Seminar in Informatik: Deklarative Programmierung

Prof. Dr. M. Hanus, Dr. H. Kuchen
Lehrstuhl für Informatik II
RWTH Aachen

Vortrag Nr. 7: Unifikation höherer Ordnung

Hans-Georg Eßer, Matr.-Nr. 184 383
esser@pool.informatik.rwth-aachen.de
Juni 1995 (Version: June 2, 1995)

Diese Ausarbeitung basiert auf dem Artikel *Higher-Order Unification Revisited: Complete Sets of Transformations* von Wayne Snyder und Jean Gallier (Journal of Symbolic Computation (1989) **8**, 101-140).

1. Einleitung und Motivation

In der Mathematik sahen wir im ersten Semester Probleme der folgenden Bauart:

$$x + 3y = 5, 2x - y = 3$$

$$\left(\begin{array}{cc|c} 1 & 3 & 5 \\ 2 & -1 & 3 \end{array}\right) \longrightarrow \left(\begin{array}{cc|c} 1 & 3 & 5 \\ 0 & -7 & -7 \end{array}\right) \longrightarrow \left(\begin{array}{cc|c} 1 & 3 & 5 \\ 0 & 1 & 1 \end{array}\right) \longrightarrow \left(\begin{array}{cc|c} 1 & 0 & 2 \\ 0 & 1 & 1 \end{array}\right)$$

Das Verfahren, das unter dem Namen Gauß-Algorithmus bekannt ist, hat hier die Lösung $x = 2, y = 1$ berechnet. Dies ist eine Lösung, und es ist die einzige Lösung. Das Gauß-Verfahren löst beliebig große *lineare Gleichungssysteme*, und für ein solches System gibt es genau drei Möglichkeiten: es kann unlösbar sein, genau eine oder unendlich viele Lösungen haben.
Eine Formalisierung des Gauß-Verfahrens könnte etwa so aussehen:
Für Matrizen definiere ein System von Transformationen:
(Add) Addiere zur i-ten Zeile ein Vielfaches der j-ten Zeile,
(Mul) Multipliziere eine Zeile mit einer Zahl $\alpha \neq 0$,
(Swap) Vertausche die i-te mit der j-ten Zeile.
Man kann dann einen Algorithmus angeben, der unter ausschließlicher Nutzung dieser Transformationen die Ausgangsmatrix in eine spezielle Diagonalform (nur Einträge in der Hauptdiagonale, dort dürfen nur noch die Zahlen 1 und 0 auftreten) umformt, an der sich dann die Lösungen ablesen lassen.
Für diese Transformationen kann gezeigt werden, daß sie korrekt und vollständig sind, d.h. alle berechneten Lösungen sind tatsächlich Lösungen des Gleichungssystem, und es geht durch die Berechnung keine Lösung verloren.

In ähnlicher Weise betrachten wir in der Informatik *Unifikationsprobleme*: hier versuchen wir, zu zwei Termen e_1, e_2 eine Substitution σ zu finden, die nur Variablen ersetzt, so daß $\sigma(e_1)$ und $\sigma(e_2)$ äquivalent bezüglich einer geeigneten Relation sind.
Wir werden (analog zum Beispiel Gauß-Algorithmus) ein Transformationssystem entwickeln, das solche Substitutionen (genannt Unifikatoren) ausrechnet.

Beispiel 1.1
Seien $e_1 = f(x, f(h, g(x)), x'))$ und $e_2 = f(x, f(h(f(y), z), y'))$.
Wir schreiben diese beiden Terme als Termpaar $\langle f(x, f(h, g(x)), x')), f(x, f(h(f(y), z), y'))\rangle$.
Nun fangen beide Terme mit dem Funktionssymbol f an. Eine Substitution kann dieses Symbol nicht ändern, also gilt: θ ist genau dann ein Unifikator für

$$\langle f(x, f(h, g(x)), x')), f(x, f(h(f(y), z), y'))\rangle,$$

wenn es paarweise die Unterterme (hier: Argumente von f) unifiziert, also

$$\langle x, x\rangle, \langle f(h, g(x)), x'), f(h(f(y), z)\rangle, \langle x', y'\rangle.$$

Aus diesem Beispiel ergibt sich die generelle Transformationsregel

$$\{\ \langle f(u_1, ..., u_n), f(v_1, ..., v_n)\rangle\ \} \cup S \implies \{\ \langle u_1, v_1\rangle, ..., \langle u_n, v_n\rangle\ \} \cup S,$$

die wir Termzerlegung nennen. Wenn wir diese Regel noch zwei mal anwenden, erhalten wir

$$\langle x, x\rangle, \langle x, f(y)\rangle, \langle g(x), z\rangle, \langle x', y'\rangle.$$

Nun ist das Paar $\langle x, x\rangle$ bereits unifiziert, und wir können auf diese Information verzichten. Es ergibt sich

$$\langle x, f(y)\rangle, \langle g(x), z\rangle, \langle x', y'\rangle.$$

Die entsprechende allgemeine Regel ist

$$\{\ \langle u, u\rangle\ \} \cup S \implies S$$

Diese beiden Transformationen vereinfachen ein System, ohne die Menge der Lösungen zu beeinflussen: Die Lösungsmenge ist invariant unter diesen Transformationen.

