4. Logic Programs

4.1. Syntax and Semantics of Logic Programs

Horn clauses $\hat{=}$ 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 = <u>sequence</u> of literals (literals can also occur repeatedly in a clause, order is important)

program/clause set = <u>sequence</u> of clauses

<u>Def 4.1.1</u> (Syntax of Logic Programs)

A non-empty finite set $\mathcal{P}$ of definite Horn clauses over a signature (Σ, Δ) is called a <u>logic program</u> over (Σ, Δ). The clauses in $\mathcal{P}$ are called <u>program clauses</u> and we distinguish the following forms of clauses:

- <u>facts</u>: clauses of the form $\{B\}$ where B is an atomic formula

• <u>rules</u>: clauses of the form $\{B, \neg C_1, \dots, \neg C_n\}$ with $n \geq 1$ for atomic formulas $B, C_1, \dots, C_n$.

A logic program is executed by submitting a

• <u>query</u> G of the form $\{\neg A_1, \dots, \neg A_k\}$ with $k \geq 1$ where $A_1, \dots, A_k$ are atomic formulas.

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

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

$$\mathcal{P} \models \exists \underbrace{X_1, \dots, X_p}_{\text{variables in } A_1, \dots, A_k} . A_1 \wedge \dots \wedge A_k$$

This is equivalent to unsatisfiability of

$\mathcal{P} \cup \{G\}$, i.e., to the unsatisfiability of

$$\mathcal{P} \cup \{\forall X_1, \dots, X_p \; \neg A_1 \vee \dots \vee \neg A_k\}$$

By Thm 3.39 (a) (Herbrand-Expansion) and compactness of prop. resolution: Equivalent to

There is a finite set of ground instantiations of $\mathcal{P} \cup \{\forall X_1, \dots, X_p \; \neg A_1 \vee \dots \vee \neg A_k\}$ that is unsatisfiable.

By completeness of SLD-resolution:

There are ground terms $t_1, \dots, t_p$ such that
$$\mathcal{P} \cup \{ (\neg A_1 \vee \dots \vee \neg A_k)[X_1/t_1, \dots, X_p/t_p] \}$$
is unsatisfiable.

Goal: Find those instantiations $t_1, \dots, t_p$ where
$$\mathcal{P} \cup \{ (\neg A_1 \vee \dots \vee \neg A_k)[X_1/t_1, \dots, X_p/t_p] \}$$ is unsatisfiable

resp.

where $\mathcal{P} \models A_1 \wedge \dots \wedge A_k \, [X_1/t_1, \dots, X_p/t_p]$

(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(renate, susanne).

married(gerd, renate).

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

?- fatherOf(gerd, Y).

Goal: for which instantiations t is
$$\mathcal{P} \cup \{ \neg \text{fatherOf}(\text{gerd}, Y) [Y/t] \} \quad \text{unsatisfiable?}$$

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

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. $\mathcal{P}$ where $\mathcal{P}$ entails the instance in G

entailment $\models$ in pred. logic, defined via interpretations

Def 4.1.3 (Declarative Semantics of a LP)

Let $\mathcal{P}$ be a LP and $G = \{\neg A_1, .., \neg A_k\}$ be a query. Then the declarative semantics of $\mathcal{P}$ w.r.t. G is defined as:

$$D[\![\mathcal{P}, G]\!] = \{\sigma(A_1 \wedge \ldots \wedge A_k) \mid \mathcal{P} \models \sigma(A_1 \wedge \ldots \wedge A_k),$$
$$\sigma \text{ is a ground substitution}\}$$

Ex. 4.1.4

$D[\![\mathcal{P}, G]\!] = \{ fatherOf(gerd, susanne) \}$

If $\mathcal{P}$ also contained the fact motherOf(renate, peter), then

$D[\![\mathcal{P}, G]\!] = \{ fatherOf(gerd, susanne), fatherOf(gerd, peter) \}$.

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 subst. in the end.

