

# Unifikation\*

Lukas Bulwahn

21.04.2005

Diese Ausarbeitung basiert auf einem Auszug von Term Rewriting and All That von Franz Baader und Tobias Nipkow [1].

## 1 Einleitung

Unter Unifikation versteht man die Gleichmachung zweier Literale. Die Unifikation ist ein Werkzeug, dass für die Resolution, also das logische Schließen aus Klauseln benötigt wird [3]. Dieses ist die theoretische Grundlage für alle logischen Programmiersprachen, wie z.B. Prolog.

## 2 Grundlagen

Begriffe, wie Signatur, Variablen und Terme, werden im Skript [2] erläutert und sind Grundlage für die weiteren Definitionen.

**Definition 2.1** Sei  $\Sigma$  eine Signatur,  $V$  die Menge der Variablen, und  $T(\Sigma, V)$  die Menge der Terme über  $\Sigma$  und  $V$ . Dann ist eine Funktion  $\sigma : V \rightarrow T(\Sigma, V)$ , so dass nur für endliche viele  $x : \sigma(x) \neq x$  gilt, eine Substitution.

Eine Substitution kann also in der Form  $\sigma = \{x_1 \mapsto \sigma(x_1), \dots, x_n \mapsto \sigma(x_n)\}$  geschrieben werden.

Beispiele:

$\sigma = \{x \mapsto f(y)\}$  und

$\gamma = \{z \mapsto x, x \mapsto z\}$  sind Substitutionen, jedoch ist

$\theta = \{f(a) \mapsto z\}$  keine Substitution.

**Definition 2.2** Eine Substitution kann von der Menge  $V$  auf die Menge aller Terme  $T(\Sigma, V)$  erweitert werden, indem man definiert:  $\hat{\sigma} : T(\Sigma, V) \rightarrow T(\Sigma, V)$  wobei gilt:  $\forall x \in V : \hat{\sigma}(x) = \sigma(x)$  und für alle sonstigen Terme  $s = f(s_1, \dots, s_n) : \hat{\sigma}(s) = f(\hat{\sigma}(s_1), \dots, \hat{\sigma}(s_n))$ .

---

\*Vortrag gehalten im Rahmen der Vorlesung *Perlen der Informatik 2*, TU München, SS 2005.

Beispiel:

$$\widehat{\sigma}(f(x, g(y))) = f(\sigma x, \widehat{\sigma}g(y)) = f(f(y), g(\sigma y)) = f(f(y), g(y))$$

Im Folgenden verwenden wir stillschweigend stets die erweiterte Definition der Substitution.

Zwei Substitutionen können nun durch die aus der Funktionentheorie bekannten Komposition verknüpft werden, welches einer Hintereinanderausführung zweier Substitutionen entspricht. Diese Verknüpfung ist offensichtlich assoziativ und die identische Abbildung über  $T(\Sigma, V)$  entspricht dem neutralen Element.

Bei der Unifikation will man den allgemeinsten Unifikator finden, dazu muss man zunächst den Begriff allgemein definieren:

**Definition 2.3** *Eine Substitution  $\sigma$  heißt allgemeiner als eine Substitution  $\sigma'$  wenn es eine Substitution  $\delta$  gibt, so dass  $\sigma' = \delta\sigma$ . Dann schreiben wir  $\sigma \preceq \sigma'$  und bezeichnen  $\sigma'$  als Instanz von  $\sigma$ .*

Beispiel:

$\sigma = \{x \mapsto f(y)\}$  und  $\sigma' = \{x \mapsto f(a), y \mapsto a\}$ , dann ist  $\sigma \preceq \sigma'$ , da mit der Substitution  $\delta = \{y \mapsto a\}$  gilt:

$$\sigma' = \delta\sigma = \{y \mapsto a\} \{x \mapsto f(y)\} = \{x \mapsto f(a), y \mapsto a\}$$

**Lemma 2.4** *Die Relation  $\preceq$  auf Substitutionen ist eine Quasi-Ordnung.*

Beweis:

Reflexivität:  $\sigma \preceq \sigma$ , da  $\sigma = \delta\sigma$  mit  $\delta$  als Identitätsabbildung.

Transitivität: Seien  $\sigma_2 = \delta_1\sigma_1$  und  $\sigma_3 = \delta_2\sigma_2$ , dann gilt auch  $\sigma_3 = \delta_2\sigma_2 = \delta_2(\delta_1\sigma_1) = (\delta_2\delta_1)\sigma_1$ .

