3.9 Logisches Programmieren und Prolog

Sprache der Logik zur Darstellung von Wissen (deklarativ) über Strukturen. Wissensherleitung durch logische Folgerung (deduktiv, Resolution usw.)

Prozedurale Sicht von Resolution

$Z = ((Y \cdot 2) + X) - Y$ Funktion von X, Y

$\mathrm{succ}(\mathrm{succ}(0))$

0 $\mathrm{goal}(X, Y, Z) :- \mathrm{mult}(Y, 2, A), \mathrm{add}(A, X, B), \mathrm{subt}(B, Y, Z).$

1 $\mathrm{add}(R, \mathrm{succ}(S), \mathrm{succ}(T)) :- \mathrm{add}(R, S, T).$

2 $\mathrm{add}(T, 0, T).$

3 $\mathrm{subt}(\mathrm{succ}(R), \mathrm{succ}(S), T) :- \mathrm{subt}(R, S, T).$

4 $\mathrm{subt}(T, 0, T).$

5 $\mathrm{mult}(R, \mathrm{succ}(S), T) :- \mathrm{mult}(R, S, U), \mathrm{add}(R, U, T).$

6 $\mathrm{mult}(R, 0, 0).$

$\mathrm{goal}, \mathrm{add}, \mathrm{subt}, \mathrm{mult}$ sind 3-stellige P-Konstanten zur Darstellung der Funktionen, die den ersten beiden Argumenten als Wert 3-Argumente zuordnen.

Wie „berechnet“ sich $((Y \cdot 2) + X) - Y$, wenn
$X \leftarrow \mathrm{succ}(\mathrm{succ}(0))$ und $Y \leftarrow \mathrm{succ}(0)$

? $-\mathrm{goal}(\mathrm{succ}(\mathrm{succ}(0)), \mathrm{succ}(0), Z)$

↯

. With $Z = \mathrm{succ}(\mathrm{succ}(\mathrm{succ}(0)))$

Horn-Logik

Erinnerung: Klauseln $\{L_1, L_2, \ldots, L_n\}$ Menge von Literalen
$= \{A_1, \ldots, A_n\} \cup \{\neg B_1, \ldots, \neg B_m\}$, A_j, B_j Atome.

Hornlogik: Klauseln mit höchstens einem positiven Literal,
d. h. $n \leq 1$. Einteilung:

$\{A, \neg B_1, \ldots, \neg B_m\}$ $m > 0$	**Regel Klausel**
$\{A\}$	**Fakt Klausel**
$\{\neg B_1, \ldots, \neg B_m\}$ $m > 0$	**Goal Klausel**
$\varnothing$	**leeres Goal**

Formeln in KLF: Endliche Mengen von Literalen.

Hornklausel	Formel von PL-1
$\{A, \neg B_1, \ldots, \neg B_m\}$	$\forall\ (A \vee \neg B_1 \vee \cdots \vee \neg B_m)$
	bzw. $(\forall\ ((B_1 \wedge \cdots \wedge B_m) \rightarrow A))$
$\{A\}$	$\forall\ (A)$
$\{\neg B_1, \ldots, \neg B_m\}$	$\forall\ (\neg B_1 \vee \cdots \vee \neg B_m)$
	bzw. $\neg\exists\ (B_1 \wedge \cdots \wedge B_m)$
$\square$	false

Notationen:

Formel	logisches Programm	Prolog
$\forall\ ((B_1 \wedge \cdots \wedge B_m) \rightarrow A)$	$A \leftarrow B_1, \ldots, B_m$	$A :- B_1, \ldots, B_m.$
$\forall\ (A)$	$A \leftarrow$	$A.$
$\neg\exists\ (B_1 \wedge \cdots \wedge B_m)$	$\leftarrow B_1, \ldots, B_m$	$?- B_1, \ldots, B_m.$

Horn-Logik (Forts.)

Eine Menge von Regeln und Fakten ist ein **logisches Programm** P (d. h. die Programmformeln sind entweder Regeln oder Fakten).

Eine Struktur $\mathcal{R}$ ist **Modell** von P, falls $\mathcal{R}$ Modell der entsprechenden Formeln in P ist.

$P \models \varphi$ hat die übliche Bedeutung.

Beachte: Sei P logisches Programm und $? - B_1, \ldots, B_m$ ein nichtleeres Ziel. Dann sind äquivalent

1. $P \cup \{? - B_1, \ldots, B_m\}$ hat kein Modell (unerfüllbar)
2. $P \models \exists(B_1 \wedge \cdots \wedge B_m)$

Herbrand Interpretationen reichen aus: d. h. Termmengen und Funktionen sind fest durch die Formeln (Programm) definiert.

