Chapter 3.
Declarative Semantics

- Last updated: April 27, 2016 -

How do we know what a goal / program means?
→ Translation of Prolog to logical formulas

How do we know what a logical formula means?
→ Models of logical formulas (Declarative semantics) ← Now
→ Proofs of logical formulas (Operational semantics) ← Later
Question

What is the meaning of this program?

bigger(elephant, horse).
bigger(horse, donkey).
is_bigger(X, Y) :- bigger(X, Y).
is_bigger(X, Y) :- bigger(X, Z), is_bigger(Z, Y).

Rephrased question: Two steps

1. How does this program translate to logic formulas?
2. What is the meaning of the logic formulas?
Semantics: Translation

How do we translate a Prolog program to a formula in First Order Logic (FOL)?

→ Translation Scheme

Can any FOL formula be expressed as a Prolog Program?

→ Normalization Steps
Translation of Prolog Programs

1. A Prolog program is translated to a set of formulas, with each clause in the program corresponding to one formula:

   \{
   \begin{align*}
   & \text{bigger( elephant, horse ),} \\
   & \text{bigger( horse, donkey ),} \\
   & \forall x. \forall y. \left( \text{bigger}(x, y) \rightarrow \text{is\_bigger}(x, y) \right), \\
   & \forall x. \forall y. \left( \exists z. \left( \text{bigger}(x, z) \land \text{is\_bigger}(z, y) \right) \rightarrow \text{is\_bigger}(x, y) \right)
   \end{align*}
   \}

2. Such a set is to be interpreted as the conjunction of all the formulas in the set:

   \begin{align*}
   & \text{bigger( elephant, horse ) } \land \\
   & \text{bigger( horse, donkey ) } \land \\
   & \forall x. \forall y. \left( \text{bigger}(x, y) \rightarrow \text{is\_bigger}(x, y) \right) \land \\
   & \forall x. \forall y. \left( \exists z. \left( \text{bigger}(x, z) \land \text{is\_bigger}(z, y) \right) \rightarrow \text{is\_bigger}(x, y) \right)
   \end{align*}
Translation of Clauses

- Each *comma* separating subgoals becomes $\land$ (conjunction).
- Each *:-* becomes $\leftarrow$ (implication)
- Each *variable in the head* of a clause is bound by a $\forall$ (universal quantifier)

<table>
<thead>
<tr>
<th>$\forall x. \forall y$</th>
<th>son$(x, y) \leftarrow$ father$(y, x) \land$ male$(x)$</th>
</tr>
</thead>
</table>

- Each *variable that occurs only in the body* of a clause is bound by a $\exists$ (existential quantifier)

<table>
<thead>
<tr>
<th>$\forall x.$</th>
<th>grandfather$(x) \leftarrow$ father$(x, Y),$ parent$(Y, Z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall x.$</td>
<td>(grandfather$(x) \leftarrow \exists y. \exists z.$ father$(x, y) \land$ parent$(y, z)$)</td>
</tr>
</tbody>
</table>
Translating Disjunction

- Disjunction is the same as two clauses:

\[
\text{disjunction}(X) : - \\
( ( a(X,Y), b(Y,Z) ) ; ( c(X,Y), d(Y,Z) ) ) .
\]

\[
\text{disjunction}(X) : - \\
a(X,Y), b(Y,Z) .
\]

\[
\text{disjunction}(X) : - \\
c(X,Y), d(Y,Z) .
\]

- Variables with the same name in different clauses are different
- Therefore, variables with the same name in different disjunctive branches are different too!
- Good Style: Avoid accidentally equal names in disjoint branches!
  - Rename variables in each branch and use explicit unification

\[
\text{disjunction}(X) : - \\
( ( X=X_1, a(X_1,Y_1), b(Y_1,Z_1) ) ; ( X=X_2, c(X_2,Y_2), d(Y_2,Z_2) ) ) .
\]

\[
\text{disjunction}(X_1) : - \\
a(X_1,Y_1), b(Y_1,Z_1) .
\]

\[
\text{disjunction}(X_2) : - \\
c(X_2,Y_2), d(Y_2,Z_2) .
\]
Declarative Semantics – in a nutshell
Meaning of Programs (in a nutshell)

Meaning of a program

Meaning of the equivalent formula.

\[
\begin{align*}
&\text{bigger( elephant, horse )} \\
&\quad \land \\
&\quad \text{bigger( horse, donkey )} \\
&\quad \land \\
&\quad \forall x. \forall y. ( \text{bigger}(x, y) \rightarrow \text{is\_bigger}(x, y) ) \\
&\quad \land \\
&\quad \forall x. \forall y. ( \exists z. (\text{bigger}(x, z) \land \text{is\_bigger}(z, y)) \rightarrow \text{is\_bigger}(x, y) )
\end{align*}
\]

Meaning of a formula

Set of logical consequences

\[
\begin{align*}
&\text{bigger( elephant, horse )} \\
&\quad \land \\
&\quad \text{bigger( horse, donkey )} \\
&\quad \land \\
&\quad \text{is\_bigger(elephant, horse)} \\
&\quad \land \\
&\quad \text{is\_bigger(horse, donkey)} \\
&\quad \land \\
&\quad \text{is\_bigger(elephant, donkey)}
\end{align*}
\]
Meaning of Programs

