Chapter 7

Negation: Declarative Interpretation

Outline

- First-Order Formulas and Logical Truth
- The Completion semantics
- Soundness and restricted completeness of SLDNF-Resolution
- Extended consequence operator
- An alternative semantics: Standard models

[Lloyd, 1987] J.W. Lloyd. Foundations of Logic Programming: Second, Extended Edition. *Springer Verlag*, 1987.

[Cavedon and Lloyd, 1989] L. Cavedon and J.W. Lloyd. A Completeness Theorem for SLDNF Resolution. *Journal of Logic Programming*, 7:177-191, 1989.

[Apt and Bol, 1994] K. Apt and R. Bol. Logic Programming and Negation: A Survey. *Journal of Logic Programming*, 19/20: 9-71, 1994.

First-Order Formulas

$\prod$, F ranked alphabets of predicate symbols and function symbols, respectively,
V set of variables

The (first-order) formulas (over $\prod$, F, and V) are inductively defined as follows:

- if $A \in TB_{\prod,F,V}$, then A is a formula
- if G_1 and G_2 are formulas, then $\neg G_1$, $G_1 \wedge G_2$ (written G_1, G_2),$G_1 \vee G_2$, $G_1 \leftarrow G_2$, and $G_1 \leftrightarrow G_2$ are formulas
- if G_1 is formula and $x \in V$, then $\forall x\, G$ and $\exists x\, G$ are formulas

Extended Notion of Logical Truth (I)

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

G true in I under σ, written $I \models_\sigma G :\Leftrightarrow$

- $I \models_\sigma p(t_1, \ldots, t_n) :\Leftrightarrow (\sigma(t_1), \ldots, \sigma(t_n)) \in p_I$
- $I \models_\sigma \neg G :\Leftrightarrow I \not\models_\sigma G$
- $I \models_\sigma G_1 \wedge G_2 :\Leftrightarrow I \models_\sigma G_1$ and $I \models_\sigma G_2$
- $I \models_\sigma G_1 \vee G_2 :\Leftrightarrow I \models_\sigma G_1$ or $I \models_\sigma G_2$
- $I \models_\sigma G_1 \leftarrow G_2 :\Leftrightarrow$ if $I \models_\sigma G_2$ then $I \models_\sigma G_1$
- $I \models_\sigma G_1 \leftrightarrow G_2 :\Leftrightarrow I \models_\sigma G_1$ iff $I \models_\sigma G_2$
- $I \models_\sigma \forall x\, G :\Leftrightarrow$ for every $d \in D$: $I \models_{\sigma'} G$
- $I \models_\sigma \exists x\, G :\Leftrightarrow$ for some $d \in D$: $I \models_{\sigma'} G$

where $\sigma' : V \rightarrow D$ with $\sigma'(x) = d$ and $\sigma'(y) = \sigma(y)$ for every $y \in V - \{x\}$

Extended Notion of Logical Truth (II)

G formula, S, T sets of formulas, I, interpretation

Let $x_1, \ldots, x_k$ be the variables occurring in G.

- $\forall x_1, \ldots, \forall x_k\, G$ universal closure of G (abbreviated $\forall G$)
- $I \models \forall G :\Leftrightarrow I \models_\sigma G$ for every state σ
- $I \models_\sigma p(t_1, \ldots, t_n) :\Leftrightarrow (\sigma(t_1), \ldots, \sigma(t_n)) \in p_I$
- G true in I (or: I model of G), written: $I \models G :\Leftrightarrow I \models \forall G$
- I model of S, written: $I \models S :\Leftrightarrow I \models G$ for every $G \in S$
- T semantic (or: logical) consequence of S, written $S \models T$ $:\Leftrightarrow$ every model of S is a model of T

Programs Never Have Negative Consequences (I)

P_{mem}:

$$\boxed{\begin{array}{l} member(x, [x|y]) \leftarrow \\ member(x, [y|z]) \leftarrow member(x, z) \end{array}}$$

Then $P_{mem} \models member(a, [a,b])$ and $P_{mem} \not\models member(a, [\,])$.

But also $P_{mem} \not\models \neg member(a, [\,])$, since

$$HB_{\{member\},\{|,[\,],a\}} \models P_{mem} \quad \text{and} \quad HB_{\{member\},\{|,[\,],a\}} \not\models \neg member(a, [\,]).$$

Nevertheless the SLDNF-tree of $P_{mem} \cup \{\neg member(a, [\,])\}$ is successful:

$$\begin{array}{ccc} \underline{\neg member(a,[\,])} & & \\ | & & \underline{member(a(,[\,])} \\ \square & & \text{failure} \\ \text{success} & & \end{array}$$

Programs Never Have Negative Consequences (II)

Problem: For every extended program P the "corresponding" Herbrand base is a model.

