Tableaux-Methode für Pl-1

Definition 5.1

Sprache: $\neg, \wedge, \vee, \rightarrow, \forall, \exists$ (zunächst ohne $=$)
Formeln der Sprache in Klassen einteilen:

- Atomare und negierte atomare Formeln.
- α-Formeln (wie in A-Logik)
  $A \wedge B, \neg(A \vee B), \neg(A \rightarrow B), \neg\neg A, \alpha_1, \alpha_2$ wie gehabt.
- β-Formeln (wie A-Logik) $\neg(A \wedge B), (A \vee B), (A \rightarrow B)$
- γ-Formeln $\forall x A, \neg\exists x A \quad (A \in \textbf{Form})$
- δ-Formeln $\exists x A, \neg\forall x A \quad (A \in \textbf{Form})$

Tableaux-Methode für Pl-1 (Forts.)

- Regeln zur Tableau-Konstruktion:
  α, β Regeln zur Verarbeitung α, β Formeln.

- γ-Regeln:

γ	$\forall x\ A$	$\neg\exists x\ A$
$\gamma[t]$	$A_x[t]$	$\neg A_x[t]$

  t beliebiger Term, Substitution erlaubt.

- δ-Regeln:

δ	$\exists x\ A$	$\neg\forall x\ A$
$\delta[y]$	$A_x[y]$	$\neg A_x[y]$

! y Variable (oder c Konstante 0-stellig) „y neu“:
y kommt noch nicht in Formeln im Ast zu $A_x[y]$ frei vor und die Substitution ist erlaubt.

Tableaux-Methode für Pl-1 (Forts.)

- **Systematische Konstruktion eines Modells**:
  Falls die Sprache keine F-Symbole enthält reichen die „Bezeichner“ für Elemente des Definitionsbereichs aus.
  Sonst wähle als Definitionsbereich für die Interpretation eine geeignete abgeschlossene Termmenge.
- Führe soviele „Terme“ ein wie notwendig.
  Existentielle Formeln benötigen „Lösungen“, solche dürfen nicht eingeschränkt sein (anderswie verwendet worden sein).
- δ-Formeln brauchen nur einmal „erfüllt“ zu werden.
  γ-Formeln hingegen müssen für alle Objekte die eingeführt wurden garantiert werden, müssen somit immer weiterhin beachtet werden.

Tableaux-Methode für Pl-1 (Forts.)

Lemma 5.2

Sei Γ eine Menge von Formeln über einer abgeschlossenen Termmenge (Universum) $\mathcal{U}$ mit

1. *Für keine Formel A gilt $A \in \Gamma$ und $\neg A \in \Gamma$.*
2. *Für jede α-Formel in Γ gilt $\alpha_1, \alpha_2 \in \Gamma$.*
3. *Für jede β-Formel in Γ gilt $\beta_1 \in \Gamma$ oder $\beta_2 \in \Gamma$.*
4. *Für jede γ-Formel in Γ gilt $\gamma[t] \in \Gamma$ für alle $t \in \mathcal{U}$.*
5. *Für jede δ-Formel in Γ gibt es ein $t \in \mathcal{U}$ mit $\delta[t] \in \Gamma$.*

$\hookrightarrow$ *Dann ist Γ erfüllbar.*

Tableaux-Methode für Pl-1 (Forts.)

! **Problem:** $\mathcal{U}$ kann unendlich sein. Erst recht wenn F-Symbole vorhanden!

- Sprachen mit „$=$“ dann Funktionskonstanten und Terme.

- ϵ-Regel

$$\frac{t_1 = t_2}{A[t_1 \leftarrow t_2]}$$

$\hookrightarrow$ In einer Formel A kann ein Term t_1 durch einen „gleichen“ Term t_2 ersetzt werden, wenn im Ast zu A der Knoten mit der Markierung $t_1 = t_2$ vorkommt. $\forall x \; x = x$ als Wurzel der $=$ Tableaux zulassen. ◀

- Problematische Regel, da sie zu einer Formelexplosion führt.

Tableaux-Methode für Pl-1 (Forts.)

Satz 5.3

*Sei $A \in$ **Form**$(\mathcal{L})$, $\Sigma \subseteq$ **Form**$(\mathcal{L})$*

a) $\models A$ gdw *es gibt ein abgeschlossenes Tableau für* $\neg A$.

b) $\Sigma \models A$ gdw *es gibt ein abgeschlossenes Tableau für* $\Sigma \cup \{\neg A\}$.

- **Systematische Tableaux-Konstruktion**:

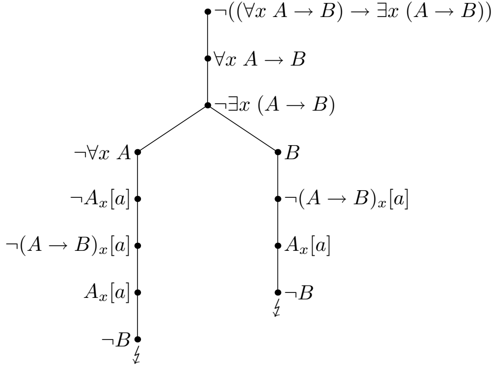
  Reihenfolge der Regelanwendungen: $\Sigma, \quad \alpha, \quad \delta, \quad \gamma, \quad \beta$

  ! Beachte: t $A_x[t]$ muss erlaubt sein.
  y neu bei δ-Regel beachten (frei).

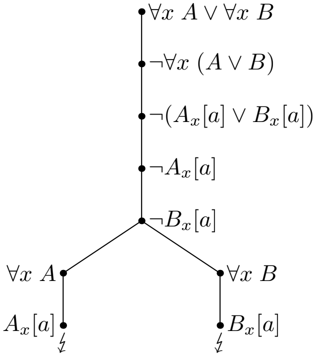
- Falls keine Funktionskonstanten in $\mathcal{L}$ Interpretation mit $\{c_1, c_2, \dots\}$ als Konstanten.

