Negation: Declarative Interpretation

- An overview

  • First Order Formulas and Logical Truth
  • Completion of Programs
  • SLDNF-resolution: Soundness and restricted completeness
  • Extended Consequence Operator
  • Standard Models


First-Order Formulas

Given ranked alphabets $F$ and $\Pi$ for function and predicate symbols, respectively, and a set $V$ of variables, the set of (first-order) formulas (over $\Pi$, $F$ and $V$) is inductively defined as follows:

- If atom $A \in TB_{\Pi,F,V}$, then $A$ is a formula

- If $G_1$ and $G_2$ are formulas, then $\neg G_1$, $G_1 \land G_2$ (written $G_1 \land G_2$), $G_1 \lor G_2$, $G_1 \iff G_2$ and $G_2 \iff G_1$ are formulas

- If $G$ is a formula and $x \in V$, then $\exists x \ G$ and $\forall x \ G$ are formulas
Extended Notion of Logical Truth (1)

Given a formula $G$, and interpretation $I$ with domain $D$, and a state $\sigma : V \rightarrow D$:

G is true in $I$ under $\sigma$, written $I |= _\sigma G$ :

- $I |= _\sigma p(t_1, ..., t_n) : \iff \sigma(t_1), ..., \sigma(t_n) \in p_i$
- $I |= _\sigma \neg G : \iff I \not|= _\sigma G$
- $I |= _\sigma G_1 \land G_2 : \iff I |= _\sigma G_1 \text{ and } I |= _\sigma G_2$
- $I |= _\sigma G_1 \lor G_2 : \iff I |= _\sigma G_1 \text{ or } I |= _\sigma G_2$
- $I |= _\sigma G_1 \rightarrow G_2 : \iff \text{if } I |= _\sigma G_2 \text{ then } I |= _\sigma G_1$
- $I |= _\sigma G_1 \leftrightarrow G_2 : \iff I |= _\sigma G_2 \text{ iff } I |= _\sigma G_1$
- $I |= _\sigma \forall x G : \iff \text{for every } d \in D: I |= _\sigma' G$
- $I |= _\sigma \exists x G : \iff \text{for some } d \in D: I |= _\sigma' G$

where $\sigma' : V \rightarrow D$ with $\sigma'(x) = d$ and $\sigma'(y) = \sigma(y)$ for every $y \in V - \{x\}$
Given a formula $G$, sets of formulas $S$ and $T$, and interpretation $I$, and where $x_1, ..., x_k$ are the variables occurring in $G$

- $\forall x_1, ..., x_k G$ is the **universal closure** of $G$ (abbreviated to $\forall G$)
- $I |= \forall G :\Leftrightarrow I |=_\sigma G$ for every state $\sigma$
- $G$ is true in $I$ (or $I$ is a **model** of $G$), written $I |= G :\Leftrightarrow I |= \forall G$
- $I$ is a **model** of $S$, written $I |= S :\Leftrightarrow I |= G$ for every $G \in S$
- $T$ is a **semantic (or logical) consequence** of $S$ written $S |= T :\Leftrightarrow$ every model of $S$ is a model of $T$  
  ($\forall I: I |= S$ implies $I |= T$)
No Negative Consequences of (Extended) Programs (1)

Consider program \( P_{\text{mem}} \):

\[
\begin{align*}
\text{member}(x, [x|y]) & \leftarrow \\
\text{member}(x, [y|z]) & \leftarrow \text{member}(x, z)
\end{align*}
\]

- Then, e.g. \( P_{\text{mem}} \models \text{member}(a, [a,b]) \) and \( P_{\text{mem}} \models \nexists \text{member}(a, []) \).
- But also, \( P_{\text{mem}} \nexists \text{member}(a, []) \) since

\[
\text{HB} \{\text{member}, [], [a]\} \models P_{\text{mem}} \text{ and } \text{HB} \{\text{member}, [], [a]\} \nexists \text{member}(a, []).\]

Nevertheless, the SLDNF tree of \( P_{\text{mem}} \cup \{\nexists \text{member}(a, [])\} \) is successful

\[
\begin{align*}
\nexists \text{member}(a, []) \\
\text{success} \\
\text{member}(a, []) \\
\text{failure}
\end{align*}
\]
No Negative Consequences of (Extended) Programs (2)

Problem:

For every extended program $P$, the corresponding Herbrand Base is a model.