Hence: No negative ground literal L can ever be a logical consequence of P.

But: SLDNF-tree of $P \cup \{L\}$ may be successful!

$\Rightarrow$ Soundness of SLDNF-resolution?

Solution: Strengthen P by completion ("replace implications by equivalences") to *comp*(P) and compare SLDNF-resolution with *comp*(P) instead of P!

Completion (Example I)

$$\begin{array}{llll} P: & happy & \leftarrow & sun, holidays \\ & happy & \leftarrow & snow, holidays \\ & snow & \leftarrow & cold, precipitation \\ & cold & \leftarrow & winter \\ & precipitation & \leftarrow & holidays \\ & winter & \leftarrow & \\ & holidays & \leftarrow & \end{array}$$

$$\begin{array}{llll} comp(P): & happy & \leftrightarrow & (sun, holidays) \vee (snow, holidays) \\ & snow & \leftrightarrow & cold, precipitation \\ & cold & \leftrightarrow & winter \\ & precipitation & \leftrightarrow & holidays \\ & winter & \leftrightarrow & true \\ & holidays & \leftrightarrow & true \\ & sun & \leftrightarrow & false \end{array}$$

Then, $comp(P) \models happy, snow, cold, precipitation, winter, holidays, \neg sun$.

Completion (Example II)

$$
\begin{array}{llll}
P: & member(x, [x|y]) & \leftarrow & \\
 & member(x, [y|z]) & \leftarrow & member(x, z) \\
 & disjoint([\,], x) & \leftarrow & \\
 & disjoint([x|y], z) & \leftarrow & member(x, z), disjoint(y, z)
\end{array}
$$

$$
\begin{array}{llll}
comp(P): & \forall x_1, x_2\ member(x_1, x_2) & \leftrightarrow & \exists x, y\ (\ x_1 = x, x_2 = [x|y]) \vee \\
 & & & \exists x, y, z\ (\ x_1 = x, x_2 = [y|z], member(x, z)) \\
 & \forall x_1, x_2\ disjoint(x_1, x_2) & \leftrightarrow & \exists x\ (\ x_1 = [\,], x_2 = x\ ) \vee \\
 & & & (\exists x, y, z\ \ x_1 = [x|y], x_2 = z, \\
 & & & \qquad \neg member(x, z), disjoint(y, z))
\end{array}
$$

plus standard equality and inequality axioms

Then, e.g. $comp(P) \models member(a, [a|b]), \neg member(a, [\,]), \neg disjoint([a], [a])$.

Completion (I)

Completion of extended program P (denoted by $comp(P)$) 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 $c = p(t_1, \ldots, t_n) \leftarrow \underline{\underline{B}}$ into

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

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

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

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

Completion (II)

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

$$p(x_1, \ldots, x_n) \leftarrow \exists \underline{z}_1 \, G_1, \ldots, p(x_1, \ldots, x_n) \leftarrow \exists \underline{z}_m \, G_m$$

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

- If $m > 0$, then replace these by the formula

$$\forall x_1, \ldots, x_n \;\; p(x_1, \ldots, x_n) \leftrightarrow \exists \underline{z}_1 \, G_1 \vee \ldots \vee \exists \underline{z}_m \, G_m$$

(If some $\exists \underline{z}_i \, G_i$ is empty, then replace it by $true$.)

- If $m = 0$, then add the formula

$$\forall x_1, \ldots, x_n \;\; p(x_1, \ldots, x_n) \leftrightarrow false$$

Completion (III)

5. Standard axioms of equality

$\forall\, [\, x = x \,]$

$\forall\, [\, x = y \rightarrow y = x \,]$

$\forall\, [\, x = y \wedge y = z \rightarrow x = z \,]$

$\forall\, [\, x_i = y \rightarrow f(x_1, \ldots, x_i, \ldots, x_n) = f(x_1, \ldots, y, \ldots, x_n) \,]$

$\forall\, [\, x_i = y \rightarrow (p(x_1, \ldots, x_i, \ldots, x_n) \leftrightarrow p(x_1, \ldots, y, \ldots, x_n)) \,]$

6. Standard axioms of inequality

$\forall\, [\, x_1 \neq y_1 \vee \ldots \vee x_n \neq y_n \rightarrow f(x_1, \ldots, x_n) \neq f(y_1, \ldots, y_n) \,]$

$\forall\, [\, f(x_1, \ldots, x_m) \neq g(y_1, \ldots, y_n) \,]$ (whenever $f \neq g$)

$\forall\, [\, x \neq t \,]$ (whenever x is proper subterm of t)

5. and 6. ensure that = must be interpreted as equality!