Beispiele

Beispiel 5.4 ($\models (\forall x A \rightarrow B) \rightarrow \exists x (A \rightarrow B)$, x nicht frei in B)



Beispiel 5.5 ($\Gamma = \{\forall x A \vee \forall x B\}$, $C \equiv \forall x(A \vee B)$. Gilt $\Gamma \models C$?)



! γ-Formeln dürfen nicht abgehakt werden, da man immer wieder neue Terme einführen kann und $\gamma[t]$ erfüllt werden muss!

Beispiele (Forts.)

Beispiel 5.6 ($\exists x \, \neg p(x,x) \models \neg \forall x \exists y p(x,y)$)

Gibt es Modelle für $\{\exists x \, \neg p(x,x), \forall x \exists y \; p(x,y)\}$

$$
\begin{array}{l}
\bullet\ \exists x \, \neg p(x,x) \\
\bullet\ \forall x \exists y \; p(x,y) \\
\bullet\ \neg p(a,a) \\
\bullet\ \exists y \; p(a,y) \\
\bullet\ p(a,b) \\
\bullet\ \exists y \; p(b,y) \\
\bullet\ p(b,c) \\
\vdots
\end{array}
$$

$\hookrightarrow$ Interpretation:

$$
\begin{array}{c|cccccc}
p(x,y) & a & b & c & d & e & \cdots \\
\hline
a & 0 & 1 & & & & \\
b & & & 1 & & & \\
c & & & & 1 & & \\
d & & & & & 1 & \\
e & & & & & & 1 \\
 & & & & & & \ddots
\end{array}
$$

- Es gibt Interpretation mit nur 2 Elementen $\neg p(a,a), p(a,b), p(b,a), p(b,b)$, die Modell sind.

$\hookrightarrow$ Vorschrift „neu“ kann aufgeweicht werden.

! Erst alle Konstanten verwenden, falls diese Wahl zu $\lightning$ führt müssen neue Konstanten einführt werden, sonst Modell gefunden.

Beispiele (Forts.)

$\exists x \; \neg p(x,x)$

$\forall x \exists y \; p(x,y)$

$\neg p(a,a)$

$\exists y \; p(a,y)$

$p(a,b)$

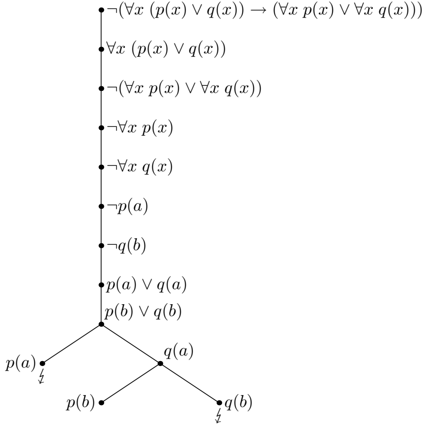
$\exists y \; p(b,y)$

$p(b,a)$

Interpretation z.B.:

	a	*b*
a	0	1
b	1	0

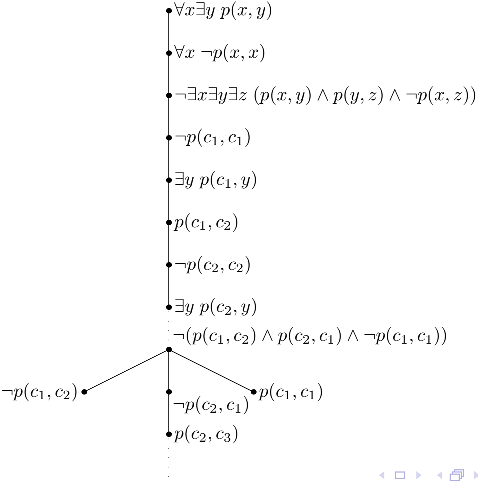
Beispiel 5.7 (Gilt $\models \forall x(p(x) \vee q(x)) \rightarrow (\forall x p(x) \vee \forall x q(x))$?)



$$I = (\{a, b\} : \quad I(p)(a) = I(q)(b) = F, \quad I(p)(b) = I(q)(a) = W)$$

Beispiel 5.8

$$\forall x \exists y\ p(x,y), \forall x\ \neg p(x,x) \models \exists x \exists y \exists z\ (p(x,y) \land p(y,z) \land \neg p(x,z))$$



Beispiele (Forts.)

Interpretation:

$p(c_1, c_1) = F \quad p(c_1, c_2) = W \quad p(c_2, c_2) = F$
$p(c_2, c_1) = F \quad p(c_2, c_3) = W \quad p(c_3, c_3) = F \ldots$

	c_1	c_2	c_3	c_4	c_5
c_1	0	1			
c_2	0	0	1		
c_3			0	1	
c_4				0	1
$\vdots$					

Beispiele Formeln mit $=$: Monoide und Gruppen

Beispiel 5.9

$$\{\forall x \forall y \forall z \; (x \cdot y) \cdot z = x \cdot (y \cdot z)$$
$$\forall x \; x \cdot 1 = x$$
$$\forall x \; x \cdot \bar{x} = 1\}$$

Gruppenaxiome

Behauptung: Es gilt: Aus Gruppenaxiome

$\models \forall x \; 1 \cdot x = x$ und $\models \forall x \; \bar{x} \cdot x = 1$

