Definite Logic Programs: Models

CSE 505 – Computing with Logic
Stony Brook University

http://www.cs.stonybrook.edu/~cse505
Logical Consequences of Formulae

• Recall: F is a *logical consequence* of P (i.e. $P \models F$) iff
  
  Every model of P is also a model of F.

• Since there are (in general) infinitely many possible interpretations, how can we check if F is a logical consequence of P?

• Solution: choose (one) "*canonical*" model I such that

  $I \models P$ and $I \models F \implies P \models F$
Definite Clauses

- A formula of the form \( p(t_1, t_2, \ldots, t_n) \), where \( p/n \) is an \( n \)-ary predicate symbol and \( t_i \) are all terms is said to be \textit{atomic}.

- If \( A \) is an atomic formula then
  - \( A \) is said to be a \textit{positive literal}
  - \( \neg A \) is said to be a \textit{negative literal}

- A formula of the form \( \forall (L_1 \lor L_2 \lor \ldots \lor L_n) \) where each \( L_i \) is a literal (negative or positive) is called a \textit{clause}.

- A clause \( \forall (L_1 \lor L_2 \lor \ldots \lor L_n) \) where exactly one literal is positive is called a \textit{definite clause} (also called \textit{Horn clause}).

- A definite clause is usually written as:
  - \( \forall (A_0 \lor \neg A_1 \lor \ldots \lor \neg A_n) \)
  - or equivalently as \( A_0 \leftarrow A_1, A_2, \ldots, A_n \).

- A \textit{definite program} is a set of definite clauses.
Herbrand Universe

• Given an alphabet A, the set of all ground terms constructed from the constant and function symbols of A is called the Herbrand Universe of A (denoted by $U_A$).

• Consider the program:

\[
p(zero) \cdot \\
p(s(s(X))) \leftarrow p(X). \\
\]

• The Herbrand Universe of the program's alphabet is: $U_A = \{zero, s(zero), s(s(zero)), \ldots\}$
Herbrand Universe: Example

- Consider the "relations" program:

  \[
  \text{parent}(\text{pam}, \text{bob}) . \quad \text{parent}(\text{bob}, \text{ann}) . \\
  \text{parent}(\text{tom}, \text{bob}) . \quad \text{parent}(\text{bob}, \text{pat}) . \\
  \text{parent}(\text{tom}, \text{liz}) . \quad \text{parent}(\text{pat}, \text{jim}).
  \]

  \[
  \text{grandparent}(X,Y) :- \\
  \quad \text{parent}(X,Z), \text{parent}(Z,Y).
  \]

- The Herbrand Universe of the program's alphabet is:

  \[
  U_A = \{ \text{pam, bob, tom, liz, ann, pat, jim} \} 
  \]
Herbrand Base

- Given an alphabet $A$, the set of all **ground atomic formulas** over $A$ is called the **Herbrand Base** of $A$ (denoted by $B_A$).

- Consider the program:

  $$
  p(\text{zero}).
  $$

  $$
  p(s(s(X))) \leftarrow p(X).
  $$

- The Herbrand Base of the program's alphabet is: $B_A = \{ p(\text{zero}), p(s(\text{zero})), p(s(s(\text{zero}))), \ldots \}$
Herbrand Base: Example

- Consider the "relations" program:

  \[
  \begin{align*}
  \text{parent(pam, bob).} & \quad \text{parent(bob, ann).} \\
  \text{parent(tom, bob).} & \quad \text{parent(bob, pat).} \\
  \text{parent(tom, liz).} & \quad \text{parent(pat, jim).} \\
  \text{grandparent(X,Y) :-} \\
  & \quad \text{parent(X,Z), parent(Z,Y).}
  \end{align*}
  \]

- The Herbrand Base of the program's alphabet is:

  \[
  B_A = \{ \text{parent(pam, pam), parent(pam, bob),} \\
  \text{parent(pam, tom), \ldots, parent(bob, pam), \ldots,} \\
  \text{grandparent(pam,pam), \ldots,grandparent(bob,pam),} \\
  \ldots \}\.
  \]
Herbrand Interpretations and Models

• A **Herbrand Interpretation** of a program $P$ is an interpretation $I$ such that:
  
  • The domain of the interpretation: $|I| = U_P$
  
  • For every constant $c$: $c_I = c$
  
  • For every function symbol $f/n$: $f_I(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$
  