- operate on configurations (pairs of negative clause and substitution)
- start with $(G, \varnothing)$ ← empty/identical substitution

goal is to reach $(\Box, \sigma)$.
Then the restriction of σ 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, σ) where G is a negative Horn clause (possibly $\Box$) and σ is a substitution.
- We have a computation step $(G_1, \sigma_1) \vdash_{\mathcal{P}} (G_2, \sigma_2)$ iff
  - $G_1 = \{\neg A_1, \dots, \neg A_k\}$ with $k \geq 1$
  - there is a program clause $K \in \mathcal{P}$ and a variable renaming ν with $\nu(K) = \{B, \neg C_1, \dots, \neg C_n\}$ and $n \geq 0$ such that
    - $\nu(K)$ has no common variables with G_1 or $RANGE(\sigma_1)$

      $\{\sigma_1(X) \mid X \in DOM(\sigma_1)\}$
    - there is an $1 \leq i \leq k$ such that A_i and B are unifiable with a mgu σ
  - $G_2 = \sigma(\{\neg A_1, \dots, \neg A_{i-1}, \neg C_1, \dots, \neg C_n, \neg A_{i+1}, \dots, \neg A_k\})$
  - $\sigma_2 = \sigma \circ \sigma_1$
- A computation of $\mathcal{P}$ with the query G is a (finite or infinite) sequence of configurations:

  $(G, \varnothing) \vdash_{\mathcal{P}} (G_1, \sigma_1) \vdash_{\mathcal{P}} (G_2, \sigma_2) \vdash_{\mathcal{P}} \dots$
- A computation $(G, \varnothing) \vdash_{\mathcal{P}} \dots \vdash_{\mathcal{P}} (\Box, \sigma)$ is called successful. If $G = \{\neg A_1, \dots, \neg A_k\}$, then the result of the computation is $\sigma(A_1 \wedge \dots \wedge A_k)$.

The <u>answer substitution</u> is σ, restricted to the variables in G.

Now we can define the <u>procedural semantics</u> of $\mathcal{P}$ w.r.t. $G = \{\neg A_1, \ldots, \neg A_k\}$:

$$P[\![\mathcal{P}, G]\!] = \{ \sigma'(A_1 \wedge \ldots \wedge A_k) \mid (G, \emptyset) \vdash_{\mathcal{P}}^{+} (\Box, \sigma),\ \sigma'(A_1 \wedge \ldots \wedge A_k) \text{ is a ground instance of } \sigma(A_1 \wedge \ldots \wedge A_k) \}$$

"+" means transitive closure, i.e. $(G, \emptyset) \vdash_{\mathcal{P}} \ldots \vdash_{\mathcal{P}} (\Box, \sigma)$

<u>Ex. 4.16</u> $\mathcal{P}$, G as in Ex. 4.12

$(\{\neg fatherOf(gerd, Y)\}, \emptyset)$

$\vdash_{\mathcal{P}} (\{\neg married(gerd, W), \neg motherOf(W, C)\}, \{Y/C, F/gerd\})$

$\vdash_{\mathcal{P}} (\{\neg motherOf(renate, C)\}, \{W/renate, Y/C, F/gerd\})$

$\vdash_{\mathcal{P}} (\Box, \{C/susanne, W/renate, Y/susanne, F/gerd\})$

Answer Subst: $\{Y/susanne\}$

Proc. Semantics has 2 indeterminisms:

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

Choices can influence success, length, result of computation:

<u>Ex 4.17</u> $\mathcal{P} = \{ \{p(X,Z), \neg q(X,Y), \neg p(Y,Z)\},$
$\{p(U,U)\},$
$\{q(a,b)\} \}$

Query $G = \{\neg p(V, b)\}$

$(\{\neg \underline{p(V,b)}\}, \emptyset)$

$\vdash_{\mathcal{P}} (\{\neg \underline{q(V,Y)}, \neg p(Y,b)\}, \{X/V, Z/b\})$ Res. with first prog. cl.

$\vdash_{\mathcal{P}} (\{\neg \underline{p(b,b)}\}, \{V/a, Y/b, X/a, Z/b\})$ – Res. with first pr. cl.