- $\forall x\ x = x$
- Axiome
- $\neg\forall x\ 1 \circ x = x$
- $\neg\forall x\ \overline{x} \circ x = 1$
- $\neg\overline{b} \circ b = 1$
- $b \circ \overline{b} = 1$
- $(\overline{b} \circ b) \circ 1 = \overline{b} \circ b)$
- $\overline{b} \circ \overline{\overline{b}} = 1$
- $(\overline{b} \circ b) \circ (\overline{b} \circ \overline{\overline{b}}) = \overline{b} \circ b$
- $(\overline{b} \circ b) \circ (\overline{b} \circ \overline{\overline{b}}) = ((\overline{b} \circ b) \circ \overline{b}) \circ \overline{\overline{b}}$
- $(\overline{b} \circ b) \circ \overline{b} = \overline{b} \circ (b \circ \overline{b})$
- $(\overline{b} \circ b) \circ \overline{b} = \overline{b} \circ 1$
- $\overline{b} \circ 1 = \overline{b}$
- $\overline{b} \circ (b \circ \overline{b}) = \overline{b}$
- $(\overline{b} \circ b) \circ \overline{b} = \overline{b}$
- $(\overline{b} \circ b) \circ (\overline{b} \circ \overline{\overline{b}}) = \overline{b} \circ \overline{\overline{b}}$
- $(\overline{b} \circ b) \circ (\overline{b} \circ \overline{\overline{b}}) = 1$
- $\overline{b} \circ b = 1$

Entscheidbare Fälle (Präfixe) Allgemeingültigkeit $\mathcal{L}$ Sprache ohne Funktionskonstanten abgeschlossene Formeln

1. $\forall x_1 \cdots \forall x_m \exists y_1 \cdots \exists y_m$ A — A quantorenfrei
2. $\forall x_1 \cdots \forall x_m \exists y \forall z_1 \cdots \forall z_n A$
3. $\forall x_1 \cdots \forall x_m \exists y_1 \exists y_2 \forall z_1 \cdots \forall z_n A$
4. $\exists y_1 \cdots \exists y_n \forall x_1 \cdots \forall x_m \exists z A$ — $n, m \geq 1$ n. entsch.
5. $\forall x_1 \cdots \forall x_m \exists y_1 \exists y_2 \exists y_3 \forall z_1 \cdots \forall z_n A$ — $m, n \geq 0$ n. entsch.

Beispiel 5.10 (Typ 2. Formel)

$$\neg\forall x_1 \forall x_2 \exists y \forall z\ A$$

$$\neg\forall x_2 \exists y \forall z\ A_{x_1}[a]$$

$$\neg\exists y \forall z\ A_{x_1,x_2}[a,b]$$

$$\neg\forall z\ A_{x_1,x_2,y}[a,b,a]$$

$$\neg\forall z\ A_{x_1,x_2,y}[a,b,b]$$

$$A_{x_1,x_2,y,z}[a,b,a,a_1]$$

$$A_{x_1,x_2,y,z}[a,b,b,a_2]$$

Resolventenmethode (PL1 - Resolutionsverfahren)

- **Formeln in Klauselnormalform auf Erfüllbarkeit testen**.

Definition 5.11

Eine Formel $A \in \mathbf{Form}(\mathcal{L})$ ist in **Klauselform** (KLF), wenn sie die Gestalt

$$A \equiv \forall x_1 \forall x_2 \cdots \forall x_n [C_1 \wedge C_2 \wedge \cdots \wedge C_k]$$

hat. Dabei sind die C_j Klauseln (d. h. Disjunktionen von Literalen ohne Wiederholungen) und die Variablen $x_1, \ldots, x_n$ sind alle Variablen, die in den C_j vorkommen.

- $\forall x_1, \ldots, \forall x_n$ Präfix $[C_1 \wedge \cdots \wedge C_k]$ Mantisse (Matrix).

$\hookrightarrow$ PKNF und Präfix enthält kein $\exists$ Quantor.

! **Beachte**: In Literalen kommen nun Variablen vor.
L und M sind konjugierte (komplementäre) Literale, wenn
$L = \bar{M}$
(A Atom, $\neg A \equiv \bar{A}$, $\bar{\bar{A}} \equiv A$).

- $p(x)$, $\neg p(y)$ sind nicht konjugiert oder komplementär. Nach Substitution z. B. $x \leftarrow a$, $y \leftarrow a$, $p(a), \neg p(a)$ sind konjugiert. Aber auch $x \leftarrow x$, $y \leftarrow x$, $p(x), \neg p(x)$.

- **Ziel:** Resolventenmethode verallgemeinern auf Formeln mit Variablen und Funktionen (d.h. Terme).

Skolemisierung

Lemma 5.12 (Skolem)

Jede Formel A der P-Logik erster Stufe kann effektiv in eine erfüllbarkeitsäquivalente Formel A' in Klauselform transformiert werden.

- *A ist erfüllbar* gdw *A' ist erfüllbar.*

- **Anwendung**: Nachweis der Allgemeingültigkeit

$\models B$ gdw $\{\neg B\}$ unerfüllbar gdw $(\neg B)'$ unerfülllbar.

Skolemisierung: Verfahren

Beweis:

A gegeben. Transformiere A wie folgt:

Schritt 1: Bilde den existentiellen Abschluss von A.
Schritt 2: Eliminiere alle überflüssigen Quantoren.
Schritt 3: Umbenennung mehrfach quantifizierter Variablen.
Schritt 4: Ersetze Operatoren, die von $\wedge, \vee, \neg$ verschieden sind. z. B. $\rightarrow$, **If** $\ldots, \leftrightarrow \ldots$
Schritt 5: Schiebe $\neg$ nach innen, bis sie vor den Atomen stehen. Insbesondere $\neg\neg A$ durch A ersetzen.
$\neg\forall x A \rightsquigarrow \exists x \neg A$, $\neg(A \vee B) \rightsquigarrow (\neg A \wedge \neg B)$
$\neg(A \wedge B) \rightsquigarrow (\neg A \vee \neg B)$

Skolemisierung: Verfahren (Fort.)

