Chapter 2.
Declarative Semantics

- Last updated: April 23, 2013 -

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?

```prolog
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. ( \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) ) \\
   \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. ( \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) )
   \end{align*}
Translation of Clauses

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

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

- Disjunction is the same as two clauses:

\[
\text{disjunction}(X) :- \\
\quad \left( \begin{array}{l}
\text{a}(X, Y), \text{b}(Y, Z) \\
\text{; } \text{c}(X, Y), \text{d}(Y, Z)
\end{array} \right)
\]

- 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

\[
\text{disjunction}(X) :- \\
\quad \left( \begin{array}{l}
\text{X=X1}, \text{a}(X_1, Y_1), \text{b}(Y_1, Z_1) \\
\text{; } \text{X=X2}, \text{c}(X_2, Y_2), \text{d}(Y_2, Z_2)
\end{array} \right)
\]

\[
\text{disjunction}(X1) :- \\
\quad \text{a}(X_1, Y_1), \text{b}(Y_1, Z_1)
\]

\[
\text{disjunction}(X2) :- \\
\quad \text{c}(X_2, Y_2), \text{d}(Y_2, Z_2)
\]
Declarative Semantics – in a nutshell
Meaning of Programs (in a nutshell)

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

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

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

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

Meaning of a program
Meaning of the equivalent formula.

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

Meaning of a formula
Set of logical consequences

\[
\text{bigger( elephant, horse )} \\
\land \\
\text{bigger( horse, donkey )} \\
\land \\
\text{is\_bigger(elephant, horse)} \\
\land \\
\text{is\_bigger(horse, donkey)} \\
\land \\
\text{is\_bigger(elephant, donkey)}
\]

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

Program

\[ \text{bigger(elephant,horse)}. \]
\[ \text{bigger(horse,donkey)}. \]
\[ \text{is}_\text{bigger}(X,Y) \leftarrow \text{bigger}(X,Y). \]
\[ \text{is}_\text{bigger}(X,Y) \leftarrow \text{bigger}(X,Z), \text{is}_\text{bigger}(Z,Y). \]

Formula

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

Model

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

Query

\[ ?- \text{bigger( elephant, X )} \]
\[ \wedge \]
\[ \text{is}_\text{bigger}(X, \text{donkey}) \]

Translation

Interpretation (logical consequence)

Matching
Declarative Semantics – the details

\[ \rightarrow \text{Interpretations of formulas} \]
\[ \rightarrow \text{Herbrand Interpretations} \]
\[ \rightarrow \text{Herbrand Model} \]
\[ \rightarrow \text{Logical Consequence} \]
Interpretations of Formulas

A formula

An Interpretation

An interpretation domain

loves: Man × Woman → Bool

Interpretations map symbols to meaning!
Interpretations of Formulas

Same formula

Slightly different Interpretation

Slightly different interpretation domain

loves: Person × Person → Bool

Interpretations map symbols to meaning!
Interpretations of Formulas

Same formula

Other Interpretation

Other interpretation domain

loves (john, mary)

targetOf: Spy × Secret → Bool

Interpretations map symbols to meaning!
Interpretations of Formulas

A formula

Our initial interpretation domain

An Interpretation

loves: Man × Woman → Bool

(brlbzf( prfkz, flurp ))
What does this tell us?

Observations

- Formulas only have a meaning with respect to an interpretation
- An interpretation maps formulas to elements of an interpretation domain
  - **constants** to constant in the domain
    - e.g. “john” to
  - **function symbols** to functions on the domain
    - function symbols do not occur in our example
  - **predicate symbols** to predicates on the domain
    - e.g. “loves/2” to “targetOf: Spy × Secret → Bool”
  - **formulas** to truth values
    - e.g. “loves(john,mary)” to “true”
What does this tell us?

Dilemma

- Too many possible interpretations!
- Which interpretation to use for proving truth?

Solution

- For universally quantified formulas there is a “standard” interpretation, the “Herbrand interpretation”, which has two nice properties:
  - If any interpretation satisfies a given set of clauses $S$ then there is a Herbrand interpretation that satisfies them
    $\Rightarrow$ It suffices to check satisfiability for the Herbrand interpretation!
  - If $S$ is unsatisfiable then there is a finite unsatisfiable set of ground instances from the Herbrand base defined by $S$.