Soundness of SLDNF-Resolution

P extended program, Q extended query, θ substitution:

- $\theta \mid_{Var(Q)}$ correct answer substitution of Q :⇔ $comp(P) \models Q\theta$
- $Q\theta$ correct instance of Q :⇔ $comp(P) \models Q\theta$

Theorem (cf. e.g. [Lloyd, 1987])

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

Corollary

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

SLDNF-Resolution is Not Complete (I): Inconsistency

P : $\boxed{p \leftarrow \neg p}$

$comp(P) \supseteq \{p \leftrightarrow \neg p\}$ "=" $\{false\}$.

Hence, $comp(P) \models p$ and $comp(P) \models \neg p$.

(because $I \not\models comp(P)$ for every interpretation I, i.e. $comp(P)$ is inconsistent)

But there is neither a successful SLDNF-derivation of $P \cup \{p\}$ nor of $P \cup \{\neg p\}$.

SLDNF-Resolution is Not Complete (II): Non-Strictness

P :

$$\begin{array}{l} p \leftarrow q \\ p \leftarrow \neg q \\ q \leftarrow q \end{array}$$

$comp(P) \supseteq \{p \leftrightarrow q \vee \neg q, q \leftrightarrow q\}$ "=" $\{p \leftrightarrow true\}$.

Hence, $comp(P) \models p$.

But there is no successful SLDNF-derivation of $P \cup \{p\}$.

SLDNF-Resolution is Not Complete (III): Floundering

P : $\boxed{p(x) \leftarrow \neg q(x)}$

$comp(P) \supseteq \{\forall x_1\ p(x_1) \leftrightarrow \exists x\ \ x_1 = x, \neg q(x), \quad \forall x_1\ q(x_1) \leftrightarrow \mathit{false}\}$
"=" $\{\forall x_1\ p(x_1) \leftrightarrow \mathit{true}, \quad \forall x_1\ q(x_1) \leftrightarrow \mathit{false}\}$.

Hence, $comp(P) \models \forall x_1\ p(x_1)$.

But there is no successful SLDNF-derivation of $P \cup \{p(x_1)\}$.

SLDNF-Resolution is Not Complete (IV): Unfairness

P :

$$r \leftarrow p, q$$
$$p \leftarrow p$$

$comp(P) \supseteq \{r \leftrightarrow p, q, \quad p \leftrightarrow p, \quad q \leftrightarrow false\}$ "=" $\{r \leftrightarrow false, \quad q \leftrightarrow false\}$.

Hence, $comp(P) \models \neg r$.

But there is no successful SLDNF-derivation of $P \cup \{\neg r\}$ w.r.t. leftmost selection rule.

Dependency Graphs

dependency graph D_P of an extended program P

$:\Leftrightarrow$

directed graph with labeled edges, where

- the nodes are the predicate symbols of P;
- the edges are either labeled by + (positive edge) or by – (negative egde);
- $p \xrightarrow{+} q$ edge in $D_P :\Leftrightarrow$

  P contains a clause $p(s_1, ..., s_m) \leftarrow \underline{\underline{L}}, q(t_1, ..., t_n), \underline{\underline{N}}$
- $p \xrightarrow{-} q$ edge in $D_P :\Leftrightarrow$

  P contains a clause $p(s_1, ..., s_m) \leftarrow \underline{\underline{L}}, \neg q(t_1, ..., t_n), \underline{\underline{N}}$

Strict, Hierarchical, Stratified Programs

P extended program, D_P dependency graph of P, p, q predicate symbols, Q extended query:

- p depends evenly (resp. oddly) on q :⇔
  there is a path in D_P from p to q with
  an even–including 0–(resp. odd) number of negative edges
- P is strict w.r.t. Q :⇔
  no predicate symbol occuring 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

Restricted Completeness of SLDNF-Resolution (I)

Theorem ([Lloyd, 1987])

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

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

Note:

Theorem does not hold, if arbitrary selection rule is fixed!

Selection rule has to be safe!

Restricted Completeness of SLDNF-Resolution (II)

Theorem ([Cavedon and Lloyd, 1989])

Let P be a stratified and allowed program and Q be an allowed query, such that P is strict w.r.t. Q.

