

Deduktive Datenbanksysteme

Problem: Transitiver Abschluss ist in PL1 nicht formulierbar (mit zustandsabhängiger Formulierung möglich)

Diskussion:

Typ 5: $q_1(\dots), \dots, q_n(\dots) : -.$

Typ 6: $q_1(\dots), \dots, q_n(\dots) : -p_1(\dots), \dots, p_m(\dots).$

\Rightarrow Übungsaufgabe

Nur Typ 1 und Typ 4: $q(\dots) : -p_1(\dots), \dots, p_n(\dots), n \geq 0$ (ist die Hornklauselform und wird bei definiten Datenbanken genutzt)

Definite Datenbanken

$$\begin{aligned} q(\dots) &: -. \text{(Fakt)} \\ q(\dots) &: -p_1(\dots), \dots, p_n(\dots). \text{(deduktive Regel, } p_{1-n} \text{ Teilziele)} \end{aligned} \tag{1}$$

- Mit IBen (Integritätsbedingungen) (+ Typ2, Typ3) ($: -p_1(\dots), \dots, p_n(\dots)$)
- Typ5 + Typ6 \Rightarrow **Disjunktive Datenbank**
- Definite Datenbank + negative Atome im Rumpf von Hornklauseln erlaubt
 \Rightarrow Volles Datalog

Formulierung von Anfragen

Klauseln vom Typ: $: -p_1(\dots), \dots, p_n(\dots)$, geschrieben $? - p_1(\dots), \dots, p_n(\dots)$

Beispiele:

- $? - ag(X, m).$
 - Bedeutung: Welche Kurse bietet 'm' an?
 - DRC: (x) / ANGEBOT(X, m)
- $? - ag(a3, m).$
 - Bedeutung: Bietet 'm' den Kurs 'a3' an?
 - DRC: () / ANGEBOT(a3, m)
- $? - ag(X, m), bl(X, s, j).$

- Bedeutung: Gib alle von 'm' angebotene Kurse, die 's' als Wiederholer belegt hat
- DRC: $(x) / \text{ANGEBOT}(x, 'm') \wedge \text{BELEGUNG}(x, 's', 'y')$
- Wie ist $(x) / \text{ANGEBOT}(x, 'm') \wedge (\exists y) \text{BELEGUNG}(x, 's', y)$ formulierbar?
 - Bedeutung: Gib die Dozenten der von s als Wiederholer belegte Kurse
 - Formulierung: $? - Ksm(X)$.
 $Ksm(X) : -ag(X, m), bl(X, s, y)$
 - Bequemer: $? - ag(X, Y*), bl(X, s, y)$, * Kennzeichnet die Ausgabevariable

In Anfragesprachen werden Vergleichsausdrücke benötigt. Dazu sind in Datalog spezielle vordefinierte Prädikate vorhanden. Für jeden Vergleichsoperator wird die Existenz eines solchen Prädikates angenommen.

Zunächst: Beschränkte Variablen in Regeln. Sei eine Regel r gegeben:

- Jede Variable, die als Argument in einem gewöhnlichen Prädikat im Rumpf von r vorkommt ist beschränkt.
- Jede Variable, die in einem Teilziel $X = c$ oder $c = X$ von r vorkommt, ist beschränkt.
- Eine Variable X ist beschränkt, wenn sie in einem Teilziel $X = Y$ oder $Y = X$ von r vorkommt mit Y ist schon als beschränkt bekannt.

Definition: sicher

Eine Regel heißt sicher, wenn alle in ihr vorkommenden Variablen beschränkt sind.

Beispiele:

- $Kls(X, Y) : -bl(Z, s, j), ag(Z, Y), X = Z$. **sicher**
- $vsj(X, Y) : -bl(Y, s, j)$. **nicht sicher** (X ist nicht beschränkt)
- $vs(X, Y) : -vs(X, Z), kp(Z, Y)$. **sicher, wenn vs terminiert**
- $kla(Z, Y) : -bl(Z, V, j), ag(Z, Y), V \neq s$. **sicher**

Bemerkung: Falls keine Build-in Prädikate erlaubt sind (/vorkommen): Eine Regel ist sicher genau dann wenn jede Variable im Kopf der Regel auch im Rumpf der Regel vorkommt.

Definition: Datalog Programm

Ein Datalog-Programm P (ohne IBen(Integritätsbedingungen)) ist eine endliche Menge von Horn-Klauseln mit Jedes $d \in P$ ist entweder

- ein Fakt $q(\dots)$. ohne Variable
- eine sichere Regel $q(\dots) : -p_1(\dots), \dots, p_n(\dots)$. mit $q \in iPraedikat$

Ein $d \in P$ heißt auch **Datalog-Klausel** Alle Fakten zu extensionalen Prädikaten sind als in DB-Relationen gespeichert zu denken.

Beispiel Datenbankzustand:

$$\begin{aligned}
 & ag(a1, m). \\
 & kp(c2, a0). \\
 & \dots \\
 & rb(a1, r1, t1) \quad (Kurs a1 im Raum r1 zu t1) \\
 & rb(a3, r2, t4)
 \end{aligned} \tag{2}$$

Angebot:

Kursnummer	Dozent
a1	m
...	...

Kursplan:

Kursnummer	Voraussetzung
c2	a0
...	...

Raumbelegung:

Kursnummer	Raum	Zeit
a1	r1	t1
...

Belegung:

Kursnummer	Teilnehmer	Wiederholer
a1	s	j
...

$$\begin{aligned}
 & vs(X, Y) : -Kp(X, Y). \\
 & vs(X, Y) : -vs(X, Z), Kp(Z, Y). \\
 & stdpl(W, X, Y, Z) : -bl(X, W, V), rb(X, Y, Z). \\
 ueberschneidungen(X, Y) : & -Kp(Z, X), Kp(Z, Y), rb(X, V, T), rb(Y, W, T), X \neq Y. \\
 & \quad (3)
 \end{aligned}$$

Deklarative Semantik

Extensionale Prädikate eines Programms (ext. Rel, ext. DB): EDB
Intentionale Prädikate eines Programms (int. Rel, int. DB): IDB

Bedeutung Bedeutung eines Datalog-Programms P: Menge derjenigen Grundatome zu den intentionalen Prädikaten von P, die logisch aus P gefolgert werden können.
(Jedes Modell von P ist auch ein Modell von $f \in F$). Mit Zielklausel $g(\dots)$, $g \in Praed$: Aus P logisch folgerbare Grundatome zu g, die von $g(\dots)$ subsummiert (überdeckt) werden. $\frac{g(a,X)}{g(a,b)}$ $\frac{g(a,X)}{g(a,c)}$

In P werden Werte aus Wertebereichen verwendet, ebenso in Darstellung der extensionalen Prädikate als DB-Relation. Daher können wir $Kost_A$ und Dom idenfifizieren. Mithilfer der Herbrand Interpretation kann die Semantik festgelegt werden (ist möglich).

Herbrand-Interpretation /-Modelle Gewöhnliche Interpretation:

$$\begin{aligned}
 Konst_A &= \{a, b\}, Dom = \{\circ, \square\} \\
 k(a) &= \circ \\
 k(b) &= \square \\
 ext(p(\cdot, \cdot)) &= \{(\circ, \square), (\square, \square)\}
 \end{aligned} \tag{4}$$

eine mögliche Herbrand-Interpretation (passt dazu)

$$\begin{aligned}
 Konst_A &= Dom = \{a, b\} \\
 k(a) &= a \\
 k(b) &= b \\
 ext(p(., .)) &= \{(a, b), (b, b)\}
 \end{aligned} \tag{5}$$

Entsprechende Herbrand-Interpretation. Betrachte alle Paare zu $p(., .)$, teste gemäß gegebener (gewöhnlicher) Interpretation in $ext(p(., .))$.

$$\begin{aligned}
 Konst_A &= \{a, b\}, Dom = \{\circ, \square\} \\
 k(a) &= \square \\
 k(b) &= \circ \\
 ext(p(., .)) &= \{(\circ, \square), (\square, \square)\}
 \end{aligned} \tag{6}$$

(a,a) wird zu $(\square, \square) \in ext(p(., .))$

Herbrand-Interpretation

$$\begin{aligned}
 Konst_A &= Dom = \{a, b\} \\
 k(a) &= a \\
 k(b) &= b \\
 ext(p(., .)) &= \{(a, a), (a, b), (b, a), (b, b)\}
 \end{aligned} \tag{7}$$

Bei beiden Interpretationen sind die gleichen Formeln gültig bei Beschränkung auf quantorenfreie Formeln ohne Variablen (und ohne Funktionen).

Beispiel Erste Interpretation: $p(a, b) \wedge p(b, a) \Rightarrow (\square, \square) \in ext(p(., .)) \wedge \dots$ bzw.
 $(a, b) \in ext(p(., .)) \wedge (b, a) \in ext(p(., .))$

Menge von Konstanten und Prädikatensymbole ist endlich, daher ist die Anzahl der möglichen Herbrand-Interpretationen endlich.

Satz von Gödel / Skolem (Herbrand, 1930)

Eine Klauselmenge P hat ein Modell genau dann wenn P hat ein Herbrand-Modell. Daraus folgt, dass ein Verfahren analog zu Wahrheitstabellen in der Aussagenlogik möglich ist.