Jeder mögliche Unifikator muß x auf einen Term abbilden, der mit dem Funktionssymbol f anfängt. Mit anderen Worten: die Bindung für x wird die Form $f(t)$ für einen Term t haben. Wir führen nun eine partielle Bindung für x ein, da wir noch nicht die endgültige Bindung kennen, wir ergänzen also $\langle x, f(x_1)\rangle$ mit einer neuen Variablen x_1:

$$\langle x, f(x_1)\rangle, \langle x, f(y)\rangle, \langle g(x), z\rangle, \langle x', y'\rangle.$$

Nun eliminieren wir x im Rest des Systems, indem wir überall x durch $f(x_1)$ ersetzen:

$$\langle x, f(x_1)\rangle, \langle f(x_1), f(y)\rangle, \langle g(f(x_1)), z\rangle, \langle x', y'\rangle.$$

Nach einem weiteren Termzerlegungsschritt erhalten wir

$$\langle x, f(x_1)\rangle, \langle x_1, y\rangle, \langle g(f(x_1)), z\rangle, \langle x', y'\rangle.$$

Insgesamt nennen wir dies einen Imitationsschritt. Die allgemeine Imitationsregel zum partiellen Auflösen nach einer Variablen ist: Wenn x im Term $f(t_1, ..., t_n)$ nicht auftritt, dann

$$\{\langle x, f(t_1, ..., t_n)\rangle\} \cup S \Longrightarrow \{\langle x, f(y_1, ..., y_n)\rangle, \langle f(y_1, ..., y_n), f(t_1, ...t_n)\rangle\} \cup S',$$

wo die y_i neue Variablen sind, die nirgends auftauchen und $S' = S[f(y_1, ..., y_n)/x]$. (Bemerkung: Wenn x in $f(t_1, ..., t_n)$ auftauchen würde, wäre das System nicht unifizierbar.)

Allgemein: Ähnlich dem Einsetzungsverfahren bei Gleichungssystemen brauchen wir eine Methode, um unser System nach einzelnen Variablen aufzulösen: Wenn wir ein Paar $\langle x, t\rangle$ haben, in dem x in t nicht vorkommt, dann können wir in allen anderen Paaren x durch t ersetzen:

$$\{\langle x, t\rangle\} \cup S \Longrightarrow \{\langle x, t\rangle\} \cup S[t/x]$$

wo $S[t/x]$ erhalten wird, indem jedes Auftreten von x in S durch t ersetzt wird. Wir nennen diese Regel Variablenelimination und erhalten in unserem Beispiel

$$\langle x, f(y)\rangle, \langle x_1, y\rangle, \langle g(f(y)), z\rangle, \langle x', y'\rangle.$$

Wir sagen: das Paar $\langle x, t\rangle$ ist in einem System S gelöst, wenn x weder in t noch in S auftritt. In diesem Sinne ist das letzte System gelöst, da alle Paare gelöst sind. Als Unifikator können wir nun $[f(y)/x, g(f(y))/z, y'/x']$ angeben.

Bis hier haben wir nur Terme erster Ordnung behandelt. Im wesentlichen wollen wir aber Terme höherer Ordnung unifizieren: dies sind Terme, bei denen Variablen auch Funktionen sein können. Dazu benutzen wir den Lambda-Kalkül, der eine Schreibweise für Funktionen erlaubt, ohne diesen einen Namen geben zu müssen. Üblicherweise schreiben wir z.B. $f(x) = x^2$ und reden dann von der Funktion f. Wenn wir nicht den Namen f vergeben wollen, sagen wir umständlich: "die Funktion, die jedes x auf x^2 abbildet". Dafür bietet uns nun der Lambda-Kalkül die Kurzschreibweise: $\lambda x.x^2$. Mit diesen neuen Lambda-Termen können wir ähnlich umgehen wie bisher, z.B.

$$F(y)[\lambda x.x^2/F] = \lambda x.x^2(y) = y^2,$$

wobei das letzte Gleichheitszeichen nicht Termgleichheit bedeutet.

Beispiel 1.2
Wir betrachten $S = \{\langle F(f(a)), f(F(a))\rangle\}$, wobei F eine funktionale Variable ist (z.B. $int \rightarrow int$). Ein Unifikator ist $\theta = [\lambda x.f(x)/F]$:

$$\theta(F(f(a))) = (\lambda x.f(x))f(a) \rightarrow f(f(a)) \leftarrow f((\lambda x.f(x))a) = \theta(f(F(a))),$$

wobei $\rightarrow$ eine Äquivalenzumformung bedeutet, die wir später definieren werden (β-Konversion). Es gibt aber noch weitere Unifikatoren, z.B. $\theta = [\lambda x.f^k(x)/F]$ für $k \geq 0$.

Im Fall erster Ordnung haben wir partielle Bindungen der Form $[f(y_1,...,y_n)/x]$ betrachtet, wo $y_1,...,y_n$ Variablen (erster Ordnung) waren. Wir müssen nun die Imitationsbindung $f(y_1,...,y_n)$ auf den Fall höherer Ordnung verallgemeinern. Dazu betrachten wir partielle Bindungen der Form

$$[\lambda x_1...x_k.a(Y_1(x_1,...,x_k),...,Y_n(x_1,...,x_k))/F],$$

wo $Y_1,...Y_n$ Variablen (höherer Ordnung) sind und a atomar (d.h. Konstante oder Variable) ist. Dabei kann a auch eine funktionale Variable sein, und an die Stelle der y_i treten Ausdrücke $Y_i(x_1,...,x_k)$, da die Unterterme auch Funktionen in $x_1,...x_k$ sein können.