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

Note:

Theorem does not hold if arbitrary selection rule is fixed!

Selection rule has to be safe and fair!

Fair Selection Rules

(extended) selection rule $\mathcal{R}$ is fair $:\Leftrightarrow$
for every SLDNF-tree $\mathcal{F}$ via $\mathcal{R}$ and for every branch ξ in $\mathcal{F}$:

- either ξ is failed
- or for every literal L occurring in a query of ξ, (some further instantiated version of) L is selected within a finite number of derivation steps

Example:

- selection rule “select leftmost literal” is unfair
- selection rule “select leftmost literal to the right of the literals introduced at the previous derivation step, if it exists; otherwise select leftmost literal” is fair

Extended Consequence Operator

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

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

In case P is a definite program, we know that

- T_P is monotonic,
- T_P is continuous,
- T_P has the least fixpoint $\mathcal{M}(P)$,
- $\mathcal{M}(P) = T_P{}^{\omega}$.

In case of extended programs all of these properties are lost!

Extended T_P-Characterization (I)

Lemma 4.3 ([Apt and Bol, 1994])
Let P be an extended program and I a Herbrand interpretation.
Then

$$I \models P \quad \text{iff} \quad T_P(I) \subseteq I.$$

Proof:

$$I \models P$$

iff for every $H \leftarrow \underline{\underline{B}} \in \mathit{ground}(P)$: $\quad I \models \underline{\underline{B}}$ implies $I \models H$

iff for every $H \leftarrow \underline{\underline{B}} \in \mathit{ground}(P)$: $\quad I \models \underline{\underline{B}}$ implies $H \in I$

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

iff $T_P(I) \subseteq I$

Extended T_P-Characterization (II)

Definition

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

Then $I_= \;:\Leftrightarrow I \cup \{=(t, t) \mid t \in HU_F\}$ is called a standardized Herbrand interpretation for F and $\prod \cup \{=\}$.

Lemma 4.4 ([Apt and Bol, 1994])

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

Then

$$I_= \models comp(P) \quad \text{iff} \quad T_P(I) = I.$$

Extended T_P-Characterization (III)

Proof Idea of Lemma 4.4:

$$I_{=} \models comp(P)$$

iff (since $I_{=}$ is a model for standard axioms of equality and inequality)

for every ground atom H : $I \models (H \leftrightarrow \vee_{(H \leftarrow \underline{\underline{B}}) \in ground(P)} \underline{\underline{B}})$

iff for every ground atom H : $H \in I \Leftrightarrow I \models \underline{\underline{B}}$ for some $H \leftarrow \underline{\underline{B}} \in ground(P)$

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

iff $T_P(I) = I$

Completion may be Inadequate

$ill \leftarrow \neg ill, infection$

$infection \leftarrow$

$comp(P) \supseteq \{ill \leftrightarrow \neg ill, infection \quad , \quad infection \leftrightarrow true\}$

is inconsistent (it has no models).

Hence, $comp(P) \models healthy$.

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

Hence, $P \not\models healthy$.

Non-Intended Minimal Herbrand Models

P_1: $\boxed{p \leftarrow \neg q}$

P_1 has three Herbrand models:

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

P_1 has no least, but two minimal Herbrand models: M_1 and M_2

However: M_1, and not M_2, is the “intuitive” model of P_1.

Supported Herbrand Interpretations

A Herbrand interpretation I is supported

$:\Leftrightarrow$

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

(Intuition: $\underline{\underline{B}}$ is an "explanation" for H)

Example:

M_1 is a supported model of P_1. ($\neg q$ is explanation for p)

M_2 is no supported model of P_1.

Also note (cf. Lemma 4.3) that $T_{P_1}(M_2) = \emptyset \subseteq M_2$, but in particular $T_{P_1}(M_1) = M_1$.

Extended T_P-Characterization (IV)

Lemma 6.2 ([Apt and Bol, 1994])

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

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

Proof Idea:

$I \models P$ and I supported

iff for every $(H \leftarrow \underline{\underline{B}}) \in ground(P)$: $I \models \underline{\underline{B}}$ implies $I \models H$

and for every $H \in I$: $I \models \vee_{(H \leftarrow \underline{\underline{B}}) \in ground(P)} \underline{\underline{B}}$

iff for every ground atom H: $I \models (H \leftarrow \vee_{(H \leftarrow \underline{\underline{B}}) \in ground(P)} \underline{\underline{B}})$

and $I \models (H \rightarrow \vee_{(H \leftarrow \underline{\underline{B}}) \in ground(P)} \underline{\underline{B}})$

iff for every ground atom H: $I \models (H \leftrightarrow \vee_{(H \leftarrow \underline{\underline{B}}) \in ground(P)} \underline{\underline{B}})$

iff $I_=$ model for $comp(P)$

iff (Lemma 4.4) $T_P(I) = I$

Non-Intended Supported Models

P_2:

$$p \leftarrow \neg q$$
$$q \leftarrow q$$

P_2 has three Herbrand models:

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

P_2 has two supported Herbrand models: M_1 and M_2

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. slide VII/35).

