Overview

⇒ Semantics

• Queries

• Proof Procedures

• Bottom-up Ground Proof Procedure

• Top-down Ground Proof Procedure
Review

• An interpretation maps any clause to either true or false
  - It is a complete mapping

• A model $I$ of $KB$ is an interpretation that maps every clause in $KB$ to true

• $KB \models g$ iff every model of $KB$ makes $g$ true
Example

- **$KB$:**
  
  \[
  \text{female(sally)}
  \]
  
  \[
  \text{person}(X) \leftarrow \text{female}(X)
  \]

- Prove $KB \models \text{person}(sally)$

- What does $KB \models \text{person}(sally)$ mean?
  
  - Means that if interpretation $I$ models $KB$ then it models $\text{person}(sally)$
  
  - Could prove this by checking all interpretations

- Let’s do proof instead
  
  - Let $I$ be a model of $KB$, prove that $I$ makes $\text{person}(sally)$ true
A Semantic Proof

• Let $I = \{D, \phi, \pi\}$ be a model of $KB = \{female(sally) \leftarrow person(X) \leftarrow female(X)\}$
  - So $\langle \phi(sally) \rangle \in \pi(female)$
  - Say $\phi(sally) = s$, so $\langle s \rangle \in \pi(female)$  \hspace{1cm} (1)
  - $person(X) \leftarrow female(X)$ must be true for $I_\rho$ for any var. assign. $\rho$  \hspace{1cm} (2)

• Consider variable assignment $\rho$ where $\rho(X) = s$
  - If $female(X)$ true for $I_\rho$ then so must $person(X)$ (from (2))  \hspace{1cm} (3)
  - $\rho(X) = s$ and $\langle s \rangle \in \pi(female)$ so $female(X)$ is true for $I_\rho$  \hspace{1cm} (4)
  - So $person(X)$ must be true for $I_\rho$ (from (3) and (4))
  - $\rho(X) = s$ so $\langle s \rangle \in \pi(person)$  \hspace{1cm} (5)
  - Since $\phi(sally) = s$, $person(sally)$ is true under $I$
• Say $\text{parent}(X, Y) \leftarrow \text{father}(X, Y)$ is in KB
  - Implicit universal quantifiers around it
  - Anytime that $\text{father}(X, Y)$ is true, so must $\text{parent}(X, Y)$

• Say $\text{grandfather}(X, Y) \leftarrow \text{father}(X, Z) \land \text{parent}(Z, Y)$ in KB
  - This clause is true for all $X, Y, Z$
  - $\forall X Y Z \ (\text{grandfather}(X, Y) \leftarrow \text{father}(X, Z) \land \text{parent}(Z, Y))$.
  - $Z$ only appears in the body

• How does $Z$ work here (variable just in the body)?
  - For any $X$ and $Y$, if we find $Z$ 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 $Z$
  - $\forall XY \ (\text{grandfather}(X, Y) \leftarrow (\exists Z \ \text{father}(X, Z) \land \text{parent}(Z, Y)))$. 
Overview

• Semantics

⇒ Queries

• Proof Procedures

• Bottom-up Ground Proof Procedure

• Top-down Ground Proof Procedure
Ground Queries

• A query is a way to ask if a body is a logical consequence of the knowledge base: \( ? \ b_1 \land ... \land b_m \)

• 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
    + Cannot tell if query is false in the intended interpretation

• Can do query-answering by:
  - Transform query \( b_1 \land ... \land b_m \) into \( \text{yes} \leftarrow b_1 \land ... \land b_m \)
  - Add (temporarily) \( \text{yes} \leftarrow b_1 \land ... \land b_m \) to KB
  - Check if \( \text{yes} \) is a logical consequence of KB
  - This lets us view queries as just finding consequences from a \( KB \)
Queries with Variables

• You might not only want to check if something is true or false, but what value makes it true

*KB*:

\[
\text{father}(\text{william}, \text{ted})
\]

\[
\text{parent}(X, Y) \leftarrow \text{father}(X, Y)
\]