In unserem Beispiel 1.2 hätte eine partielle Bindung für F, die das Symbol f imitiert, die Form $\lambda x.f(Y(x))$, so daß wir $\{\langle F(f(a)), f(F(a))\rangle\}$ in

$$\{\langle F, \lambda x.f(Y(x))\rangle, \langle f(Y(f(a))), f(f(Y(a)))\rangle\}$$

transformieren würden. (Hierbei haben wir bereits einen β- Reduktionsschritt angewandt.) Nach Termzerlegung ergibt sich

$$\{\langle F, \lambda x.f(Y(x))\rangle, \langle Y(f(a)), f(Y(a))\rangle\}.$$

Leider reicht diese Imitationsregel nicht aus, um die Bindungen aufzubauen: Suchen wir nun eine partielle Bindung für Y, stehen wir vor dem gleichen Problem wir bei F, wenn wir immer weiter imitieren, erhalten wir eine nicht abbrechende Folge von Transformationen. Dieses Problem ergibt sich, da Terme höherer Ordnung Variablen als erstes Symbol haben können, und unsere Transformationen müssen Bindungen wie z.B. $\lambda x.x$ finden können. Wenn wir für $\lambda x_1...x_k$ kurz $\lambda\overline{x_k}$ schreiben, erhält unsere neue Regel für das Finden partieller Bindungen (grob) die Form:

$$\{\langle\lambda\overline{x_k}.F(u_1,...,u_n), \lambda\overline{x_k}.a(v_1,...,v_m)\rangle\} \cup S \Longrightarrow$$
$$\{\langle F, t\rangle\} \cup \sigma\Big(\{\langle\lambda\overline{x_k}.F(u_1,...,u_n), \lambda\overline{x_k}.a(v_1,...,v_m)\rangle\} \cup S\Big),$$

wo a ein Funktionssymbol, eine Konstante oder Variable ist, und t entweder eine Imitationsbindung (d.h. $t = \lambda\overline{y_n}.a(Y_1(\overline{y_n}),...,Y_m(\overline{y_n}))$) oder eine Projektionsbindung (d.h. $t = \lambda\overline{y_n}.y_i(Y_1(\overline{y_n}),...,Y_q(\overline{y_n}))$ für ein $i, 1 \leq i \leq n$) ist und $\sigma = [t/F]$.
In unserem Beispiel können wir $\{\langle F(f(a)), f(F(a))\rangle\}$ durch eine Projektionsbindung in

$$\{\langle F, \lambda x.x\rangle, \langle F(f(a)), f(F(a))\rangle\}$$

transformieren. Wenn wir dann die Substitution $[\lambda x.x/F]$ (mit folgender β-Reduktion) anwenden, erhalten wir

$$\{\langle F, \lambda x.x\rangle, \langle f(a), f(a)\rangle\}.$$

Nach Entfernen des trivialen Paars erhalten wir das gelöste System $\{\langle F, \lambda x.x\rangle\}$.
Genauer untersuchen wir diese Problematik in Kapitel 4.

2. Grundlegendes

Definition 2.1: Typen, Typ-Konstruktor $\to$
Es sei $\mathcal{T}_0$ eine Menge von *Grundtypen* (z.B. int, bool, usw.). Dann definieren wir induktiv die Menge der *Typen* $\mathcal{T}$ als die kleinste Menge, die $\mathcal{T}_0$ und mit $\alpha, \beta \in \mathcal{T}$ auch $(\alpha \to \beta)$ enthält. Dabei sei $(\alpha \to \beta)$ der Typ von Funktionen, die Objekte von Typ α auf Objekte vom Typ β abbilden. Wir nehmen an, daß der Konstruktor $\to$ rechtsassoziativ ist, und für $(\alpha_1 \to (\alpha_2 \to ...(\alpha_n \to \beta)...))$ schreiben wir kurz $\alpha_1, ..., \alpha_n \to \beta$.

Definition 2.2: Funktionskonstanten, Variablen, Atome, λ-Terme
Sei Σ eine Menge von Symbolen, die wir *Funktionskonstanten* nennen, jedes Symbol $f \in \Sigma$ habe einen eindeutigen Typ $\tau(f) \in \mathcal{T}$. Zu jedem Typ $\alpha \in \mathcal{T}$ existiere eine abzählbar unendliche Menge V_α von *Variablen* dieses Typs. Setze $V = \bigcup_{\tau \in \mathcal{T}} V_\tau$.
$\mathcal{A} := V \cup \Sigma$ heißt Menge der *Atome.*
Dann ist die Menge $\mathcal{L}$ der λ-*Terme* induktiv definiert durch:
(i) $\mathcal{A} \subseteq \mathcal{L}$
(ii) Regel der Funktionsanwendung:
Für $e_1 \in \mathcal{L}$ vom Typ $\alpha \to \beta$ und $e_2 \in \mathcal{L}$ vom Typ α ist $(e_1 e_2) \in \mathcal{L}$ vom Typ β.
(iii) Für $e \in \mathcal{L}$ vom Typ β und $x \in V_\alpha$ ist $(\lambda x.e) \in \mathcal{L}$ vom Typ $\alpha \to \beta$.
(iv) $\mathcal{L}$ ist die kleinste solche Menge.
Wir bezeichnen mit $\tau(e)$ den Typ eines Terms e.