- Hence, there is no negative ground literal $\neg A$ as a logical consequence of $P$.

- But SLDNF tree of $P_{\text{mem}} \cup \{\neg A\}$, may be successful ! (?)

Hence, is SLDNF-resolution sound? What does soundness mean?

Solution:

Strengthen $P$ by completion ("replace implication by "equivalence") to $\text{comp}(P)$ and compare SLDNF-resolution and $\text{comp}(P)$, instead of $P$!
Completed Definitions (Example 1)

\begin{align*}
P: & \quad \text{happy} \leftarrow \text{snow, holidays} \\
    & \quad \text{happy} \leftarrow \text{sun} \\
    & \quad \text{snow} \leftarrow \text{cold, precipitation} \\
    & \quad \text{cold} \leftarrow \\
    & \quad \text{precipitation} \leftarrow \\
\end{align*}

Whereas \( P \models \text{precipitation, cold, snow} \) \% the least Herbrand Model

\begin{align*}
\text{comp}(P): & \quad \text{happy} \leftrightarrow (\text{snow, holidays}) \setminus \text{sun} \\
    & \quad \text{snow} \leftrightarrow \text{cold, precipitation} \\
    & \quad \text{cold} \leftrightarrow \text{true} \\
    & \quad \text{precipitation} \leftrightarrow \text{true} \\
    & \quad \text{holidays} \leftrightarrow \text{false} \\
    & \quad \text{sun} \leftrightarrow \text{false} \\
\end{align*}

\( \text{comp}(P) \models \text{precipitation, cold, snow, \neg holidays, \neg sun, \neg happy} \)
Completed Definitions (Example 2)

\[
\text{P:} \quad \begin{align*}
\text{member}(x,[x|y]) & \leftarrow \\
\text{member}(x,[y|z]) & \leftarrow \text{member}(x,z) \\
\text{disjoint}([],x) & \leftarrow \\
\text{disjoint}([x|y],z) & \leftarrow \neg \text{member}(x,z), \text{disjoint}(y,z)
\end{align*}
\]

\[
\text{comp}(P): \quad \begin{align*}
\forall x_1 \forall x_2 \quad & \text{member}(x_1,x_2) \leftarrow \\
\exists x,y \quad & (x_1 = x, \ x_2 = [x|y]) \ \uparrow \\
\exists x,y,z \quad & (x_1 = x, \ x_2 = [y|z], \ \text{member}(x,z)) \\
\forall x_1 \forall x_2 \quad & \text{disjoint}(x_1,x_2) \leftarrow \\
\exists x \quad & (x_1 = [], \ x_2 = x) \ \uparrow \\
\exists x,y,z \quad & (x_1 = [x|y], \ x_2 = z, \\
& \neg \text{member}(x,z), \ \text{disjoint}(y,z))
\end{align*}
\]

( plus the standard axioms for equality and inequality)

Then, e.g.  \( \text{comp}(p) \models \text{member}(a,[a,b]), \neg \text{member}(a,[ ]), \neg \text{member}(a,[b,c]) \)
Completion (1)

- The completion of extended program $P$ (denoted by $\text{comp}(P)$), is the set of formulas constructed from $P$ by the following 6 steps:

1. Associate with every $n$-ary predicate symbol $p$ a sequence of pairwise distinct variables $x_1, \ldots, x_n$ which do not occur in $P$.

2. Transform each clause $c = p(t_1, \ldots, t_n) \leftarrow B$ into

   $$ p(x_1, \ldots, x_n) \leftarrow x_1 = t_1, \ldots, x_n = t_n, B $$

3. Transform each resulting formula $p(x_1, \ldots, x_n) \leftarrow G$ into

   $$ p(x_1, \ldots, x_n) \leftarrow \exists z \ G $$

   where $z$ is a sequence of the elements of $\text{Vars}(c)$.  