Stratifications

P extended program and D_P dependency graph of P:

- predicate symbol p defined in P
  $:\Leftrightarrow$ P contains a clause $p(t_1, ..., t_n) \leftarrow \underline{\underline{B}}$
- $P_1 \cup ... \cup P_n = P$ stratification of P $:\Leftrightarrow$
  - $P_i \neq \emptyset$ 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 $p \xrightarrow{+} q$ in D_P: q not defined in $\bigcup_{j=i+1}^{n} P_j$
  - for every p defined in P_i and edge $p \xrightarrow{-} q$ in D_P: q not defined in $\bigcup_{j=i}^{n} P_j$

Lemma 6.5 ([Apt and Bol, 1994])

An extended program is stratified iff it admits a stratification.

Note: A stratified program may have different stratifications.

Example (I)

P:

$$
\begin{array}{|rl|}
\hline
zero(0) & \leftarrow \\
positive(x) & \leftarrow num(x), \neg zero(x) \\
num(0) & \leftarrow \\
num(s(x)) & \leftarrow num(x) \\
\hline
\end{array}
$$

$P_1 \cup P_2 \cup P_3$ is a stratification of P, where

$$P_1 = \{num(0) \leftarrow \quad , \quad num(s(x)) \leftarrow num(x)\}$$

$$P_2 = \{zero(0) \leftarrow\}$$

$$P_2 = \{positive(x) \leftarrow num(x), \neg zero(x)\}$$

Example (II)

P:

$$
\begin{array}{rl}
num(0) & \leftarrow \\
num(s(x)) & \leftarrow num(x) \\
even(0) & \leftarrow \\
even(x) & \leftarrow \neg odd(x),\ num(x) \\
odd(s(x)) & \leftarrow even(x)
\end{array}
$$

P admits no stratification.

Standard Models (Stratified Progams)

I Herbrand interpretation, $\prod$ set of predicate symbols:

$I \mid \prod \;:\Leftrightarrow\; I \cap \{p(t_1, ..., t_n) \mid p \in \prod, t_1, ..., t_n \text{ ground terms}\}$

Let $P_1 \cup ... \cup P_n$ be stratification of extended program P.

$M_1 :\Leftrightarrow$ least Herbrand model of P_1 such that

$$M_1 \mid \{p \mid p \text{ not defined in } P\} = \emptyset$$

$M_2 :\Leftrightarrow$ least Herbrand model of P_2 such that

$$M_2 \mid \{p \mid p \text{ defined nowhere or in } P_1\} = M_1$$

$\vdots$

$M_n :\Leftrightarrow$ least Herbrand model of P_n such that

$$M_n \mid \{p \mid p \text{ defined nowhere or in } P_1 \cup ... \cup P_{n-1}\} = M_{n-1}$$

We call $M_P = M_n$ the standard model of P.

Example (I)

Let $P_1 \cup P_2 \cup P_3$ with

$P_1 = \{num(0) \leftarrow \quad , \quad num(s(x)) \leftarrow num(x)\}$

$P_2 = \{zero(0) \leftarrow\}$

$P_3 = \{positive(x) \leftarrow num(x), \neg zero(x)\}$

be stratification of P.

Then:

$M_1 = \{num(t) \mid t \in HU_{\{s,0\}}\}$

$M_2 = \{num(t) \mid t \in HU_{\{s,0\}}\} \cup \{zero(0)\}$

$M_3 = \{num(t) \mid t \in HU_{\{s,0\}}\} \cup \{zero(0)\} \cup \{positive(t) \mid t \in HU_{\{s,0\}} - \{0\}\}$

Hence $M_P = M_3$ is the standard model of P.

Properties of Standard Models

Theorem 6.7 ([Apt and Bol, 1994])

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*.

Corollary

For a stratified program *P*, *comp*(*P*) admits a Herbrand model.

Objectives

- First-Order Formulas and Logical Truth
- The Completion semantics
- Soundness and restricted completeness of SLDNF-Resolution
- Extended consequence operator
- An alternative semantics: Standard models