

THEORETISCHE INFORMATIK UND LOGIK

18. Vorlesung: Unifikation

Markus Krötzsch

Lehrstuhl Wissensbasierte Systeme

TU Dresden, 23. Juni 2017

Resolution für Prädikatenlogik

Ein konkreter Algorithmus zum logischen Schließen:

- (1) Logische Konsequenz auf Unerfüllbarkeit reduzieren
- (2) Formeln in Klauselform umwandeln
 - Formel bereinigen
 - Negationsnormalform bilden
 - Pränexform bilden
 - Skolemform bilden
 - Konjunktive Normalform bilden
- (3) Resolutionsverfahren anwenden
 - Unifikation zum Finden passender Klauseln
 - Bilden von Resolventen bis zur Terminierung

Konjunktive Normalform in Prädikatenlogik

Eine Formel ist in **konjunktiver Normalform (KNF)** wenn sie in Pränexform $\mathcal{Q}x_1 \dots \mathcal{Q}x_n.F$ ist, wobei F eine Konjunktion von Disjunktionen von Literalen ist, welche keine Quantoren enthält. Das heißt F hat die Form:

$$(L_{1,1} \vee \dots \vee L_{1,m_1}) \wedge (L_{2,1} \vee \dots \vee L_{2,m_2}) \wedge \dots \wedge (L_{n,1} \vee \dots \vee L_{n,m_n})$$

wobei $L_{i,j}$ Literalen sind. **Klauseln** sind Disjunktionen von Literalen.

Wir stellen die KNF in der Prädikatenlogik wie folgt her:

- (1) Formel bereinigen
- (2) Bilden der Negationsnormalform
- (3) Bilden der Pränexform
- (4) Skolemisieren
- (5) Erschöpfende Anwendung der folgenden Ersetzung auf Teilformeln im quantorenfreien Teil der Formel:

$$F \vee (G \wedge H) \mapsto (F \vee G) \wedge (F \vee H)$$

Beispiel: Konjunktive Normalform

$$\begin{aligned} & \forall x_1, x_2, x_3, x_4, x_5. \\ & \left(\left((W(x_1) \wedge \neg L(x_1)) \vee (L(x_1) \wedge \neg W(x_1)) \right) \right. \\ & \quad \wedge (\neg W(x_2) \vee (W(x_3) \vee L(x_4))) \\ & \quad \left. \wedge (\neg L(x_5) \vee (\neg W(f_6(x_1, x_2, x_3, x_4, x_5)) \wedge \neg L(f_7(x_1, x_2, x_3, x_4, x_5)))) \right) \\ \equiv & \forall x_1, x_2, x_3, x_4, x_5. \\ & \left((W(x_1) \vee L(x_1)) \wedge (\neg L(x_1) \vee L(x_1)) \right. \\ & \quad \wedge (W(x_1) \vee \neg W(x_1)) \wedge (\neg L(x_1) \vee \neg W(x_1)) \\ & \quad \wedge (\neg W(x_2) \vee (W(x_3) \vee L(x_4))) \\ & \quad \left. \wedge (\neg L(x_5) \vee (\neg W(f_6(x_1, x_2, x_3, x_4, x_5)) \wedge \neg L(f_7(x_1, x_2, x_3, x_4, x_5)))) \right) \\ \equiv & \forall x_1, x_2, x_3, x_4, x_5. \\ & \left((W(x_1) \vee L(x_1)) \wedge (\neg L(x_1) \vee L(x_1)) \right. \\ & \quad \wedge (W(x_1) \vee \neg W(x_1)) \wedge (\neg L(x_1) \vee \neg W(x_1)) \\ & \quad \wedge (\neg W(x_2) \vee (W(x_3) \vee L(x_4))) \\ & \quad \left. \wedge (\neg L(x_5) \vee \neg W(f_6(x_1, x_2, x_3, x_4, x_5)) \wedge (\neg L(x_5) \vee \neg L(f_7(x_1, x_2, x_3, x_4, x_5)))) \right) \end{aligned}$$

Klauselform

Die **Klauselform** ist eine vereinfachte Schreibweise der KNF:

- Allquantoren werden weggelassen
- Klauseln werden als Mengen von Literalen geschrieben
- Konjunktionen von Klauseln werden als Mengen von Mengen von Literalen geschrieben