$\vdash_{\mathcal{P}} (\{\neg q(b,Y'), \neg \underline{p(Y',b)}\}, \{X'/b, Z'/b, V/a, Y/b, X/a, Z/b\})$

$\vdash_{\mathcal{P}} (\{\neg q(b,b)\}, \{U/b, Y'/b, \dots\})$

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

$(\Box, \{U/b, V/a, \dots\})$

Answer Subst: $\{V/a\}$. $\rightsquigarrow$ $p(a,b) \in \mathcal{P}[\![\mathcal{P}, G]\!]$.

Moreover, one could have used the 2nd prog. clause in the first res step:

$(\{\neg p(V,b)\}, \emptyset)$

$\vdash_{\mathcal{P}} (\Box, \{U/b, V/b\})$.

Answer subst: $\{V/b\}$ $\rightsquigarrow$ $p(b,b) \in \mathcal{P}[\![\mathcal{P}, G]\!]$.

<u>Thm 4.18</u> (Equivalence of declarative and procedural semantics)

Let $\mathcal{P}$ be a LP and G be a query.

Then $\mathcal{D}[\![\mathcal{P}, G]\!] = \mathcal{P}[\![\mathcal{P}, G]\!]$.

<u>Proof</u>: 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 $\mathcal{P}$

• in each step, extend the facts of $\mathcal{P}$ by those statements that can be inferred by one more application of a rule from $\mathcal{P}$.

Formally: use a function $trans_{\mathcal{P}}(M)$. M: set of atomic ground formulas. $p(t_1, \dots, t_n)$

It returns M extended by those ground atomic formulas that can be deduced from M by one application of a rule from $\mathcal{P}$.

Then: Set of all true statements about $\mathcal{P}$:

$$\emptyset \cup trans_{\mathcal{P}}(\emptyset) \cup \underbrace{trans_{\mathcal{P}}(trans_{\mathcal{P}}(\emptyset))}_{trans^2_{\mathcal{P}}(\emptyset)} \cup trans^3_{\mathcal{P}}(\emptyset) \cup \dots$$

<u>Def 4.1.9.</u> (Transformation $trans_{\mathcal{P}}$)

Let $\mathcal{P}$ be a LP over a signature (Σ, Δ).

Then $trans_{\mathcal{P}}$ is a function $trans_{\mathcal{P}}: \underbrace{Pot(At(\Sigma, \Delta, \emptyset))}_{\text{Set containing all subsets of } At(\Sigma,\Delta,\emptyset)} \to Pot(At(\Sigma, \Delta, \emptyset))$

with

$trans_{\mathcal{P}}(M) = M \cup \{A' \mid \{A', \neg B_1', \dots, \neg B_n'\}$ is a ground instance

of a clause $\{A, \neg B_1, \ldots, \neg B_n\} \in \mathcal{P}$
and $B_1', \ldots, B_n' \in M\}$

Ex 4.1.10

$trans_{\mathcal{P}}^{0}(\emptyset) = \emptyset$

$trans_{\mathcal{P}}^{1}(\emptyset) = \{motherOf(ren, sus), married(gerd, ren)\}$

$trans_{\mathcal{P}}^{2}(\emptyset) = \{$ —— " ———————,

fatherOf(gerd, renate)$\}$

$trans_{\mathcal{P}}^{3}(\emptyset) = trans_{\mathcal{P}}^{2}(\emptyset)$

Ex 4.1.11 In general, the iteration of applying $trans_{\mathcal{P}}$ repeatedly can go on infinitely long.

p(a).

p(f(X)) :- p(X).

$trans_{\mathcal{P}}(\emptyset) = \{p(a)\}$

$trans_{\mathcal{P}}^{2}(\emptyset) = \{p(a), p(f(a))\}$

$trans_{\mathcal{P}}^{3}(\emptyset) = \{p(a), p(f(a)), p(f(f(a)))\}$

$\vdots$

$$\underbrace{\bigcup_{i \in \mathbb{N}} trans_{\mathcal{P}}^{i}(\emptyset)} = \{p(f^{i}(a)) \mid i \in \mathbb{N}\}$$

We call this set $M_{\mathcal{P}}$.

We use $M_P = \bigcup_{i \in \mathbb{N}} trans_P^i(\emptyset)$ to define the semantics of P.

