Example

\[ \text{KB}: \text{female(sally)} \]

\[ \text{Prove } \text{KB} \models \text{person(sally)} \]

What does \( \text{KB} \models \text{person(sally)} \) mean?

- Means that if interpretation \( I \) models \( \text{KB} \) then it models \( \text{person(sally)} \).

- Could be proven by checking all interpretations.

Let's do proof by contradiction instead.

- Assume that \( I \) is a model of \( \text{KB} \) but not a model of \( \text{person(sally)} \).

Review

- An interpretation maps any clause to either true or false.
- It is a complete mapping.
- A model \( I \) of \( \text{KB} \) is an interpretation that maps every clause in \( \text{KB} \) to true.
- \( \text{KB} \models g \) iff every model of \( \text{KB} \) makes \( g \) true.

Overview

- Semantics
- Queries
- Proof Procedures
- Bottom-up Ground Proof Procedure
- Top-down Ground Proof Procedure

Questions
Overview

• Semantics ⇒ Queries

• Proof Procedures

• Bottom-up Ground Proof Procedure

• Top-down Ground Proof Procedure

More on Variables in Clauses (pg. 42)

• Say parent(X,Y) ← father(X,Y) is in KB
  - Implicit universal quantifiers around it
  - Anytime that father(X,Y) is true, so must parent(X,Y)

• Say grandfather(X,Y) ← father(X,Z) ∧ parent(Z,Y) is in KB
  - This clause is true for all X, Y, Z
  - ∀X Y Z (grandfather(X,Y) ← father(X,Z) ∧ parent(Z,Y)).
  - Z only appears in the body

• How does Z work here (variable just in the body)?
  - For any X and Y, if we find a Z that makes body true, head must be true
  - Now it seems that Z is just existentially quantified
  - We just need to find one Z for each X and Y, not ensure it is true for all Z

Proof by Contradiction: A Semantic Proof

• Assume I = \{D, φ, π\} is a model of KB = \{female(sally) \lor person(X) ← female(X)\}
  - So \(<φ(sally)> ∈ π(female)\)
  - Say φ(sally) = F, so <F> ∈ π(female) (1)
  - person(X) ← female(X) must be true for I ρ for any var. assignment ρ (2)

• Assume person(sally) is not true under I
  - So <F> \notin π(person) (3)

• Consider variable assignment ρ where ρ(X) = F
  - If female(X) is true for I ρ then person(X) must be true for I ρ (from (2)) (4)
  - ρ(X) = F and <F> ∈ π(female) so female(X) is true for I ρ (5)
  - So person(X) must be true for I ρ (from (4) and (5))
  - ρ(X) = F so <F> ∈ π(person) (6)

• Contradiction from (3) and (6)
Overview

• Semantics
• Queries ⇒ Proof Procedures

• Bottom-up Ground Proof Procedure
• Top-down Ground Proof Procedure

Queries with Variables

• You might not only want to check if something is true or false, but what value makes it true.
  - KB:
    - father(william,ted)
    - parent(X,Y) ← father(X,Y)

• Example:
  - ?parent(X,ted)
  - Who is Ted's parent?
  - Could transform this to yes ← parent(X,ted)
  - But, let's capture the variables in the body:

Ground Queries

• A query is a way to ask if a body is a logical consequence of KB:

  ? b₁ ∧ ... ∧ bₘ

• Ground query (no variables) has the answer:
  - "yes" if the body is a logical consequence of the KB
  - "no" if the body is not a logical consequence of the KB

• We do not distinguish between it being false in all models or just some.

• You might not only want to check if something is true or false.

Ground Queries

• This Rests Your Queries as Finding Consequences From KB
  - Check if \( \exists x, \text{father}(x,\text{ted}) \) in KB
  - Add (temporarily) \( \text{yes}(\text{yes}(\text{yes}(\text{yes}))) \) to KB
  - Transform query
  - yes ← yes ← yes ← yes ...