Beispiel: Unser Beispiel kann damit wie folgt geschrieben werden:

$$\{ \{W(x_1), L(x_1)\}, \\ \{\neg L(x_1), L(x_1)\}, \\ \{W(x_1), \neg W(x_1)\}, \\ \{\neg L(x_1), \neg W(x_1)\}, \\ \{\neg W(x_2), W(x_3), L(x_4)\}, \\ \{\neg L(x_5), \neg W(f_6(x_1, x_2, x_3, x_4, x_5))\}, \\ \{\neg L(x_5), \neg L(f_7(x_1, x_2, x_3, x_4, x_5))\} \}$$

Idee der Resolution

Aussagenlogische Resolution verallgemeinert die Verkettung von Implikationen. Beispiel:

$$\begin{array}{cc} A \rightarrow B & B \rightarrow C \\ \hline \{\neg A, B\} & \{\neg B, C\} \\ \hline \{\neg A, C\} \\ A \rightarrow C \end{array}$$

Dies ist auch in der Prädikatenlogik ein gültiger Schluss:

$$\begin{array}{cc} \forall x.(A(x) \rightarrow B(x)) & \forall x.(B(x) \rightarrow C(x)) \\ \hline \{\neg A(x), B(x)\} & \{\neg B(x), C(x)\} \\ \hline \{\neg A(x), C(x)\} \\ \forall x.(A(x) \rightarrow C(x)) \end{array}$$

Unifikation

Terme in Klauselform

Allerdings ist es nicht immer so einfach.

Schon die alten Griechen wussten:

$$\frac{\forall x.(\text{Mensch}(x) \rightarrow \text{Sterblich}(x)) \quad \text{Mensch}(\text{sokrates})}{\text{Sterblich}(\text{sokrates})}$$

Aber die entsprechenden Klauseln sind:

$$\{\neg \text{Mensch}(x), \text{Sterblich}(x)\} \quad \{\text{Mensch}(\text{sokrates})\}$$

Problem: Wie kann man Resolution über unterschiedliche Atome $\text{Mensch}(x) \neq \text{Mensch}(\text{sokrates})$ durchführen?

Lösung: x ist universell quantifiziert und kann für beliebige Elemente stehen, also auch für Sokrates – logisch gilt:

$$\forall x.(\text{Mensch}(x) \rightarrow \text{Sterblich}(x)) \quad \text{Mensch}(\text{sokrates}) \rightarrow \text{Sterblich}(\text{sokrates}) \\ \{\neg \text{Mensch}(x), \text{Sterblich}(x)\} \quad \models \quad \{\neg \text{Mensch}(\text{sokrates}), \text{Sterblich}(\text{sokrates})\}$$

Funktionsterme in Klauseln

Was passiert mit Funktionen?

$$\frac{\forall x.(\text{Mensch}(x) \rightarrow \exists y.\text{hatVater}(x, y)) \quad \forall z, v.(\text{hatVater}(z, v) \rightarrow \text{hatKind}(v, z))}{\forall x.(\text{Mensch}(x) \rightarrow \exists y.\text{hatKind}(y, x))}$$

Entsprechende Klauseln:

$$\frac{\{\neg \text{Mensch}(x), \text{hatVater}(x, f(x))\} \quad \{\neg \text{hatVater}(z, v), \text{hatKind}(v, z)\}}{\{\neg \text{Mensch}(x), \text{hatKind}(f(x), x)\}}$$

Passen $\text{hatVater}(x, f(x))$ und $\neg \text{hatVater}(z, v)$ zusammen?

Ja, wenn wir z durch x ersetzen und v durch $f(x)$, denn es gilt:

$$\frac{\forall z, v.(\text{hatVater}(z, v) \rightarrow \text{hatKind}(v, z)) \quad \{\neg \text{hatVater}(z, v), \text{hatKind}(v, z)\}}{\forall x.(\text{hatVater}(x, f(x)) \rightarrow \text{hatKind}(f(x), x)) \quad \{\neg \text{hatVater}(x, f(x)), \text{hatKind}(f(x), x)\}}$$

Zusammenfassung und Verallgemeinerung

Für die Anwendung von Resolution benötigt man jeweils zwei gleiche Atome

(einmal negiert und einmal nicht-negiert)

Wir erreichen das in Prädikatenlogik durch folgende Methoden:

- Wir ersetzen (universell quantifizierte) Variablen durch andere Terme \rightsquigarrow **Substitution**
- Damit wollen wir erreichen, dass Terme (und letztlich Atome) gleich werden \rightsquigarrow **Unifikation**

Substitutionen

Eine **Substitution** ist eine endliche Menge der Form

$$\{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\},$$

wobei $x_1, \dots, x_n \in \mathbf{V}$ paarweise verschiedene Variablen und $t_1, \dots, t_n \in \mathbf{T}$ beliebige Terme sind.