Desweiteren ist  $\sigma \sim \sigma'$  wenn  $\sigma \preceq \sigma'$  und  $\sigma' \preceq \sigma$ .

Die Substitutionen  $\sigma$  und  $\sigma'$  sind nicht unbedingt identisch, jedoch unterscheiden sie sich nur durch eine Umbenennung der Variablen.

Für die folgenden Beweise werden noch die weiteren zwei Definitionen benötigt, die der Vollständigkeit halber hier noch erläutert werden:

**Definition 2.5** *Sei  $\Sigma$  eine Signatur,  $V$  die Menge der Variablen,  $T(\Sigma, V)$  die Menge der Terme über  $\Sigma$  und  $V$  und  $t \in T(\Sigma, V)$ . Dann definiert man  $|t| := 1$  falls  $t = x \in V$  und für  $t = f(t_1, \dots, t_n) : |t| := 1 + \sum_{i=1}^n |t_i|$ .*

Beispiel:

$$|f(x, g(y, z, h(n)))| = 1 + |x| + |g(y, z, h(n))| = 2 + 1 + |y| + |z| + |h(n)| = 5 + 1 + |n| = 7$$

**Definition 2.6** Sei  $x = (x_1, \dots, x_n)$  und  $y = (y_1, \dots, y_n)$  Listen aus  $n$  Elementen, dann ist  $x >_{lex} y \Leftrightarrow \exists k \leq n (\forall i < k : x_i = y_i) \wedge x_k > y_k$ .

Beispiel:

$$(0, 0, 9) <_{lex} (1, 2, 7) <_{lex} (1, 3, 5) <_{lex} (2, 1, 1) <_{lex} (2, 1, 2)$$

### 3 Problemstellung

Beispiel:

- $f(x) =^? f(a)$ , hier ist  $\sigma = \{x \mapsto a\}$  eine Lösung.
- $x =^? f(y)$ ,  $\sigma_1 = \{x \mapsto f(y)\}$ ,  $\sigma_2 = \{x \mapsto f(a), y \mapsto a\}$  und  $\sigma_3 = \{x \mapsto f(x), y \mapsto x\}$  sind mögliche Lösungen.
- $h(x, g(x, x)) =^? h(f(a), g(x, y))$  hat  $\sigma = \{x \mapsto f(a), y \mapsto f(a)\}$  als eine Lösung.
- $f(x) =^? g(x)$  hat keine Lösung.
- $x =^? f(x)$  und  $f(x, x) =^? f(y, g(y))$  haben ebenfalls keine Lösung.

Folgende Definition beschreibt formal das zu lösende Problem bei der Unifikation.

**Definition 3.1** Das Unifikationsproblem ist eine endliche Menge von Gleichungen  $S = \{s_1 =^? t_1, \dots, s_n =^? t_n\}$ . Eine Lösung bzw. ein Unifikator von  $S$  ist eine Substitution  $\sigma$  mit  $\forall i \in \{1, \dots, n\} : \sigma s_i = \sigma t_i$ .  $U(S)$  bezeichnet die Menge aller Unifikatoren von  $S$ .

$S$  ist unifizierbar falls eine Substitution existiert, also  $U(S) \neq \emptyset$ .

**Definition 3.2** Eine Substitution  $\sigma$  ist ein allgemeinsten Unifikator (engl. most general unifier) von  $S$  wenn  $\sigma \in U(S)$  und  $\forall \sigma' \in U(S) : \sigma \preceq \sigma'$ .

Beispiel:

Für  $x =^? f(y)$  möchten wir zeigen, dass  $\sigma = \{x \mapsto f(y)\}$  ein allgemeinsten Unifikator ist. Wir erinnern uns, dass wir bereits für  $\sigma' = \{x \mapsto f(a), y \mapsto a\}$   $\sigma \preceq \sigma'$  gezeigt haben.

Sei  $\theta$  ein beliebiger Unifikator, dann ist zu zeigen, dass  $\sigma \preceq \theta$ . Dazu zeigt, man das für alle Variablen  $\theta = \theta\sigma$  gilt.

Für den Unifikator  $\theta$  gilt  $\theta x = \theta f(y) = \theta\sigma x$ ,  $\theta y = \theta\sigma y$  und außerdem für jede nicht in der Gleichung vorhandene Variable  $\theta z = \theta\sigma z$ .

Betrachten wir nochmal das zweite Beispiel und die Lösung  $\sigma_3$ , so stellt man fest, dass die mehrfache Anwendung von  $\sigma_3$  den Term weiterhin verändert. Im Weiteren möchten wir aber nur Unifikatoren betrachten, die sich wohl verhalten.

