4. Logic Programs

4.1. Syntax and Semantics of Logic Programs

4.2. Universality of Logic Programming

4.3. Indeterminism of Logic Programming

4.1. Syntax and Semantics of Logic Programs

Horn clauses \( \subseteq \) clauses in logic programs

But in logic programming, the order of literals in a clause and of program clauses in a program plays a role.

Therefore, from now on:

Clause = sequence of literals (literals can also occur repeatedly in a clause, order is important)

Program/clause set = sequence of clauses

Def 4.1.1. (Syntax of Logic Programs)

A non-empty finite set \( \mathcal{S} \) of definite Horn clauses over a signature \( (\Sigma, \Delta) \) is called a logic program over \( (\Sigma, \Delta) \). The clauses in \( \mathcal{S} \) are called program clauses and we distinguish the following forms of clauses:

- Facts: clauses of the form \( \{ B \} \) where \( B \) is an atomic formula
• rules: clauses of the form \( \{ B, \neg C_1, \ldots, \neg C_n \} \) with \( n \geq 1 \) for atomic formulas \( B, C_1, \ldots, C_n \).

A logic program is executed by submitting a query \( G \) of the form \( \{ \neg A_1, \ldots, \neg A_k \} \) with \( k \geq 1 \) where \( A_1, \ldots, A_k \) are atomic formulas.

As usual: clause stands for universally quantified disjunction of its literals.

Calling a LP \( \mathcal{P} \) with query \( G = \{ \neg A_1, \ldots, \neg A_k \} \) means that one wants to prove:

\[
\mathcal{P} \vdash \exists X_1, \ldots, X_p. A_1 \land \ldots \land A_k
\]

variables in \( A_1, \ldots, A_k \)

This is equivalent to unsatisfiability of

\[
\mathcal{P} \cup \{ G \}, \text{ i.e., to the unsatisfiability of }
\]

\[
\mathcal{P} \cup \{ \forall X_1, \ldots, X_p. \neg A_1 \lor \ldots \lor \neg A_k \}
\]

By Thm 33.1(a) (Herbrand—Expansion) and compactness of propositional resolution: Equivalent to there is a finite set of ground instantiations of \( \mathcal{P} \cup \{ \forall X_1, \ldots, X_p. \neg A_1 \lor \ldots \lor \neg A_k \} \) that is unsatisfiable.
By completeness of SLD-resolution:

There are ground terms $t_1, ..., t_p$ such that

$$P \cup \{ (\neg A_1 \lor \ldots \lor \neg A_k) [X_n/t_n, \ldots , X_p/t_p] \}$$

is unsatisfiable.

**Goal:** Find those instantiations $t_1, ..., t_p$ where

$$P \cup \{ (\neg A_n \lor \ldots \lor \neg A_k) [X_n/t_n, \ldots , X_p/t_p] \}$$

is unsatisfiable

resp.

where

$$P \subseteq A_n \land \ldots \land A_k [X_n/t_n, \ldots , X_p/t_p]$$

(i.e., we also want to know the answer substitutions)

Answer substitutions are constructed during the SLD-resolution proof.

**Ex 4.12** Consider the LP:

motherOf (rente, susanne).
moved (gerd, rente).

fatherOf (F, C) :- married (F, W), motherOf (W, C).

?- fatherOf (gerd, Y).

**Goal:** for which instantiations $t$ is

$$P \cup \{ \neg fatherOf (gerd, Y) [Y/t] \}$$

unsatisfiable?

To find this out: SLD-resolution on $P \cup \{ G \}$.

Answer substitution: compose all used mgu's and restrict them to the variables occurring in the
Initial query.
Here: \{ Y/ susanne \}.

We have defined the syntax of LP.
Now: define the semantics of LP.
3 different (but equivalent) possibilities:
4.1.1. declarative semantics
4.1.2. procedural (or operational) semantics
4.1.3. fixpoint (or denotational) semantics

4.1.1. Declarative Semantics of Logic Prog.

Idea: use the semantics of predicate logic.
All ground instances of a query \( G \) are "true" in a logic prog. \( S \) where \( S \) entails the instance in \( G \).

\[ \text{entailment } \models \text{ in pred. logic, defined via interpretations} \]

**Def 4.13** (Declarative Semantics of a LP)

Let \( S \) be a LP and \( G = \{ \neg A_1, \ldots, \neg A_n \} \) be a query.
Then the declarative semantics of \( S \) w.r.t. \( G \) is defined as:

**D**\( S, G \) \( \models \) \( \{ \sigma(A_1, \ldots, nA_n) \mid S \models \sigma(A_1, \ldots, nA_n), \sigma \text{ is a ground substitution} \} \)

**Ex. 4.14**
\[ DII \mathcal{S}, GII = \{ \text{fatherOf}(gard, susane) \} \]

If \( \mathcal{S} \) also contained the fact \( \text{motherOf}(renate, petr) \), then
\[ DII \mathcal{S}, GII = \{ \text{fatherOf}(gard, susane), \text{fatherOf}(gard, petr) \} \]

4.1.2. Procedural Semantics of LP

Idea: provide an example interpreter which does the "right" thing. In this way, one can define the meanings of programs.