Die **Anwendung einer Substitution** auf einen Ausdruck A (Formel oder Term) führt zu einem Ausdruck $A\{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$, der aus A entsteht, wenn man jedes freie Vorkommen einer Variablen x_i in A **simultan** durch t_i ersetzt.

Eine Formel bzw. ein Term $A\sigma$, der durch Anwendung einer Substitution σ auf A entsteht, nennt man **Instanz von A (unter σ)**

Beispiele: $p(x, y, z)\{x \mapsto f(y), y \mapsto a\} = p(f(y), a, z)$
 (und nicht $p(f(a), a, z)$)
 $(q(x, y) \rightarrow \forall x.r(x, y))\{x \mapsto z, y \mapsto f(x)\} = (q(z, f(x)) \rightarrow \forall x.r(x, f(x)))$
 (und nicht $(q(z, f(x)) \rightarrow \forall x.r(z, f(x)))$)!

Komposition von Substitutionen

Wir können Substitutionen hintereinander ausführen:

Für Substitutionen $\sigma = \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ und $\theta = \{y_1 \mapsto s_1, \dots, y_m \mapsto s_m\}$ ist die **Komposition $\sigma \circ \theta$** die folgende Substitution:

$$\{x_1 \mapsto t_1\theta, \dots, x_n \mapsto t_n\theta\} \cup \{y_i \mapsto s_i \mid y_i \in \{y_1, \dots, y_m\} \setminus \{x_1, \dots, x_n\}\}$$

Satz: Für alle Terme oder Formeln A und Substitutionen σ und θ gilt: $A(\sigma \circ \theta) = (A\sigma)\theta$.

Beweis: Per struktureller Induktion über den Aufbau von Termen.

I-Anfang: Satz gilt für Konstanten $A \in \mathbf{C}$ und Variablen $A \in \mathbf{V}$ per Definition.

I-Hypothese: Satz gilt für Terme t_1, \dots, t_ℓ .

I-Schritt: Damit gilt für größere Funktionsterme $f(t_1, \dots, t_\ell)(\sigma \circ \theta) = f(t_1(\sigma \circ \theta), \dots, t_\ell(\sigma \circ \theta)) \stackrel{IH}{=} f((t_1\sigma)\theta, \dots, (t_\ell\sigma)\theta) = (f(t_1, \dots, t_\ell)\sigma)\theta$.

Beweis für Formeln analog (aber mit mehr Fällen im I-Schritt). \square

Unifikationsprobleme

Ein **Unifikationsproblem** ist eine endliche Menge von Gleichungen der Form $G = \{s_1 \doteq t_1, \dots, s_n \doteq t_n\}$.

Eine Substitution σ ist ein **Unifikator** für G falls $s_i\sigma = t_i\sigma$ für alle $i \in \{1, \dots, n\}$ gilt.

Beispiele:

$\{f(x) \doteq f(\text{sokrates})\}$ hat den Unifikator $\{x \mapsto \text{sokrates}\}$
 $\{x \doteq f(y), y \doteq g(z)\}$ hat den Unifikator $\{x \mapsto f(g(z)), y \mapsto g(z)\}$,
aber auch z.B. $\{x \mapsto f(g(a)), y \mapsto g(a), z \mapsto a\}$
und $\{x \mapsto f(g(f(b))), y \mapsto g(f(b)), z \mapsto f(b)\}$
 $\{f(x) \doteq g(x)\}$ hat keinen Unifikator
 $\{x \doteq f(x)\}$ hat keinen Unifikator

Eindeutigkeit des mgu

Satz: Die allgemeinsten Unifikatoren von G sind bis auf Umbenennung von Variablen identisch.

(ohne Beweis)

Beispiel: Das Problem $\{f(x) \doteq f(z)\}$ hat die allgemeinsten Unifikatoren $\{x \mapsto z\}$ und $\{z \mapsto x\}$, aber auch z.B. $\{x \mapsto v, z \mapsto v\}$ ($v \in \mathbf{V}$). Dagegen ist $\{x \mapsto a, z \mapsto a\}$ mit $a \in \mathbf{C}$ ein nicht-allgemeinster Unifikator.