Offen ist nur noch die Interpretation der P-Konstanten.

$$\mathcal{R} \leftrightarrow I(\mathcal{R}) = \{r(t_1, \ldots, t_n) \mid r\ P\text{-Konstante, } n\text{-stellig}, \\ t_1, \ldots, t_n \in H_P \text{ Grundterme}, \\ (t_1, \ldots, t_n) \in r_{\mathcal{R}}\}$$

Semantik logischer Programme (deklarative Semantik).

Sei P ein logisches Programm. Unter den Herbrand Interpretationen die Modelle von P sind, gibt es ein minimales Herbrand Modell M_P:

$$M_P = \{r(t_1, \ldots, t_n) \mid t_1, \ldots, t_n \text{ Grundterme und } \\ P \models r(t_1, \ldots, t_n)\}$$

M_P lässt sich rekursiv definieren.

Horn-Logik (Forts.)

3.50 Satz

Sei $\exists X_1 \cdots \exists X_k (B_1 \wedge \cdots \wedge B_m)$ eine abgeschlossene existentielle Formel und P logisches Programm. Dann sind äquivalent:

1. $P \models \exists X_1 \cdots \exists X_k (B_1 \wedge \cdots \wedge B_m)$
2. $P \models (B_1 \wedge \cdots \wedge B_m)[X_1/t_1, \ldots, X_k/t_k]$ für Grundterme $t_1, \ldots, t_k$
3. M_p ist Modell von $\exists X_1 \cdots \exists X_k (B_1 \wedge \cdots \wedge B_m)$
4. $M_p \models (B_1 \wedge \cdots \wedge B_m)[X_1/t_1, \ldots, X_k/t_k]$ für Grundterme $t_1, \ldots, t_k$

Grundlage für M_p ist die Semantik (Bedeutung) von P.
(Beachte der Satz gilt nicht für universelle Formeln!)

3.51 Beispiel

P über Signatur $0, \mathrm{succ}$ (Fkt. Symbole), add (3 St. Pr.-Konstante)

$$P: \quad \mathrm{add}(X, 0, X).$$
$$\mathrm{add}(X, \mathrm{succ}(Y), \mathrm{succ}(Z)) :- \mathrm{add}(X, Y, Z).$$

Offenbar ist

$$M_p = \{\mathrm{add}(\mathrm{succ}^n(0), \mathrm{succ}^m(0), \mathrm{succ}^{n+m}(0))\ n, m \in \mathbb{N}\}$$

Wie werden existentielle Anfragen beantwortet

(Frage 1) $? - \mathrm{add}(\mathrm{succ}^3(0), \mathrm{succ}^8(0), Z)$

JA mit $Z = \mathrm{succ}^{11}(0)$

(Frage 2) $? - \mathrm{add}(X, \mathrm{succ}^8(0), Z)$

$(P \stackrel{?}{\models} \exists X \exists Z \ \mathrm{add}(X, \mathrm{succ}^8(0), Z))$

JA mit $X = 0, Z = \mathrm{succ}^8(0)$
mit $X = \mathrm{succ}(0), Z = \mathrm{succ}^9(0) \ldots$
$\Big\} Z = \mathrm{succ}^8(X)$

allgemeinste Lösung.

(Frage 3) $? - \mathrm{add}(\mathrm{succ}^3(0), Y, Z)$

$(P \stackrel{?}{\models} \exists Y \exists Z \ \mathrm{add}(\mathrm{succ}^3(0), Y, Z))$

JA mit $Y = 0, Z = \mathrm{succ}^3(0)$
mit $Y = \mathrm{succ}(0), Z = \mathrm{succ}^4(0)$
mit $Y = \mathrm{succ}^2(0), Z = \mathrm{succ}^5(0) \ldots$

$Z = \mathrm{succ}^3(Y)$ ist jedoch keine allgemeinste Lösung:
Da $\forall Y \ \mathrm{add}(\mathrm{succ}^3(0), Y, \mathrm{succ}^3(Y))$ nicht logische Folgerung von P ist (Übung!)
(Sie ist jedoch in M_p gültig!! Induktives Theorem)

Lösungssubstitutionen

(Frage 4) $? - \mathrm{add}(X, Y, \mathrm{succ}^3(0))$

$(P \stackrel{?}{\models} \exists X \exists Y \, \mathrm{add}(X, Y, \mathrm{succ}^3(0))$

$X = 0$	$Y = \mathrm{succ}^3(0)$
$\mathrm{succ}(0)$	$\mathrm{succ}^2(0)$
$\mathrm{succ}^2(0)$	$\mathrm{succ}(0)$
$\mathrm{succ}^3(0)$	0

Lösungssubstitutionen:
Sei $G = ? - B_1, \ldots, B_m$ ein goal, P logisches Programm, σ Substitution, $\sigma|_G$ Einschränkung von σ auf die Variablen, die in G vorkommen.