    $\Rightarrow$ Unsatisfiability can be checked finitely
Herbrand Interpretations

**Herbrand Base** = all positive ground literals in \( D \)

**Herbrand Universe** = all ground terms that can be constructed from the constants and function symbols in \( D \)

**Formula**

\[ p(\,c_1, f(\,c_2)\,). \]

**Herbrand Interpretation**

\[ \{ \, p : \text{HU} \times \text{HU} \rightarrow \text{true} \, \} = \text{Pred} \]

\[ \{ \, c_1, c_2 \, \} = \text{Const} \]

\[ \{ \, f/1 \, \} = \text{Func} \]

**Interpretation domain domain \( D \)**

\[ p(\,c_1, c_1) \]
\[ p(\,c_1, c_2) \]
\[ p(\,c_1, f(\,c_1)) \]
\[ \{ \, p(\,c_1, f(\,c_2)) \, \} \]
\[ p(\,c_1, f(\,f(\,c_1))) \]
\[ p(\,c_1, f(\,f(\,c_2))) \]
\[ \ldots \]

\[ p(\,c_2, c_2) \]
\[ p(\,c_2, c_1) \]
\[ p(\,c_2, f(\,c_1)) \]
\[ p(\,c_2, f(\,c_2)) \]
\[ \ldots \]

\[ f(\,c_1), f(f(\,c_1)), \ldots \]
\[ f(\,c_2), f(f(\,c_2)), \ldots \]

\[ c_1 \]
\[ c_2 \]
Herbrand Interpretations of Formulas with Variables

**Formula**

\[ p( c_1, f( X ), c_2 ). \]

**Herbrand Interpretation**

**Interpretation domain D**

\[ \{ p: HU \times HU \times HU \to true \} = \text{Pred} \]
\[ \{ c_1, c_2 \} = \text{Const} \]
\[ \{ f/1 \} = \text{Func} \]

\[ HU = \text{Herbrand Universe} \]
\[ = \text{all ground terms that can be constructed from the constants and function symbols in } D \]

\[ \text{Herbrand Base} \]
\[ = \text{all positive ground literals in } D \]
Herbrand Models (1)

- The Interpretation Domain (D) of a program P consists of three sets:
  - **Const** contains all constants occurring in P
  - **Func** contains all function symbols occurring in P
  - **Pred** contains a predicate $p: \text{HU} \times \ldots \times \text{HU} \rightarrow \text{true}$

  for each predicate symbol $p$ of arity $n$ occurring in the program P

- The Herbrand Universe (HU) of a program P is the set of all ground terms that can be constructed from the function symbols and constants in P

- The Herbrand Base of a program P is the set of all positive ground literals that can be constructed by applying the predicate symbols in P to arguments from the Herbrand Universe of P
Herbrand Models (2)

- **A Herbrand Interpretation** maps each formula in a program P to the elements of the Herbrand Base that are its **logical consequences**
  - Each *ground fact* is mapped to true.
  - Each *ground instantiation* of a non-ground fact is mapped to true.
  - Each *ground instantiation* of the head literal of a rule that is a logical consequence of the rule body is mapped to true.

- **The Herbrand Model** of a program P is the **subset of the Herbrand Base of P that is true** according to the Herbrand Interpretation.
  - It is the **set of all logical consequences** of the program.

- **The Herbrand Model** of P can be **constructed by fixpoint iteration**:
  - Initialize the model with the ground instantiations of facts in P
  - Add all new facts that follow from the intermediate model and P
  - … until the model does not change anymore (= fixpoint is reached)
Constructing Models by Fixpoint Iteration

Program

\[ p : - q. \]
\[ q : - p. \]
\[ p : - r. \]
\[ r. \]

Formulas

\[ p \leftarrow q \land \]
\[ q \leftarrow p \land \]
\[ p \leftarrow r \land \]
\[ r \]

Model(s)

\[ M_0 \]
\[ r. \]
\[ M_1 \]
\[ r. p. \]
\[ r. p. q. \]
\[ M_2 \]
\[ r. p. q. \]
\[ M_3 \]
\[ r. p. q. \]
\[ M_4 = M_3 \]

Clauses contributing model elements in the respective iteration

\[ r \% r \]
\[ p \% r \]
\[ r \% p \]
\[ r \% q \]
\[ p \% r \]
\[ q \% p \]
\[ p \% q \]
Model-based Semantics $\rightarrow$ 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
Declarative Semantics Assessed

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

Perfect for understanding the meaning of a program

---

**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
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