Beispiel $F = \{p(a) \Rightarrow q(b), p(a) \wedge q(b)\}, q(b)?$

p	q	$p(a) \Rightarrow q(b)$ erfüllt?	$p(a) \wedge q(b)$ erfüllt?	$p(a) \Rightarrow q(b)$ und $p(a) \wedge q(b)$ erfüllt?
{}	{}	✓	-	-
{}	{a}	✓	-	-
{}	{b}	✓	✓	✓
{}	{a, b}	✓	✓	✓
{a}	{}	-	✓	-
...
{b}	{b}	✓	✓	✓

Jedes Modell von F ist auch ein Modell von q(b), d.h. q(b) kann aus F logisch gefolgert werden. Gilt bei Klauselmengen, aber **Vorsicht bei allgemeinen Formeln**.

Beispiel: $\{p(a), (\exists X)(\neg p(X))\}$ Formelmenge, keine Klauselmenge
Modell (vgl. Übung):

$$\begin{aligned} Dom &= \{0, 1\} \\ k(a) &= 0 \\ ext(p(.)) &= \{(0)\} \end{aligned} \tag{8}$$

Aber: Es gibt kein durch `ext` bestimmtes Herbrand-Modell:

1. $ext(p(.)) = \{(a)\}, Konst = Dom = \{a\}$
2. $ext(p(.)) = \{\}$

Herbrand-Modell muss genügend viele Elemente enthalten, damit der Satz von Gödel / Skolem gelten kann. **Skolemisierung** bedeutet, dass man alle Existenzquantoren durch Funktionen ersetzt:

$$(\forall x_1, \dots, x_n)(\exists y)(F) \rightsquigarrow (\forall x_1, \dots, x_n)(F[f(x_1, \dots, x_n)/y]) \tag{9}$$

Bemerkung: Skolemisierung

Jede Formel der PL1 Logik kann man in einer erfüllbarkeitsäquivalente Formel in Skolemform umformen:

1. Pränexnormalform

2. Umformungen à la $(\forall x_1, \dots, x_n)(\exists y)(F) \rightsquigarrow (\forall x_1, \dots, x_n)(F[f(x_1, \dots, x_n)/y])$ mit jeweils einem neuen Funktionssymbol

Dies ist eine Art ‘‘Materialisierung’’ der durch den Existenzquantor gebundenen Variablen.

Beispiel (von oben) $\{p(a), \neg p(y)^1\}$ erfüllbar $\iff \{p(a), (\exists X)(\neg p(X))\}$ erfüllbar

Vorsicht: Semantische Äquivalenz von Formeln und ihren Skolem-Normalformen im Allgemeinen nicht gegeben.

Skolem-NF: $(\exists X)(p(X)) : p(a)$ Interpretation I mit

$$\begin{aligned} Dom &= \{1, 2\} \\ k(a) &= 1 \\ ext(p(.)) &= \{(2)\} \end{aligned} \tag{10}$$

$\rightsquigarrow \models_I (\exists X)(p(X))$ Belegung von X mit allen Elementen aus Dom, d.h. auch mit 2. Aber $\not\models_I p(a)$, da $(1) \notin ext(p(.))$.

Im Kontext von Datalog

- Herbrand-Universum: Konst
- Herbrand-Basis (HB): Menge aller Grundatome
($\sim EDB^2, IDB^3$ (Fakten in der Datenbank und solche die Ableitbar sind) $\subset HB$)
- Herbrand-Interpretation: (Konst, id, ext), d.h. jedes Konstantensymbol wird als es selbst interpretiert. (Verglichen mit relationaler Interpretation, dort k ist Bijektion)
- Jede Herbrand-Interpretation ist eindeutig bestimmt durch ext (Extension, Ausprägung), da Konst und id unveränderlich sind
- Jedes Prädikat ist eindeutig bestimmt durch die Angabe der Grundatome, für die es ‘‘wahr’’ liefert.
- extentional: genaue Antwort in Tupel
- intentional: Beispielsweise Formeln als Antworten

¹neue Variable

²gegeben durch Datenbankzustand

³muss ausgerechnet / gefolgert werden

Definition: Herbrand-Interpretation

Einfache Definition: Eine Herbrand-Interpretation ist eine Teilmenge der Herbrand-Basis.

Beispiel Kursverwaltung

$$\begin{aligned} \text{Aus DB-Rel. KURSPLAN } & \rightsquigarrow kp \in ePräd \\ vs(X, Y) : -kp(X, Y) & \rightsquigarrow vs \in iPräd \end{aligned} \tag{11}$$

Gesucht: Durch Programm P bestimmte Grundatome (Fakten).

Definition: Grundatom

Ein Grundatom f ist eine logische Folgerung einer Menge D von Datalog Klauseln (z.B. $D \models f$) \diamondsuit_{Def} . Jedes Herbrand Modell von D ist auch ein Modell von f.

Da f ein Grundatom ist gilt $D \models f \implies f$ ist in jedem Herbrand-Modell von D enthalten. Das heißt $f \in \bigcap\{I | I \text{ Herbrand-Modell von } D\}$.

Sei $f \in \bigcap\{I | I \text{ Herbrand-Modell von } D\}$, dann ist f ein Grundatom und jedes Modell von D auch in Modell von f.

Definition: Menge aller Konsequenzen

$$cons(D) =_{def} \{f \in HB_D | D \models f\}$$

Satz 1.1

$$cons(D) = \bigcap\{I | I \text{ Herbrand-Modell von } D\}$$

Aufgrund der Eigenschaften unserer Regel ist der Schnitt / $cons(D)$ ein Modell, dies gilt es zu beweisen:

Da $cons(D) \subseteq HB_D$, ist $cons(D)$ eine Herbrand-Interpretation.

Satz 1.2

$cons(D)$ ist ein Herbrand-Modell von D.

Beweis

z.Z. Jedes $d \in D$ ist gültig in dieser Interpretation, also $\text{cons}(D)$ ($\models_{\text{cons}(D)} d$).

Beweis Falls d ein Grundatom ist gehört d zu jedem Herbrandmodell von D , daraus folgt $d \in \text{cons}(D)$.

Sei d eine Regel $q(\dots) : -p_1(\dots), \dots, p_m(\dots)$. Sei ϱ eine Belegung für die Variable von d . Annahme: $(\forall 1 \leq i \leq m)(||p_i(\cdot)||^\varrho \in \text{cons}(D))$, sonst d gültig unter $\text{cons}(D)$. Für jedes Herbrand-Modell I von D gilt, dann $||p_i(\cdot)||^\varrho \in I, i = 1, \dots, m$ und damit $||q(\cdot)||^\varrho \in \text{cons}(D)$, da d eine Horn-Klausel ist. (Der Schluss ist z.B. für $d = q_1(\dots), q_2(\dots) : -p_1(\dots), \dots, p_n(\dots)$ nicht möglich.)

Damit $D \models_{\text{cons}(D)} ||q(\dots)||^\varrho \in \text{cons}(D)$. Wir erhalten, dass $\text{cons}(D)$ ein Herbrand-Modell von D ist. Da d beliebig gewählt wird folgt der Satz.

$\text{cons}(D)$ ist offensichtlich eindeutig bestimmt und das kleinste Herbrand-Modell von D . Damit: Seantik eines Datalog-Programms P ist gegeben durch das kleinste Herbrand-Modell von P oder (äquivalent) durch $\text{cons}(P) = \{f \in HB_P \mid O \models f\}$.

Beispiel $r \in ePräd, p, q \in iPräd, Konst = \{1, 2, 3\}$.

Fakten:

$$\begin{aligned} r(1). \\ p(X) : -q(X). \\ q(X) : -r(X). \end{aligned} \tag{12}$$

Herbrand-Modelle:

$$\begin{aligned} &\{r(1), p(1), q(1)\} \\ &\{r(1), p(1), q(1), q(2), p(2)\} \\ &\{r(1), p(1), q(1), q(2), p(2), p(3)\} \end{aligned} \tag{13}$$

Fixpunkt-Semantik

Deklarative Semantik liefert keinen brauchbaren Algorithmus. Die Fixpunkt-Semantik führt direkt zu einem Algorithmus für die Schrittweise Berechnung des kleinsten Herbrand-Modells.

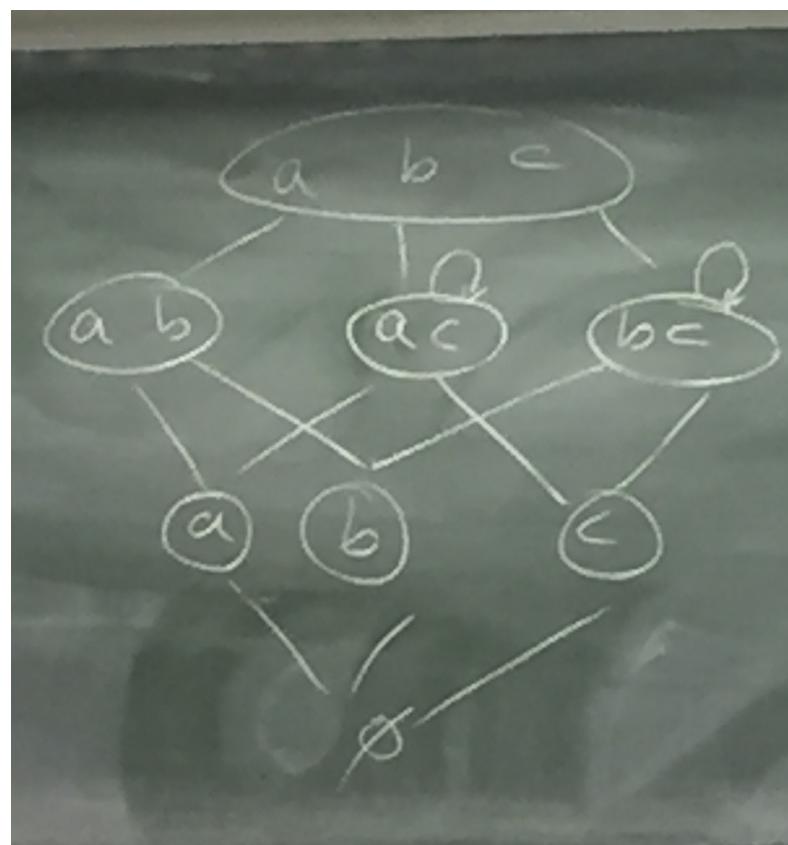


Figure 1:

$$\begin{aligned}
 c &\in ac \\
 \tau(c) &\subseteq \tau(ac) \\
 \tau(c) &= c! \\
 \tau(c) &= ac
 \end{aligned} \tag{14}$$

$$\begin{aligned}
 c &\in bc \\
 \tau(c) &\subseteq \tau(bc) \\
 \tau(c) &= c! \\
 \tau(c) &= bc
 \end{aligned} \tag{15}$$

Dies ist ein Widerspruch

P Datalog-Programm τ_P liefert Fakten von P vereinigt mit Fakten, die in einem Schritt aus der Argumentenmenge von τ_P mit den Regeln von P abgeleitet werden können.