Completion (2)

4. For every n-ary predicate symbol $p$, let

$$p(x_1, \ldots, x_n) \leftarrow \exists z_1 G_1, \ldots, p(x_1, \ldots, x_n) \leftarrow \exists z_m G_m$$

be all implications obtained in step 3 ($m \geq 0$).

4.a) If $m > 0$, then replace these by the formula

$$\forall x_1, \ldots, x_n p(x_1, \ldots, x_n) \leftrightarrow \exists z_1 G_1 \lor \ldots \lor \exists z_m G_m$$

(if some $G_i$ is empty, replace it by $true$.)

4.b) If $m = 0$, i.e predicate $p$ had no defining clause, then add the formula

$$\forall x_1, \ldots, x_n p(x_1, \ldots, x_n) \leftrightarrow false.$$
5. Add the **standard axioms of equality**

5.1 $\forall [ x = x ]$ \hspace{2cm} \% reflexivity

5.2 $\forall [ x = y \rightarrow y = x ]$ \hspace{2cm} \% simmetry

5.3 $\forall [ x = y , y = z \rightarrow x = z ]$ \hspace{2cm} \% transitivity

5.4 $\forall [ x_i = y \rightarrow f(x_1, ..., x_i, ..., x_n) = f(x_1, ..., y, ..., x_n) ]$ \hspace{1cm} \% substitutivity

5.5 $\forall [ x_i = y \rightarrow p(x_1, ..., x_i, ..., x_n) \leftrightarrow p(x_1, ..., y, ..., x_n) ]$ \hspace{1cm} \% substitutivity

6. Add the **standard axioms of inequality**

6.1 $\forall [ x_1 \neq y_1 \ldots \neq x_m \neq y_m \rightarrow f(x_1, ..., x_i, ..., x_n) \neq f(y_1, ..., y_i, ..., y_n) ]$

6.2 $\forall [ f(x_1, ..., x_i, ..., x_n) \neq g(y_1, ..., y_i, ..., y_n) ]$ \hspace{1cm} (when $f \neq g$)

6.3 $\forall [ x \neq t ]$ (when $x$ is a proper subterm of $t$)

- Notice that axioms 6.1 to 6.3 are a restriction of FO equality to the **UNA** (**Unique Names Assumption**) – “different names denote different entities”
Soundness of SLDNF-Resolution

Given extended program \( P \), extended query \( Q \) and substitution \( \theta \):

- \( \theta \mid_{\text{var}(Q)} \) is a **correct answer substitution** of \( Q \) :\( \iff \) \( \text{comp}(P) \models Q \theta \)

- \( Q\theta \) is a **correct instance** of \( Q \) :\( \iff \) \( \text{comp}(P) \models Q \theta \)

**Theorem (Lloyd, 1987)**

If there exists a successful SLDNF-derivation of \( P \cup \{Q\} \) with CAS \( \theta \), then

\[
\text{comp}(P) \models Q \theta.
\]

**Corollary (Lloyd, 1987)**

If there exists a successful SLDNF-derivation of \( P \cup \{Q\} \), then

\[
\text{comp}(P) \models \exists Q.
\]
Incompleteness of SLDNF (Inconsistency)

Thus, $\text{comp}(P) \supseteq \{ p \leftrightarrow \neg p \} = \{ \text{false} \}$

Hence, $\text{comp}(p) \models p$ and $\text{comp}(p) \models \neg p$

- In this case $\text{comp}(P)$ is inconsistent, as it has no model, since for every $I$, $I \models \neg \text{comp}(P)$

In this case of inconsistency, SLDNF-resolution is incomplete, since there is neither a successful SLDNF-derivation for $P \cup \{p\}$, nor for $P \cup \{\neg p\}$,
Incompleteness of SLDNF (Non Strictness)

Thus, \( \text{comp}(P) \supseteq \{ p \leftrightarrow q \land \neg q, \ \neg q \leftrightarrow q \} = \{ p \leftrightarrow \text{true} \} \)

Hence, \( \text{comp}(p) \models p. \)

In this case, of non-strictness (\( p \) depends on \( q \) both evenly and oddly, see slide 18), SLDNF-resolution is incomplete, since there is no successful SLDNF-derivation for \( P \cup \{ p \} \).