Satz: Wenn ein Unifikationsproblem lösbar ist (d.h., einen Unifikator hat), dann hat es auch einen allgemeinsten Unifikator.

Wir zeigen das durch Angabe eines Algorithmus für die mgu-Berechnung.

Unifikatoren vergleichen

Das Problem $\{x \doteq f(y), y \doteq g(z)\}$ hat viele Unifikatoren – gibt es einen „besten“?

Eine Substitution σ ist **allgemeiner** als eine Substitution θ , in Symbolen $\sigma \leq \theta$, wenn es eine Substitution λ gibt, so dass $\sigma \circ \lambda = \theta$.

Der **allgemeinste Unifikator** für ein Unifikationsproblem G ist ein Unifikator σ für G , so dass $\sigma \leq \theta$ für alle Unifikatoren θ für G .

(Die englische Bezeichnung des allgemeinsten Unifikators ist most general unifier (mgu).)

Beispiel: Für $G = \{x \doteq f(y), y \doteq g(z)\}$ ist $\sigma = \{x \mapsto f(g(z)), y \mapsto g(z)\}$ ein allgemeinster Unifikator. Dagegen ist $\theta = \{x \mapsto f(g(f(b))), y \mapsto g(f(b)), z \mapsto f(b)\}$ ein Unifikator, der nicht allgemeinst ist, denn es gilt $\sigma\{z \mapsto f(b)\} = \theta$.

Gelöste Unifikationsprobleme

Ein Unifikationsproblem $G = \{x_1 \doteq t_1, \dots, x_n \doteq t_n\}$ ist in **gelöster Form**, wenn x_1, \dots, x_n paarweise verschiedene Variablen sind, die nicht in den Termen t_1, \dots, t_n vorkommen.

In diesem Fall definieren wir eine Substitution $\sigma_G := \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$.

Satz: Wenn G ein Unifikationsproblem in gelöster Form ist, dann ist σ_G ein allgemeinster Unifikator für G .

Beweis: Sei $G = \{x_1 \doteq t_1, \dots, x_n \doteq t_n\}$. Dann ist $x_i\sigma_G = t_i = t_i\sigma_G$, wobei die zweite Gleichheit gilt, da t_i keine Variable aus x_1, \dots, x_n enthält. Also ist σ_G Unifikator für G .

Für einen beliebigen Unifikator θ von G gilt: $\theta = \sigma_G \circ \theta$:

- Für $x_i \in \{x_1, \dots, x_n\}$ ist $x_i\theta = t_i\theta = (x_i\sigma_G)\theta$
- Für $y \notin \{x_1, \dots, x_n\}$ ist $y\theta = y\sigma_G\theta$

Also ist σ_G ein allgemeinster Unifikator. □

Lösen von Unifikationsproblemen

Wir bringen Unifikationsprobleme schrittweise in gelöste Form:

Unifikationsalgorithmus

Eingabe: Unifikationsproblem G

Ausgabe: allgemeinsten Unifikator für G , oder „nicht unifizierbar“

Wende die folgenden Umformungsregeln auf G an, bis keine Regel mehr zu einer Änderung führt:

- **Löschen:** $\{t \doteq t\} \cup G' \rightsquigarrow G'$
- **Zerlegung:**
 $\{f(s_1, \dots, s_n) \doteq f(u_1, \dots, u_n)\} \cup G' \rightsquigarrow \{s_1 \doteq u_1, \dots, s_n \doteq u_n\} \cup G'$
- **Orientierung:** $\{t \doteq x\} \cup G' \rightsquigarrow \{x \doteq t\} \cup G'$ falls $x \in \mathbf{V}$ und $t \notin \mathbf{V}$
- **Eliminierung:** $\{x \doteq t\} \cup G' \rightsquigarrow \{x \doteq t\} \cup G' \setminus \{x \doteq t\}$ falls $x \in \mathbf{V}$ nicht in t vorkommt

Wenn G dann in gelöster Form ist, dann gib σ_G aus.
Andernfalls gib aus „nicht unifizierbar“.