Ableitung in einem Schritt

Definition: Substitution

Eine Substitution ist eine endliche Menge der Form

$$\{X_1/t_1, \dots, X_n/t_n\}, X_1, \dots, X_n \text{ unterschiedliche Variablen, } t_1, \dots, t_n \text{ Terme, } X_i \neq t_i \tag{16}$$

Falls alle t_i Konstanten sind, ist dies eine **Grundsubstitution**.

Sei θ eine Substitution, t ein Term (Variable oder Konstante), so gilt

$$t\theta =_{def} \begin{cases} t_i, & \text{falls } t/t_i \in \theta \\ t, & \text{sonst} \end{cases} \tag{17}$$

Sei L ein Literal, $L\theta$ bezeichnet dasjenige Literal, das aus L entsteht, indem alle Variablen X_i mit L_i für die $X_i/t_i \in \theta$ gilt, simultan durch t_i ersetzt werden. Analog $d\theta$ für eine Datalog-Klausel d.

Beispiel

$$L = p(X, Y, a), \theta = \{X/a, Y/X\} L\theta = p(a, X, a) \quad (18)$$

Sei D eine Menge von Datalog Klauseln. In einem Schritt aus D ableitbare Menge von Fakten:

$$fakt_1(D) = \{f \in HB_D \mid (\exists RegelL_0 : -L_1, \dots, L_n)(\exists Sicht\theta)(\{L_1\theta, \dots, L_n\theta\} \subseteq D \wedge f = L_0\theta)\} \quad (19)$$

(Annahme: built-in Prädikate geeignet berücksichtigt \Rightarrow Algorithmus FAKT_1)

Definiere $\tau_P | 2^{HB} \Rightarrow 2^{HB}$. $\tau_P(I) =_{def} P_F \cup fakt_1(P_R \cup I)$. P_f = Menge der Fakten von P. P_P Menge von Regeln von P.

Satz 1.3

Für jedes Datalog-Programm P ist τ_P eine monotone Transformation auf $(2^{HB}, \subseteq)$.

Beweis

Seien $I_1, I_2 \in HB_P$ mit $I_1 \subseteq I_2$. z.Z.: $\tau_P(I_1) \subseteq \tau_P(I_2)$.

Sei $f \in \tau_P(I_1)$. Falls $f \in P_F$, dann gilt auf $f \in \tau_p(I_2)$. $f \in fakt_1(P_R \cup I_1)$ da $I_1 \subseteq I_2$, gilt $P_R \cup I_1 \subseteq P_R \cup I_2$ und somit $f \in fakt_1(P_R \cup I_2)$ wg. Monotonie von Fakt_1 auf $(2^{HB}, \subseteq)$.

Beispiel

$$\begin{aligned} P &= p(1) \\ &= p(2). \\ &= r(1). \end{aligned} \quad (20)$$

$$\begin{aligned} q(X) &: -s(X), r(X). \\ s(X) &: -p(X). \end{aligned}$$

$$\rightsquigarrow^{T_p(\emptyset)}$$

$$\begin{aligned} p_1 \\ p_2 \\ r_1 \end{aligned} \quad (21)$$

$\rightsquigarrow^{T_p} (\dots)$

$$\begin{array}{ll}
 p_1 \\
 p_2 \\
 r_1 \\
 s(1)(\text{fakt_1}) \\
 s(2)(\text{fakt_1})
 \end{array} \tag{22}$$

$\rightsquigarrow^{T_p} (\dots)$

$$\begin{array}{ll}
 p_1 \\
 p_2 \\
 r_1 \\
 s(1) \\
 s(2) \\
 q(1)(\text{fakt_1})
 \end{array} \tag{23}$$

Satz 1.4

Für jedes Datalog-Programm P gilt $\text{cons}(P) = \text{lf}_p(\tau_p)$ (lf = least fixpunkt).

Beweis

:

1) $\text{cons}(P)$ ist ein Fixpunkt von τ_P .

$\tau_p(\text{cons}(P)) = P_F \cup \text{fakt_1}(P_R \cup \text{cons}(P))$. $\text{cons}(P)$ ist kleinstes Herbrand-Modell von P, d.h. alle Regeln sind gültig unter $\text{cons}(P) \Rightarrow \text{fakt_1}(P_R \cup \text{cons}(P)) = \text{cons}(P) \setminus P_F$. (und kein Fakt kann aus $\text{cons}(P)$ entfernt werden, ohne dass die Modelliereigenschaft verloren geht.) $\Rightarrow \tau_P(\text{cons}(P)) = P_F \cup \text{cons}(P) \setminus P_F = \text{cons}(P)$

2) $\text{cons}(P)$ ist minimaler Fixpunkt von τ_P .

Annahme: $(\exists I \not\subseteq \text{cons}(P))(\tau_P(I) = I)$.

$\text{cons}(P)$ ist minimales Herbrand-Modell $\Rightarrow I$ ist kein Herbrand-Modell. Da $P_F \subseteq I$ wg. Annahme $\tau_P(I) = I$ folgt mindestens eine Regel ist nicht erfüllt, d.h.

$(\exists h_0, \dots, h_n \in P_R)(\exists \text{Substitution} \theta)(\{h_1\theta, \dots, L_n(\theta)\} \subseteq I \wedge L_0\theta \in I)$. Aber $L_0 \in \text{fakt}_1(P_R \cup I)$ nach Definitio von fakt_1. Da auch $L_0\theta \in P_F$ wegen $P_F \subseteq I$, folgt $\tau_P(I) \neq I$. Noch Fixpunkttheorem (Kuaster / Tarski) ist minimaler Fixpunkt von τ_P kleinstter Fixpunkt von τ_P .

Vorgehensweise bei Fixpunktberechnung

bottom-up

Bezeichnung: Forward-chaining

Da " \Leftarrow " die natürliche Richtung für die Anwendung von Regeln ist. Da Regeln sicher sind ist $L_i\theta$ stets ein Fakt θ_i . Betrachte Eignung der Methode zur Anfrageauswertung, z.B. ?- vs(c4, y). "Alle Kurse, die Voraussetzung von Kurs c4 sind"

Ineffizient Berechnung des l_f_p und anschließende Suche zu $vs(c4, y)$ passende Fakten \Rightarrow Vermeide möglichst Berechnung nicht relevanter Fakten (später).

Starte mit Zielklausel, Suche ein Regel die zu einem Atom der Klausel passt. Ersetze das Atom durch angepassten Rumpf der gewählten Regel, neue Anfrage, iterieren.

Bezeichnung: Backward-chaining

Beispiel

$$\begin{aligned} P &= p(1) \\ &= p(2). \\ &= r(1). \end{aligned} \tag{24}$$

$$\begin{aligned} q(X) &: -s(X), r(X). \\ s(X) &: -p(X). \end{aligned}$$

Prozedurale Semantik

Beweistheoretische Sicht, übertragen auf Datalog Programm P aufgefasst als Theorie 1. Stufe

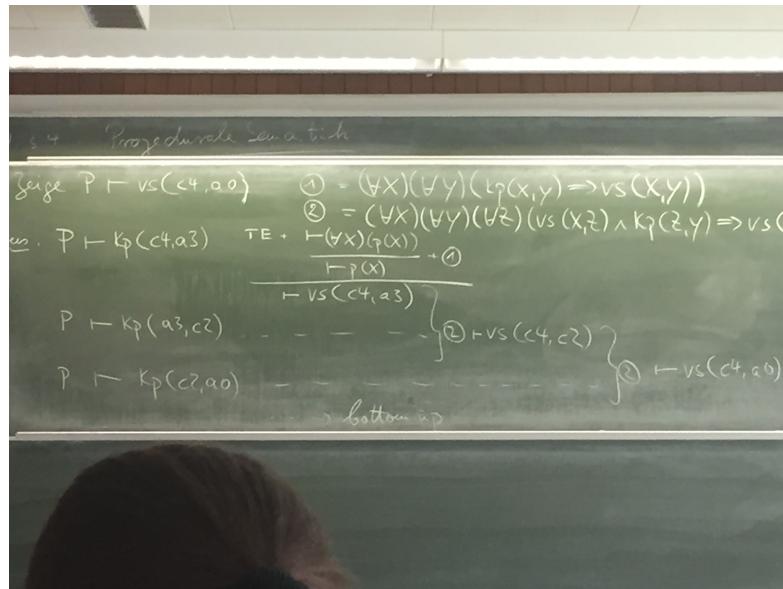


Figure 2:

Semantik: $\{f \in HB_P \mid P \vdash f\}$

Wie können Ableitungen (Beweise) von Fakten systematisch durchgeführt werden?