Notice that in the absence of one of the first two clauses, there would be no incompleteness, since \( p \leftrightarrow \text{true} \) would not be in \( \text{comp}(P) \).
Incompleteness of SLDNF (Floundering)

Thus, \( \text{comp}(P) \supset \{ \forall x_1 p(x_1) \leftrightarrow \exists x (x_1 = x, \neg q(x)) , \forall x_1 q(x_1) \leftrightarrow \text{false} \} \) "="

\{ \forall x_1 p(x_1) \leftrightarrow \text{true} , \forall x_1 q(x_1) \leftrightarrow \text{false} \}

Hence, \( \text{comp}(p) \models \forall x_1 p(x_1) \).

In this case, of floundering, SLDNF-resolution is incomplete, since there is no successful SLDNF-derivation for \( P \cup \{p(x_1)\} \)

**Note:** SLDNF blocks query \( \neg q(x) \), contrary to Prolog. Prolog is not incomplete in this case, but for the wrong reasons!
Incompleteness of SLDNF (Unfairness)

Thus, \( \text{comp}(P) \models \{ r \leftarrow p, q \}, \ p \leftarrow p, \ q \leftarrow \text{false} \) \( \models \) \( \{ r \leftarrow \text{false}, \ q \leftarrow \text{false} \} \)

Hence, \( \text{comp}(P) \not\models \neg r \).

In this case, SLDNF-resolution might be incomplete, since there is no successful SLDNF-derivation for \( P \cup \{ \neg r \} \) when the leftmost selection rule (as in Prolog) is adopted (unfairness of the selection rule).
A dependency graph $D_p$ of an extended program $P$

$:\iff$

directed graph with labelled edges, where

- The nodes are the predicate symbols of $P$
- The edges are either positive (labelled $+$) or negative (labelled $-$);

- $p \rightarrow^+ q$ edge in $D_p$ $:\iff$

  $P$ contains a clause $p(s_1, \ldots, s_m) \leftarrow L, q(t_1, \ldots, t_n), R$

- $p \rightarrow^- q$ edge in $D_p$ $:\iff$

  $P$ contains a clause $p(s_1, \ldots, s_m) \leftarrow \neg L, \neg q(t_1, \ldots, t_n), R$
Strict, Hierarchical and Stratified Programs

Given an extended program $P$, with dependency graph $D_p$, with predicate symbols $p, q$, and an extended query $Q$:

- **$p$ depends evenly / oddly on $q$ :↔**
  
  There is a path in $D_p$ from $p$ to $q$ with an even (including 0) / odd number of negative edges.

- **$P$ is strict wrt. $Q$ :↔**
  
  No predicate symbol occurring in $Q$ depends both evenly and oddly on a predicate symbol in the head of a clause in $P$.

- **$P$ is hierarchical :↔**
  
  No cycle exists in $D_p$

- **$P$ is stratified :↔**
  
  No cycle with a negative edge exists in $D_p$
Strict, Hierarchical and Stratified Examples (1)

\[ P_1: \ p \leftarrow \neg p \]

- \( p \) depends on itself (oddly)
- \( P_1 \) is strict (no even and odd dependency of \( p \) on any predicate)
- \( P_1 \) is not hierarchical (there is a cycle in \( P_1 \))
- \( P_1 \) is not stratified (there is a cycle with a negative edge in \( P_1 \))

\[ P_2: \]
\[ p \leftarrow q \]
\[ p \leftarrow \neg q \]
\[ q \leftarrow q \]

- \( p \) depends evenly (0) and oddly (1) on \( q \), which depends on itself (evenly)
- \( P_2 \) is not strict (there is an even and an odd dependency of \( p \) on \( q \))
- \( P_2 \) is not hierarchical (there is a cycle in \( P_2 \), as \( q \) depends on itself)
- \( P_2 \) is stratified (there is no cycle with a negative edge in \( P_2 \))
Strict, Hierarchical and Stratified Examples (2)

\[ P_3: \quad p(x) \leftarrow \neg q(x) \]

- \( p \) depends oddly on \( q \)
- \( P_3 \) is strict (no even and odd dependency of predicate \( p/1 \) on any predicate)
- \( P_1 \) is hierarchical (there is no cycle in \( P_3 \))
- \( P_1 \) is stratified (there is no cycle with a negative edge in \( P_3 \))

\[ P_4: \quad r \leftarrow p, q \]
\[ \quad p \leftarrow p \]