Schritt 6: Schränke Quantoren auf ihren Wirkungsbereich ein:

$$Qx(A * B) \rightsquigarrow A * QxB \quad x \text{ nicht frei in } A$$
$$\rightsquigarrow QxA * B \quad x \text{ nicht frei in } B$$

Schritt 7: Eliminiere existentielle Quantoren durch Einführung von **Skolem Funktionen**. Wähle dabei die erste Teilformel (von links) der Form $\exists y B(y)$ und ersetze sie durch $B_y[f(x_1, \dots, x_n)]$, $n \geq 0$, wobei

a) $x_1 \dots x_n$ alle unterschiedlichen freien Variablen in $\exists y B(y)$ sind, die links von $\exists y B(y)$ universell quantifiziert sind.

b) f eine „frische“ n-stellige Funktionskonstante.

Schritt 8: Schiebe $\forall$ Quantoren nach links.

Schritt 9: Bringe Matrix in KNF und simplifiziere.

Skolemisierung

- **Korrektheit** $A_j \equiv \cdots \exists y B(y) \rightsquigarrow A_{j+1} \equiv \cdots B_y[f(x_1, \dots, x_n)]$
- **Behauptung:** A_j ist erfüllbar gdw A_{j+1} ist erfüllbar.
  Sei I Modell für A_j: Für jede mögliche Belegung der Variablen $x_1, \dots, x_n$ muss es ein oder mehrere Werte für y geben, mit denen A_j wahr wird.
  I' wie I zusätzlich eine n-stellige Funktionskonstante $f : f(x_1, \dots, x_n)$ gibt für Wertetupel für $x_1 \dots x_n$ ein (den) y Wert als Funktionsergebnis.

Beispiele

Beispiel 5.13

$$\forall x \, \{p(x) \to \exists z \, \{\neg \forall y \, [q(x,y) \to p(f(x_1))] \land \forall y \, [q(x,y) \to p(x)]\}\}$$

1. Existentieller Abschluss und Elimination von überf. Quantoren:
$$\exists x_1 \forall x \, \{p(x) \to \{\neg \forall y \, [q(x,y) \to p(f(x_1))] \land \forall y \, [q(x,y) \to p(x)]\}\}$$
2. Umbenennung von y:
$$\exists x_1 \forall x \, \{p(x) \to \{\neg \forall y \, [q(x,y) \to p(f(x_1))] \land \forall z \, [q(x,z) \to p(x)]\}\}$$
3. Elimination von $\to$:
$$\exists x_1 \forall x \, \{\neg p(x) \lor \{\neg \forall y \, [\neg q(x,y) \lor p(f(x_1))] \land \forall z \, [\neg q(x,z) \lor p(x)]\}\}$$

Beispiele (Forts.)

4. $\neg$ nach innen:
$\exists x_1 \forall x \, \{\neg p(x) \vee \{\exists y \, [q(x,y) \wedge \neg p(f(x_1))] \wedge \forall z \, [\neg q(x,z) \vee p(x)]\}\}$
5. Quantoren auf Geltungsbereich einschränken:
$\exists x_1 \forall x \, \{\neg p(x) \vee \{[\exists y \, q(x,y) \wedge \neg p(f(x_1))] \wedge [\forall z \, \neg q(x,z) \vee p(x)]\}\}$
6. Eliminieren der Ex.-Quantoren $\exists x_1$ und $\exists y$ (0 - stellige bzw. 1-stellige Funktionseinführung):

$$\forall x \, \{\neg p(x) \vee \{[q(x,g(x)) \wedge \neg p(f(a))] \wedge [\forall z \, \neg q(x,z) \vee p(x)]\}\}$$

Beispiele (Forts.)

7. Quantoren nach links:

$$\forall x \forall z \, \{\neg p(x) \vee \{[q(x, g(x)) \wedge \neg p(f(a))] \wedge [\neg q(x, z) \vee p(x)]\}\}$$

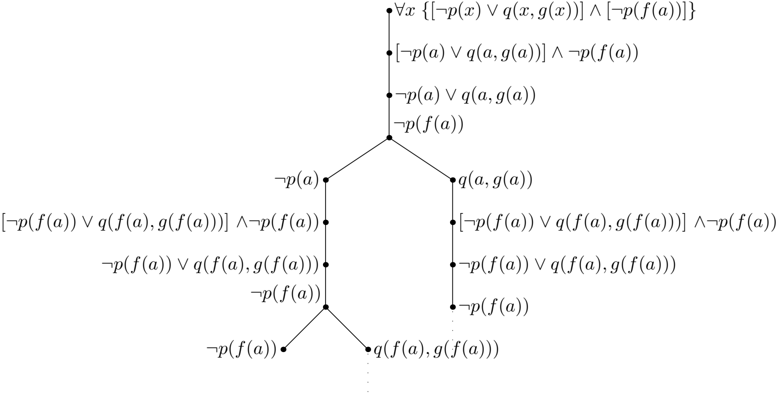
8. KNF + Simplifikation:

$$\forall x \forall z \, \{[\neg p(x) \vee q(x, g(x))] \wedge [\neg p(x) \vee \neg p(f(a))] \wedge [\neg p(x) \vee \neg q(x, z) \vee p(x)]\}$$

$$\rightsquigarrow A' \equiv \forall x \, \{[\neg p(x) \vee q(x, g(x))] \wedge \neg p(f(a))\}$$

- $A' \equiv \forall x[[\neg p(x) \vee q(x, g(x))] \wedge [\neg p(f(a))]]$ Erfüllbar?