Regeln:

$$\begin{aligned}
 (1) & vs(X, Y) : -Kp(X, Y). \\
 (2) & vs(X, Y) : -vs(X, Z), kp(Z, Y).
 \end{aligned} \tag{25}$$

$$P = \{Kp(c4, a3), \dots, kp(c2, a0), (1), (2)\}$$

Zeige $P \vdash vs(c4, a0)$.

$$\begin{aligned}
 (1) &= (\forall X)(\forall Y)(kp(X, Y) \Rightarrow vs(X, Y)) \\
 (2) &= (\forall x)(\forall Y)(\forall Z)(vs(X, Z) \wedge kp(Z, Y) \Rightarrow vs(X, Y))
 \end{aligned} \tag{26}$$

Beweis

Idee

- Erzeuge alle Beweismuster (Suchbäume)
- Suche Belegungen für die Variable, so dass alle Blätter zu Fakten aus P werden
- jede solche Belegung erzeugt einen Beweisbaum.

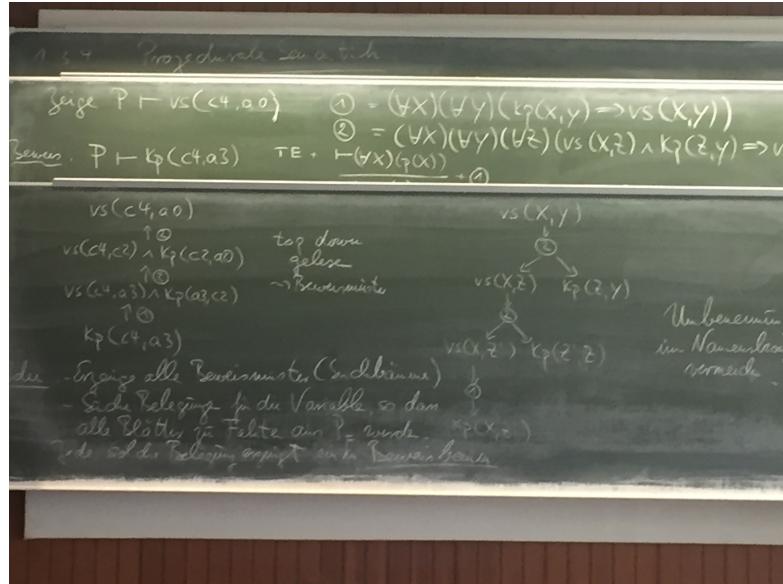


Figure 3:

Zunächst Suche nach “passenden” Köpfen von Regeln erfordert Definition.

Definition: Unifizierbar

Seien L_1 und L_2 heißen **unifizierbar**, wenn $(\exists \text{ Substitution } \Theta)(L_1\Theta = L_2\Theta)$. Θ heißt dann **Unifikator**.

Beispiel $L_1 = vs(X, Z)$

$L_2 = cs(X, Y)$

$\Theta = \{Z/Y\}$ und $\Theta = \{Y/Z\}$ sind Unifikatoren von dem Paar L_1, L_2 , aber auch $\Theta = \{X/a, Y/a, Z/a\}$. Die ersten beiden sind spezifischer als das letzte

$L_1 = q(X, Y, c) L_2 = q(W, b, Z)$ Unifikation z.B.:

$$\begin{aligned}\Theta_1 &= \{X/WY/bZ/c\} \\ \Theta_2 &= \{X/T, W/T, y/b, Z/c\} \\ \Theta_3 &= \{X/a, W/a, Y/b, Z/c\}\end{aligned}\tag{27}$$

Nicht unifizierbar: $L_1 = q(X, c, U) L_2 = q(W, G, Z)$ oder $L_1 = q(X, a, X) L_2 = q(b, Y, Y)$

Definition: Komposition

Sei $\Theta = \{X_1/t_1, \dots, X_n/t_n\}$, $\varsigma = \{Y_1/n_1, \dots, Y_m/n_m\}$ Substitutionen.
Die Komposition $\Theta\varsigma$ von Θ und ς erhält man aus

$$X_1/t_1\varsigma, \dots, X_m/t_m\varsigma, Y_1/n_q, \dots, Y_m/n_m \quad (28)$$

Durch Streichen von Elementen der Form Z/Z sowie Y_i/n_i mit $Y_i = X_j$ für ein $j, j \in \{1, \dots, n\}$

Beispiel $\theta = \{X/a, Y/W\}\varsigma = \{X/bmY/V, W/Z\}$
 $\Theta\varsigma = \{X/a, YZ, W/Z\}$

Definition: allgemeinere Substitution

Seien Θ, ς Substitutionen, Θ heißt allgemeiner als $\varsigma \diamond (\exists \text{Substitution } \delta) (\Theta\delta = \varsigma)$.
Seine L_1, L_2 Literale. Ein allgemeinster Unifikator (mgu) von $L_1 \text{ in } L_2$ ist ein Unifikator von $L_1 \text{ und } L_2$, der allgemeiner als alle anderen Unifikatoren ist.

Beispiel Θ_2 ist allgemeiner als Θ_3 ; betrachte $\delta = \{T/a\}$, es gilt $\Theta_2\delta = \Theta_3$.
 Θ_1 ist allgemeiner als Θ_2 , $\delta = \{W/T\}$. mgu ist i.A. nicht eindeutig bestimmt.
 $L_1 = p(X, X)L_2 = p(V, W)$ mgu:

$$\begin{aligned} & \{X/W, V/W\} \\ & \{X/V, W/V\} \\ & \{V/X, W/X\} \end{aligned} \quad (29)$$

Beispiel $L_1 = q(X, Y, c)^4L_2 = q(W, b, Z)^5$

Suchbaum

(Beweismuster zu einem Programm P)

- Wurzel ist mit einem “Ziel” $g = p(t_1, \dots, t_n), p \in iPr \cdot \cdot \cdot ad$, bekannt.

⁴t1, t2, t3

⁵k1 k2 k3

Figure 4:

- Knoten entlang eines Pfades von der Wurzel aus sind abwechselnd mit Atome und Regeln benannt. (Ziel-, Regelknoten)
- Alle Blattknoten sind mit Atomen benannt
- Sei k ein mit einem Atom $a(s_1, \dots, s_l)$ benannten Knoten (keine Blattknoten), dann ist der unmittelbare Nachfolger von k mit einer Regel von P benannt, deren Kopf mit $q(sq, \dots, s_l)$ unifizierbar ist
- sei k' ein mit einer Regel $r = L_0 : -L_1, \dots, L_m$ benannte Knoten der unmittelbare Vorgänger von k' sei mit $q(s_1, \dots, s_l)$ benannt. Dann hat k' m unmittelbare Nachfolger, die mit $I_1mgu(q(s_1, \dots, s_l), I_0), \dots, I_m mgu(q(s_1, \dots, s_l), I_0)$ benannt sind. Dabei sind I_0, \dots, I_m Atome, die aus L_1, \dots, L_m durch eventuelle Umbenennung von Variablen hevorgehen (die Variablen seien stets so umbenannt, dass sie im Baum eindeutig sind).

Beispiel

Suchbaum → Beweisbaum

Gegeben Sei der Suchbaum S und eine Grundsubstitution Θ mit folgenden Eigenschaften:

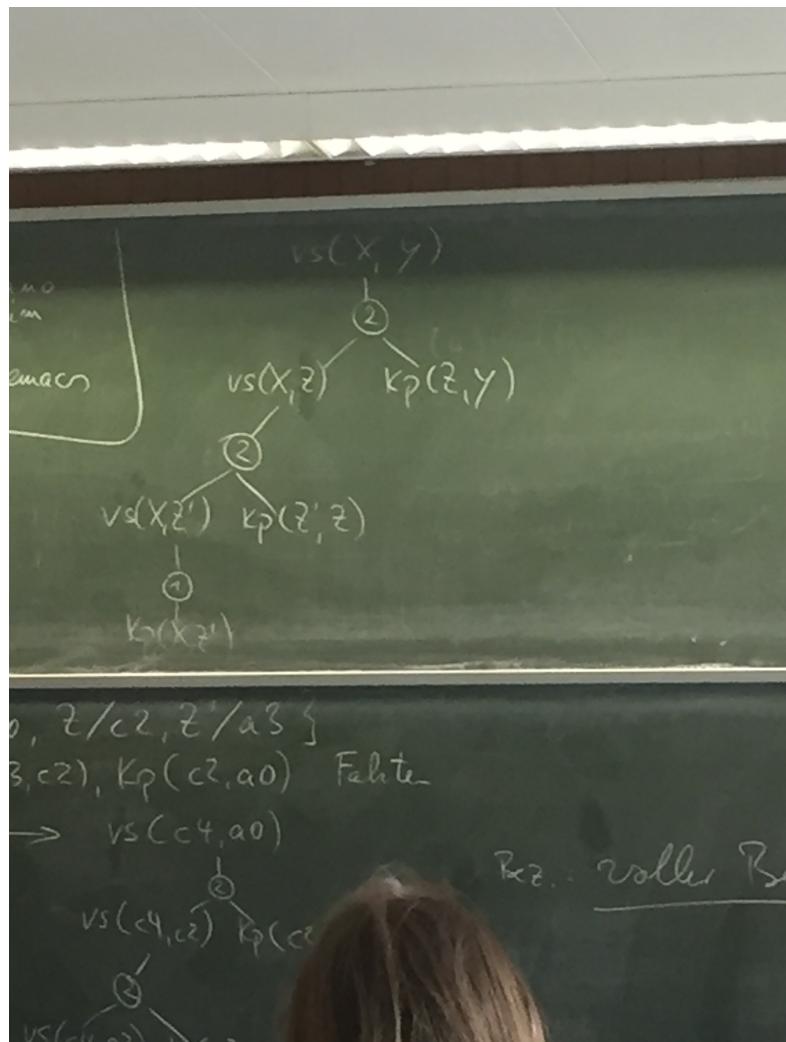


Figure 5:

- Θ ordnet jeder Variable in S ein Konstantensymbol aus der Menge der in P vorkommenden Konstantensymbole zu.
- Für jedes Blatt von S gilt, dass Θ , angewandt auf die Benennung des Blattes, einen Fakt aus P_F liefert (build-in-Prädikate seien entsprechend berücksichtigt)

Beweisbaum B entsteht aus S durch Anwendung von Θ auf alle Benennungen von Zielknoten. B repräsentiert einen Beweis für $g\Theta$, g Benennung der Wurzel von S .