**Definition 3.3** Eine Substitution  $\sigma$  ist idempotent wenn  $\sigma = \sigma\sigma$ .

Nicht idempotente Substitutionen wie  $\sigma_3$  sind bei der Wahl des Unifikators ausgeschlossen.

**Theorem 3.4** Falls ein Unifikationsproblem  $S$  eine Lösung hat, dann existiert ein idempotenter allgemeinsten Unifikator.

Dieses Theorem wird nun durch den folgenden Algorithmus bewiesen.

## 4 Der Unifikations-Algorithmus

**Definition 4.1** Ein Unifikationsproblem  $S = \{x_1 =? t_1, \dots, x_n =? t_n\}$  ist in gelöster Form falls alle  $x_i$  paarweise verschiedene Variablen sind, welche nicht mehr in den Termen  $t_i$  vorkommen. In diesem Fall definieren wir die Substitution  $\vec{S} = \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ .

**Lemma 4.2** Ist  $S$  in gelöster Form, dann ist  $\sigma = \sigma\vec{S}$  für alle  $\sigma \in U(S)$ .

Beweis: Sei  $S = \{x_1 = t_1, \dots, x_n = t_n\}$ . Wir zeigen, dass  $\sigma$  und  $\sigma\vec{S}$  sich für alle Variablen gleich verhalten, also  $\forall x \in V : \sigma x = \sigma\vec{S}x$ . Dazu unterscheiden wir nun zwei verschiedene Fälle, nämlich entweder ist eine Variable  $x$  in den zu lösenden Gleichungen enthalten (Fall 1) oder nicht (Fall 2).

1.  $x \in \{x_1, \dots, x_n\}$ : Sei  $x = x_k$  mit  $1 \leq k \leq n$ . Dann ist  $\sigma x = \sigma t_k = \sigma\vec{S}x$ . Das erste Gleichheitszeichen gilt, da  $\sigma \in U(S)$  und damit aus Definition 3.1 stets  $\sigma x_k = \sigma t_k$  gilt. Das zweite Gleichheitszeichen gilt, da die Substitution  $\vec{S}$  die Variable  $x$  genau auf  $t_k$  abbildet.
2.  $x \notin \{x_1, \dots, x_n\}$ : Dann gilt  $\sigma x = \sigma\vec{S}x$ . Dies ist sofort ersichtlich, da die Substitution  $\vec{S}$  nicht vorhandene Variablen  $x$  einfach wieder auf sich selbst abbildet, also  $\vec{S}x = x$ .

□

**Lemma 4.3** Falls  $S$  in gelöster Form ist, ist  $\vec{S}$  ein idempotenter allgemeinsten Unifikator von  $S$ .

Beweis: Die Idempotenz ist einleuchtend, da keine der Variablen  $x_i$  in den Termen  $t_i$  vorkommen. Daher gilt aber auch  $\vec{S}x_i = t_i = \vec{S}t_i$  und damit  $\vec{S} \in U(S)$ . Die zweite Bedingung für einen allgemeinsten Unifikator folgt direkt aus dem vorherigen Lemma.

□

Einen allgemeinsten Unifikator  $\vec{S}$  finden wir also genau dann wenn wir die Gleichungen  $S$  in gelöste Form gebracht haben. Im Folgenden beschreiben wir einen Algorithmus um  $S$  in gelöste Form zu bringen.

Dazu betrachten wir die folgenden vier Regeln:

- Delete:  $\{t =^? t\} \uplus S \implies S$  - Diese Regel entfernt überflüssige Gleichungen aus der Menge  $S$ .
- Decompose:  $\{f(t_1, \dots, t_n) = f(u_1, \dots, u_n)\} \uplus S \implies \{t_1 =^? u_1, \dots, t_n =^? u_n\} \cup S$  - Diese Regel zerlegt eine Gleichung von zwei Termen in neue Gleichungen. Die Anzahl der neu entstehenden Gleichungen entspricht dabei der Stelligkeit von  $f$ .
- Orient:  $\{t =^? x\} \uplus S \implies \{x =^? t\} \cup S$  wenn  $t \notin V$  - Diese Regel vertauscht die Seiten der Gleichung. Dabei ist zu beachten, dass dies nur gemacht wird, falls  $t$  selbst keine Variable ist.
- Eliminate:  $\{x =^? t\} \uplus S \implies \{x =^? t\} \cup x \mapsto t(S)$  wenn  $x \in \text{Var}(S) - \text{Var}(t)$  - Diese Regel wendet eine bereits in gelöste Form gebrachte Gleichung auf die anderen Gleichungen an, um damit die gelöste Variable aus den noch zulösenden Gleichungen zu eliminieren. Die Bedingung beschreibt die Notwendigkeit, dass die Gleichung  $x =^? t$  wirklich in gelöster Form ist.