Beispiel

$\{x \doteq f(a), g(x, x) \doteq g(x, y)\}$	Eliminierung
$\{x \doteq f(a), g(f(a), f(a)) \doteq g(f(a), y)\}$	Zerlegung
$\{x \doteq f(a), f(a) \doteq f(a), f(a) \doteq y\}$	Löschen
$\{x \doteq f(a), f(a) \doteq y\}$	Orientierung
$\{x \doteq f(a), y \doteq f(a)\}$	gelöste Form

Die zugehörige Substitution $\{x \mapsto f(a), y \mapsto f(a)\}$ ist Unifikator des ursprünglichen Unifikationsproblems

Korrektheit des Unifikationsalgorithmus (1)

Satz: Der Unifikationsalgorithmus berechnet für jedes Unifikationsproblem einen allgemeinsten Unifikator, falls es einen Unifikator gibt, und liefert „nicht unifizierbar“ wenn es keinen gibt.

Beweis: Wir zeigen nacheinander drei Aussagen, aus denen die Behauptung folgt.

Bei der Eingabe beliebiger Unifikationsprobleme G gilt:

- (1) Wenn der Algorithmus eine Substitution σ ausgibt, dann ist σ ein allgemeinsten Unifikator für G
- (2) Wenn der Algorithmus „nicht unifizierbar“ ausgibt, dann hat G keinen Unifikator
- (3) Der Algorithmus terminiert, d.h. erzeugt nach endlich vielen Schritten eine der beiden Ausgaben

Korrektheit des Unifikationsalgorithmus (2)

Beweis (Fortsetz.): Als erstes zeigen wir eine Hilfsaussage (\ddagger):

Wenn ein Unifikationsproblem G_1 in einem Ersetzungsschritt in ein Problem G_2 umgewandelt werden kann, dann haben G_1 und G_2 die gleichen Unifikatoren.

Für **Löschen**, **Orientierung** und **Zerlegung** ist das leicht zu sehen.

Für **Eliminierung** betrachten wir $G_1 = \{x \doteq t\} \cup G'$ und die Substitution $\sigma = \{x \mapsto t\}$. Es ist also $G_2 = \{x \doteq t\} \cup G' \sigma$.

θ Unifikator für G_1

gdw. $x\theta = t\theta$ und θ Unifikator für G'

gdw. $x\theta = t\theta$ und $\sigma \circ \theta$ Unifikator für G'

Begründung: $\sigma = \sigma_{\{x \doteq t\}}$ und daher $\theta = \sigma \circ \theta$
für alle Unifikatoren θ von $\{x \doteq t\}$ (gezeigt auf Folie 16)

gdw. $x\theta = t\theta$ und θ Unifikator für $G' \sigma$

gdw. θ Unifikator für $\{x \doteq t\} \cup G' \sigma = G_2$

Korrektheit des Unifikationsalgorithmus (3)

Beweis (Fortsetz.): (1) Wenn der Algorithmus eine Substitution σ ausgibt, dann ist σ ein allgemeinsten Unifikator für G .

- Gemäß Hilfsaussage (\ddagger) erhält jeder Umformungsschritt die Unifikatoren und damit auch die allgemeinsten Unifikatoren
- Per Induktion gilt also: jedes Unifikationsproblem, welches der Algorithmus erzeugt, hat den gleichen allgemeinsten Unifikator wie G
- Wenn der Algorithmus einen Unifikator ausgibt, dann ist dies ein allgemeinsten Unifikator für eine gelöste Form (Satz auf Folie 16)

Damit folgt die Behauptung.

Korrektheit des Unifikationsalgorithmus (5)

Beweis (Fortsetz.): (3) Der Algorithmus terminiert, d.h. erzeugt nach endlich vielen Schritten eine der beiden Ausgaben.

Ansatz: Wir ordnen Unifikationsprobleme so an, dass das aktuelle Problem in jedem Schritt kleiner wird, und argumentieren, dass dies nicht ewig so weitergehen kann.

Dazu definieren wir für jedes Problem G' ein Tripel natürlicher Zahlen $\kappa(G') = \langle v, g, r \rangle$:

- v ist die Zahl der nicht gelösten Variablen in G'
Eine Variable x ist gelöst, wenn sie in G' nur in einer Gleichung $x \doteq t$ vorkommt und dabei nicht in t enthalten ist
- g ist die Gesamtzahl der Vorkommen von Funktionssymbolen, Konstanten und Variablen in G'
- r ist die Zahl der Gleichungen $s \doteq x \in G'$ mit Variable $x \in \mathbf{V}$ auf der rechten Seite

Korrektheit des Unifikationsalgorithmus (4)

Beweis (Fortsetz.): (2) Wenn der Algorithmus „nicht unifizierbar“ ausgibt, dann hat G keinen Unifikator.