Beispiel:

$$\begin{aligned}\Theta = & \{X/c4, Y/a0, Zc2, Z'/a3\} \\ & kp(c4, a3), kp(a3, c2), kp(c2, a0) \text{ Fakten}\end{aligned}$$

Bezeichnung : Voller Beweisbaum

Systematische Erzeugung der Suchbäume

Sukzessive alle Bäume der Tiefe 1, 2, ... erzeugen (breadth first)

Definition: Tiefe eines Baums maximale Anzahl von Zielknoten auf einem Pfad von einem Blattknoten zur Wurzel. Entsprechend Knoten der Tiefe i , Ebene i eines Baumes. Zusätzlich: Spezielle Suchbäume (Tiefe 0) für Fakten aus P .

Beispiel : Ziel $g = vs(c4, y)$

- **Tiefe 1:** $vs(c4, y)$, Θ , kein Beweisbaum
- **Tiefe 2:** $vs(c4, y)$
 $kp(c4, y)$, $\{y/a3\}$

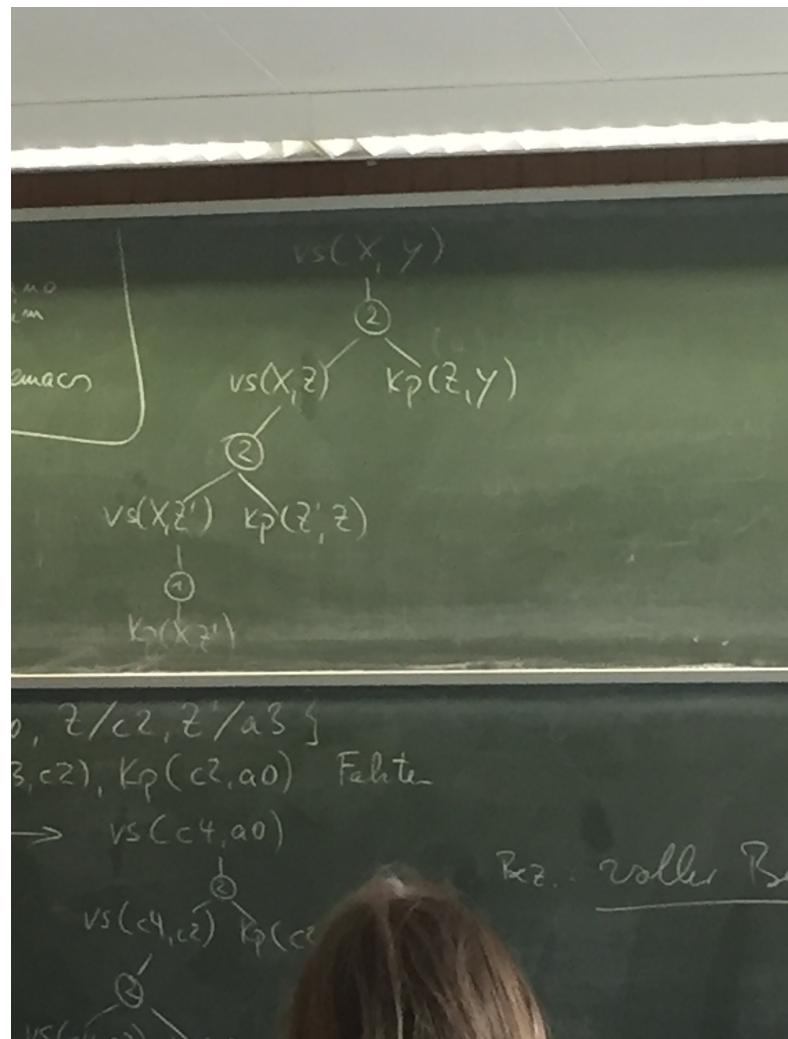


Figure 6:

Offensichtlich

- Zu jedem endlichen Beweise, der mit Grundatomen “Startet”, kann ein entsprechender Beweisbaum auf den beschriebenen Weg erhalten werden.
- Zu jedem Fakt $f \in \text{cons}(P)$ gibt es einen endlichen Beweis, der mit Grundatomen startet, siehe Fixpunktberechnung \rightarrow voller Beweisbaum

\Rightarrow Methode Suchbaum / Beweisbaum ist vollständig.

Da ein Beweisbaum mit $g\Theta$ als Benennung der Wurzel einen Beweis für $g\Theta$ darstellt

\Rightarrow Methode Suchbaum / Beweisbaum ist korrekt.

Satz 1.5

Sei P ein Datalog-Programm. Die Suchbaum / Beweisbaum Methode, angewandt auf alle Ziele $q(X_1, \dots, X_{\text{Stelligkeit}(q)})$, q intentionales Prädikatesymbol von P , liefert $\text{cons}(P)$ als Ergebnis

Frage: Abbruchkriterium bei der Erzeugung von Suchbäumen

Es gilt

- Beweisbäume sind isomorph zu den Suchbäumen, aus denen sie erzeugt wurden
- Zu jedem Fakt $f \in \text{cons}(P)$ gibt es einen vollen Beweisbaum, in dem auf jeder Zielknotenebene ein neuer Fakt (neues Teilziel) auftritt

Beweis Sei B ein voller Beweisbaum, der diese Eigenschaft nicht erfüllt. Dann gibt es ein i , so dass in B auf Ebene i nur Fakten auftreten, die schon auf vorhergehenden Ebenen auftreten. Die Teilbäume von B mit Wurzel auf Ebene i können “höher gehängt” werden \Rightarrow Es gibt einen äquivalenten, vollen Beweisbaum mit geringerer Tiefe.

- Es gibt nur endlich viele verschiedene Grundatome, die aus den Prädikaten und Konstantensymbolen gebildet werden können. Konkret: Seien $ap(p)$ die Anzahl verschiedener Prädikatensymbole die in p vorkommen
- $ak(p)$ die Anzahl verschiedener Konstantensymbole, die in p vorkommen

Dann ist $\max.fakt(P) = ap(P) * ak(P)^{\max.st(P)}$ eine obere Schranke für die Anzahl verschiedener Grundatome. **Damit:**

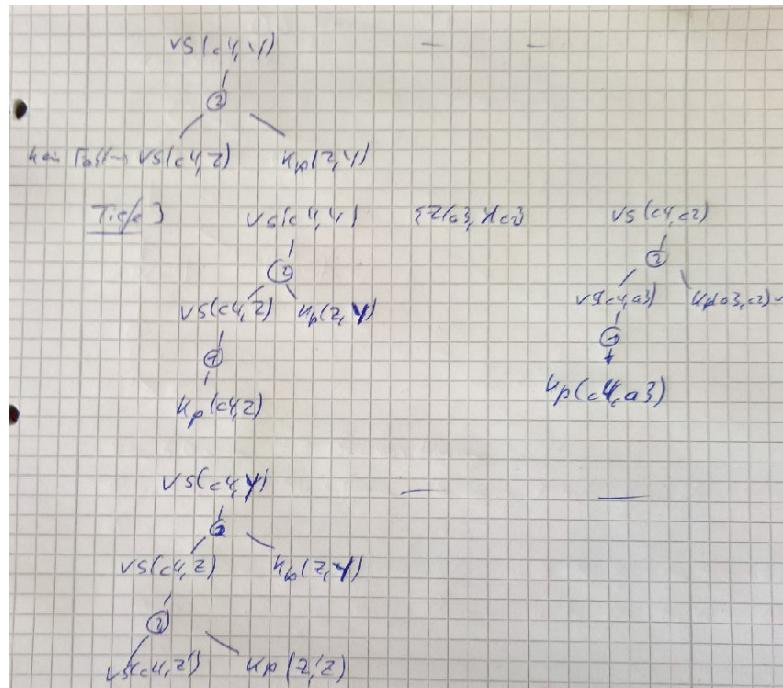


Figure 7:

Satz 1.6

Die Suchbaum / Beweisbaum Methode bleibt vollständig für ein Programm P, wenn nur Bäume mit max. Tiefe $\text{max_fakt}(P)$ betrachtet werden. Statt breadth first andere Vorgehensweise denkbar:

- Depth first mir Abbruch bei $\text{max_fakt}(P)$ und backtracking
- Dynamische Abbruchbedingungen, abhängig von erzeugten Faktenmenge (limit 10)
- Erkennen und Ausnutzen identischer (Teil-) Zielknoten (\rightsquigarrow gerichteter, azyklischer Graph)

Zusammenfassen von Such und Beweisbäumen, Berücksichtigung von Fakten aus $P_F \Rightarrow$ frühzeitiges Erkennen von Suchbäumen zu denen keine Beweisbäume existieren

Sei g das Ziel (Wurzelbenennung), dann erzeugt ein Baum der mit der leeren Menge

endet, genau ein Ergebnis.