  • For every predicate symbol $p/n$: $p_I \subseteq (U_P)^n$ (i.e. some subset of $n$-tuples of ground terms)

• A **Herbrand Model** of a program $P$ is a Herbrand interpretation that is a model of $P$.
Herbrand Models

- All Herbrand interpretations of a program give the same “meaning” to the constant and function symbols.
- Different Herbrand interpretations differ only in the “meaning” they give to the predicate symbols.
- We often write a Herbrand model simply by listing the subset of the Herbrand base that is true in the model.
- Example: Consider our numbers program, where
  \{p(zero), \ p(s(s(zero))), \ p(s(s(s(s(zero)))))\}, \ldots\}
  represents the Herbrand model that treats
  \p_I = \{zero, s(s(zero)), s(s(s(s(zero))))\}, \ldots\}
  as the meaning of p.
Sufficiency of Herbrand Models

- Let $P$ be a definite program. If $I'$ is a model of $P$ then $I = \{ A \in B_p \mid I' \models A \}$ is a Herbrand model of $P$.

Proof (by contradiction):
Let $I$ be a Herbrand interpretation.
Assume that $I'$ is a model of $P$ but $I$ is not a model.
Then there is some ground instance of a clause in $P$:

$$A_0 \leftarrow A_1, \ldots, A_n.$$ 

which is not true in $I$ i.e., $I \models A_1, \ldots, I \models A_n$ but $I \not\models A_0$

By definition of $I$ then, $I' \models A_1, \ldots, I' \models A_n$ but $I' \not\models A_0$

Thus, $I'$ is not a model of $P$, which contradicts our earlier assumption.
Definite programs only

- Let P be a definite program. If I' is a model of P then
  \[ I = \{ A \in B_P \mid I' \models A \} \]
  is a Herbrand model of P.

This property holds only for definite programs!

- Consider \( P = \{ \neg p(a), \exists X. p(X) \} \)
  - There are two Herbrand interpretations: \( I_1 = \{ p(a) \} \) and \( I_2 = \{ \} \)
    - The first is not a model of P since \( I_1 \not\models \neg p(a) \)
    - The second is not a model of P since \( I_2 \not\models \exists X. p(X) \)
  - But there are non-Herbrand models, such as I:
    - \(| I | = \mathbb{N} \) (the set of natural numbers)
    - \( a_I = 0 \)
    - \( p_I = \text{"is odd"} \)
Properties of Herbrand Models

1. For any definite program, if every Herbrand Model of P is also a Herbrand Model of F, then P ⊨ F.

2. If M is a set of Herbrand Models of a definite program P, then $\bigcap M$ is also a Herbrand Model of P.

3. For every definite program P there is a unique least model $M_p$ such that:
   a) $M_p$ is a Herbrand Model of P and,
   b) for every Herbrand Model M, $M_p \subseteq M$.

4. $M_p = \text{the set of all ground logical consequences of P.}$
Properties of Herbrand Models

• If $M_1$ and $M_2$ are Herbrand models of $P$, then $M = M_1 \cap M_2$ is a model of $P$.

• Assume $M$ is not a model.

• Then there is some clause $A_0 : \neg A_1, \ldots, A_n$ such that $M \models A_1, \ldots, M \models A_n$ but $M \not\models A_0$

• Which means $A_0 \notin M_1$ or $A_0 \notin M_2$ by def. of $\cap$

• But $A_1, \ldots, A_n \in M_1$ as well as $M_2$.

• Hence one of $M_1$ or $M_2$ is not a model.
Properties of Herbrand Models

• There is a unique least Herbrand model

• Let $M_1$ and $M_2$ are two incomparable minimal Herbrand models, but $M = M_1 \cap M_2$ is also a Herbrand model (previous theorem), and $M \subseteq M_1$ and $M \subseteq M_2$

• Thus $M_1$ and $M_2$ are not minimal.
Least Herbrand Model

- The *least Herbrand model* $M_p$ of a definite program $P$ is the set of all ground logical consequences of the program:

$$M_p = \{A \in B_p \mid P \models A\}$$

- First, $M_p \supseteq \{A \in B_p \mid P \models A\}$ (i.e., $M_p$ is a superset of the logical consequences $\{A \in B_p \mid P \models A\}$):
  - By definition of logical consequence, $P \models A$ means that $A$ has to be in every model of $P$ and hence also in the least Herbrand model.
Least Herbrand Model