• Example: \( \text{?parent}(X, \text{ted}) \)
  - Who is Ted’s parent?
  - Could transform this to \( \text{yes} \leftarrow \text{parent}(X, \text{ted}) \)
  - But, lets capture the variables in the body: \( \text{yes}(X) \leftarrow \text{parent}(X, \text{ted}) \)

• An answer is either
  - **instance** of ‘yes’ that is a logical consequence of *KB*: \( \text{yes}(\text{william}) \)
  - or **no** if no instance is a logical consequence of KB
Overview

• Semantics
• Queries

⇒ Proof Procedures

• Bottom-up Ground Proof Procedure
• Top-down Ground Proof Procedure
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
  - Can’t just check the user’s intended interpretation
    + Computer can only access the \( KB \)
Proof Procedures

• **Proof**: a mechanically derivable demonstration that a formula logically follows from a $KB$

• **Proof procedure**: an algorithm that constructs proofs
  - $KB \vdash g$ means $g$ can be derived from $KB$ with the proof procedure

• Proof procedure can be nondeterministic
  - So as to simplify the specification
  - Still need to specify an actual implementation

• Properties of Proof Procedure
  - **Soundness**: if $KB \vdash g$ then $KB \models g$
  - **Completeness**: if $KB \models g$ then $KB \vdash g$

• Terminology:
  - semantic proof: $\models$, logically follows, logically entails, models
  - syntactic proof: $\vdash$, derives
Two Types of Proof Procedures

- Bottom-Up Forward-Chaining
- Top-Down Backward-Chaining

KB → Query
Overview

• Semantics
• Queries
• Proof Procedures

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

• For now, only consider ground facts and ground rules
  - no variables

• Bottom-up or forward chaining procedure:
  starts from $KB$ and works towards query

• Forward chaining rule
  - If $h \leftarrow b_1 \land ... \land b_m$ is a clause in the $KB$
  - and each $b_i$ has been derived
  - then $h$ can be derived

• Forward chaining rule also works if $h$ is a fact in $KB$ ($m = 0$)
  - Lets you derive $h$

• Call the set of derivables the consequence set $(C)$
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?
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?
Is it Sound?

- Does everything in C logically follow from KB?
- Proof by contradiction: assume $KB \vdash g$ but $KB \not\models g$
  - $g$ is the result of a finite number of derivations
  - Without loss of generality, assume $g$ is first one in derivation such that $KB \not\models g$
  - 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 first bad one, 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 the $KB$
  - Using definition of $\leftarrow$, can show that $g$ must logically follow from $KB$
  - Contradiction
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 ∈ C
  - Otherwise, I(h) is false
  - I is an interpretation because it defines a subset of ground atoms as being true, and the rest as false
- I is an interpretation, but is it also a model of KB?
  - i.e. for every g ∈ KB, is I(g) true?
• Proof by Contradiction: Let \( g \in KB \) but where \( I(g) \) is false
  - Since \( g \in KB \), \( g \) 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 \( KB \)
    bottom up algorithm must have applied this rule and hence \( h \in C \)
  - Hence \( h \) is true in \( I \)
  - Contradiction
Final Step in Completeness Proof

- Let \( g \) be atomic and \( KB \models g \)
  - Need to make sure that \( KB \vdash g \)

- Since \( KB \models g \), \( g \) must be in every model of \( KB \)

- So, it is in the interpretation defined by the Consequence set

- Since \( g \) is atomic and it is true in the interpretation,
  it must be in consequence set

- So \( KB \vdash g \)
Overview

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

$\Rightarrow$ Top-down Ground Proof Procedure
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

\[
\begin{align*}
\text{yes} & \leftarrow a_1 \land \ldots \land a_m \\
a_i & \leftarrow b_1 \land \ldots \land b_p \\
\text{yes} & \leftarrow a_1 \land \ldots \land a_{i-1} \land b_1 \land \ldots \land b_p \land a_{i+1} \land \ldots \land a_m
\end{align*}
\]
Now for some definitions

- **Answer clause** is \( \text{yes} \leftarrow a_1 \land \ldots \land a_m \)