Beobachtung: In diesem Fall erzeugt der Algorithmus ein Problem G' , das eine der folgenden Gleichungen enthält:

- (a) $x \doteq t$, wobei die Variable $x \in \mathbf{V}$ in t vorkommt
- (b) $f(\dots) \doteq g(\dots)$ mit $f \neq g$

Begründung: G' muss eine Gleichung $s \doteq u$ enthalten, die nicht in gelöster Form ist.

- Fall 1: $s \doteq u$ hat auf einer Seite ein Variable. Dann hat sie die Form (a), da sonst Orientierung oder Löschen möglich wäre.
- Fall 2: $s \doteq u$ hat auf keiner Seite eine Variable. Dann hat sie die Form (b), da sonst Zerlegung möglich wäre

Aus der Beobachtung folgt (2), da Gleichungen (a) und (b) keinen Unifikator haben. Laut (\ddagger) hat damit auch G keinen Unifikator.

Korrektheit des Unifikationsalgorithmus (6)

Beweis (Fortsetz.): (3) Der Algorithmus terminiert, d.h. erzeugt nach endlich vielen Schritten eine der beiden Ausgaben.

Wir ordnen Unifikationsprobleme G' **lexikographisch** bezüglich der Tripel $\kappa(G')$.

Für G_1 und G_2 mit $\kappa(G_i) = \langle v_i, g_i, r_i \rangle$ gilt $G_1 > G_2$, falls:

- $v_1 > v_2$ oder
- $v_1 = v_2$ und $g_1 > g_2$ oder
- $v_1 = v_2$ und $g_1 = g_2$ und $r_1 > r_2$.

Beispiel: $\langle 4, 2, 1 \rangle > \langle 3, 42, 23 \rangle > \langle 3, 42, 22 \rangle > \langle 3, 41, 1000 \rangle$.

In dieser Ordnung gibt es keine unendlichen absteigenden Ketten immer kleiner werdender Unifikationsprobleme: Die Ordnung ist **wohlfundiert**. (ohne Beweis)

Korrektheit des Unifikationsalgorithmus (7)

Beweis (Fortsetz.): (3) Der Algorithmus terminiert, d.h. erzeugt nach endlich vielen Schritten eine der beiden Ausgaben.

Die Behauptung folgt nun, da jede Regelanwendung zu einem $>$ -kleineren Unifikationsproblem führt:

- v : Zahl der nicht gelösten Variablen
- g : Gesamtzahl von Funktionssymbolen, Variablen und Konstanten
- r ist die Zahl der Gleichungen $s \doteq x \in G'$ mit Variable $x \in \mathbf{V}$

Der Effekt der Regeln ist dabei wie folgt:

	v	g	r	
Löschen	\geq	$>$		Da es keine unendlichen Ketten immer kleinerer Probleme geben kann, muss der Algorithmus irgendwann terminieren
Zerlegen	\geq	$>$		
Orientieren	\geq	$=$	$>$	
Eliminierung	$>$			

Beispiel: Terminierungsordnung

		v	g	r
$\{x \doteq f(a), g(x, x) \doteq g(x, y)\}$	Eliminierung	2	9	0
$\{x \doteq f(a), g(f(a), f(a)) \doteq g(f(a), y)\}$	Zerlegung	1	12	0
$\{x \doteq f(a), f(a) \doteq f(a), f(a) \doteq y\}$	Löschen	1	10	1
$\{x \doteq f(a), f(a) \doteq y\}$	Orientierung	1	6	1
$\{x \doteq f(a), y \doteq f(a)\}$	gelöste Form	0	6	0

Es gilt: $\langle 2, 9, 0 \rangle > \langle 1, 12, 0 \rangle > \langle 1, 10, 1 \rangle > \langle 1, 6, 1 \rangle > \langle 0, 6, 0 \rangle$

Zusammenfassung und Ausblick

Bei der prädikatenlogischen Resolution müssen Atome unifiziert werden

Unifikation von Termen findet mit Hilfe von Substitutionen statt

Der Unifikationsalgorithmus erlaubt uns, allgemeinste Unifikatoren zu berechnen, falls diese existieren

- Was erwartet uns als nächstes?
- Der Resolutionsalgorithmus und seine Korrektheit
 - Herbrand, genialer Mathematiker aber unglücklicher Bergsteiger
 - Logik über endlichen Modellen und ihre praktische Anwendung