- \( r \) depends evenly on \( q \) and \( p \), that depends evenly on itself
- \( P_4 \) is strict (no predicate depends evenly and oddly on another predicate)
- \( P_2 \) is not hierarchical (there is a cycle in \( P_4 \), the self dependency of \( p \))
- \( P_2 \) is stratified (there is no cycle with a negative edge in \( P_1 \))
Restricted Completeness of SLDNF-resolution (1)

Theorem (Lloyd, 1987)

Let $P$ be a hierarchical and allowed program and $Q$ an allowed query.

If $\text{comp}(P) \models Q^\theta$ for some $\theta$ such that $Q^\theta$ is ground, then there is a successful SLDNF-derivation of $P \cup \{Q\}$ with CAS $\theta$.

**Note 1:** A SLDNF-derivation might not be found by an interpreter with an arbitrary selection rule (due to trapping in infinite derivations).

**Note 2:** The theorem applies to safe interpreters that adopt selection rules that are safe (do not select negative literals that are not ground), unlike most Prolog interpreters.
Restricted Completeness of SLDNF-resolution (2)

Theorem (Lloyd, 1987)

Let \( P \) be a **stratified** and **allowed** program and \( Q \) an allowed query, such that \( P \) is **strict** wrt. \( Q \)

If \( \text{comp}(P) \models Q^\theta \) for some \( \theta \) such that \( Q^\theta \) is ground, then there is a successful SLDNF-derivation of \( P \cup \{Q\} \) with \( \text{CAS} \theta \).

**Notes:** The selection rule must be **safe** and **fair** (not trapped in infinite derivations).
An extended selection rule $R$ is **fair** if:

- Either $\xi$ is failed; or
- For every literal $L$ occurring in a query of $\xi$, (some further instantiated version of $L$) $L$ is selected within a finite number of derivation steps.

Examples:

- Selection rule “select leftmost literal” is unfair (depth-first search).
- Selection rule “select leftmost literal to the right of the literals introduced in the previous derivation step, if it exists, otherwise select leftmost literal” is fair (breadth-first search).
Let $P$ be an extended program and $I$ a Herbrand interpretation. Then

$$T_P(I) := \{ H \mid H \leftarrow B \in \text{ground}(P), I \models B \}$$

When $P$ is a definite program, then

- $T_P$ is monotonic
- $T_P$ is continuous
- $T_P$ has the least fixpoint $M(P)$

- $M(P) = T_P \uparrow \omega$

In case of extended programs all these properties are lost, since $T_P$ is not monotonic.
Extended Consequence Operator is Not Continuous

\[ P_a: \ p \leftarrow \neg q \]

Let \( I = \{ q \} \)
- Then \( T_P(I) = \{ \} \),
- Hence, it is not the case that \( I \subseteq T_P(I) \).
- Therefore, \( T_P \) is not monotonic (nor continuous) for \( P_a \) (due to negation).

\[ P_b: \ p \leftarrow p \]

Let \( I_1 = \{ \} \) and \( I_2 = \{ p \} \)
- Then \( T_P(I_1) = I_1 = \{ \} \), and \( T_P(I_2) = I_2 = \{ p \} \),
- Therefore, \( T_P \) is monotonic (and continuous) for \( P_b \). (and all with no negation)
Extended $T_P$-Characterization (1)

**Lemma 4.3** ([ApBo94]):

Let $P$ be an extended program and $I$ a Herbrand interpretation. Then

$$I |= P \text{ iff } T_P(I) \subseteq I$$

**Proof.**

$$I |= P$$

iff for every $H \leftarrow B \in \text{ground}(P)$ : $I |= B$ implies $I |= H$

iff for every $H \leftarrow B \in \text{ground}(P)$ : $I |= B$ implies $H \in I$

iff for every ground atom $H$ : $H \in T_P(I)$ implies $H \in I$

iff $T_P(I) \subseteq I$
Extended $T_P$-Characterization (2)

**Definition**

Let $F$ and $\Pi$ be ranked alphabets of function and predicate symbols, respectively, let $= \notin \Pi$ be a binary predicate symbol (for “equality”), and let $I$ be a Herbrand interpretation for $F$ and $\Pi$.