Solution: perform SLD-resolution and collect the used mgu's to obtain the answer subs. in the end.

- operate on configurations (pairs of negative clause and substitution)

- start with \((G, \theta)\)
  \(\theta\) - empty/identical substitution

  goal is to reach \((\square, \emptyset)\).

  Then the restriction of \(\emptyset\) to the variables in \( G \) is the answer substitution.

- Computation: sequence of configurations where the step from one config. to the next is done by SLD-resolution.

- 3 modifications of SLD-resolution:
  - standardized SLD-resolution: only rename variables in prog. clauses, not in negative clauses
  - binary resolution: only resolve one literal in each clause in each resolution step
- clauses are regarded as sequences of literals (instead of sets). Thus: a literal can occur multiple times in a clause.

**Def 4.15 (Procedural Semantics of LP)**

Let $\mathcal{P}$ be a LP.

- A configuration is a pair $(G, \sigma)$ where $G$ is a negative Horn clause (possibly $\bot$) and $\sigma$ is a substitution.
- We have a computation step $(G_1, \sigma_1) \xrightarrow{\mathcal{P}} (G_2, \sigma_2)$ if
  - $G_1 = \{ \neg A_1, \ldots, \neg A_k \}$ with $k \geq 1$
  - there is a program clause $K \in \mathcal{P}$ and a variable renaming $\nu$ with $\nu(K) = \{ B, \neg C_1, \ldots, \neg C_n \}$ and $n \geq 0$ such that
    - $\nu(K)$ has no common variables with $G_1$ or $\text{RANGE}(G_1)^+$
    - $\{ \nu(X) \mid X \in \text{DOM}(G_1) \}$
  - there is an $1 \leq i \leq k$ such that $A_i$ and $B$ are unifiable with a mgu $\rho$
  - $G_2 = \nu \left( \{ \neg A_1, \ldots, \neg A_{i-1}, \neg C_1, \ldots, \neg C_n, \neg A_i, \ldots, \neg A_k \} \right)$
  - $\sigma_2 = \sigma \circ \nu^{-1}$
- A computation of $\mathcal{P}$ with the query $G$ is a (finite or infinite) sequence of configurations:
  - $(G, \emptyset) \xrightarrow{\mathcal{P}} (G_1, \sigma_1) \xrightarrow{\mathcal{P}} (G_2, \sigma_2) \xrightarrow{\mathcal{P}} \ldots$
- A computation $(G, \emptyset) \xrightarrow{\mathcal{P}} \ldots \xrightarrow{\mathcal{P}} (\bot, \sigma)$ is called successful. If $G = \{ \neg A_1, \ldots, \neg A_k \}$, then the result of the computation is $\nu \left( A_1 \land \ldots \land A_k \right)$.  

LP 2015 Seite 6
The answer substitution is $\delta$, restricted to the variables in $G$.

Now we can define the procedural semantics of $S$ w.r.t. $G = \{\neg A_1, \ldots, \neg A_n\}$:

$$\text{PI}$, $G$ = $\{ \delta'(A_1 \land \ldots \land A_n) \mid (G, \emptyset) \vdash^+ (\Box, \sigma) \}$

"$\vdash^+$" means transitive closure, i.e.

$$(G, \emptyset) \vdash_0 \ldots \vdash_0 (\Box, \sigma)
\delta'(A_1 \land \ldots \land A_n) \text{ is a ground instance of } \delta(\Box, \sigma)$$

Ex. 4.16 $S$, $G$ as in Ex. 4.12

$$(\{\neg \text{fatherOf}(\text{gerd}, y)\}, \emptyset)$$
$$\vdash (\{\neg \text{married}(\text{gerd}, w), \neg \text{motherOf}(w, c)\}, \{y/c, F/\text{gerd}3\})$$
$$\vdash (\{\neg \text{motherOf}(\text{renate}, c\}), \{w/\text{renate}, y/c, F/\text{gerd}3\})$$
$$\vdash (\Box, \{c/\text{susanne}, w/\text{renate}, y/\text{susanne}, F/\text{gerd}3\})$$

Answer Subst: $\{ y/\text{susanne} \}$

Proc. Semantics has 2 indeterminisms:

1. choice of prog. clause $K$ for the next resolution step
2. choice of literal $A_i$ in the current goal for the next res. step.