  • Can do query-answering:
  
  • Knowledge base, yes \( \models \) yes if yes is a logical consequence of the KB
  
  • Yes is a way to ask if a body is a logical consequence of the KB
Proof Procedures

- Proof:
a mechanically derivable demonstration that a formula logically follows from a KB
- Proof procedure:
an algorithm that constructs proofs
- \( \text{KB} \vdash g \) means \( g \) can be derived from \( \text{KB} \) with the proof procedure
- Proof procedure can be nondeterministic
- To do so in practice we need an actual implementation

Properties of Proof Procedure

- Soundness: if \( \text{KB} \vdash g \) then \( \text{KB} \models g \)
- Completeness: if \( \text{KB} \models g \) then \( \text{KB} \vdash g \)

Terminology:
- Semantic proof: \( \models \), logically follows, logically entails, models
- Syntactic proof: \( \vdash \), derives

Semantics Is Not Enough

- We have KB - We know what conclusions are valid to make
- KB \( \models g \) iff \( g \) is true in all models of KB
- Can extend this so user can ask queries with variables as well
- But, don't yet have a mechanical way of checking if KB \( \models g \)
- Checking all interpretations is very expensive

- We can access the KB.
Non-deterministic Specification

• Haven't specified the exact order that things should be done in
• What order should we pick clauses from $KB$ to try?

Bottom-up Ground Proof Procedure

• For now, only consider ground facts and ground rules
• Bottom-up or forward chaining procedure starts from $KB$ and works towards query
• Forward chaining rule:
  - If $h \leftarrow b_1 \land \ldots \land b_m$ is a clause in $KB$ and each $b_i$ has been derived, then $h$ can be derived
  - If $h$ is a fact in $KB$, then $h$ can be derived
  - Call the set of derivables the consequence set ($C$)

Semantics

• Queries
• Proof Procedures
• Bottom-up Ground Proof Procedure
• Top-down Ground Proof Procedure

Overview
Is it Complete?

• Does \( C \) have every ground atom that logically follows from \( KB \)?
• We need to prove something about consequence sets
• Let \( C \) be the final consequent set generated by the algorithm
  - Will stop because finite number of constants and predicate symbols
  - Will stop with same \( C \), no matter what order \( C \) was generated

Define \( I \) such that for atom \( h \)
  - \( I(h) \) is true if \( h \in C \)
  - Otherwise, \( I(h) \) is false
  - \( I \) is an interpretation because it defines a subset of ground atoms as true, and the rest as false

• \( I \) is an interpretation, but is it a model of \( KB \)?
  - i.e., for every \( g \in KB \), is \( I(g) \) true?
  - Proof by contradiction: assume \( KB \vdash g \) but \( KB \nvDash g \)
    - \( g \) is the result of a finite number of derivations
    - Without loss of generality, assume \( g \) is first one in derivation such that \( KB \nvDash g \)
    - Now \( g \) was derived by a cause
      - \( g \leftarrow b_1 \land \ldots \land b_m \) in \( KB \) where the \( b_i \)’s have already been derived
    - Since \( g \) was derived, all \( b_i \)’s logically follow from \( KB \)
    - So \( b_1 \land \ldots \land b_m \) logically follows from \( KB \) (from definition of \( \land \))
  - \( g \leftarrow b_1 \land \ldots \land b_m \) logically follows from \( KB \) since it is in \( KB \)
  - Using definition of \( \leftarrow \), can show that \( g \) must logically follow from \( KB \)
  - Contradiction

Is it Sound?

• Does \( C \) have every ground atom that logically follows from \( KB \)?

Example:

- \( a \leftarrow b \land c \)
- \( b \leftarrow d \land e \)
- \( b \leftarrow g \land e \)
- \( c \leftarrow e \)
- \( d \)
- \( e \)

- What is the consequence set?
- Does \( C \) contain \( a, b, c \)?