Beispiele (Forts.) - Tableaux:



Terme: $a, f(a), g(a), g(f(a)), g(g(a)), g(g(f(a))), g(g(g(a)))$.

$$\mathcal{T} = \{a, g^n(a), g^n(f(a)) \quad n \geq 0\} \text{ Definitionsbereich}$$

- Erfüllbar:

$$I = (\{a\}, f(a) := a, g(a) := a, p(a) = 0, q(a, a) = 1)$$

! **Bei nicht Erfüllbarkeit**: im endlich abgeschlossenen Tableau kommen nur endlich viele Grundterme (Terme ohne Variablen) vor, d. h. eine endliche Menge von Grundinstanzen der Matrix ist unerfüllbar.

$\rightsquigarrow$ **Herbrand-Interpretationen**.

Herbrand-Universum

Definition 5.14 (Herbrand-Universum)

Sei $A \in$ **Form**. Das **Herbrand-Universum** H_A ist die Menge aller Terme, die aus den Individuenkonstanten (0-stellige Funktionskonstanten) und den Funktionskonstanten, die A vorkommen, gebildet werden können.

Enthält A keine Individuenkonstante, so wird eine hinzugenommen.

Herbrand-Universum (Forts.)

Beispiel 5.15

- $A \equiv \forall x\{[\neg p(x) \vee q(x, g(x))] \wedge \neg p(f(a))\}$

$\hookrightarrow \quad H_A = \{a, f(a), g(a), g(g(a)), g(f(a)), \cdots\}$

Die Menge der Grundterme in a, f 1-stellig, g 1-stellig.

- $B \equiv \forall x \exists y[p(f(x), y, g(x, y))]$

$\hookrightarrow H_B \equiv \{a, f(a), g(a, a), f(g(a, a)), g(a, f(a)), \ldots\}$

Das Herbrand-Universum (Forts.)

Definition 5.16 (Herbrand-Interpretation)

Eine **Herbrand-Interpretation** für eine (abgeschlossene) Formel A ist eine Interpretation $I = (H_A, \dots)$, die

a) jeder Individuenkonstante a in A den Term $a \in H_A$ zuordnet.

b) jeder n-stelligen Funktionskonstante f, die in A vorkommt, eine Funktion $I(f) : H_A^n \to H_A$ zuordnet, die die Terme $t_1, \dots, t_n \in H_A$ auf den Term $f(t_1, \dots, t_n) \in H_A$ abbildet.

Der Satz von Herbrand

Satz 5.17

- *Eine Formel A in Klauselform ist erfüllbar* gdw *A ist in einer Herbrand-Interpretation erfüllbar.*
- *A ist unerfüllbar* gdw *A unerfüllbar in allen Herbrand-Interpretationen.*

- Festlegung der Definitionsbereiche der Interpretationen.
  Sei $A \equiv \forall x_1 \cdots \forall x_n [C_1 \wedge \cdots \wedge C_k]$.
  Eine **Grundinstanz** einer Klausel C_j $(1 \leq j \leq k)$ von A ist eine Klausel, die man erhält, wenn man alle Individuenvariablen in C_j durch Terme aus H_A ersetzt, d. h. eine Grundsubstitution $x_i \leftarrow t_i$ auf C_j anwendet. Solche Klauseln, die keine Individuenvariablen enthalten heißen **Grundklauseln**.

Herbrand-Prozeduren

Satz 5.18 (von Herbrand)

Eine Formel A in Klauselform ist genau dann unerfüllbar, wenn es eine endliche Konjunktion von Grundinstanzen ihrer Klauseln gibt, die unerfüllbar ist.

! Konjunktionen von Grundklauseln sind wie aussagenlogische Formeln in KNF zu behandeln.

$\leadsto$ Tableaux-Davis-Putnam-(Grund)-Resolution anwendbar.

Herbrand-Prozeduren (Forts.)

$\hookrightarrow$ **Herbrand-Prozeduren**
A Formel in KLF mit n-Klauseln und Herbrand Universum H_A.

- Erzeuge systematisch alle Grundklauseln G_j aus den Klauseln C_j $(1 \leq j \leq n)$ von A.
- Prüfe, ob die Konjunktion (aller) Grundklauseln unerfüllbar ist.

▶ **Probleme**:

- Systematische Erzeugung der Grundklauseln.
- Auswahl von Grundklauseln, die man auf Erfüllbarkeit testet.
- Welches Verfahren wählt man.

Grundresolventenmethode

- $A \equiv [\forall x \exists y\ p(x,y) \wedge \forall y \exists z\ q(y,z)] \rightarrow \forall x \exists y \exists z$ $(p(x,y) \wedge q(y,z))$
  gilt $\models A$? Transformiere $\neg A$ in KLF
- $B \equiv \forall x \forall y \forall y_1 \forall z [p(x,f(x)) \wedge q(y,g(y)) \wedge$ $(\neg p(a,y_1) \vee \neg q(y_1,z))]$
- $H_B = \{a, f(a), g(a), f(f(a)), f(g(a)), \ldots$
- **Bilde Grundinstanzen**:
  $[p(a,f(a)) \wedge q(a,g(a)) \wedge [\neg p(a,a) \vee \neg q(a,a)]]$
  $\wedge$
  $\vdots$
  $\wedge[p(a,f(a)) \wedge q(f(a),g(f(a))) \wedge [\neg p(a,f(a)) \vee \neg q(f(a),$ $g(f(a)))]$

Grundresolventenmethode (Forts.)