Choices can influence success, length, result of computation:

Ex. 4.17 $S = \{\{p(x, z), \neg p(x, y), \neg p(y, z)\}$,

$\{p(u, v)\}$,

$\{q(a, b)\}$.
Query \( G = \{ \neg p(V, b) \} \)
\( (\{ \neg p(V, b) \}, \emptyset) \)
\( \vdash p (\{ \neg q(V, y), \neg p(y, b) \}), \{ x / V, z / b \} ) \)  \hspace{1cm} \text{Res. with first prog. cl.}
\( \vdash p (\{ \neg q(b, y) \}, \{ V / a, y / b, x / a, z / b \}) \)  \hspace{1cm} \text{Res. with first res. cl.}
\( \vdash p (\{ \neg q(b, y) \}, \{ x / b, z / b, V / a, y / b, x / a, z / b \}) \)
\( \vdash p (\{ \neg q(b, y) \}, \{ u / b, y / b, \ldots \}) \)

Finite failing computation (doesn’t end in \( \Box \)).

If after the first 2 computation steps one would have used the 2nd prog. clause, one would have reached
\( (\Box, \{ u / b, V / a, \ldots \}) \)

Answer Subst: \( \{ V / a \} \). \( \neg p(a, b) \in \Pi II \bar{3}, 6 \bar{II} \).

Moreover, one could have used the 2nd prog. clause in the first res step:
\( (\{ \neg q(V, b) \}, \emptyset) \)
\( \vdash p (\Box, \{ u / b, V / b \}) \).

Answer Subst: \( \{ V / b \} \). \( \neg p(b, b) \in \Pi II \bar{3}, 6 \bar{II} \).

Theorem 4.18 (Equivalence of declarative and procedural semantics)

Let \( B \) be a LP and \( G \) be a query.

Then \( DES, G \bar{II} = \Pi II \bar{3}, 6 \bar{II} \).

Proof: Based on soundness & completeness of
SLD-resolution. Moreover, one has to keep track of the substitutions.

4.1.3. Fixpoint Semantics of LP

Idea: only regard the program \( P \) in each step, extend the facts of \( P \) by those statements that can be inferred by one more application of a rule from \( P \).

Formally: use a function \( \text{trans}_P(\cdot) \). It returns \( M \) extended by those ground atomic formulas that can be deduced from \( M \) by one application of a rule from \( P \).

Then: Set of all true statements about \( P \):

\[
\emptyset \cup \text{trans}_P(\emptyset) \cup \text{trans}_P(\text{trans}_P(\emptyset)) \cup \text{trans}_P^3(\emptyset) \cup \ldots
\]

\[
\text{trans}_P^\infty(\emptyset)
\]

Definition (Transformation \( \text{trans}_P \))

Let \( P \) be a LP over a signature \((\Sigma, \Delta)\). Then \( \text{trans}_P \) is a function \( \text{trans}_P : \text{Pot}(\text{At}(\Sigma, \Delta, \Theta)) \rightarrow \text{Pot}(\text{At}(\Sigma, \Delta, \Theta)) \) with

\[
\text{trans}_P(M) = M \cup \{ A^1 \mid \neg B_1', \ldots, \neg B_n' \} \text{ is a ground instance}
\]
of a clause \( \{ A, \neg B_1, \ldots, \neg B_n \} \in \mathcal{S} \)
and \( B_1', \ldots, B_n' \in M^3 \)

Ex 4.1.10

\[
\text{trans}^0_{\mathcal{S}}(\varnothing) = \varnothing
\]

\[
\text{trans}^1_{\mathcal{S}}(\varnothing) = \{ \text{motherOf}(\text{reu}, \text{sus}), \text{married}(\text{grod}, \text{reu}) \}
\]

\[
\text{trans}^2_{\mathcal{S}}(\varnothing) = \{ \text{fatherOf}(\text{grod}, \text{remote}) \}
\]

\[
\text{trans}^3_{\mathcal{S}}(\varnothing) = \text{trans}^2_{\mathcal{S}}(\varnothing)
\]

Ex 4.1.11 In general, the iteration of applying \( \text{trans} \) repeatedly can go on infinitely long.