Funktionsanwendung sei standardmäßig linksassoziativ, so daß $(...(e_1 e_2) e_3)...e_n)$ kurz $(e_1 e_2 ... e_n)$ geschrieben werden kann.
Eine Folge von λ-Abstraktionen $\lambda x_1.(\lambda x_2.(...\lambda x_n.e)...))$ schreiben wir als $\lambda x_1 x_2 ... x_n.e$, wo e Funktionsanwendung oder atomar ist.
Klammern werden wir weglassen, wenn dabei die Bedeutung klarbleibt.
Der Punkt bezieht soviel Rechtskontext wie möglich mit ein, so daß z.B. $\lambda x.stu$ als $(\lambda x.((st)u))$ interpretiert wird.

Definition 2.3: Matrix, Binder, Termgröße, Scope, bindend, gebunden, frei
In einem Term $\lambda x_1 ... x_n.e$ (e entweder Funktionsanwendung oder Atom) nennen wir e die *Matrix* des Terms und $\lambda x_1 ... x_n$ den *Binder* des Terms (also "Term" = "Binder"."Matrix"). $x_1, ..., x_n$ treten in diesem Term *bindend* auf.
Die *Größe* $|u|$ eines Terms u ist die Anzahl der atomaren Unterterme von u.
Eine Variable x tritt *gebunden* im Term e auf, wenn e einen Unterterm $\lambda x.e'$ enthält. In diesem Fall heißt e' der *Gültigkeitsbereich* oder *Scope* dieses bindenden Auftretens von x.
Eine Variable x tritt *frei* im Term e auf, wenn x ein Unterterm von e ist, aber nicht im Scope eines bindenden Auftretens von x auftritt. Die Menge der freien Variablen von e heißt $FV(e)$.

Definition 2.4: Ordnung, Sprache der Ordnung n
Für Typen $\varphi \in \mathcal{T}$ ist die *Ordnung* $Ord(\varphi)$ induktiv definiert als:
$Ord(\varphi) := 1$, falls $\varphi \in \mathcal{T}$;
$Ord(\varphi) := max(Ord(\alpha) + 1, Ord(\beta))$, falls $\varphi = \alpha \to \beta$.

Die *Ordnung* einer Funktionskonstante oder Variablen ist die Ordnung ihres Typs. Eine *Sprache der Ordnung* n ist eine Sprache, in der Funktionskonstanten höchstens Ordnung $n+1$ und (freie oder gebundene) Variablen höchstens Ordnung n haben.
Dies ist eine Verallgemeinerung der Konvention, daß ein Term erster Ordnung eine Variable oder Konstante ist, ein Term zweiter Ordnung eine Funktion von Variablen (erster Ordnung), etc.

Konvention
Im folgenden stehen $\alpha, \beta, \gamma, \varphi$ stets für Typen, b, c für Konstanten eines Grundtyps, f, g, h für Konstanten vom Funktionstyp (2. Ordnung), x, y, z für Variablen beliebigen Typs und a für Atome. Freie Variablen vom Funktionstyp nennen wir F, G, H, Y. λ-Terme heißen e, r, s, t, u, v, w. Wenn aus dem Zusammenhang klar wird, von welchem Typ ein Ausdruck ist, lassen wir diese Information weg.

Die "Rechenregeln" des Lambda-Kalküls sind die folgenden:

Definition 2.5: Lambda-Konversion, Redex, Normalform, Äquivalenz
Für Terme u, t und eine Variable x sei $u[t/x]$ der Term, der aus u entsteht, wenn man in u jedes freie Auftreten von x durch t ersetzt; $BV(t)$ sei die Menge der gebundenen Variablen in t. Dann gibt es die folgenden drei Regeln der Lambda-Konversion:
(i) α-Konversion:

$$y \notin FV(t) \cup BV(t) \;\Rightarrow\; (\lambda x.t) \succ_\alpha (\lambda y.(t[y/x]))$$

(d.h. wenn y als Variable in t nicht vorkommt, dann kann x durch y ersetzt werden. In üblicher Notation wird durch $f(x) = x^2 - x$ dieselbe Funktion definiert wie durch $f(y) = y^2 - y$.)
(ii) β-Konversion:

$$((\lambda x.s)t) \succ_\beta s[t/x]$$

(d.h.: $s(x)|_{x=t}$ und $s(t)$ sind gleich.)
(iii) η-Konversion:

$$x \notin FV(t) \;\Rightarrow\; (\lambda x.(tx)) \succ_\eta t$$

(d.h. t ist genau die Funktion, die jedem x den Wert $t(x)$, oder in unserer Schreibweise (tx) zuordnet.)
Die Terme auf der linken Seite heißen *Redexe* . Ein Term t, der keinen β-Redex enthält, heißt *β-Normalform* , $\eta-$ und $\beta\eta$-Normalformen werden ähnlich definiert.
Falls $e[s]$ ein Lambda-Term e mit Unterterm s an einer festen Stelle ist und $e[t]$ den Term bezeichnet, der aus $e[s]$ hervorgeht, indem man diesen Unterterm s durch t ersetzt, wobei s und t vom gleichen Typ sind (formal: $e[s] = a_1a_2...a_msb_1b_2...b_n, e[t] = a_1a_2...a_mtb_1b_2...b_n, \tau(s) = \tau(t)$), dann können wir die Relation $\rightarrow_\alpha$ definieren durch