$\sigma = \{X_1 \leftarrow t_1, \ldots, X_n \leftarrow t_n\}$ ist eine **korrekte Lösungssubstitution** für $P \cup \{G\}$ gdw $X_1, \ldots, X_n$ kommen in G vor und $P \models \forall \, ((B_1 \wedge \cdots \wedge B_m)\sigma)$.

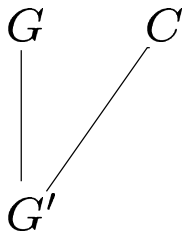
(Beachte: nicht äquivalent zu $M_p \models \forall \, ((B_1 \wedge \cdots \wedge B_m)\sigma)$ nur, falls variablenfrei).

Wie operationalisiert man die Bestimmung von korrekten Lösungssubstitutionen ? $\rightsquigarrow$ **Operationale Semantik** Varianten der Resolution.

SLD-Resolution
(selective linear Resolution for definitive clauses)

Sei G goal ? $-A_1, \ldots, A_m$ und $C \equiv A :- B_1, \ldots, B_q$ eine Programmformel ($q = 0$ erlaubt). Seien weiterhin G und C variablendisjunkt und sei μ ein MGU von A_k und A. (Die Klauseln können resolviert werden).

Das goal
$G' \equiv ?- A_1\mu, \ldots, A_{k-1}\mu, B_1\mu, \ldots, B_q\mu, A_{k+1}\mu, \ldots, A_k\mu$
ist eine SLD-Resolvente von G und C über μ.



SLD als Regel

P-Klausel, goal $\rightsquigarrow$ goal$'$

SLD-Ableitungen, wie üblich definieren: Eine Ableitung der Form

$G_0, G_1, \ldots, G_n, \ldots$ Programmformeln $C_0, C_1, \ldots, C_n, \ldots$ Substitutionen $\mu_0, \mu_1, \ldots, \mu_n, \ldots$, so dass

G_{n+1} SLD-Resolvente von G_n und C_n über μ_n (Hierbei kommen die Variablen in C_{n+1} nicht in $G_0, G_1, \ldots, G_n, C_0, \ldots, C_n, \mu_0, \ldots, \mu_n$ vor).

3.52 Definition Sei P logisches Programm, G goal. Eine SLD-Widerlegung von $P \cup \{G\}$ ist eine endliche SLD-Ableitung. Bis zum Ziel G_n aus G, wobei G_n leer ist. $C_0, \ldots, C_{n-1}$ sind Varianten (Umbenennungen) von Programmformeln aus P.

$\mu = (\mu_0\mu_1 \cdots \mu_{n-1})|_G$ berechnete **Lösungssubstitution**.