$g\Theta_1, \dots, \Theta_n, (\Theta_1, \dots, \Theta_n)$ die in dieser Reihenfolge angewandten Unifikatoren (von der Wurzel aus) $\Theta = \Theta_1, \dots, \theta_n$ heißt Antwortsubstitution.

Resolutionsmethode

Für allgemeine Klauselformen entwickelte Methode zum automatischen Beweisen.

Literatur J Allen Robinson: A machine-oriented logic base on the resolution principle. Journal of the ACM 12, S.23-41, 1965

In Logikprogrammierung (Horn-Klauseln): SLD-Resolution (Linear Resolution with Selected Function for Definite Clauses).

- S (Selection function): Legt fest, wie Literale aus einem Ziel (Klausel mit einem oder mehreren negierten Literalen) auszuwählen sind
- L (Linear): In jedem Schritt wird genau ein Literal ausgewählt für die Unifikation mit dem Kopf einer Regel einer einem Fakt
- D (Definit): Beschränkung auf Programme mit definiten Klauseln

Bemerkung Methode steht für ganze Klasse von Algorithmen. Methode auch bekannt unter LUSH (Linear resolution with Unrestricted Selection function for Horn clauses)

Übergang Such / Beweisbaum \rightsquigarrow Resolution Nachweis der Unerfüllbarkeit einer Klausel

Seien $E = \{\neg M_1, \dots, \neg M_i, \dots, \neg M_K\}$, $r = L_0 : -L_1, \dots, L_m$ eine Regel und $S(E) = \neg M_i$. Es existiere $mgu(M_i, L_0) = 0$.

Resolvente von E und r: $\{\neg M_1\Theta, \dots, \neg M_{i-1}\Theta, \neg L_1\Theta, \dots, L_m\Theta \neg M_{i+1}\Theta, \dots, \neg M_K\Theta\}$

- Unerfüllbarkeit: Anwendung der Methode ende mit \square (leere Klausel, entspricht Falsch)

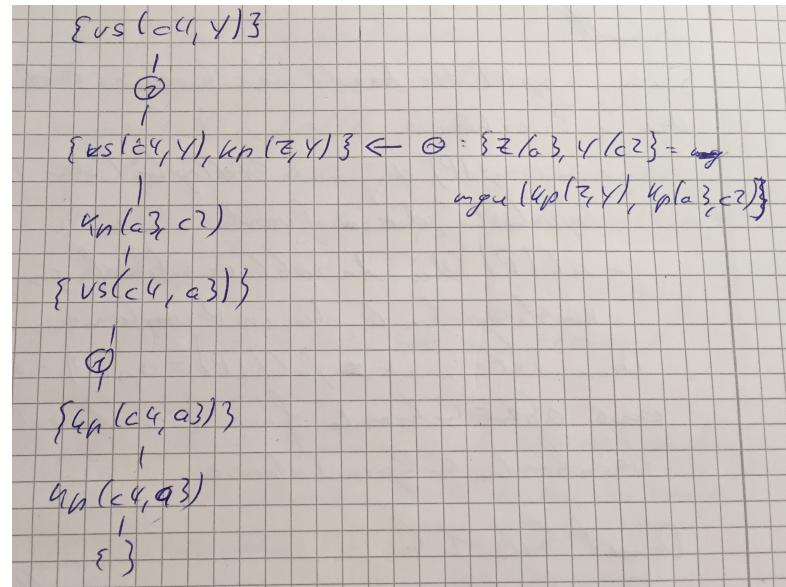


Figure 8:

- Ergebnis: Sei $g = \{\neg p(t_1, \dots, t_n)\}$ das Ziel, $\{p(t_1, \dots, t_n)\Theta/\Theta$ Antwortsubstitution für eine Ableitung von g , die mit \square endet }
- Ohne Beweis.

Satz 1.7

Die SLD-Resolution liefert korrekte und vollständige Ergebnisse für alle Datalog-Programme.

Beispiel m alle Kurse $\neq a3$ mit ihren Studierenden, die Voraussetzung von $c4$ sind.

$$\begin{aligned}
 r_1 &= vs(X, Y) : -Kp(X, Y). \\
 &\quad vs(c4, Y) : -vs(v4, Z), Kp(Z, Y). \\
 &\quad m(Z, W) : -vs(c4, Z), ag(Z, X), bl(Z, W, U), Z \neq a3.
 \end{aligned}$$

1.Schritt

- $r_1 \checkmark$

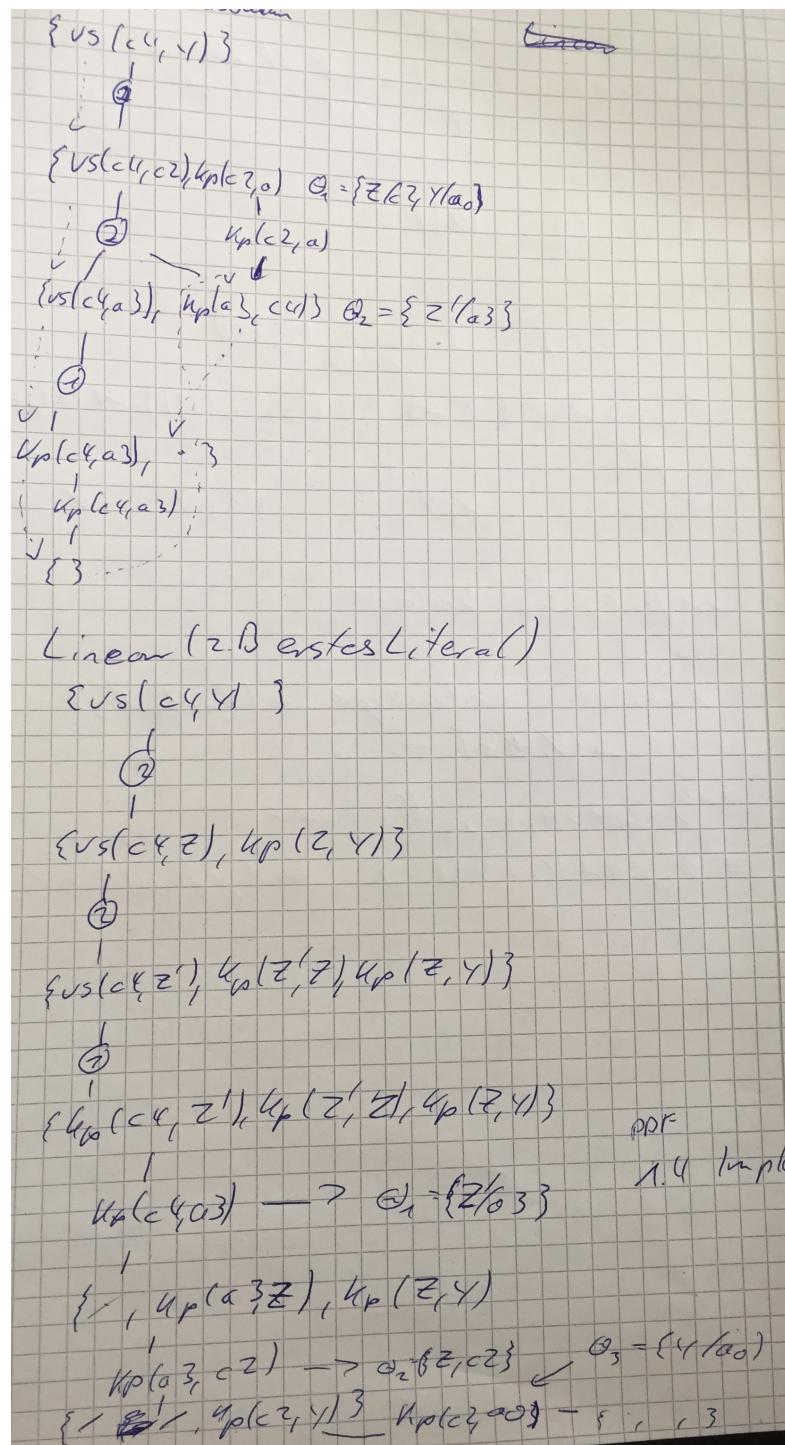


Figure 9:

Seite 3 unter Ben

$$u(v, w, x) := \exists \sum_{\tau} y_1 u(s, z, y_1, \cancel{y_2, y_3}, v=a, w=x)$$

$S_1 = \overline{\tau}$

$S_2 = \overline{\tau}_{2,3} (\overline{\tau}_{\tau=6}(u))$

$D_v = \{a\}$

$D_w = \overline{\tau}_1(\tau) \quad (v \in_b \underset{x \rightarrow w}{\tau_1}(\tau))$

$G_{v=a \wedge w=x} (\tau \Delta \overline{\tau}_{2,3} (\overline{\tau}_{\tau=6}(u)) \Delta \{a\} \Delta \overline{\tau}_1(\tau))$
kann es fallen

Typen: $x, y \quad z, y \quad v, w$

rectified rules Seite 5

Bsp

Annahme: weiterer Regl $u(z_1, z_2, z_3) :- \dots$ in P

$\rightarrow Fina(u, v, w, x) := \dots$

Umb $v \rightarrow y_1, w \rightarrow y_2, x \rightarrow y_3$ ($G_{v=a \wedge w=x} \dots$)

$\equiv G_{y_1=a \wedge y_2=y_3 \wedge \dots}$ analog f. $u(z_1, z_2, z_3)$

Figure 10:

- $r_2 \rightsquigarrow \tilde{r}_2 = vs(Q, Y) : -vs(C4, Z), kp(Z, Y), Q = c4.$
- $r_3 \checkmark$