- M_P is a <u>fixpoint</u> of $trans_P$: $trans_P(M_P) = M_P$

  This means: M_P already contains all true statements about P.

- M_P is the <u>least</u> fixpoint of $trans_P$: for all other fixpoints M of $trans_P$, we have $M_P \subseteq M$

  ↑ smallest w.r.t. $\subseteq$

  This means: M_P only contains those statements that are enforced by P (i.e., that are really true in P).

Now: Prove formally that

$M_P = \bigcup_{i \in \mathbb{N}} trans_P^i(\emptyset)$

is the least fixpoint of $trans_P$. (A similar construction can be used to define the semantics of other prog. languages.)

A. Properties of $\subseteq$

- reflexive $\quad M_1 \subseteq M_1$
- transitive $\quad M_1 \subseteq M_2$ and $M_2 \subseteq M_3$ implies $M_1 \subseteq M_3$
- antisymmetric $\quad M_1 \subseteq M_2$ and $M_2 \subseteq M_1$ implies $M_1 = M_2$

— "ordering"

Moreover, $\subseteq$ is a <u>complete</u> reflexive ordering.

- $\subseteq$ 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 <u>chain</u>)

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

$$\bigcup_{i \in \mathbb{N}} M_i .$$

<u>Reason</u>: $\bigcup_{i \in \mathbb{N}} M_i$ is an <u>upper bound</u> 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 $M_0 \subseteq M'$, $M_1 \subseteq M'$, ...

$\rightsquigarrow \bigcup_{i \in \mathbb{N}} M_i \subseteq M'$.

<u>Lemma 4.1.12</u> The subterm relation $\subseteq$ on $Pot(At(\Sigma, \Delta, \emptyset))$ is a complete reflexive order.

<u>Proof</u>: See above

B. Properties of $trans_\rho$

$trans_\rho$ has 2 important properties:

- $trans_\rho$ is <u>monotonic</u>: $M_1 \subseteq M_2$ implies
$$trans_\rho(M_1) \subseteq trans_\rho(M_2)$$
- $trans_\rho$ is <u>Continuous</u> (stetig):

$$\begin{tikzcd}
M_0 \arrow[d] & \subseteq & M_1 \arrow[d] & \subseteq & \dots & \arrow[r, "lub"] & M \arrow[d] \\
trans_\rho(M_0) & \subseteq & trans_\rho(M_1) & \subseteq & \dots & \arrow[r, "lub"] & trans_\rho(M)
\end{tikzcd}$$

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

<u>Lemma 4.1.13</u> (Monotonicity and Continuity of $trans_\rho$)

(a) $trans_\rho$ is monotonic, i.e., if $M_1 \subseteq M_2$ then $trans_\rho(M_1) \subseteq trans_\rho(M_2)$.

(b) $trans_\rho$ is continuous, i.e.,

for every chain $M_0 \subseteq M_1 \subseteq M_2 \subseteq \dots$

we have $trans_\rho\left(\bigcup_{i \in \mathbb{N}} M_i\right) = \bigcup_{i \in \mathbb{N}} trans_\rho(M_i)$.

<u>Proof</u>: (a) follows immediately from the definition of $trans_\rho$.
We now show (b).

First, show $trans_\rho(\bigcup_{i\in\mathbb{N}} M_i) \supseteq \bigcup_{i\in\mathbb{N}} trans_\rho(M_i)$.

This follows from monotonicity of $trans_\rho$:

$$\bigcup_{i\in\mathbb{N}} M_i \supseteq M_i$$

$$\leadsto trans_\rho(\bigcup_{i\in\mathbb{N}} M_i) \supseteq trans_\rho(M_i) \text{ for all } i\in\mathbb{N}$$

$$\leadsto trans_\rho(\bigcup_{i\in\mathbb{N}} M_i) \supseteq \bigcup_{i\in\mathbb{N}} trans_\rho(M_i)$$

Now we show $trans_\rho(\bigcup_{i\in\mathbb{N}} M_i) \subseteq \bigcup_{i\in\mathbb{N}} trans_\rho(M_i)$.

Let $A' \in trans_\rho(\bigcup_{i\in\mathbb{N}} M_i)$.