\[
p(a).
\]

\[
p(f(X)) := p(X).
\]

\[
\text{trans}^0_{\mathcal{S}}(\varnothing) = \{ p(a) \}
\]

\[
\text{trans}^1_{\mathcal{S}}(\varnothing) = \{ p(a), p(f(a)) \}
\]

\[
\text{trans}^2_{\mathcal{S}}(\varnothing) = \{ p(a), p(f(a)), p(f(f(a))) \}
\]

\[
\vdots
\]

\[
\bigcup_{i \in \mathbb{N}} \text{trans}^i_{\mathcal{S}}(\varnothing) = \{ p(f^i(a)) \mid i \in \mathbb{N} \}
\]

We call this set \( M_{\mathcal{S}} \).
We use $M_\emptyset = \bigcup_{i \in \mathbb{N}} \text{trans}_p^i (\emptyset)$ to define the semantics of $p$.

- $M_\emptyset$ is a fixpoint of $\text{trans}_p$ : $\text{trans}_p (M_\emptyset) = M_\emptyset$
  This means: $M_\emptyset$ already contains all true statements about $p$.

- $M_\emptyset$ is the least fixpoint of $\text{trans}_p$ : for all other fixpoints $M$ of $\text{trans}_p$, we have $M_\emptyset \subseteq M$
  This means: $M_\emptyset$ only contains those statements that are enforced by $p$ (i.e., that are really true in $\mathcal{P}$).

Note: Prove formally that $M_\emptyset = \bigcup_{i \in \mathbb{N}} \text{trans}_p^i (\emptyset)$ is the least fixpoint of $\text{trans}_p$. (A similar construction can be used to define the semantics of other prog. languages.)

A. Properties of $\subseteq$

- reflexive $M_\emptyset \subseteq M_\emptyset$
- transitive $M_\emptyset \subseteq M_1$ and $M_1 \subseteq M_2$ implies $M_\emptyset \subseteq M_2$
- antisymmetric $M_\emptyset \subseteq M_1$ and $M_1 \subseteq M_2$ implies $M_\emptyset = M_2$
"ordering"

Moreover, \( \leq \) is a complete reflexive ordering.

- \( \leq \) must have a smallest element: \( \emptyset \)
- every chain has a least upper bound, i.e.:
  Whenever there are sets \( M_0, M_1, \ldots \) with
  \( M_0 \subseteq M_1 \subseteq M_2 \subseteq \ldots \) (a so-called chain)
  then there exists a least upper bound (lub) \( M \).

This means: \( M_i \subseteq M \) for all \( i \in \mathbb{N} \)
and for all other upper bounds \( M' \), we have
\( M \subseteq M' \).

Solution: lub of \( M_0, M_1, \ldots \) is
\[ \bigcup_{i \in \mathbb{N}} M_i \,.
\]

Reason: \( \bigcup_{i \in \mathbb{N}} M_i \) is an upper bound of \( M_0, M_1, \ldots \)
because \( M_i \subseteq \bigcup_{i \in \mathbb{N}} M_i \).

It is the lub: If there were another upper bound \( M' \) of \( M_0, M_1, \ldots \),
then \( M_0 \leq M', M_1 \leq M', \ldots \)
\[ \bigcup_{i \in \mathbb{N}} M_i \leq M' \,.
\]

Lemma 4.1.12 The subterm relation \( \leq \) on
\[ \text{Prt}(\text{At}(\Sigma, \Delta, \Theta)) \] is a complete reflexive order.
Proof: see above

B. Properties of trans_p

trans_p has 2 important properties:

- trans_p is monotonic: \( M_1 \leq M_2 \) implies \( \text{trans}_p(M_1) \leq \text{trans}_p(M_2) \)

- trans_p is continuous (sketch):

\[
\begin{align*}
M_0 & \leq M_1 \leq \ldots \downarrow \downarrow \\
\text{trans}_p(M_0) & \leq \text{trans}_p(M_1) \leq \ldots \Downarrow \\
\text{trans}_p(M) &
\end{align*}
\]

Continuity means: the black and the green step yield the same solution.

Lemma 4.1.13 (Monotonicity and Continuity of trans_p)

(a) trans_p is monotonic, i.e., if \( M_1 \leq M_2 \) then \( \text{trans}_p(M_1) \leq \text{trans}_p(M_2) \).

(b) trans_p is continuous, i.e.,

for every chain \( M_0 \leq M_n \leq M_2 \leq \ldots \)

we have \( \text{trans}_p(\bigcup_{i \in \mathbb{N}} M_i) = \bigcup_{i \in \mathbb{N}} \text{trans}_p(M_i) \).

Proof: (a) follows immediately from the definition of trans_p. We now show (b).
First, show \( \forall i \in \mathbb{N}, \text{trans}_p(U, M_i) \supseteq U \text{trans}_p(M_i) \).