- Does \( C \) contain \( g \)?
- Does \( C \) contain \( q \)?
- Does \( C \) contain \( a, b \)?
Overview

- Semantics
- Queries
- Proof Procedures
  - Bottom-up Ground Proof Procedure
  - Top-down Ground Proof Procedure
  - Procedures
  - Queries

Final Step in Completeness Proof

- Let $g$ be atomic and $\text{KB} \models g$
- Need to make sure that $\text{KB} \vdash g$
- Since $\text{KB} \models g$, $g$ must be in every model of $\text{KB}$
- So, $g$ is in the interpretation defined by the consequence set
- Since $g$ is atomic and it is true in the interpretation, $g$ must be in the consequence set
- So, it is in the interpretation defined by the consequence set
- Let $\text{KB} \models g$
- Since $\text{KB} \models g$, $g$ must be in every model of $\text{KB}$
- $g$ is in the interpretation defined by the consequence set
- Since $g$ is atomic and it is true in the interpretation, $g$ must be in the consequence set
- So, $g$ is in the interpretation defined by the consequence set
- $g$ is in the consequence set
- Hence $\text{KB} \models g$

Proof that Consequence Set is a Model

- Proof by Contradiction: Let $g \in \text{KB}$ but where $I(g)$ is false
  - Since $g \in \text{KB}$, $g$ must have the form $h \leftarrow b_1 \land \ldots \land b_m$
  - So $h \leftarrow b_1 \land \ldots \land b_m$ is false in $I$
  - Remember, definition of $\leftarrow$ comes from Datalog, not $I$
  - So $h$ must be false in $I$ and $b_1 \land \ldots \land b_m$ must be true in $I$
  - If $b_1 \land \ldots \land b_m$ is true in $I$, each individually must be true in $I$
  - Remember, definition of $\land$ comes from Datalog, not $I$
  - So, all of the $b_i$ must be in $C$ (due to how we defined $I$)
  - Since all $b_i$ in $C$ and $h \leftarrow b_1 \land \ldots \land b_m$ is in $\text{KB}$, bottom-up algorithm must have applied this rule and hence $h \in C$
  - Hence $h$ is true in $I$

Final Step in Completeness Proof

- Hence $\text{KB} \models g$
- Since $\text{KB} \models g$, $g$ must be in every model of $\text{KB}$
- So all of the $b_i$ must be in $C$ (due to how we defined $I$
- Remember, definition of $\leftarrow$ comes from Datalog, not $I$
- So $h \leftarrow b_1 \land \ldots \land b_m$ is false in $I$
- Remember, definition of $\land$ comes from Datalog, not $I$
- So, all of the $b_i$ must be in $C$ (due to how we defined $I$
- Since all $b_i$ in $C$ and $h \leftarrow b_1 \land \ldots \land b_m$ is in $\text{KB}$, bottom-up algorithm must have applied this rule and hence $h \in C$
- Hence $h$ is true in $I$

Proof that Consequence Set is a Model

- Hence $\text{KB} \models g$
- Since $\text{KB} \models g$, $g$ must be in every model of $\text{KB}$
- So all of the $b_i$ must be in $C$ (due to how we defined $I$
- Remember, definition of $\leftarrow$ comes from Datalog, not $I$
- So $h \leftarrow b_1 \land \ldots \land b_m$ is false in $I$
- Remember, definition of $\land$ comes from Datalog, not $I$
- So, all of the $b_i$ must be in $C$ (due to how we defined $I$
- Since all $b_i$ in $C$ and $h \leftarrow b_1 \land \ldots \land b_m$ is in $\text{KB}$, bottom-up algorithm must have applied this rule and hence $h \in C$
- Hence $h$ is true in $I$
Example

Now for some definitions

Top-Down Ground Proof Procedure

- Alternative to bottom-up (forward-chaining)
- Top-down (backward-chaining)
  - Start with goal, work toward facts in KB

Definite Clause Resolution for Ground Case

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

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

\[
\text{yes} \leftarrow a_1 \land \ldots \land a_{i-1} \land b_1 \land \ldots \land b_p \land a_{i+1} \land \ldots \land a_m
\]
Any top-down proof can be converted to a bottom-up proof.

Any bottom-up proof can be converted to a top-down proof.

There are many other ways of doing proofs - e.g. Unit resolution - We will explore some of these later in the course.