- Teste, ob erfüllbar:
  Grundresolventenmethode

1.	$p(a, f(a))$	
2.	$q(f(a), g(f(a)))$	
3.	$\neg p(a, f(a)) \vee \neg q(f(a), g(f(a)))$	
4.	$\neg q(f(a), g(f(a)))$	1R3
5.	$\square$	2R4

? Wie wählt man die Klauseln und Substitutionen geschickt!

Substitutionen

1.	$p(x, f(x))$	$x \leftarrow a \quad p(a, f(a))$
2.	$q(y, g(y))$	Substitutionen
3.	$\neg p(a, y_1) \vee \neg q(y_1, z)$	$y_1 \leftarrow f(a)$
		$\neg p(a, f(a)) \vee \neg q(f(a), z)$
4.	$\neg q(f(a), z)$	1R3 $\quad x \leftarrow a \quad y_1 \leftarrow f(a)$
5.	$\square$	2R4 $\quad y \leftarrow f(a) \quad z \leftarrow g(f(a))$

Definition 5.19 (Erinnerung)

Eine Substitution θ ist eine endliche Menge der Form $\{\langle v_1, t_1\rangle, \dots, \langle v_n, t_n\rangle\}$, v_i ist Individuenvariable $v_i \neq v_j, t_i$ Terme ($\mathbf{Term}(\mathbb{V}, \mathbb{F})$, $\mathbb{F}$ Funktionskonstanten $t_i \not\equiv v_i$, $i = 1, \dots, n$. $\{v_1, \dots, v_n\}$ ist der Definitionsbereich der Substitution $\langle v_i, t_i\rangle$

- „Bindung" für v_i.
- θ induziert Abbildungen $\theta : \mathbf{Term}(\mathbb{V}, \mathbb{F}) \to \mathbf{Term}(\mathbb{V}, \mathbb{F})$
  bzw. $\theta : \mathbf{AForm} \to \mathbf{AForm}$

  Wie üblich:
  - $\theta(v_i) \equiv t_i \quad i = 1, \dots, n$
  - $\theta(v) \equiv v \quad$ sonst
  - $\theta(f(t_1, \dots, t_n)) \equiv f(\theta(t_1), \dots, \theta(t_n))$
  - $\theta(p(t_1, \dots, t_n)) \equiv p(\theta(t_1), \dots, \theta(t_n))$
  - $\theta(t = t') \equiv \theta(t) = \theta(t')$

Substitutionen (Forts.)

- Komposition von Substitutionen: wie üblich.
  $\theta\sigma$ erst θ, dann σ.
  Identität als Substitution erlaubt.
- Betrachte $t_1 \equiv f(g(x), y)$, $t_2 \equiv f(z, g(a))$
  Frage: Gibt es Substitution θ mit $t_1\theta = t_2\theta$ d.h.

$$\theta(f(g(x), y)) \equiv \theta(f(z, g(a)))$$

- θ heißt dann **Unifikator** von t_1 und t_2.
  $\{x \leftarrow a \quad y \leftarrow g(a) \quad z \leftarrow g(a)\}$ sind
  $\{y \leftarrow g(a) \quad z \leftarrow g(x)\}$ Unifikatoren
  $\rightsquigarrow$ **Unifikationsalgorithmen**
  Allgemeinster Unifikator: σ (MGU) (Most General Unifier).
  Eigenschaft: Ist θ Unifikator, so gibt es τ mit $\theta = \sigma\tau$.

Unifikation

Definition 5.20 (Unifikator)

Sei $S = A_1 \vee \cdots \vee A_n \quad \{A_1, \ldots, A_n\}$ eine Disjunktion (Menge) atomarer Formeln A_j $(1 \leq j \leq n)$. Eine Substitution θ heißt **Unifikator** für S, falls $A_1\theta \equiv A_2\theta \equiv \cdots \equiv A_n\theta$ (insbesondere müssen die Prädikatskonstanten der A_i alle identisch sein).

- Gibt es für S einen solchen Unifikator, dann heißt S **unifizierbar**.
- Ein Unifikator θ heißt **allgemeinster Unifikator** oder **MGU** für S, wenn es für jeden Unifikator σ von S eine Substitution τ gibt, so dass $\sigma = \theta\tau$.

Unifikationsalgorithmen

Satz 5.21

Sei $S = \{A_1, \ldots, A_n\}$ *Menge von atomaren Formeln, dann ist es entscheidbar ob* S *unifizierbar ist und ein MGU* σ *lässt sich berechnen.*

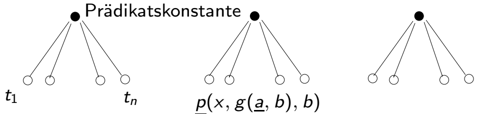
Beweis:

Unifikationsalgorithmen für atomare Formeln in Präfix Notation.

Idee: Bestimme „Disagreement set" durch Tiefensuche.

Unifikationsalgorithmen (Forts.)

Darstellung atomarer Formeln



$\underline{p}(x, g(\underline{a}, b), b)$

$\underline{p}(x, g(\underline{f(y, z)}, x), y)$ $\underline{p}(x, g(\underline{g(h(x), a)}, y), h(x))$

$\hookrightarrow$ DS: $\{\underline{f}(y, z), \underline{a}, \underline{g}(h(x), a)\}$ nicht unifizierbar.

- DS: $\{\dots, \underline{v}, \dots, \underline{t}, \dots\}$, v kommt nicht in t vor.
  Ändere: $\sigma : v \leftarrow t$.
- Berechne DSσ dann weiter bis entweder ein Unifikator bestimmt wurde oder nicht-unifizierbar als Ergebnis vorliegt.

! Es gibt sehr effiziente Unifikationsverfahren (lineare Zeit). Siehe hierfür Literatur.

