB is true and g is false

PDCL Semantics: Logical Consequence

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?

$KB \models p$ $KB \models q$ $KB \models r$ $KB \models s$

PDCL Semantics: Logical Consequence

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

$$KB = \left\{ \begin{array}{l} p \leftarrow q \\ q \\ r \leftarrow s \end{array} \right.$$

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 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 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 $\leftarrow$ sw3_up $\wedge$ power $\wedge$ unlit_l2.
- sw3_up.
- power $\leftarrow$ 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, $KB \vdash_P g$ means g can be derived from knowledge base KB with the proof procedure.
- Recall $KB \models g$ 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
    - $KB \vdash_S g$ if g holds in all such models

Soundness of a proof procedure

Definition (soundness)

A proof procedure P is sound if KB $\vdash_P$ g implies KB $\models$ 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 $\vdash_P$ g)
  then g is true in all models of KB (KB $\models$ 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 sound:

  If KB $\vdash_S$ g, then it is true in all models, i.e. KB $\models$ 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 KB $\models$ g, **then** KB $\vdash_S$ g (because KB $\vdash_U$ g for any g)

- Proof procedure U is not sound:

Proof by counterexample: KB = {a $\leftarrow$ b.}

KB $\vdash_U$ a, but not KB $\models$ 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 $\vdash$ 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 "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" 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

$C := \{\};$

repeat

    select clause $h \leftarrow b_1 \wedge \ldots \wedge b_m$ 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.

KB $\vdash$ g if $g \in C$ at the end of this procedure.

Bottom-up proof procedure: example

C := {};
repeat
  select clause $h \leftarrow b_1 \wedge \ldots \wedge b_m$ 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.

$a \leftarrow b \wedge c$

$a \leftarrow e \wedge f$

$b \leftarrow f \wedge k$

$c \leftarrow e$

$d \leftarrow k$

e.

$f \leftarrow j \wedge e$

$f \leftarrow c$

$j \leftarrow c$

{}

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.
```

$a \leftarrow b \wedge c$

$a \leftarrow e \wedge f$

$b \leftarrow f \wedge k$

$c \leftarrow e$

$d \leftarrow k$

$e.$

$f \leftarrow j \wedge e$

$f \leftarrow c$

$j \leftarrow c$

{}

{e}

{c,e}

{c,e,f}

{c,e,f,j}

{a,c,e,f,j}

Done.

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 KB $\vdash_P$ g implies KB $\models$ g.

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

```
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.
```

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 KB (KB $\models$ g)

Soundness of bottom-up proof procedure BU

$C := \{\};$
repeat
    select clause $h \leftarrow b_1 \wedge \ldots \wedge b_m$ 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.

For soundness of bottom-up proof procedure BU: prove

If $g \in C$ at the end of BU procedure,
then g is true in all models of KB ($KB \models g$)

By contradiction: Suppose there is a g such that $KB \vdash g$ but not $KB \models g$.

– Let h be first atom added to C that's not true in every model of KB
  - In particular, suppose I is a model of KB in which h isn't true.
– There must be a clause in KB of form $h \leftarrow b_1 \wedge \ldots \wedge b_m$
– 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 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 $h \leftarrow b_1 \wedge \ldots \wedge b_m$ 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 $\models$ g implies KB $\vdash$ 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 $\models$ 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.

Learning Goals Up To Here

- 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