$$e[s] \rightarrow_\alpha e[t] \quad\Longleftrightarrow\quad s \succ_\alpha t,$$

genauso definieren wir Relationen $\rightarrow_\beta$ und $\rightarrow_\eta$. Außerdem definieren wir $\rightarrow_{\beta\eta} := \rightarrow_\beta \cup \rightarrow_\eta$ (d.h. $e_1 \rightarrow_{\beta\eta} e_2 \iff e_1 \rightarrow_\beta e_2 \vee e_1 \rightarrow_\eta e_2$). Dann erklären wir für jede dieser Relationen $\rightarrow$ auf übliche Weise die symmetrische Hülle $\leftrightarrow$, die transitive Hülle $\rightarrow^+$, und die symmetrische, reflexive und transitive Hülle $\overset{*}{\longleftrightarrow}$. Die Relationen $\overset{*}{\longleftrightarrow}_\beta$, $\overset{*}{\longleftrightarrow}_\eta$ und $\overset{*}{\longleftrightarrow}_{\beta\eta}$ heißen β-, η- und *$\beta\eta$-Äquivalenz.*

Man zeigt leicht, daß diese Regeln der Lambda-Konversion typerhaltend sind.

Definition 2.6: substituierbar
Ein Term s heißt *für x in t substituierbar*, wenn für jeden Unterterm $\lambda y.t'$ von t gilt: $y \in FV(s) \Rightarrow x \notin FV(t')$.

(Wenn man Ersetzungen $t[s/x]$ zuläßt, wo s nicht für x in t substituierbar ist, gibt es Probleme mit der β-Konversion.)

Konvention
Im folgenden betrachten wir nur Terme, für die die Mengen der freien Variablen und der gebundenen Variablen disjunkt sind; alle Vergleiche von Lambda-Termen werden modulo α-Konversion erfolgen (d.h. wir interessieren uns nicht für simple Umbenennungen von Variablen). Dies erlaubt uns die Darstellung von Lambda-Bindern mit generischen Variablen $x_1, ..., x_k$, wo dies keine Verwirrung erzeugt.
Außerdem ändern wir unsere Notation für β-Konversion dahingehend ab, daß wir die k Schritte in

$$(\lambda x_1...x_k.u)v_1...v_k \rightarrow^k_\beta u[v_1/x_1, ..., v_k/x_k]$$

als einen Schritt betrachten und

$$(\lambda x_1...x_k.u)v_1...v_k \rightarrow_\beta u[v_1/x_1, ..., v_k/x_k]$$

schreiben.

Definition 2.7: getypter β- bzw. $\beta\eta$-Kalkül
Unter dem *getypten β-Kalkül* bzw. *getypten $\beta\eta$-Kalkül* verstehen wir den Kalkül, in dem nur die β-Regel bzw. β- und η-Regel zugelassen sind.

Der Quelltext gibt eine abstrakte Methode zur Unifikation höherer Ordnung, die die grundlegenden logischen Themen so klar wie möglich präsentiert. Es wird nur Unifikation von Termen im getypten $\beta\eta$-Kalkül entwickelt.

Es folgen zwei der wesentlichen Ergebnisse, die diesen Kalkül betreffen.

Satz 2.8: Starke Normalisierung
Jede Folge von $\beta\eta$-Reduktionen ist endlich.
(d.h., daß ausgehend von einem Term t_0 für jede Folge von Reduktionsschritten $t_0 \rightarrow_{\beta\eta} t_1 \rightarrow_{\beta\eta} ... \rightarrow_{\beta\eta} t_i \rightarrow_{\beta\eta} ...$ ein n existiert, so daß t_n keinen $\beta\eta$-Redex mehr enthält, und damit bricht die Folge nach dem n-ten Schritt ab.)

Beweis: Für η-Reduktion ist anschaulich klar, daß es nur endlich viele Reduktionsschritte geben kann, da sich der Term mit jeder Anwendung verkürzt: $|(\lambda x.(tx))|_Z = |t|_Z + 8$. (Hier

ist $|t|_Z$ die Anzahl der Zeichen in t)
Zur β-Reduktion: Wenn der Term t Klammertiefe n hat, dann hat $((\lambda x.s)t)$ Klammertiefe $n+1$. Andererseits hat $s[t/x]$ nur Klammertiefe n, d.h. mit jedem β-Reduktionsschritt wird die Klammertiefe des Terms, auf den die Reduktion angewandt wird, um 1 erniedrigt. Dies kann aber nur endlich oft geschehen, so daß auch Folgen von β-Reduktionen nur endlich sein können. $\boxed{\text{Q.E.D.}}$

Satz 2.9: Satz von Church und Rosser
Wenn zwei Lambda-Terme s und t $\beta\eta$-äquivalent sind ($s \overset{*}{\longleftrightarrow}_{\beta\eta} t$), dann gibt es einen Term u, so daß $s \rightarrow^*_{\beta\eta} u$ und $t \rightarrow^*_{\beta\eta} u$.
(d.h.: wenn zwei Terme $\beta\eta$-äquivalent sind, dann können sie auf den gleichen Term reduziert werden.)

Wir erhalten daraus das Resultat, daß es zu jedem Term eine (bis auf α-Konversion) eindeutige $\beta\eta$-Normalform gibt. Man kann zwei Terme auf Äquivalenz testen, indem man ihre Normalformen ausrechnet und vergleicht.