PL1-Resolutionsverfahren

Definition 5.22 (Allgemeine Resolventenregel)

Seien C_1 und C_2 Klauseln ohne gemeinsame Variablen.
Seien $A_1 \vee \cdots \vee A_k$ und $\neg B_1 \vee \neg B_2 \vee \cdots \vee \neg B_l$ Teildisjunktionen von C_1 bzw. C_2, so dass $A_1, \ldots, A_k, B_1, \ldots, B_l$ unifizierbar sind, mit allgemeinsten Unifikator θ und sei

$$A_i\theta \equiv B_j\theta \equiv p(r_1, \ldots, r_n) \text{ [\textbf{Faktor}]}$$

Die **Resolvente** von C_1 und C_2 ist dann die Klausel $C_1\theta \backslash p(r_1, \ldots, r_n) \cup C_2\theta \backslash \neg p(r_1, \ldots, r_n)$ als Menge von Literalen.

Resolutionsverfahren (Forts.)

Satz 5.23 (Robinson)

Eine Formel A in Klauselform ist genau dann unerfüllbar, wenn die leere Klausel $\square$ aus A mit der Resolventenregel hergeleitet werden kann.

- [Annahme: Klauseln in A sind variablendisjunkt] Immer erreichbar da
  $$\forall \bar{x}[C_1(\bar{x}) \land \cdots \land C_k(\bar{x})] \models\!\!\!\!\;\dashv \forall \bar{x}_1 \forall \bar{x}_2 \cdots \forall \bar{x}_k \; [C_1(\bar{x}_1) \land \cdots \land C_k(\bar{x}_k)]$$
- Beachte allgemeine Resolventenregel $k, l \geq 1$
- Viele Varianten: Unit-Resolution, lineare Resolventenregel....

! Ziel ist es, so schnell wie möglich die leere Klausel herzuleiten, d. h. soviel Literale wie möglich zu „faktorisieren“.

Beispiel

Beispiel 5.24

$A \equiv \neg\exists y \forall z\, [p(z,y) \leftrightarrow \neg\exists x\, [p(z,x) \wedge p(x,z)]]$

- Frage gilt $\models A$?
  $\neg A \equiv \neg\neg\exists y \forall z\, [p(z,y) \leftrightarrow \neg\exists x\, [p(z,x) \wedge p(x,z)]]$

$\hookrightarrow$ KLF($\neg A$)

$$\begin{aligned} \forall z \forall x\, [\quad & [\neg p(z,a) \vee \neg p(z,x) \vee \neg p(x,z)] \wedge \\ & [p(z,f(z)) \vee p(z,a)] \wedge \\ & [p(f(z),z) \vee p(z,a)]] \end{aligned}$$

Beispiel (Forts.)

- Klauseln (nach Umbenennen der Variablen, variablendisjunkt)
  1. $\neg p(z_1, a), \neg p(z_1, x_1), \neg p(x_1, z_1)$
  2. $p(z_2, f(z_2)), p(z_2, a)$
  3. $p(f(z_3), z_3), p(z_3, a)$
- Resolvente von 2. und 1. mit Teildisjunktionen $\neg p(z_1, a), \neg p(z_1, x_1), \neg p(x_1, z_1)$ in 1. und $p(z_2, a)$ in 2. Der Unifikator $\langle x_1, a\rangle, \langle z_2, a\rangle, \langle z_1, a\rangle$ MGU.

- 4. $p(a, f(a))$

Beispiel (Forts.)

- Resolvente 3. und 1. mit Teildisjunktionen
  $\neg p(z_1, a), \neg p(z_1, x_1), \neg p(x_1, z_1),\ p(z_3, a)$
- DS:

$$\left.\begin{array}{lll} \{z_1, z_1, x_1, z_3\} & x_1 \leftarrow z_1 & z_3 \leftarrow z_1 \\ \{a, z_1\} & z_1 \leftarrow a & \end{array}\right\} \begin{array}{ll} x_1 \leftarrow a & z_1 \leftarrow a \\ z_3 \leftarrow a & \end{array}$$

  - 5. $p(f(a), a)$

- Resolvente 4., 1. mit Teildisjunktionen
  $p(a, f(a)), \neg p(x_1, z_1) \qquad x_1 \leftarrow a, z_1 \leftarrow f(a)$

  - 6. $\neg p(f(a), a)$

Beispiel (Forts.)

- Resolvente 5., 6.

  - 7. $\square$

$\hookrightarrow$ Also gilt
$\models A \equiv \neg\exists y \forall z\, [p(z,y) \leftrightarrow \neg\exists x\, [p(z,x) \wedge p(x,z)]]$

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
  $2 \equiv \mathrm{succ}(\mathrm{succ}(0))$

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

Logisches Programmieren und Prolog

- $\text{goal}, \text{add}, \text{subt}, \text{mult}$ 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 $X \leftarrow \text{succ}(\text{succ}(0))$ und $Y \leftarrow \text{succ}(0)$
  $? -\text{goal}(\text{succ}(\text{succ}(0)), \text{succ}(0), Z)$

? $\lightning$

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

Horn-Logik

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

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

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

Horn-Logik

- 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

$\hookrightarrow$ **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. $P \models \exists (B_1 \wedge \cdots \wedge B_m)$

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

Horn-Logik (Forts.)

- Offen ist nur noch die Interpretation der P-Konstanten.

$$\mathcal{R} \leftrightarrow I(\mathcal{R}) = \{r(t_1, \ldots, t_n) \quad \begin{array}{|l} r \quad P\text{-Konstante, } n\text{-stellig} \\ t_1, \ldots, t_n \in H_P \text{ Grundterme} \\ (t_1, \ldots, t_n) \in r_{\mathcal{R}}\} \end{array}$$

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:

- $$M_P = \{r(t_1, \ldots, t_n) \mid \begin{array}{l} t_1, \ldots, t_n \text{ Grundterme und} \\ P \models r(t_1, \ldots, t_n)\} \end{array}$$

- M_P lässt sich rekursiv definieren.