Then $I_\approx := I \cup \{ t = t \mid t \in HU_F \}$ is called a standardized Herbrand interpretation for $F$ and $\Pi$.

**Lemma 4.4** ([ApBo94]):

Let $P$ be an extended program and $I$ a Herbrand interpretation. Then

$$I_\approx \models \text{comp}(P) \iff T_P(I) = I$$
Proof sketch of Lemma 4.4:

\( I_\subseteq \models \text{comp}(P) \)

iff for every ground atom \( H \):

\( I \models H \iff \forall (H \leftarrow B \in \text{ground}(P) \ B) \)

(since \( I_\subseteq \) is a model for standard axioms of equality and inequality)

iff for every ground atom \( H \):

\( H \subseteq I \iff I \models B \) for some \( H \leftarrow B \in \text{ground}(P) \)

iff for every ground atom \( H \):

\( H \subseteq I \iff H \in T_P(I) \)

iff \( T_P(I) = I \)
Extended $\mathcal{T}_P$-Characterization: Example (1)

Let

\[
\begin{align*}
I_0 &= \{ \}, \\
I_1 &= \{ p \}, \\
I_2 &= \{ q \}, \\
I_3 &= \{ p, q \},
\end{align*}
\]

Then

\[
\begin{align*}
\mathcal{T}_P(I_0) &= \{ p \}, \\
\mathcal{T}_P(I_1) &= \{ p \}, \\
\mathcal{T}_P(I_2) &= \{ \}, \\
\mathcal{T}_P(I_3) &= \{ \}.
\end{align*}
\]

- $\mathcal{T}_P(I_1) = I_1$. Hence, $I_1 \models \text{comp}(P_a)$ and $I_1 \models P_a$

- $\mathcal{T}_P(I_2) \subseteq I_2$, but $\mathcal{T}_P(I_2) \neq I_2$. Hence, $I_2 \models P_a$, but $I_2 \nRightarrow \text{comp}(P_a)$

- $\mathcal{T}_P(I_3) \subseteq I_3$, but $\mathcal{T}_P(I_3) \neq I_3$. Hence, $I_3 \models P_a$, but $I_3 \nRightarrow \text{comp}(P_a)$

- $\mathcal{T}_P(I_0) \nsubseteq I_0$. Hence, $I_0 \nmodels P_a$
**Extended $T_p$-Characterization: Example (2)**

| $P_b$: | $p \leftarrow \neg q$
| \hline | $q \leftarrow q$
| \hline | $\text{comp}(P_b) = \{ p \leftrightarrow \neg q \}$

Let $I_0 = \{ \}$, $I_1 = \{ p \}$, $I_2 = \{ q \}$, $I_3 = \{ p, q \}$.

Then $T_p(I_0) = \{ p \}$, $T_p(I_1) = \{ p \}$, $T_p(I_2) = \{ q \}$, $T_p(I_3) = \{ q \}$.

- $T_p(I_1) = I_1$. Hence, $I_1 \models \text{comp}(P_b)$ and $I_1 \models P_b$.
- $T_p(I_2) = I_2$. Hence, $I_2 \models \text{comp}(P_b)$ and $I_1 \models P_b$.
- $T_p(I_3) \subseteq I_3$, but $T_p(I_3) \neq I_3$. Hence, $I_3 \models P_b$, but $I_3 \nmodels \text{comp}(P_b)$.
- $T_p(I_0) \nsubseteq I_0$. Hence, $I_0 \nmodels P_b$. 

Foundations of Logic and Constraint Programming
Completion may be Inadequate

Thus, $\text{comp}(P) \supseteq \{ \text{ill} \leftrightarrow \neg \text{ill}, \text{infection} \leftrightarrow \text{true} \}$ "="

$\{\text{ill} \leftrightarrow \neg \text{ill}, \text{infection} \leftrightarrow \text{true} \}$

is inconsistent (it has no models).