Beispiele:

- $S = \{h(x, g(x, x)) =^? h(f(a), g(x, y))\}$   
 $\implies \{x =^? f(a), g(x, x) =^? g(x, y)\}$   
 $\implies \{x =^? f(a), g(f(a), f(a)) =^? g(f(a), y)\}$   
 $\implies \{x =^? f(a), f(a) =^? f(a), f(a) =^? y\}$   
 $\implies \{x =^? f(a), f(a) =^? y\}$   
 $\implies \{x =^? f(a), y =^? f(a)\}$

Die gefundene Substitution  $\vec{S} = \{x \mapsto f(a), y \mapsto f(a)\}$  ist also ein allgemeinster Unifikator.

- $S = \{f(x, x) =^? f(y, g(y))\}$   
 $\implies \{x =^? y, x =^? g(y)\}$   
 $\implies \{x =^? y, y =^? g(y)\}$

Hier terminiert der Algorithmus ohne dass  $S$  in gelöste Form gebracht werden konnte. Wie wir später zeigen, gibt es keinen Unifikator für  $S$ .

Der Algorithmus um die Menge der Gleichungen  $S$  in gelöster Form zu bringen, ist die beliebige Anwendung der vier Regeln bis keine mehr anwendbar ist.

Ist  $S$  nun in gelöster Form, dann gebe  $\vec{S}$  aus - welches wie bereits bewiesen ein allgemeinster Unifikator von  $S$  ist - und falls nicht in gelöster Form gebe „fail“ aus.

## 5 Termination des Algorithmus

**Lemma 5.1** *Der Algorithmus terminiert für jede Eingabe.*

Beweis: Wir bezeichnen eine Variable  $x$  als gelöst, wenn sie genau einmal in  $S$  vorhanden ist und dann auf der linken Seite einer Gleichung der Form  $x = ?t$  ist, wobei  $x \notin \text{Var}(t)$ . Um die Termination nun zu beweisen, betrachten wir folgendes Tripel natürlicher Zahlen  $(n_1, n_2, n_3)$  mit

- $n_1$  ist die Anzahl der noch nicht gelösten Variablen in  $S$ .
- $n_2$  ist die Größe von  $S$ , die wie folgt definiert ist:  $\sum_{s=?t \in S} (|s| + |t|)$
- $n_3$  ist die Anzahl der Gleichungen  $t = ?x$  in  $S$ .

Betrachten wir nun wie die vier Regeln das Tripel verändern, so stellen wir fest, dass das Tripel mit jedem weiteren Schritt lexikographisch kleiner wird. Die Regel „Eliminate“ entfernt eine Variable aus  $S$ , damit wird  $n_1$  kleiner und die anderen Regeln werden  $n_1$  nicht mehr vergrößern.

Die Regeln „Delete“ und „Decompose“ verkleinern  $n_2$  und die Regel „Orient“ verkleinert offensichtlich  $n_3$ . Damit ist auch die Termination bewiesen, denn solange eine Regeln anwendbar ist, wird das Tripel lexikographisch kleiner bis die Form  $(0, n_2, 0)$  erreicht ist, und man die gesuchte Substitution ablesen kann, oder der Algorithmus terminiert ohne gefundene Lösung.

□

## 6 Korrektheit und Vollständigkeit

Für die Korrektheit zeigen wir, dass durch Anwenden der Regeln die Menge der Unifikatoren erhalten bleibt.

**Lemma 6.1** *Wenn  $S \implies T$ , dann ist  $U(S) = U(T)$ .*

Beweis: Dazu betrachten wir nun wieder die vier Regeln:

Für Delete ist offensichtlich, dass die Menge der Unifikatoren erhalten bleibt, gleiches gilt auch für Orient und Decompose.

Betrachten wir nun Eliminate:

Sei  $\theta := \{x \mapsto t\}$ . Nach dem Lemma gilt für die Gleichung in gelöster Form  $x = ?t$ , dass  $\sigma = \sigma\theta$  wenn  $\sigma x = \sigma t$ .