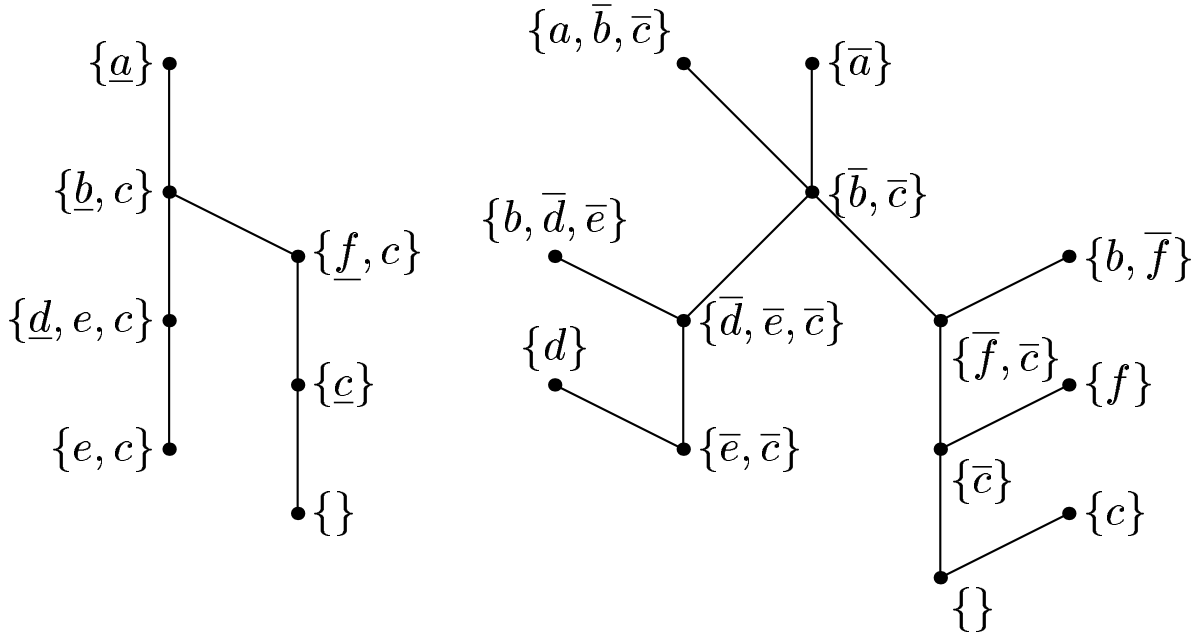
Beispiel

3.53 Bemerkung und Beispiel

• Korrektheit und Vollständigkeit lassen sich beweisen!
Siehe etwa Leitsch: Resolutionskalküle

• Beispiel

	Klauseln
$a :- b, c.$	$\{a, \bar{b}, \bar{c}\}$
$a :- d.$	$\{a, \bar{d}\}$
$b :- d, e.$	$\{b, \bar{d}, \bar{e}\}$
$b :- f.$	$\{b, \bar{f}\}$
$c.$	$\{c\}$
$c :- d, f.$	$\{c, \bar{d}, \bar{f}\}$
$d.$	$\{d\}$
$f.$	$\{f\}$



Operationale Semantik von PROLOG

Prolog: Logik + Kontrolle

Fixiere Reihenfolge der SLD-Schritte

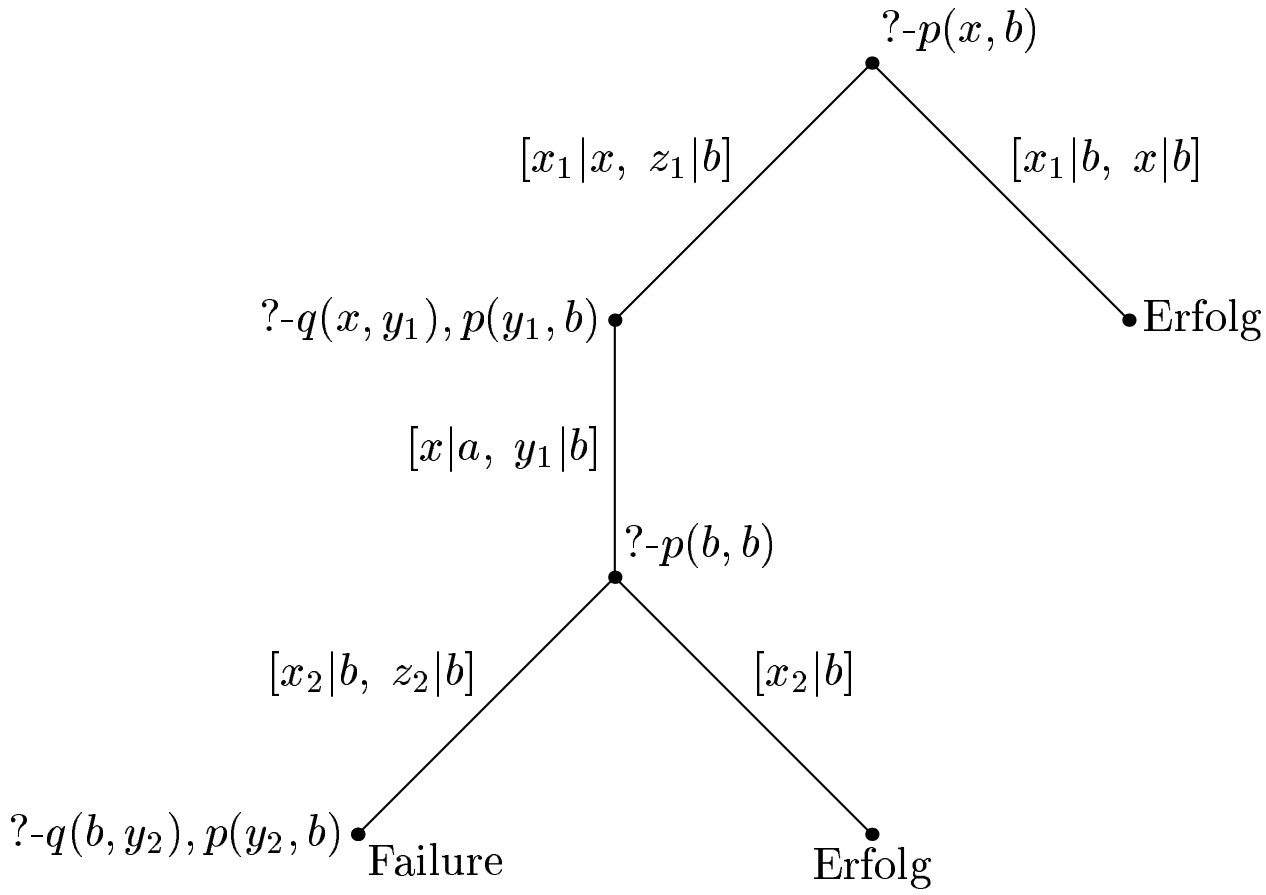
- Ordne Programmformeln (Liste)
- Goals als Listen - erstes Literal
- Bilde stets Resolventen mit Kopf der ersten Programmformel mit erstem Literal des Goals, **normale SLD-Resolution**.
- Probleme: Laufzeiten sind abhängig von den Reihenfolgen! Oft hilft ein Umordnen der Literale im Goal oder in den Programmklauseln.