Hence, $\text{comp}(P) \models \text{healthy}$

But, $I = \{ \text{ill}, \text{infection} \}$ is the only Herbrand model of $P$.

Hence, $P \not\models \text{healthy}$
Non-Intended Herbrand Models

Because $T_p$ is not monotonic nor continuous, it is not possible, in general, to get a fixpoint, nor a least model from an extended program. Nevertheless, there are *minimal* models, although not necessarily unique.

\[ P_1: \ p \leftarrow \neg q \]

$P_1$ has three Herbrand models

\[ M_1 = \{p\}, \quad M_2 = \{q\}, \quad \text{and} \quad M_3 = \{p, q\} \]

$P_1$ has *no* least, but *two* minimal models: $M_1$ and $M_2$.

However: $M_1$ and not $M_2$, is the "intended" model of $P_1$. 


A Herbrand interpretation $I$ is supported:

\[ I \models B \]

For every $H \in I$ there exists some $H \leftarrow B \in \text{ground}(P)$ such that $I \models B$

(intuitively: $B$ is an explanation for $H$)

Example:

\[
P_1: \quad p \leftarrow \neg q
\]

$M_1 = \{p, \neg q\}, \quad M_2 = \{\neg p, q\}, \quad \text{and} \quad M_3 = \{p, q\}$

- $M_1$ is a supported model of $P_1$ ($\neg q$ is the explanation for $p$)
- $M_2$ is not a supported model of $P_1$ (there is no explanation for $q$)

Note that $T_p(M_2) = \emptyset \subseteq M_2$ and that $T_p(M_1) = M_1$. 
Extended $T_P$-Characterization (4)

Lemma 6.2 ([ApBo94]):

Let $P$ be an extended program and $I$ a Herbrand interpretation. Then

$$I \models P \text{ and } I \text{ supported} \iff T_P(I) = I$$

Proof (sketch):

$I \models P \text{ and } I \text{ supported}$
- iff for every $H \leftarrow B \in \text{ground}(P): \ I \models B \implies I \models H$ and
  - for every $H \in I$: $I \models (H \leftarrow B \in \text{ground}(P) \ B)$
  - iff for every ground atom $H$: $I \models (H \leftarrow B \in \text{ground}(P) \ B)$
    and $I \models (H \rightarrow B \in \text{ground}(P) \ B)$
- iff for every $H \in I$: $I \models (H \leftarrow B \in \text{ground}(P) \ B)$
- $I_\vDash$ is a model of $\text{comp}(P)$
- iff $T_P(I) = I$ (cf. Lemma 4.4)
Non-Intended Supported Models

\[ P_2: \quad p \leftarrow \neg q \]
\[ q \leftarrow q \]

- \( P_2 \) has three Herbrand models

\[ M_1 = \{p\}, \quad M_2 = \{q\}, \quad \text{and} \quad M_3 = \{p, q\} \]

- \( P_2 \) has two supported Herbrand models: \( M_1 \) and \( M_2 \).

In fact, both \( M_1 \) and \( M_2 \) are minimal models for \( \text{comp}(P_2) = \{p \leftarrow \neg q\} \)

- However: \( M_1 \) and not \( M_2 \), is the “intended” model of \( P_2 \).

\( M_1 \) is called the standard model of \( P_2 \) (cf. later slide)

- In general, it is not possible to define standard models, unless the programs are stratified!
Stratifications

Let $P$ be an extended program and $D_p$ its dependency graph,

- Predicate symbol $p$ is **defined** in $P$ $\iff$ $P$ contains a clause $p(t_1, \ldots, t_m) \leftarrow B$

- $P_1 \cup \ldots \cup P_n = P$ is a stratification of $P$ $\iff$
  - $P_i \neq \emptyset$ for every $i \in 1..n$
  - $P_i \cap P_j = \emptyset$ for every $i,j \in 1..n$, with $i \neq j$
  - for every $p$ defined in $P_i$ and edge $p \rightarrow^* q$ in $D_p$:
    - $q$ is not defined in $\bigcup_{i+1 \leq j < n} P_j$
  - for every $p$ defined in $P_i$ and edge $p \rightarrow q$ in $D_p$:
    - $q$ is not defined in $\bigcup_{i \leq j < n} P_j$