- **Answer** is answer clause with \( m = 0 \)

- **Derivation** of a query \( ?q_1 \land \ldots \land q_k \) from \( KB \) is a sequence of answer clauses \( \gamma_0, \gamma_1, \ldots, \gamma_n \)
  - \( \gamma_0 \) is the answer clause corresponding to the original query
  - \( \gamma_i \) is obtained by resolving \( \gamma_{i-1} \) with a clause in \( KB \)
  - \( \gamma_n \) is the answer

- **Nondeterminism**
  - In choosing which clause from \( KB \) to resolve with
  - Can find all derivations by systematically considering all different choices (see Chapter 4)
Example

- $KB$
  - $a \leftarrow b \land c.$
  - $b \leftarrow d \land e.$
  - $b \leftarrow g \land e.$
  - $c \leftarrow e.$
  - $d.$
  - $e.$
  - $f \leftarrow a \land g.$
  - $?a.$
## Bottom-Up versus Top-Down

### KB

<table>
<thead>
<tr>
<th>KB</th>
<th>Top-Down</th>
<th>KB Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a \leftarrow b \land c )</td>
<td>( yes \leftarrow a )</td>
<td>( a \leftarrow b \land c )</td>
</tr>
<tr>
<td>( b \leftarrow d \land e )</td>
<td>( yes \leftarrow b \land c )</td>
<td>( b \leftarrow d \land e )</td>
</tr>
<tr>
<td>( b \leftarrow g \land e )</td>
<td>( yes \leftarrow d \land e \land c )</td>
<td>( d )</td>
</tr>
<tr>
<td>( c \leftarrow e )</td>
<td>( yes \leftarrow e \land c )</td>
<td>( c )</td>
</tr>
<tr>
<td>( d )</td>
<td>( yes \leftarrow e )</td>
<td>( c \leftarrow e )</td>
</tr>
<tr>
<td>( e )</td>
<td>( yes \leftarrow c )</td>
<td>( c \leftarrow e )</td>
</tr>
<tr>
<td>( f \leftarrow a \land g )</td>
<td>( yes \leftarrow )</td>
<td>( e )</td>
</tr>
</tbody>
</table>

?\( a \)
## Bottom-Up versus Top-Down

<table>
<thead>
<tr>
<th>KB</th>
<th>Top-Down</th>
<th>KB Rule</th>
<th>Bottom-Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a \leftarrow b \land c$</td>
<td>yes $\leftarrow a$</td>
<td>$a \leftarrow b \land c$</td>
<td>$a \in C$</td>
</tr>
<tr>
<td>$b \leftarrow d \land e$</td>
<td>yes $\leftarrow b \land c$</td>
<td>$b \leftarrow d \land e$</td>
<td>$C = {e,c,d,b}$</td>
</tr>
<tr>
<td>$b \leftarrow g \land e$</td>
<td>yes $\leftarrow d \land e \land c$</td>
<td>$b \leftarrow d \land e$</td>
<td>$C = {e,c,d}$</td>
</tr>
<tr>
<td>$c \leftarrow e$</td>
<td>yes $\leftarrow e \land c$</td>
<td>$d$</td>
<td>$C = {e,c}$</td>
</tr>
<tr>
<td>$d$</td>
<td>yes $\leftarrow e$</td>
<td>$c$</td>
<td>$C = {e,c}$</td>
</tr>
<tr>
<td>$e$</td>
<td>yes $\leftarrow c$</td>
<td>$c \leftarrow e$</td>
<td>$C = {e,c}$</td>
</tr>
<tr>
<td>$f \leftarrow a \land g$</td>
<td>yes $\leftarrow$</td>
<td>$e$</td>
<td>$C = {e}$</td>
</tr>
</tbody>
</table>

?a
Bottom-up versus Top-down

- Any top-down proof can be converted to a bottom-up proof.
- Any bottom-up proof can be converted to a top-down proof.
- So, top-down proof procedure is complete and sound

- There are many other ways of doing proofs
  - e.g. Unit resolution
  - We will explore some of these later in the course
  - But top-down and bottom-up are sufficient for datalog