2.Schritt

- $E_{r_1} = \text{KURSPLAN}$
- $E_{r_2} = \sigma_{Q=c4} \underbrace{\Pi_2(\varsigma_{@1=c4}(VS))}_{Z} \bowtie^6 \underbrace{\text{KURSPLAN}}_Z, Y \bowtie \underbrace{\{c4\}_Q}_Q$
- $E_{r_3} = \sigma_{Z \neq a3} \underbrace{\Pi_2(\varsigma_{@1=c4}(VS))}_{Z} \bowtie \text{ANGEBOT} \bowtie \text{BELEGUNG}$

3.Schritt vereinfacht $Q \Rightarrow X^7$, damit \tilde{r}_1 und \tilde{r}_2 passen.

$$\begin{aligned} & vs(X, Y) \\ & vs(X, Y) \\ & m(Z, W) \end{aligned}$$

4.Schritt

$$\begin{aligned} VS &= \underbrace{\text{KURSPLAN}}_{XY} \cup \underbrace{(\Pi_2(\varsigma_{@1=c4}(VS)) \bowtie \text{KURSPLAN} \bowtie \{c4\}_X)}_X Y \\ M &= (\varsigma_{Z \neq a3}(\Pi_2(\varsigma_{@1=c4}(VS)) \bowtie \text{ANGEBOT} \bowtie \text{BELEGUNG}))[z, w] \end{aligned}$$

Schreibweise für Gleichungssysteme

$$R_i = E_i \underbrace{(R_1, \dots, R_n)}_{DB-Relationen}, i = 1, \dots, n$$

Vergleich der Ausruckskraft⁸ von Datalog und Relationale Algebra:

- Reines Datalog ohne Rekursion entspricht RA^+ (monotone Relationale Algebra)
- Reines Datalog ohne Rekursion entspricht RA^+ ohne Gleichungssysteme (*i* RA^+)

⁶Join Operation

⁷kommt nicht im Rumpf von \tilde{r}_2 vor

⁸genauere Definition später

Beispiel: Kursplan (kp):

X	Y
c4	a3
a3	c2
c4	a2
c2	a0

$$vs(X, Y) : -Kp(X, Y). \quad vs(X, Y) : -\underbrace{vs(X, Z)}_{XZ}, \underbrace{kp(Z, Y)}_{ZY}. \rightsquigarrow vs = \Pi_{1,3}(VS \bowtie KURSPLAN) \cup KU$$

Jacobs entspricht Gauss-Seidel

$$R^0 = \emptyset$$

$$Q^1 = \emptyset$$

$$R^1 = KP \leftarrow (\emptyset_{X2} \bowtie KP_{ZY})[X, Y]$$

$$Q^2 = KP$$

$$R^2 = \{(c4, c2), (a3, a0)\} \cup KP \quad (KP_{XZ}) \bowtie KP_{ZY})[X, Y] \cup KP_{X,Y}$$

$$Q^3 = R^2$$

$$R^3 = \{(c4, a0)\} \cup R^2 \quad (((KP_{XZ} \bowtie KP_{ZY})^9)[X, Y] \cup KP_{XY})_{XZ})[X, Y] \cup KP_{XY}$$

Anmerkung

Berücksichtigung der “neuen” Information für VS hätte genügt. Grund: $(VS' \cup VS'') \bowtie KURSPLAN = VS' \bowtie KURSPLAN \cup VS'' \bowtie KURSPLAN$

Nach Schritt 4

$$\begin{aligned} R_1 \bowtie R_2 \cup R_2 \bowtie R_3 &\Rightarrow^{mit \delta-Berücksichtigung \ 10} \\ \delta R_1 \bowtie \cup R_2 \bowtie R_3 \cup R_1 \bowtie \delta R_2 \cup \delta R_2 \bowtie R_3 \cup R_1 \bowtie R_2 \cup R_2 \bowtie \delta R_3 \\ &\Rightarrow^{11} \delta R_1 \bowtie R_2 \cup R_1 \bowtie \delta R_2 \cup \delta R_2 \bowtie R_3 \cup R_w \bowtie \delta R_3 \end{aligned}$$

⁹Es wird zu viel berechnet beim Join. Siehe unten

¹¹Ist suboptimal, weil wieder zu viel berechnet wird

$$\textcircled{1} \quad g = \text{vs}(c4, y)$$

P: $\begin{array}{l} \text{vs}(x, y) - k_p(x, y). \\ \text{vs}(x, y) - \text{vs}(x, z), k_p(z, y). \end{array} \rightsquigarrow \begin{array}{l} \text{vs}(c4, y) - k_p(c4, y). \\ \text{vs}(c4, y) - \text{vs}(c4, z), k_p(z, y) \end{array}$

Figure 11:

$$\textcircled{1} \quad g = \text{vs}(c4, y)$$

P: $\begin{array}{l} \text{vs}(x, y) - k_p(x, y). \\ \text{vs}(x, y) - \text{vs}(x, z), k_p(z, y). \end{array}$

top down: $\begin{array}{c} \text{vs}(c4, y) \\ \text{vs}(c4, z), k_p(z, y) \quad k_p(c4, y) \\ \text{vs}(c4, z) \quad k_p(z, z) \quad k_p(c4, z) \\ \text{vs}(c4, z) \quad k_p(z, z) \quad k_p(c4, z) \end{array}$

\times learn w. \rightarrow What 'c4' am I?

magic vs (c4),
 $\text{vs}(x, y) - k_p(x, y)$,
 $\text{vs}(x, y) - \text{magic vs}(x), \text{vs}(x, z), k_p(z, y)$

Figure 12:

$$\delta R^0 = KP$$

$$R^0 = KP$$

$$\delta Q^1 = KP$$

$$\delta R^1 = \{(c4, c2), (a3, a0)\} \cup KP$$

$$\delta R^1 = \{(c4, c2), (a3, a0)\} \cup KP$$

$$R^1 = Kp \cup \{(c4, c2), (a3, a0)\}$$

$$\delta Q^2 = \{(c4, c2), (a3, a0)\}$$

$$\delta R^2 = \delta Q^2 \bowtie Kp \cup Kp = \{(c4, a0)\} \cup Kp$$

$$R^2 = Kp \cup \{(c4, c2), (a3, a0)\} \cup \{(c4, a0)\}$$

$$\delta Q^3 = \{(c4, a0)\}$$

$$\delta R^3 = \delta Q^3 \bowtie Kp \cup Kp = Kp$$

$$\delta R^3 = \emptyset$$

$$R^3 = R^2 \cup \emptyset$$

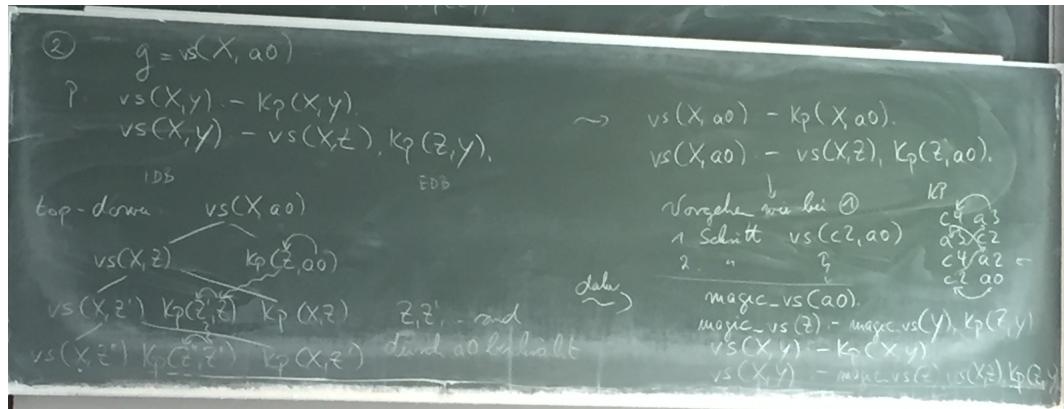


Figure 13:

Beispiel

$$r_2 = vs^{fb}(X, Y) : -vs^{fb}_(X, Z), Kp(Z, Y).$$

i) ✓

ii) ... magic_r2_vs^{fb}_1(X, Z) ...

iii) ... magic_r2_vs^{fb}_1(Z) ...

iv) vs^{fb}(Y)

magic_r2_vs^{fb}_1(Z) : -magic_vs^{fb}(Y), kp(Z, Y).

$$r_0 = query^f(X : -vs^{fb}_(X, a0)).$$

i) ...

ii) query^f(X) : -magic_r0_vs^{fb}_1(X, a0)

iii) query^f(X) : -magic_r0_vs^{fb}_1(a0)

iv) -

v) vs^{fb}(Y)

magic_r0_vs^{fb}_1(a0) : -magic_query^f().