Meaning of a program

Meaning of the equivalent formula.

\[
\text{bigger( elephant, horse )} \\
\wedge \\
\text{bigger( horse, donkey )} \\
\wedge \\
\forall x. \forall y. ( \text{bigger}(x, y) \rightarrow \text{is}_\text{bigger}(x, y) ) \\
\wedge \\
\forall x. \forall y. ( \exists z. ( \text{bigger}(x, z) \wedge \text{is}_\text{bigger}(z, y)) \rightarrow \text{is}_\text{bigger}(x, y) )
\]

Meaning of a formula

Set of logical consequences

\[
\text{bigger( elephant, horse )} \\
\wedge \\
\text{bigger( horse, donkey )} \\
\wedge \\
\text{is}_\text{bigger}(\text{elephant, horse}) \\
\wedge \\
\text{is}_\text{bigger}(\text{horse, donkey}) \\
\wedge \\
\text{is}_\text{bigger}(\text{elephant, donkey})
\]

Model = Set of logical consequences = What is true according to the formula
Semantics of Programs and Queries (in a nutshell)

<table>
<thead>
<tr>
<th>Program</th>
<th>Formula</th>
<th>Model</th>
<th>Query</th>
</tr>
</thead>
</table>
| `bigger(elephant, horse).`  
`bigger(horse, donkey).`  
`is_bigger(X,Y) :-`  
  `bigger(X,Y).`  
`is_bigger(X,Y) :-`  
  `bigger(X,Z),`  
  `is_bigger(Z,Y).` | `bigger( elephant, horse )`  
`^`  
`bigger( horse, donkey )`  
`^`  
`∀x.∀y.(is_bigger(x, y) ⇐ bigger(x, y) )`  
`^`  
`∀x.∀y.( ∃z.(is_bigger(x, y) ⇐ bigger(x, z) ∧ is_bigger(z, y)))` | `bigger( elephant, horse )`  
`^`  
`bigger( horse, donkey )`  
`^`  
`is_bigger(elephant, horse)`  
`^`  
`is_bigger(horse, donkey)`  
`^`  
`is_bigger(elephant, donkey)` | `?-`  
  `bigger( elephant, X )`  
`^`  
`is_bigger(X, donkey)` |

Translation  
Interpretation  
(logical consequence)  
Matching
Model-based Semantics → Algorithm

Model-based semantics
- Herbrand interpretations and Herbrand models
- Basic step = “Entailment” (Logical consequence)
- A formula is true if it is a logical consequence of the program

Algorithm = Logic + Control
- Logic = Clauses
- Control =
  - Bottom-up fixpoint iteration to build the model
  - Matching of queries to the model

<table>
<thead>
<tr>
<th>Program</th>
<th>Formula</th>
<th>Model</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>bigger(elephant,horse). bigger(horse,donkey). ...</td>
<td>bigger( elephant, horse ) ^ bigger( horse, donkey ) ^ ...</td>
<td>bigger( elephant, horse ) ^ bigger( horse, donkey ) ^ ...</td>
<td>?- bigger( elephant, X ) ^ is_bigger(X, donkey)</td>
</tr>
</tbody>
</table>

Translation  Interpretation (logical consequence)  Matching
Constructing Models by Fixpoint Iteration

Program:

\[
\begin{align*}
  p & \leftarrow \neg q, \\
  q & \leftarrow \neg p, \\
  p & \leftarrow \neg r, \\
  r.
\end{align*}
\]

Formulas:

\[
\begin{align*}
  p & \leq q \land \\
  q & \leq p \land \\
  p & \leq r \land \\
  r.
\end{align*}
\]

Model(s):

\[
M_0 \rightarrow M_1 \rightarrow M_2 \rightarrow M_3 \rightarrow M_4 = M_3
\]

Clauses contributing model elements in the respective iteration:

\[
\begin{align*}
  r & \% r \\
  p & \% p \\
  p & \% p \\
  q & \% q \\
  p & \% p \\
  q & \% q \\
  p & \% p
\end{align*}
\]
## Declarative Semantics Assessed

<table>
<thead>
<tr>
<th><strong>Pro</strong></th>
<th><strong>Contra</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>- Simple</td>
<td>- Inefficient</td>
</tr>
<tr>
<td>◆ Easy to understand</td>
<td>◆ Need to build the whole model in the worst case</td>
</tr>
<tr>
<td>- Thorough formal foundation</td>
<td>- Inapplicable to infinite models</td>
</tr>
<tr>
<td>◆ implication (entailment)</td>
<td>◆ Never terminates if the query is not true in the model</td>
</tr>
</tbody>
</table>

![Perfect for understanding the meaning of a program]

![Bad as the basis of a practical interpreter implementation]
Chapter Summary

- Translation to logic
  - From clauses to formulas

- Declarative / Model-based Semantics
  - Herbrand Universe
  - Herbrand Interpretation
  - Herbrand Model

- Operational interpretation
  - Model construction by fix-point iteration
  - Matching of goals to the model

- Assessment
  - Strength
  - Weaknesses