Semantics of Datalog Languages

CS240B Spring 2002 Notes

From Section 8.8 of Advanced Database Systems—Morgan Kaufmann, 1997
Syntax of FOL—the alphabet

1. **Constants.**

2. **Variables.** In addition identifiers beginning with upper case, \( x, y \) and \( z \) also represent variables in this section.

3. **Functions.** Such as \( f(t_1, \ldots, t_n) \) where \( f \) is an \( n \)-ary functor and \( t_1, \ldots, t_n \) are the arguments.

4. **Predicates.**

5. **Quantifiers.** The existential quantifier \( \exists \) and the universal quantifier \( \forall \).

6. **Parentheses and punctuation symbols, used liberally as needed to avoid ambiguities.**
Syntax of First Order Logic–cont.

A *Term* is defined inductively as follows:

- A variable is a term
A *Term* is defined inductively as follows:

- A variable is a term
- A constant is a term
A Term is defined inductively as follows:

- A variable is a term
- A constant is a term
- If $f$ is an $n$-ary functor and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.
Well-Formed Formulas (WFFs)

1. If $p$ is an $n$-ary predicate and $t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is a formula (called an atomic formula or, more simply, an atom).

2. If $F$ and $G$ are formulas, then so are $\neg F$, $F \lor G$, $F \land G$, $F \leftarrow G$, $F \rightarrow G$ and $F \leftrightarrow G$.

3. If $F$ is a formula and $x$ is a variable, then $\forall x (F)$ and $\exists x (F)$ are formulas. When so, $x$ is said to be quantified in $F$.

$$\exists G_1(\text{took}(N, cs101, G_1)) \land \exists G_2(\text{took}(N, cs143, G_2)) \land \exists M(\text{student}(N, M, \text{junior}))$$
A WFF $F$ is said to a *closed formula* if every variable occurrence in $F$ is quantified. The formula in the previous example is not closed. But the following one is.

$$\forall x \forall y \forall z \ (p(x, z) \lor \neg q(x, y) \lor \neg r(y, z))$$
Definite Clauses

A Definite Clause is a WFF which:
- is closed,
- all its variables are universally quantified, and
- is a disjunction of one positive atom and zero or more negated atoms.

A definite clause is representable with the rule notation:

\[ \forall x \forall y \forall z p(x, z) \leftarrow q(x, y), r(y, z). \]
Positive Programs

- A definite clause with an empty body is called a *unit clause*.
- The notation used for unit clauses is “$A.$” instead of the more precise notation “$A ← .$”
- A *fact* is a unit clause without variables.

A unit clause (everybody loves himself) and three facts:

```
loves(X, X).
loves(marc, mary).
loves(mary, tom).
hates(marc, tom).
```

*A positive logic program is a set of definite clauses.*
Herbrand Interpretations for program $P$

- The *Herbrand Universe* for $P$, denoted $U_P$, is the set of all terms that can be recursively constructed by letting the arguments of the functions be constants in $P$ or elements in $U_P$.

- The *Herbrand Base* of $P$ is defined as the set of atoms that can be built from the predicates by replacing their arguments with elements from $U_P$.

- An *Herbrand Interpretation* is defined by assigning to each $n$-ary predicate $q$ an $n$-relation $Q$, where $q(a_1, ..., a_n)$ is true iff $(a_1, ..., a_n) \in Q$.

- Also, every subset of the *Herbrand Base* of $P$ defines an Herbrand interpretation of $P$. 

– p.8
Example

\[
\begin{align*}
\text{anc}(X, Y) & \leftarrow \text{parent}(X, Y). \\
\text{anc}(X, Z) & \leftarrow \text{anc}(X, Y), \text{parent}(Y, Z) \\
\text{parent}(X, Y) & \leftarrow \text{father}(X, Y). \\
\text{parent}(X, Y) & \leftarrow \text{mother}(X, Y). \\
\text{mother}(\text{anne}, \text{silvia}). & \quad \text{mother}(\text{anne}, \text{marc}).
\end{align*}
\]

Here: \( U_P = \{\text{anne, silvia, marc}\} \), and

\[
B_P = \{\text{parent}(x, y) | x, y \in U_P\} \cup \{\text{father}(x, y) | x, y \in U_P\} \\
\{\text{mother}(x, y) | x, y \in U_P\} \cup \{\text{anc}(x, y) | x, y \in U_P\}
\]
Example—cont.

\[
\text{anc}(X, Y) \leftarrow \quad \text{parent}(X, Y).
\]

\[
\text{anc}(X, Z) \leftarrow \quad \text{anc}(X, Y), \text{parent}(Y, Z)
\]

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

\[
\text{parent}(X, Y) \leftarrow \quad \text{mother}(X, Y).
\]

\[
\text{mother}(\text{anne}, \text{silvia}). \quad \text{mother}(\text{anne}, \text{marc}).
\]

- Herbrand Base: 4 binary predicates, and for each, 3 possible assignments for each argument: \( B_P = 4 \times 3 \times 3 = 36 \).
- Herbrand Interpretations (HIs): There are \( 2^{|B_P|} \) subsets of \( B_P \)—\( 2^{36} \) for this program.
- With infinite universe we have an infinite number of interpretations.
The Models of a Program
Section 8.9 in *Advanced Database Systems*
Morgan Kaufmann, 1997
Ground Instances of a Rule

Let $r$ be a rule in a program $P$. $\text{ground}(r)$ denotes the set of ground instances of $r$ (i.e., all the rules obtained by assigning to the variables in $r$, values from the Herbrand universe $U_P$).

$$\text{parent}(X, Y) \leftarrow \text{mother}(X, Y).$$

With 2 variables and $U_P = 3$, $\text{ground}(r)$ has $3 \times 3$ rules:

$$\text{parent}(\text{anne}, \text{anne}) \leftarrow \text{mother}(\text{anne}, \text{anne}).$$
$$\text{parent}(\text{anne}, \text{marc}) \leftarrow \text{mother}(\text{anne}, \text{marc}).$$
$$\ldots$$
$$\text{parent}(\text{silvia}, \text{silvia}) \leftarrow \text{mother}(\text{silvia}, \text{silvia}).$$
The ground version of a program $P$, denoted $\text{ground}(P)$, is the set of the ground instances of its rules:

$$\text{ground}(P) = \{\text{ground}(r) \mid r \in P\}$$
Models of a Program

Let $I$ be an interpretation for a program $P$. If an atom $a \in I$ we say that $a$ is true, otherwise we say that $a$ is false. Conversely for negated atoms $\neg a$.

*Satiation**: A rule $r \in P$ is said to hold true in interpretation $I$, or to be satisfied in $I$, if every instance of $r$ is satisfied in $I$.

*Model*. An interpretation $I$ that satisfies all the rules in $\text{ground}(P)$ is said to be a model for $P$.
A model $M$ for a program $P$ is said to be a *minimal model* for $P$ if there exists no other model $M'$ of $P$ where $M' \subset M$.

A model $M$ for a program $P$ is said to be its *least model* if every model $M'$ of $P$ has the property that $M' \supseteq M$.

**Model Intersection Property.** Let $P$ be a positive program, and $M_1$ and $M_2$ be two models for $P$. Then, $M_1 \cap M_2$ is also a model for $P$.

**Theorem:** Every positive program has a least model.