Horn-Logik (Forts.)

Satz 5.25

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!)

Horn-Logik (Forts.)

Beispiel 5.26

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

$$
\begin{array}{ll}
P: & \mathrm{add}(X, 0, X). \\
 & \mathrm{add}(X, \mathrm{succ}(Y), \mathrm{succ}(Z)) : -\mathrm{add}(X, Y, Z).
\end{array}
$$

$\hookrightarrow$ 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) $? -\mathrm{add}(X, \mathrm{succ}^8(0), Z)$
  $(P \overset{?}{\models} \exists X \exists Z \; \mathrm{add}(X, \mathrm{succ}^8(0), Z))$
  JA $\left.\begin{array}{l}\text{mit } X = 0, Z = \mathrm{succ}^8(0) \\ \text{mit } X = \mathrm{succ}(0), Z = \mathrm{succ}^9(0) \dots\end{array}\right\} Z = \mathrm{succ}^8(X)$
  ist allgemeinste Lösung.

Wie werden existentielle Anfragen beantwortet? (Forts.)

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

  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)

Wie werden existentielle Anfragen beantwortet? (Forts.)

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

Lösungssubstitutionen:

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

Lösungssubstitutionen

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

$\sigma = \{X_1 \leftarrow t_1, \dots, X_n \leftarrow t_n\}$ ist eine **korrekte Lösungssubstitution** für $P \cup \{G\}$ gdw $X_1, \dots, X_n$ 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 $?-A_1, \dots, A_m$ und $C \equiv A :- \ B_1, \dots, B_q$ 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
- $G' \equiv ?- A_1\mu, \dots, A_{k-1}\mu, B_1\mu, \dots, B_q\mu, A_{k+1}\mu, \dots, A_k\mu$ ist eine **SLD-Resolvente** von G und C über μ.

$$\begin{tikzcd} G \arrow[d, dash] & C \arrow[dl, dash] \\ G' & \end{tikzcd}$$

SLD als Regel

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

SLD-Resolution: SLD-Ableitungen

- **SLD-Ableitungen**, wie üblich definieren:
  SLD-Ableitungen sind Ableitung der Form

  $G_0, G_1, \dots, G_n, \dots$ Programmformeln $C_0, C_1, \dots, C_n, \dots$
  Substitutionen $\mu_0, \mu_1, \dots, \mu_n, \dots$, 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
  $G_0, G_1, \dots, G_n, C_0, \dots, C_n, \mu_0, \dots, \mu_n$ vor).

Lösungssubstitution

Definition 5.27 (Lösungssubstitution)

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.

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

Bemerkung und Beispiel

Bemerkung 5.28

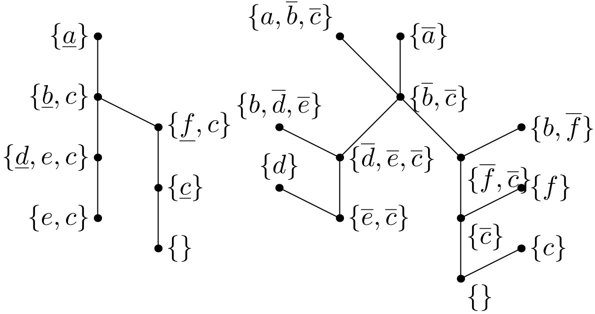
- *Korrektheit und Vollständigkeit lassen sich beweisen!*

Siehe etwa Leitsch: Resolutionskalküle

Beispiel 5.29

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

Beispiel: Goal ? - a



Operationale Semantik von PROLOG

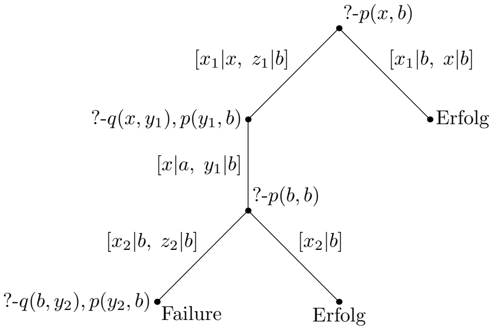
- **Prolog: Logik + Kontrolle**
- Fixiere Reihenfolge der SLD-Schritte
  - Ordne Programmformeln (Liste)
  - Goals als Listen - erstes Literal
  - Bilde stets Resolventen mit Kopf der ersten Programmformel mit erstem Literal des Goals, **normale SLD-Resolution**.
  - **Probleme**: Laufzeiten sind abhängig 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 (Forts.)

- **Umordnung der Literale in Programmformeln**

$$\begin{array}{ll} 1 & p(X,Z) :- p(Y,Z), q(X,Y). \\ 2 & p(X,X). \\ 3 & q(a,b). \\ \multicolumn{2}{l}{G \equiv ?-\ p(X,b)} \end{array}$$

$\longrightarrow$ Depth First $\rightsquigarrow$ kein Ergebnis.

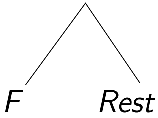
$\rightsquigarrow$ Umordnung des SLD-Baums

Beispiel 5.30 (Listen über Σ)

Gegeben Signatur Σ definiere Listen über Σ:

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



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

Beispiel: Listen über Σ (Forts.)

- **Operationen auf Listen**
  - append([ ], M, M).
  - append([$X|L$], M, [$X|N$]) :− append(L, M, N).

    ? −append([a, b], [a, Y], Z)

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

$\hookrightarrow$ **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.