- Second, $M_p \subseteq \{A \in B_p \mid P \models A\}$ (i.e., $M_p$ is a subset of the logical consequences $\{A \in B_p \mid P \models A\}$):
  - Assume that $A$ is in $M_p$. Hence, $A$ is in every Herbrand model of $P$ by def. of $M_p$.
  - Assume that it is not true in some non-Herbrand model of $P$: $I' \models \neg A$
  - By sufficiency of Herbrand models (i.e., If $I'$ is a model of $P$ then $I = \{A \in B_p \mid I' \models A\}$ is a Herbrand model of $P$), there is some Herbrand model $I$ such that $I \models \neg A$.
  - Hence $A$ cannot be an element of $I$. This contradicts that $A$ is in every Herbrand model of $P$ by def. of $M_p$.
Construction of Least Herbrand Models

• **Immediate consequence operator:**
  • Given an interpretation $I \subseteq Bp$, construct $I'$ such that
    
    $I' = \{ A_0 \in Bp \mid A_0 \leftarrow A_1, \ldots, A_n \text{ is a ground instance of a clause in } P \text{ and } A_1, \ldots, A_n \in I \}$
  • $I'$ is said to be the *immediate consequence of* $I$ written as $I' = Tp(I)$, where $Tp$ is called the **immediate consequence operator**
  • Consider the sequence:
    
    $\emptyset, Tp(\emptyset), Tp(Tp(\emptyset)), \ldots, Tp^i(\emptyset), \ldots$
  • $Mp \supseteq Tp^i(\emptyset)$ for all $i$ (Mp is a superset of all $Tp^i(\emptyset)$).
  • Let $Tp \uparrow \omega = \bigcup_{i=0,\infty} Tp^i(\emptyset)$
    • Then $Mp = Tp \uparrow \omega$
### Computing Least Herbrand Models: An Example

parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

\[ \text{anc}(X,Y) :\begin{array}{l}
\text{parent}(X,Y) \\
\text{parent}(X,Z), \text{anc}(Z,Y) 
\end{array} \]

<table>
<thead>
<tr>
<th>(M_1)</th>
<th>(\emptyset)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M_2 = T_P(M_1) = {\text{parent}(pam, bob),)</td>
<td>{\text{parent}(pam, bob), \text{parent}(tom, bob), \text{parent}(tom, liz), \text{parent}(bob, ann), \text{parent}(bob, pat), \text{parent}(pat, jim) }</td>
</tr>
<tr>
<td>(M_3 = T_P(M_2) = {\text{anc}(pam, bob), \text{anc}(tom, bob), \text{anc}(tom, liz), \text{anc}(bob, ann), \text{anc}(bob, pat), \text{anc}(pat, jim) } \cup M_2 )</td>
<td></td>
</tr>
<tr>
<td>(M_4 = T_P(M_3) = {\text{anc}(pam, ann), \text{anc}(pam, pat), \text{anc}(tom, ann), \text{anc}(tom, pat), \text{anc}(bob, jim) } \cup M_3 )</td>
<td></td>
</tr>
<tr>
<td>(M_5 = T_P(M_4) = {\text{anc}(pam, jim), {\text{anc}(tom, jim) } \cup M_4 )</td>
<td></td>
</tr>
<tr>
<td>(M_6 = T_P(M_5) = M_5 )</td>
<td></td>
</tr>
</tbody>
</table>
Computing $M_p$: Practical Considerations

- Computing the least Herbrand model, $M_p$, as the **least fixed point** of $T_p$:
  - terminates for *Datalog* programs (programs w/o function symbols)
  - may not terminate in general (because it could be infinite)
    - For programs with function symbols, computing logical consequence by first computing $M_p$ is **impractical**.
- Even for Datalog programs, computing least fixed point directly using the $T_p$ operator is wasteful (known as *Naive* evaluation)
- Note that $T_p^i(\emptyset) \subseteq T_p^{i+1}(\emptyset)$ for all $i$.
- We can calculate $\Delta T_p^{i+1}(\emptyset) = T_p^{i+1}(\emptyset) - T_p^i(\emptyset)$ [The difference between the sets computed in two successive iterations]
  - This strategy is known as **semi-naive** evaluation.