Chapter 2.
Declarative Semantics

- Last updated: May 29, 2017 -

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?

\[
\begin{align*}
\text{bigger(elephant, horse).} \\
\text{bigger(horse, donkey).} \\
\text{is\_bigger(X, Y) \colon= bigger(X, Y).} \\
\text{is\_bigger(X, Y) \colon= bigger(X, Z), is\_bigger(Z, Y).}
\end{align*}
\]

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)?

Can any FOL formula be expressed as a Prolog Program?
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:

   \{ 
   \text{bigger( elephant, horse ),}
   \text{bigger( horse, donkey ),}
   \forall x. \forall y. ( \text{bigger}( x, y ) \rightarrow \text{is\_bigger}( x, y ) ),
   \forall x. \forall y. ( \exists z. ( \text{bigger}( x, z ) \land \text{is\_bigger}( z, y ) ) \rightarrow \text{is\_bigger}( x, y ) ) 
   \}

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

   \text{bigger( elephant, horse )} \land
   \text{bigger( horse, donkey )} \land
   \forall x. \forall y. ( \text{bigger}( x, y ) \rightarrow \text{is\_bigger}( x, y ) ) \land
   \forall x. \forall y. ( \exists z. ( \text{bigger}( x, z ) \land \text{is\_bigger}( z, y ) ) \rightarrow \text{is\_bigger}( x, y ) )
Translation of Clauses

- Each comma separating subgoals becomes $\land$ (conjunction).

- Each $\leftarrow$ becomes $\leftarrow$ (implication)

- Each variable in the head of a clause is bound by a $\forall$ (universal quantifier)

$$\forall x. \forall y \quad \text{son}(x, y) \leftarrow \text{father}(y, x) \land \text{male}(x)$$

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

$$\forall x. \exists y. \exists z. \quad \text{grandfather}(x) \leftarrow \text{father}(x, y) \land \text{parent}(y, z)$$
Translating Disjunction

- Disjunction is the same as two clauses:
  
  \[
  \text{disjunction}(X) : - \\
  \quad ( ( \mathit{a}(X,Y), \mathit{b}(Y,Z) ) \\
  \quad ; ( \mathit{c}(X,Y), \mathit{d}(Y,Z) ) \\
  \quad ).
  \]

- 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.

  \[
  \text{disjunction}(X) : - \\
  \quad ( ( \mathit{X} = \mathit{X}1, \mathit{a}(\mathit{X}1,\mathit{Y}1), \mathit{b}(\mathit{Y}1,\mathit{Z}1) ) \\
  \quad ; ( \mathit{X} = \mathit{X}2, \mathit{c}(\mathit{X}2,\mathit{Y}2), \mathit{d}(\mathit{Y}2,\mathit{Z}2) ) \\
  \quad ).
  \]

  \[
  \text{disjunction}(\mathit{X}1) : - \\
  \quad \mathit{a}(\mathit{X}1,\mathit{Y}1), \mathit{b}(\mathit{Y}1,\mathit{Z}1). \\
  \quad \text{disjunction}(\mathit{X}2) : - \\
  \quad \mathit{c}(\mathit{X}2,\mathit{Y}2), \mathit{d}(\mathit{Y}2,\mathit{Z}2). 
  \]
Declarative Semantics – in a nutshell
**Meaning of Programs (in a nutshell)**

**Meaning of a program**
Meaning of the equivalent formula.

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

**Meaning of a formula**
Set of logical consequences

- \( \text{bigger( elephant, horse )} \)
- \( \wedge \)
- \( \text{bigger( horse, donkey )} \)
- \( \wedge \)
- \( \text{is\_bigger}(\text{elephant}, \text{horse}) \)
- \( \wedge \)
- \( \text{is\_bigger}(\text{horse}, \text{donkey}) \)
- \( \wedge \)
- \( \text{is\_bigger}(\text{elephant}, \text{donkey}) \)
Meaning of Programs

Meaning of a program

Meaning of the equivalent formula.

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

Meaning of a formula

Set of logical consequences

\[
\begin{align*}
\text{bigger( elephant, horse )} \\
\land \\
\text{bigger( horse, donkey )} \\
\land \\
\text{is\_bigger}(\text{elephant, horse}) \\
\land \\
\text{is\_bigger}(\text{horse, donkey}) \\
\land \\
\text{is\_bigger}(\text{elephant, donkey})
\end{align*}
\]
# Semantics of Programs and Queries (in a nutshell)

### Program

- `bigger(\text{elephant}, \text{horse})`.
- `bigger(\text{horse}, \text{donkey})`.
- `\text{is\_bigger}(X, Y) \leftarrow \text{bigger}(X, Y)`.
- `\text{is\_bigger}(X, Y) \leftarrow \text{bigger}(X, Z), \text{is\_bigger}(Z, Y)`.

### Formula

- `\text{bigger}(\text{elephant}, \text{horse})`.
- `\text{bigger}(\text{horse}, \text{donkey})`.
- `\forall x. \forall y. (\text{is\_bigger}(x, y) \leftarrow \text{bigger}(x, y))`.
- `\forall x. \forall y. (\exists z. (\text{is\_bigger}(x, y) \leftarrow \text{bigger}(x, z) \land \text{is\_bigger}(z, y)))`.

### Model

- `\text{bigger}(\text{elephant}, \text{horse})`.
- `\text{bigger}(\text{horse}, \text{donkey})`.
- `\text{is\_bigger}(\text{elephant}, \text{horse})`.
- `\text{is\_bigger}(\text{horse}, \text{donkey})`.
- `\text{is\_bigger}(\text{elephant}, \text{donkey})`.

### Query

- `?- \text{bigger}(\text{elephant}, X) \land \text{is\_bigger}(X, \text{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

---

**Program**
- bigger(elephant, horse).
bigger(horse, donkey).
...

**Formula**
- bigger(elephant, horse).
^ bigger(horse, donkey).
^ ...

**Model**
- bigger(elephant, horse).
^ bigger(horse, donkey).
^ ...

**Query**
?- bigger(elephant, X).
^ is_bigger(X, donkey)

---

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

Program:

\[
\begin{align*}
p &: q. \\
qu &: p. \\
p &: r. \\
r &.
\end{align*}
\]

Formulas:

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

Model(s):

\[
\begin{align*}
M_0 &
\rightarrow M_1 &
\rightarrow M_2 &
\rightarrow M_3 &
\rightarrow M_4 = M_3
\end{align*}
\]

Clauses contributing model elements in the respective iteration:

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

**Pro**
- Simple
  - Easy to understand
- Thorough formal foundation
  - Implication (entailment)

Perfect for understanding the meaning of a program

Excellent query language

**Contra**
- Inefficient
  - Need to build the whole model in the worst case
- Inapplicable to infinite models
  - Never terminates if the query is not true in the model

Bad as the basis of a practical interpreter implementation

Cannot express execution order, side-effects (e.g. I/O), …

No programming language
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