Sätze 2.8 und 2.9 gelten entsprechend für β-Konversion bzw. η-Konversion.

Beispiel, Def.: β-Normalform
Für einen Term t sei $t\downarrow$ seine eindeutige β-Normalform. Dann gilt: $s \overset{*}{\longleftrightarrow}_{\beta} t \iff s\downarrow = t\downarrow$.

Konvention, Def.: Kopf
Wir setzen i.a. voraus, daß Terme in β-Normalform sind, wenn nichts anderes erwähnt wird. Jeder Term in β-Normalform kann in der Form $\lambda x_1...x_n(ae_1...e_m)$ (für ein $n \geq 0$) dargestellt werden, wo der *Kopf* a atomar ist, d.h. a ist eine Funktionskonstante, eine gebundene Variable oder eine in diesem Term freie Variable, und die Terme $e_1, ..., e_m$ haben die gleiche Form. In Analogie zur Notation der Terme erster Ordnung schreiben wir einen solchen Term $\lambda x_1...x_n.a(e_1, ..., e_m)$. Wir führen außerdem eine Kurzschreibweise ein, die an die Notation von Vektoren angelehnt ist: $\lambda x_1...x_n.e = \lambda\overline{x_n}.e$ und $\lambda\overline{x_n}.f(e_1, ..., e_m) = \lambda\overline{x_n}.f(\overline{e_m})$.

Definition 2.10: starr, flexibel
Ein Term, dessen Kopf eine Funktionskonstante oder eine gebundene Variable ist, heißt ein *starrer* Term. Ist der Kopf eine freie Variable, heißt er ein *flexibler* Term.

Beispiel: Der Term $\lambda x.F(\lambda y.y(x, a), c)$ ist flexibel, aber sein Unterterm $\lambda y.y(x, a)$ ist starr.

Wie bereits erwähnt, beschäftigen wir uns hier nur mit der Unifikation von Termen im $\beta\eta$-Kalkül, und da wir im folgenden untersuchen, auf welche Weise Substitution und folgende β-Reduktion zwei Terme identisch machen, brauchen wir nicht explizit auf die Rolle der η-Reduktion eingehen. Die formale Rechtfertigung dafür ist das nächste Lemma:

Lemma 2.11
Für zwei Terme s und t gilt: $s \rightarrow^*_{\beta\eta} t \iff \exists u\ s \rightarrow^*_{\beta} u \rightarrow^*_{\eta} t$.

Dies ermöglicht uns, in zwei Schritten auf $\beta\eta$-Äquivalenz zu testen: (1) Reduktion der Terme auf β-Normalform und (2) Test dieser Normalformen auf η-Äquivalenz ($s \xleftrightarrow{*}_{\beta\eta} t \iff s\!\downarrow \xleftrightarrow{*}_{\eta} t\!\downarrow$). Somit läßt sich η-Konversion "ausfaktorisieren", indem wir η-Äquivalenzklassen von Termen betrachten. Im folgenden suchen wir nach einer Darstellung dieser Klassen durch kanonische Vertreter.

Definition 2.12: η-expandiert
Sei $e = \lambda x_1 \ldots x_n.a(e_1, \ldots, e_m)$ ein Term in β-Normalform vom Typ $\alpha_1, \ldots, \alpha_n, \alpha_{n+1}, \ldots, \alpha_{n+k} \rightarrow \beta$ mit Grundtyp $\beta \in \mathcal{T}_0$. Die *η-expandierte Form* von e, bezeichnet mit $\eta[e]$, erhält man durch Hinzunahme k neuer Variablen passenden Typs zu Binder und Matrix des Terms und rekursives Anwenden der selben Entwicklung auf die Unterterme, so daß man

$$\eta(e) = \lambda x_1 \ldots x_n x_{n+1} \ldots x_{n+k}.a(\eta[e_1], \ldots, \eta[e_m], \eta[x_{n+1}], \ldots, \eta[x_{n+k}])$$

erhält, wo $\tau(x_{n+i}) = \alpha_{n+i}$ für $1 \leq i \leq k$.

Dies ist die Normalform unter der inversen Operation zur η-Reduktion (so daß $\eta[e] \rightarrow_{\eta}^{*} e$) und ist *nur* definiert, wenn e bereits in β-Normalform ist. Man kann zeigen, daß in η-expandierter Form jedes Atom auf so viele Argumente angewandt wird, wie sein Typ erlaubt, und daß die Typen der Matrizen aller Unterterme Grundtypen sind. Diese Form ist nützlicher als die η-Normalform, weil so die Typen des Terms und aller Unterterme expliziter sind. Daher ist es eine angenehme syntaktische Konvention für die Darstellung der Kongruenzklassen aller Terme, die modulo der η-Regel gleich sind. Durch Terminduktion kann man zeigen, daß diese expandierten Terme immer existieren und (bis auf α-Konversion) eindeutig sind, so daß für zwei Terme s und t in β-Normalform gilt: $s \xleftrightarrow{*}_{\eta} t \iff \eta[s] = \eta[t]$. Damit erhält der Satz von Church und Rosser die folgende Form:

Satz 2.13:
Für Terme s und t gilt: $s \xleftrightarrow{*}_{\beta\eta} t \iff \eta[s\!\downarrow] = \eta[t\!\downarrow]$.