Then $\{A', \neg B_1', \dots, \neg B_n'\}$ is a ground instance of a clause $\{A, \neg B_1, \dots, \neg B_n\} \in \rho$ and $B_1', \dots, B_n' \in \bigcup_{i\in\mathbb{N}} M_i$.

Since $M_0 \subseteq M_1 \subseteq \dots$, there exists a $j\in\mathbb{N}$ such that $B_1', \dots, B_n' \in M_j$.

$$\leadsto A' \in trans_\rho(M_j) \subseteq \bigcup_{i\in\mathbb{N}} trans_\rho(M_i). \quad \boxed{\text{}}$$

Now we can show that M_ρ is indeed the least fixpoint of $trans_\rho$. (This theorem holds in general: every continuous function f over a complete ordering has a least fixpoint, which is the lub of the chain $\emptyset, f(\emptyset), f^2(\emptyset), \dots$. Here, $\emptyset$ is the smallest element of the ordering.)

Thm 4.1.14 (Fixpoint Theorem, Kleene+Tarski)

For every LP $\mathcal{P}$, the function $trans_{\mathcal{P}}$ has a least fixpoint $lfp(trans_{\mathcal{P}})$. Here:

$$lfp(trans_{\mathcal{P}}) = \bigcup_{i \in \mathbb{N}} trans_{\mathcal{P}}^{i}(\emptyset).$$

Proof: 1. $\bigcup_{i \in \mathbb{N}} trans_{\mathcal{P}}^{i}(\emptyset)$ is a fixpoint of $trans_{\mathcal{P}}$.

$$trans_{\mathcal{P}}\Big(\underline{\bigcup_{i \in \mathbb{N}} trans_{\mathcal{P}}^{i}(\emptyset)}\Big)$$

$$= \bigcup_{i \in \mathbb{N}} trans_{\mathcal{P}}^{i+1}(\emptyset) \qquad \text{(since } trans_{\mathcal{P}} \text{ is continuous)}$$

$$= \emptyset \cup \bigcup_{i \in \mathbb{N}} trans_{\mathcal{P}}^{i+1}(\emptyset)$$

$$= \bigcup_{i \in \mathbb{N}} trans_{\mathcal{P}}^{i}(\emptyset).$$

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

Let M be another fixpoint of $trans_{\mathcal{P}}$.

We want to show: $\bigcup_{i \in \mathbb{N}} trans_{\mathcal{P}}^{i}(\emptyset) \subseteq M$.

It suffices to show: $trans_{\mathcal{P}}^{i}(\emptyset) \subseteq M$ for all $i \in \mathbb{N}$.

Prove this by induction on i.

Ind Base: $i = 0$

—

$trans_{\mathcal{P}}^{0}(\emptyset) = \emptyset \subseteq M$ ✓

Ind Step : $i > 0$

Ind Hypothesis: $trans_{\mathcal{P}}^{i-1}(\emptyset) \subseteq M$

By monotonicity of $trans_{\mathcal{P}}$: $trans_{\mathcal{P}}^{i}(\emptyset) \subseteq trans_{\mathcal{P}}(M) = M$

because M is a fixpoint of $trans_{\mathcal{P}}$. ☒

Finally, we can define the fixpoint semantics of LP:

Def 4.1.15 (Fixpoint Semantics of LP)

Let $\mathcal{P}$ be a LP, let $G = \{\neg A_1, \dots, \neg A_k\}$ be a query.
Then the fixpoint semantics of $\mathcal{P}$ w.r.t. G is defined as:

$$\mathcal{F}[\![\mathcal{P}, G]\!] = \{\sigma(A_1 \wedge \dots \wedge A_k) \mid \sigma(A_i) \in lfp(trans_{\mathcal{P}}) \text{ for all } 1 \le i \le k\}.$$

Thm 4.1.16 (Equivalence of all 3 semantics-definitions)

Let $\mathcal{P}$ be a LP, G be a query.
Then $\mathcal{D}[\![\mathcal{P}, G]\!] = \mathcal{P}[\![\mathcal{P}, G]\!] = \mathcal{F}[\![\mathcal{P}, G]\!]$.

Proof: see course notes.