Beispiele

- P sei gegeben durch:

  1 $p(X, Z) :- q(X, Y), p(Y, Z).$
  2 $p(X, X).$
  3 $q(a, b).$

$G \equiv ?- p(X, b)$ SLD(P, G) Baum aller N-SLD Resolventen



Beispiele (Fort.)

• **Umordnung der Literale in Programmformeln**

1 $p(X, Z) :- p(Y, Z), q(X, Y).$
2 $p(X, X).$
3 $q(a, b).$

$G \equiv ?- \ p(X, b)$

SLD(P, G) enthält ∞-Pfade.

Depth First $\rightsquigarrow$ kein Ergebnis.

Umordnung der Programmformeln

$\rightsquigarrow$ Umordnung des SLD-Baums

• Beispiel: **Listen über** Σ

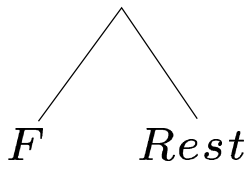
Gegeben Signatur Σ definiere Listen über Σ:

2-stellige Funktion Infix-Notation $[\cdot|\cdot] : L(\Sigma)^2 \rightarrow L(\Sigma)$

Konstante [ ] bezeichne die leere Liste.

Rekursive Definition:

- [ ] ist Liste über Σ
- $[F|Rest]$ ist Liste über Σ, falls F Σ-Term oder Liste über Σ
- $Rest$ Liste über Σ



Beispiel: Listen (Forts.)

$[F_1, F_2, \ldots, F_n|Rest] := [F_1|[F_2|\cdots|[F_n|Rest]\cdots]]$

$[F_1, \ldots, F_n] := [F_1, \ldots, F_n|[\ ]]$

Operationen auf Listen

•append$([\ ], M, M)$.

•append$([X|L], M, [X|N]) :-$ append(L, M, N).

? $-$append$([a, b], [a, Y], Z)$

Ergibt: $.Z \leftarrow [a, b, a, Y]$

Weitere Operationen

element$(X, [X|L])$.
element$(X, [Y|L]) :-$ unequal(Y, X), element(X, L).
mirror$([\ ], [\ ])$.
mirror$([X|L], M) :-$ mirror(L, N), append$(N, [X], M)$.
delete$([\ ], X, [\ ])$.
delete$([X|L], X, M) :-$ delete(L, X, M)
delete$([Y|L], X, [Y|M]) :-$ unequal(Y, X), delete(L, X, M).
delete1$([X|L], X, L)$.
delete1$([Y|L], X, [Y|M]) :-$ unequal(Y, X), delete1(L, X, M).

Schlussaufgabe: Formalisiere und beweise folgenden Satz:

If the professor is happy if all his students like logic, then he is happy if he has no students.