Definition 2.14: $\mathcal{L}_{exp}, \mathcal{L}_{\eta}$
$\mathcal{L}_{exp}$ sei die Menge aller η-expandierten Formen, d.h. $\mathcal{L}_{exp} := \{\ \eta[e\!\downarrow] \mid e \in \mathcal{L}\ \}$. $\mathcal{L}_{\eta}$ ist die kleinste Teilmenge von $\mathcal{L}$, die $\mathcal{L}_{exp}$ enthält und unter Funktionsanwendung und Lambda-Abstraktion abgeschlossen ist, d.h. für $e_1, e_2 \in \mathcal{L}_{\eta}$ sind auch $(e_1 e_2)$ und $\lambda x.e_1$ in $\mathcal{L}_{\eta}$.

Die wesentlichen Eigenschaften von $\mathcal{L}_{exp}$ und $\mathcal{L}_{\eta}$, die uns erlauben, uns auf η-expandierte Formen zu beschränken, sind:

Lemma 2.15
Für jede Variable x und jedes Paar e, e' von Termen passenden Typs gilt:
(1) $e, e' \in \mathcal{L}_{exp} \implies (\lambda x.e) \in \mathcal{L}_{exp}$ und $(ee')\!\downarrow \in \mathcal{L}_{exp}$;
(2) $e \in \mathcal{L}_{\eta} \implies e\!\downarrow \in \mathcal{L}_{exp}$;
(3) $e, e' \in \mathcal{L}_{\eta} \implies (\lambda x.e) \in \mathcal{L}_{\eta}$ und $(ee') \in \mathcal{L}_{\eta}$;
(4) $e \in \mathcal{L}_{\eta}, e \rightarrow_{\beta}^{*} e' \implies e' \in \mathcal{L}_{\eta}$;

(5) $e, e' \in \mathcal{L}_\eta \implies e'[e/x] \in \mathcal{L}_\eta$.

Diese Abschlußeigenschaften von $\mathcal{L}_\eta$ (von denen nicht alle von der Menge der η-Normalformen erfüllt werden) erlauben formal, daß wir die η-Regel in den folgenden Abschnitten implizit lassen, indem wir unsere Methode der Unifikation höherer Ordnung in der Sprache $\mathcal{L}_\eta$ entwickeln, und daß wir explizit β-Konversion nur als Berechnungsregel betrachten.

Wir formalisieren nun den generellen Begriff der *Substitution* von Lambda-Termen für freie Variablen im $\beta\eta$-Kalkül. Danach zeigen wir, wie diese über $\mathcal{L}_{exp}$ spezialisiert werden kann.

Definition 2.16: Substitution, Träger, einführen
Eine *Substitution* ist eine (totale) Funktion $\sigma : V \to \mathcal{L}$, so daß $\sigma(x) \neq x$ nur für endlich viele $x \in V$ gilt, und σ typerhaltend ist, d.h. $\tau(\sigma(x)) = \tau(x)$ für alle $x \in V$.
Ist σ eine Substitution, dann heißt $D(\sigma) := \{ x \mid \sigma(x) \neq x \}$ der *Träger (domain, support)* von σ. Die Substitution mit leerem Träger heißt *Identitätssubstitution* oder einfach *Identität* und wird mit Id bezeichnet. Die Menge der von σ *eingeführten* Variablen ist $I(\sigma) := \bigcup_{x \in D(\sigma)} FV(\sigma(x))$.

Ein trickreicher Punkt dieser Definition ist, daß Substitutionen totale Funktionen sind, die fast überall trivial sind (d.h. nur auf einer endlichen Menge von Variablen nicht-trivial).
Wenn der Träger einer Substitution σ die Menge $\{ x_1, ..., x_n \}$ ist und $t_i = \sigma(x_i)$, dann schreiben wir $\sigma = [t_1/x_1, ..., t_n/x_n]$ und $\sigma(u) = u[t_1/x_1, ..., t_n/x_n]$.

Definition 2.17: umbenennend, Einschränkung
Eine Substitution ρ heißt *umbenennend weg von W*, falls $\rho(x)$ für jedes $x \in D(\rho)$ eine Variable ist, $I(\rho) \cap W = \emptyset$ und für $x, y \in D(\rho)$ gilt: $\rho(x) \xleftrightarrow{*}_\eta \rho(y) \implies x = y$. Wenn W unwichtig ist, heißt ρ einfach *umbenennend* . Die *Einschränkung* einer Substitution σ auf eine Menge W', bezeichnet als $\sigma|_{W'}$, ist die Substitution σ', für die gilt: $\sigma' = \sigma$ auf W', $\sigma' = Id$ sonst.
Statt "umbennende Substitution" sagen wir auch kurz *Umbenennung*.

Da $\mathcal{L}$ frei erzeugt ist, hat jede Substitution $\sigma : V \to \mathcal{L}$ eine eindeutige Fortsetzung $\hat{\sigma} : \mathcal{L} \to \mathcal{L}$, die wie folgt definiert wird:

Definition 2.18: Fortsetzung $\hat{\sigma}$
Sei $\sigma_{-x} := \sigma|_{D(\sigma)\setminus\{x\}}$. Für jede Substitution σ definiere die Fortsetzung $\hat{\sigma}$ durch:
(1) $\hat{\sigma}(x) = \sigma(x), \forall x \in V$;
(2) $\hat{\sigma}(a) = a, \forall a \in \Sigma$;
(3) $\hat{\sigma}(\lambda x.e) = \lambda x.\hat{\sigma_{-x}}(e)$;
(4) $\hat{\sigma}((e_1 e_2)) = (\hat{\sigma}(e_1)\hat{\sigma}(e_2))$.