**Lemma 6.5 ([ApBo94]):**

An extended program is stratified iff it admits a (not necessarily unique) stratification.
Stratifications Example (1)

- \( P_1 \cup P_2 \cup P_3 \) is a stratification of \( P \), where
  - \( P_1 = \{ \text{num}(0) \leftarrow \}, \ \text{num}(s(x)) \leftarrow \text{num}(x) \} \)
  - \( P_2 = \{ \text{zero}(0) \leftarrow \} \)
  - \( P_3 = \{ \text{positive}(x) \leftarrow \text{num}(x), \neg \text{zero}(x) \} \)

\[
\begin{align*}
\text{zero}(0) & \leftarrow \\
\text{positive}(x) & \leftarrow \text{num}(x), \neg \text{zero}(x) \\
\text{num}(0) & \leftarrow \\
\text{num}(s(x)) & \leftarrow \text{num}(x)
\end{align*}
\]
Stratifications Example (2)

\[
\begin{align*}
\text{even}(0) & \leftarrow \\
\text{even}(x) & \leftarrow \neg \text{odd}(x), \text{num}(x) \\
\text{odd}(s(x)) & \leftarrow \text{even}(x) \\
\text{num}(0) & \leftarrow \\
\text{num}(s(x)) & \leftarrow \text{num}(x)
\end{align*}
\]

\[P \text{ admits no stratification}\]

since

\[
\text{strat(even)} > \text{strat(odd)} \quad \text{but} \quad \text{strat(odd)} \geq \text{strat(even)}
\]
Standard Models (Stratified Programs)

Let

- \( I \) be an Herbrand interpretation and \( \Pi \) a set of predicate symbols:

- \( I \upharpoonright \Pi \iff I \cap \{ p(t_1, \ldots, t_m) \mid p \in \Pi, t_1, \ldots, t_m \text{ ground terms} \} \)

- \( P_1 \cup \ldots \cup P_n \) be a stratification of extended program \( P \).

Then

- \( M_1 \iff \) least Herbrand model of \( P_1 \) such that
  \[
  M_1 \upharpoonright \{ p \mid p \text{ not defined in } P \} = \emptyset
  \]

- \( M_2 \iff \) least Herbrand model of \( P_2 \) such that
  \[
  M_2 \upharpoonright \{ p \mid p \text{ defined nowhere or in } P_1 \} = M_1
  \]

- \( M_n \iff \) least Herbrand model of \( P_n \) such that
  \[
  M_n \upharpoonright \{ p \mid p \text{ defined nowhere or in } P_1 \cup \ldots \cup P_{n-1} \} = M_{n-1}
  \]

We call \( M_P = M_n \) the **standard model** of \( P \).
Standard Models: Example

Let $P_1 \cup P_2 \cup P_3$ with
- $P_1 = \{ \text{num}(0) \leftarrow, \text{num}(s(x)) \leftarrow \text{num}(x) \}$
- $P_2 = \{ \text{zero}(0) \leftarrow \}$
- $P_3 = \{ \text{positive}(x) \leftarrow \text{num}(x), \neg \text{zero}(x) \}$

be a stratification of $P$. Then
- $M_1 = \{ \text{num}(t) \mid t \in \text{HU}_{\{s,0\}} \}$
- $M_2 = \{ \text{num}(t) \mid t \in \text{HU}_{\{s,0\}} \} \cup \{\text{zero}(0)\}$
- $M_3 = \{ \text{num}(t) \mid t \in \text{HU}_{\{s,0\}} \} \cup \{\text{zero}(0)\} \cup \{\text{positive}(t) \mid t \in \text{HU}_{\{s,0\}} - \{0\} \}$

Hence $M_P = M_3$ is the standard model of $P$
Properties of Standard Models

**Theorem 6.7** ([ApBo94]):

Consider a **stratified** program $P$. Then

- $M_P$ does not depend on the chosen stratification of $P$,
- $M_P$ is a minimal model of $P$,
- $M_P$ is a supported model of $P$,
- $M_P$ is a supported model of $P$,

**Corollary:**

For a **stratified** program $P$, $\text{comp}(P)$ admits a Herbrand Model.