$r_0 \rightsquigarrow {}^{12}magic_r0_vs^{fb}_1(a0) : -$

$r_2 \rightsquigarrow {}^{12}magic_r2_vs^{fb}_1(Z) : -magic_vs^{fb}(Y), Kp(Z, Y).$

¹²“Sideways information passing”

$$\begin{aligned} p(X) &: \neg q(X) \\ p(X) \cup q(X) \\ q(X) &: \neg \neg p(X) \end{aligned}$$

$P = \{ag(a3, o) : \neg ag(a3, m)\}$.
“Falls ‘m’ Kurs a3 nicht anbietet, bietet o den Kurs an”
Mit CWA:

$$\begin{aligned} ag(a3, m) &\notin ANGEBOT \\ ag(a3, o) &\notin ANGEBOT \end{aligned}$$

Da beide keine logische Folgerung von P (müsste in allen Modellen gültig sein)
Nimm Modelle $\{ag(a3, o)\}$.

$$\begin{aligned} \{ag(a3, o)\} \\ \Rightarrow^{CWA^{13}} \neg ag(a3, m) \text{ und } \neg ag(a3, o) \text{ sind gültig.} \\ \text{Aber } \neg ag(a3, m) \Rightarrow ag(a3, 0) \rightsquigarrow \text{Widerspruch zur CWA} \end{aligned}$$

Beachte folgende HB-Interpretation von P.

$$\begin{aligned} I_1 &= \{ag(a3, o)\} \text{ und } I_2 = \{ag(a3, m)\} (I_1, I_2 \text{ sind minimale Modelle}) \\ I_3 &= \{ag(a3, o), ag(a3, m)\} \text{ ist ebenfalls Modell von P} \end{aligned}$$

In I_1 gilt mit CWA: $\neg ag(a3, m)$
In I_2 gilt mit CWA: $\neg ag(a3, o)$

Grund für Problem: Implikation $\neg ag(a3, m) \Rightarrow ag(a3, o)$ äquivalent zu $ag(a3, m) \vee ag(a3, 0)$. Welches Modell soll ausgezeichnet werden? Semantik?

¹³Closed World Assumption

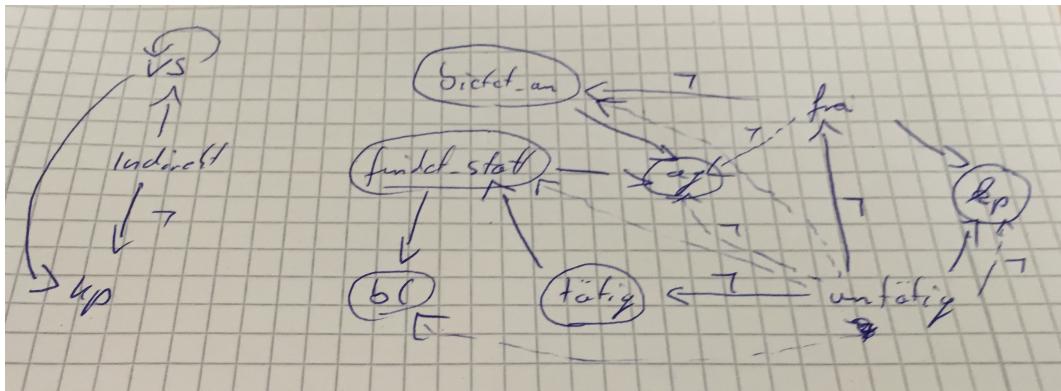


Figure 14:

Offensichtlich: I Modell von P \diamond I Modell von $P' = \{ag(a3, m) : -ag(a3, o)\}$
Gehe wie bei Fuxpunktberechnung vor:

P Bekannte Fakten $EDB = \{ag(a1, m), ag(c4, q), ag(a3, d), \dots\}$
 $ag(a3, m) \notin EDB$
CWA: $\neg ag(a3, m)$ gilt $\Rightarrow^{\text{Regel}} ag(a3, 0)$

P' $ag(a3, o) \notin EDB$
CWA: $\neg ag(a3, 0)$ gilt $\Rightarrow^{\text{Regel}} ag(a3, m)$

Daraus folgt: Wähle $I_1(I_2)$ als Modell (Semantik) von P(P') \rightsquigarrow Auswertungsrichtung gemäß Implikation legt Semantik fest. D.h. Semantik ist abhängig von der Art des Aufschreibens der Klauseln. Daraus ergibt sich die Frage nach einer besseren Semantikdefinition.

$$\begin{aligned}
& r(1), s(1), s(2). \\
U(X) : -r(X). & r(1) \rightsquigarrow u(1) \\
q(X) : -s(X), \neg u(X). & \Leftarrow s(1) \rightsquigarrow q(1) \\
& \Leftarrow s(2) \rightsquigarrow q(2)
\end{aligned}$$

Disjunktives Datalog

Einige Bemerkungen zu disjunktivem Datalog. Negationen im Rumpf sind umgeschrieben genau ein Oder (Disjunktion).

Form 6: $q_1(\dots), \dots, q_n(\dots) : -p_1(\dots), \dots, p_n(\dots)$.

Negation in Rümpfen erlaubt \Rightarrow Frage: Sind Disjunktionen im Kopf von Regeln notwendig?

Beispiel: Tim ist 18 oder¹⁴ 16 Jahre alt: $alter(tim, 18), alter(tim, 16) : -.$

Logisch äquivalent zu:¹⁵

$$\begin{aligned} alter(tim, 18) : & \neg\neg alter(tim, 16). \\ alter(tim, 16) : & \neg\neg alter(tim, 18).^{\text{16}} \end{aligned}$$

\rightsquigarrow Falls auf “Tim ist nicht 18 Jahre alt.” etwas mit CWA geschlossen werden kann, folgt definitiv “Tim ist 16 Jahre alt.”.

Mehrere Semantiken in Literatur vorgeschlagen, etwa:

“General Closed World Assumption” (GCWA)

Idee: Weil mehrere minimale Herbrand Modelle möglich sind \rightsquigarrow Schluss auf das Negative, $\neg p$, genau dann möglich wenn p in keinem minimalen Modell enthalten ist.

Beispiel: $\{alter(tim, 18)\}, \{alter(tim, 16)\}$ sind die beiden minimalen Modelle $\{alter(tim, 18), alter(tim, 16)\}$ ist auch ein Modell, aber nicht minimal.

Minimale Modelle: $\{q, r\}, \{q, s\}$

$$\begin{aligned} P = \{p, q : & - \\ & q : - \\ & r, s : -.\} \end{aligned}$$

p ist in keinem minimalen Modell enthalten $\rightsquigarrow \neg p$ kann unter GCWA angenommen werden.

¹⁴ Altagserfahrung: Exklusiv, Theoretisch liegt dieses Wissen um exklusivität nicht vor.

¹⁵ Tim ist 18 oder 16 Jahr alt, mehr wissen wir nicht (oder formal gesehen inklusives Oder)

¹⁶ Tim ist 18 Jahre alt, wenn er nicht 16 Jahre alt ist

Eigenschaften GCWA erfüllt die Eigenschaften der

- + Konsistenz: Negative Information, die auf Grund der GCWA hinzugefügt wird, führt nicht zu Widerspruch in der Theorie.
- + Stabilität: Negative Information, die auf Grund der GCWA hinzugefügt wird, verringert die positive Information nicht und vergrößert sie auch nicht.
- + Reduktion: Für definite Programme entspricht die GCWA der CWA.
- - Monotonie: CWA garantiert Monotonie, GCWA nicht.

Monoton

Seien P, P' definite oder disjunkte Programme. Sei R eine Negationsregel mit $R(P)$ definiert die Menge der negativen Atome, die gemäß R für P angenommen werden können.

R ist monoton $\Diamond P \subseteq P' \Rightarrow R(P') \subseteq R(P)$

Beispiel

$$\begin{aligned} P &= \{p(a) \vee p(b)\} \\ P' &= P \cup \{p(a)\} \\ P'' &= P' \cup \{p(b)\} \end{aligned}$$

Success:

$$\begin{aligned} \text{Success von } P &= \{p(a) \vee p(b)\} \\ \text{Success von } P' &= \{p(a) \vee p(b), p(a)\} \\ \text{Success von } P'' &= \{p(a) \vee p(b), p(a), p(b)\} \end{aligned}$$

Unknown:

$$\begin{aligned} \text{Unknown von } P &= \{p(a), p(b)\} \\ \text{Unknown von } P' &= \emptyset \\ \text{Unknown von } P'' &= \emptyset \end{aligned}$$

Mit GCWA:

$$\begin{aligned} \text{Durch GCWA von } P &= \emptyset \\ \text{Durch GCWA von } P' &= \{\neg p(b)\} \\ \text{Durch GCWA von } P'' &= \emptyset \end{aligned}$$

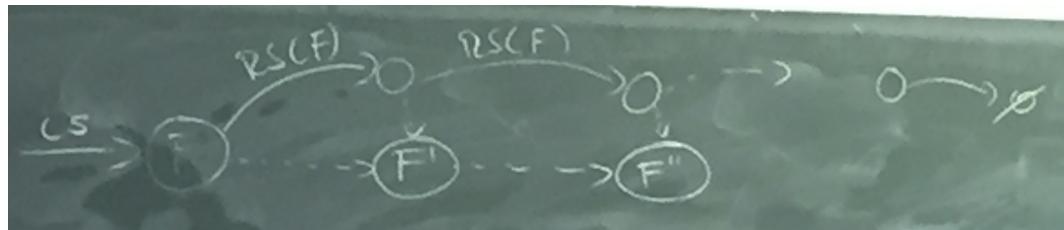


Figure 15:

KOMponenten		
Berechnung	Teil	Anzahl
Flügel	Strebe	5
Flügel	Fahrgestell	1
Flügel	Niete	100
Fahrgestell	Niete	8
Fahrgestell	Schrauer	3
Schrauer	Niete	4
Strebe	Niete	10

FLÜGELTEILE		
	Teil	Anzahl
Strebe	Strebe	5
	Fahrgestell	1
	Niete	100
Niete	Niete	50
	Niete	8
	Schrauer	3
Schrauer	Niete	12

Figure 16:

Dadurch ist zu sehen, dass die Monotonie verletzt (zwischen P' und P''), damit ist die GCWA nicht monoton.

$ANCESTRY(X, Y) : -MOTHERHOOD(X, Y).$

$ANCESTRY(X, Y) : -MOTHERHOOD(X, Z), ANCESTRY(Z, Y).$