This follows from monotonicity of \( \text{trans}_p \):

\[
\forall i \in \mathbb{N}, UM_i \supseteq M_i
\]

\[
\forall i \in \mathbb{N}, \text{trans}_p(U, M_i) \supseteq \text{trans}_p(M_i)
\]

\[
\forall i \in \mathbb{N}, \text{trans}_p(U, M_i) \supseteq U \text{trans}_p(M_i)
\]

Now we show \( \forall i \in \mathbb{N}, \text{trans}_p(U, M_i) \subseteq U \text{trans}_p(M_i) \).

Let \( A' \in \text{trans}_p(U, M_i) \). Then \( \{ A', \neg B_1', \ldots, \neg B_n' \} \) is a ground instance of a clause \( \{ A, \neg B_1, \ldots, \neg B_n \} \in \mathcal{P} \) and

\[
B_1', \ldots, B_n' \in UM_i.
\]

Since \( M_0 \subseteq M_i \subseteq \ldots \), there exists a \( j \in \mathbb{N} \) such that

\[
B_1', \ldots, B_n' \in M_j.
\]

\[
\forall i \in \mathbb{N}, A' \in \text{trans}_p(M_j) \subseteq U \text{trans}_p(M_j).
\]

Now we can show that \( M_0 \) is indeed the least fixpoint of \( \text{trans}_p \). (This theorem holds in general: every continuous function \( f \) over a complete ordering has a least fixpoint, which is the lub of the chain \( \emptyset, f(\emptyset), f^2(\emptyset), \ldots \). Here, \( \emptyset \) is the smallest element of the ordering.)
Theorem 4.1.14 (Fixpoint Theorem, Kleene-Tarski)

For every LP $p$, the function $\text{trans}_p$ has a least fixpoint $\text{lfp}(\text{trans}_p)$. Here:

\[
\text{lfp}(\text{trans}_p) = \bigcup_{i \in \mathbb{N}} \text{trans}_p^i(\emptyset).
\]

Proof: 1. $\bigcup_{i \in \mathbb{N}} \text{trans}_p^i(\emptyset)$ is a fixpoint of $\text{trans}_p$.

\[
\begin{align*}
\text{trans}_p \left( \bigcup_{i \in \mathbb{N}} \text{trans}_p^i(\emptyset) \right) \\
= \bigcup_{i \in \mathbb{N}} \text{trans}_p^{i+1}(\emptyset) & \quad \text{(since trans}_p \text{ is continuous)} \\
= \emptyset \cup \bigcup_{i \in \mathbb{N}} \text{trans}_p^{i+1}(\emptyset) \\
= \bigcup_{i \in \mathbb{N}} \text{trans}_p^i(\emptyset).
\end{align*}
\]

2. $\bigcup_{i \in \mathbb{N}} \text{trans}_p^i(\emptyset)$ is smaller or equal to any other fixpoint $M$ of $\text{trans}_p$.

Let $M$ be another fixpoint of $\text{trans}_p$.

We want to show: $\bigcup_{i \in \mathbb{N}} \text{trans}_p^i(\emptyset) \subseteq M$.

It suffices to show: $\text{trans}_p^i(\emptyset) \subseteq M$ for all $i \in \mathbb{N}$.

Prove this by induction on $i$.

**Ind Base**: $i = 0$
\[ \text{trans}_p(\emptyset) = \emptyset \subseteq M \]

Ind Step: \( i > 0 \)

Ind Hypothesis: \( \text{trans}_p^{i-1}(\emptyset) \subseteq M \)

By monotonicity of \( \text{trans}_p \):
\[ \text{trans}_p^i(\emptyset) \subseteq \text{trans}_p(M) = M \]

because \( M \) is a fixpoint of \( \text{trans}_p \).

Finally, we can define the fixpoint semantics of LP:

**Def 4.1.15 (Fixpoint Semantics of LP)**

Let \( S \) be a LP, let \( G = \{
\neg A_1, \ldots, \neg A_n, T\} \) be a query.

Then the fixpoint semantics of \( S \) w.r.t. \( G \) is defined as:

\[ F[S, G] = \{ \sigma(A_1, \ldots, A_n) \mid \sigma(A_i) \in \text{lfp}(\text{trans}_p) \text{ for all } 1 \leq i \leq n \} \]

**Thm 4.1.16 (Equivalence of all 3 semantics definitions)**

Let \( S \) be a LP, \( G \) be a query.

Then \( \text{DIL} S, G \equiv \text{PIL} S, G \equiv F[S, G] \).

Proof: see course notes.