Daher folgt, dass  $\sigma \in U(\{x = ?t\} \uplus S) \leftrightarrow \sigma x = \sigma t \wedge \sigma \in U(S) \leftrightarrow \sigma x = \sigma t \wedge \sigma \theta \in U(S) \leftrightarrow \sigma x = \sigma t \wedge \sigma \in U(\theta S) \leftrightarrow \sigma \in U(\{x = ?t\} \cup \theta S)$ .

□

**Lemma 6.2** *Wenn der Algorithmus eine Substitution  $\sigma$  ausgibt, dann ist  $\sigma$  ein idempotenter allgemeinsten Unifikator von  $S$ .*

Beweis: Die Termination haben wir in Abschnitt 4 gezeigt und die Korrektheit haben wir mit dem vorherigen Lemma bewiesen.

Um die Vollständigkeit zu beweisen, müssen wir folgende zwei Eigenschaften von Termen erläutern.

**Lemma 6.3** *Eine Gleichung  $f(s_1, \dots, s_m) = g(t_1, \dots, t_n)$  mit  $f \neq g$  hat keine Lösung.*

Beweis: Intuitiv ist klar, dass man keine Substitution finden wird, die  $f$  in  $g$  abbildet und daher ist stets  $\sigma(f(s_1, \dots, s_m)) = f(\sigma(s_1, \dots, \sigma(s_m))) \neq g(\sigma(t_1, \dots, \sigma(t_n))) = \sigma(g(t_1, \dots, t_n))$ .

□

**Lemma 6.4** *Eine Gleichung  $x =^? t$  mit  $x \in \text{Var}(t)$  und  $x \neq t$  hat keine Lösung.*

Beweis: Wenn  $x \neq t$ , kann  $t$  nur noch von der Form  $f(t_1, \dots, t_n)$  sein, wobei für mindestens ein  $t_i$   $x$  enthalten muss. Daher kann eine Substitution  $\sigma$  den Term nicht unifizieren, da  $|\sigma(x)| \leq |\sigma(t_i)| < |\sigma(t)|$ .

□

Dies sind genau die zwei Fälle bei dem der Algorithmus nun nicht erfolgreich terminiert, was im folgenden Lemma nun festgehalten wird.

**Lemma 6.5** *Falls  $S$  lösbar ist, scheitert der Algorithmus nicht.*

Beweis: Betrachtet man eine beliebigen Ausgabe  $\vec{S}$ , dann reicht es zu zeigen, dass  $S$  in gelöster Form sein muss.

Welche Eigenschaften hat nun  $S$  bei Termination des Algorithmus?

Nun,  $S$  kann keine Gleichung von der Form  $f(s_1, \dots, s_n) =^? f(t_1, \dots, t_n)$  enthalten, da hier noch die Regel „Decompose“ angewendet werden kann.

Sie kann ebenfalls keine Gleichung der Form  $f(s_1, \dots, s_m) = g(t_1, \dots, t_n)$ , da wir bewiesen haben, dass  $S$  dann nicht lösbar ist, und es existiert auch keine Gleichung der Form  $t =^? x$  mit  $t \notin V$ . Daher muss auf der linken Seite stets eine Variable  $x$  sein.

„Delete“ entfernt ebenfalls alle Gleichungen der Form  $x =^? x$ . Übrig bleibt also nur noch Gleichungen der Form  $x =^? t$ . Diese enthalten nun wegen Lemma 6.4 nun auch keine  $x$  in den rechten Termen  $t$ .

Aufgrund der Regel „Eliminate“ kommt keine Gleichung mehrfach in  $S$  vor und damit befindet sich  $S$  nach Definition in gelöster Form.

□

Das Theorem 3.4 ist damit bewiesen.

Die zwei Fälle, für die  $S$  wegen Lemma 6.3 und 6.4 keine Lösung besitzt, können als entsprechende Regeln in den Algorithmus eingefügt werden:

Clash :  $\{f(s_1, \dots, s_m) = g(t_1, \dots, t_n)\} \uplus S$  mit  $f \neq g \implies \perp$  - Beendet den Algorithmus vorzeitig, da die Situation von Lemma 6.3 eingetreten ist.

Occurs-Check :  $\{x =? t\}$  mit  $x \in Var(t)$  und  $x \neq t \implies \perp$  - Beendet wegen Lemma 6.4.

## Literatur

- [1] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
- [2] T. Nipkow et al. *Skript zur Vorlesung Perlen der Informatik gehalten im WS 2004/05 an der TU München*. 2005.
- [3] P. Rechenberg and G. Pomberger. *Informatik-Handbuch*. Carl Hanser Verlag München Wien, 1997.