Somit läßt eine Substitution außer den *freien* Variablen eines Terms alles fest. Im folgenden werden wir σ und $\hat{\sigma}$ identifizieren.
Man beachte: Durch unsere Annahme, daß die Mengen der freien und gebundenen Varia-

blen stets disjunkt sind, kann es durch Anwendung einer Substitution nie zu einem "variable capture" kommen.
Man kann leicht zeigen, daß der Typ eines Terms durch Substitutionen nicht geändert wird.

Bemerkung: Es ist wichtig, daß mit $\sigma(e)$ stets das Resultat nach Anwenden der Substitution σ auf e *ohne* β-Reduktion gemeint ist; mit $\sigma(e)\downarrow$ bezeichnen wir das Resultat von Anwendung der Substitution und anschließender β-Reduktion. Diese unübliche Trennung zwischen Substitution und anschließender β-Reduktion ist nützlich, da wir später die genauen Effekte dieser beiden Operationen auf Lambda-Terme untersuchen wollen.

Definition 2.19: Vereinigung, Komposition zweier Substitutionen
Die *Vereinigung* der zwei Substitutionen σ und θ mit disjunkten Trägern ($D(\sigma) \cap D(\theta) = \emptyset$) ist definiert durch:
$(\sigma \cup \theta)(x) = \sigma(x), x \in D(\sigma)$;
$(\sigma \cup \theta)(x) = \theta(x), x \in D(\theta)$;
$(\sigma \cup \theta)(x) = x$, sonst.
Die *Komposition (Hintereinanderausführung)* von σ und θ ist die Substitution $\sigma \circ \theta$, die durch $\sigma \circ \theta(x) := \hat{\theta}(\sigma(x))$ definiert ist. *Achtung:* Die Schreibweise ist anders als gebräuchlich: die am weitesten links stehende Substitution wird zuerst ausgeführt.

Definition 2.20: gleich über W, allgemeiner über W, unabhängig
Es sei W eine Menge von Variablen. Zwei Substitutionen σ, θ heißen *gleich über* W, in Zeichen $\sigma = \theta\ [W]$, falls $\forall x \in W, \sigma(x) = \theta(x)$. σ und θ heißen *β-gleich über* W, in Zeichen $\sigma =_\beta \theta\ [W]$, falls $\forall x \in W, \sigma(x) \overset{*}{\longleftrightarrow}_\beta \theta(x)$ bzw. $\sigma(x)\downarrow = \theta(x)\downarrow$. Die Relationen $=_\eta$ und $=_{\beta\eta}$ werden entsprechend unter Nutzung von $\overset{*}{\longleftrightarrow}_\eta$ und $\overset{*}{\longleftrightarrow}_{\beta\eta}$ definiert.
σ heißt *allgemeiner als θ über* W, in Zeichen $\sigma \leq \theta\ [W]$, falls es eine Substitution η gibt, so daß $\theta = \sigma \circ \eta\ [W]$, und wir definieren $\sigma \leq_\beta \theta\ [W]$, falls es eine Substitution η' gibt, so daß $\theta =_\beta \sigma \circ \eta'\ [W]$. Analog werden $\leq_\eta$ und $\leq_{\beta\eta}$ definiert. Wenn W die Menge aller Variablen ist, lassen wir das "$[W]$" weg.
σ und θ heißen *unabhängig,* falls weder $\sigma \leq_{\beta\eta} \theta$ noch $\theta \leq_{\beta\eta} \sigma$.

Der Vergleich von Substitutionen modulo β-, η- und $\beta\eta$-Konversion wird durch das folgende Lemma gerechtfertigt, welches durch einfache Terminduktion bewiesen werden kann:

Lemma 2.21
Es seien σ und θ zwei beliebige Substitutionen, so daß entweder $\sigma =_\beta \theta$, $\sigma =_\eta \theta$ oder $\sigma =_{\beta\eta} \theta$. Dann gilt für jeden Term u (jeweils) $\sigma(u) \overset{*}{\longleftrightarrow}_\beta \theta(u)$, $\sigma(u) \overset{*}{\longleftrightarrow}_\eta \theta(u)$ bzw. $\sigma(u) \overset{*}{\longleftrightarrow}_{\beta\eta} \theta(u)$.

Wir zeigen nun, daß wir den Substitutionsbegriff völlig mit der Sprache $\mathcal{L}_\eta$ in Zusammenhang bringen können, ohne Unklarheiten zu verursachen.

Definition 2.22: normalisiert
Eine Substitution θ heißt *normalisiert,* falls für jede Variable $x \in D(\theta)$ gilt: $\theta(x) \in \mathcal{L}_{exp}$.

O.B.d.A. können wir annehmen, daß es in keiner normalisierten Substitution eine Bindung $\eta(x)/x$ für eine Variable x gibt. Eine normalisierte, umbenennende Substitution hat die Form $[\eta[y_1]/x_1, ..., \eta[y_n]/x_n]$; wendet man diese Substitution an und β-reduziert dann, so ist der Effekt eine Umbenennung der Variablen $x_1, ..., x_n$ in $y_1, ..., y_n$. Das folgende Korollar von Lemma 2.15 zeigt, warum es sinnvoll ist, normalisierte Substitutionen zu verwenden:

Korollar 2.23
Ist θ eine normalisierte Substitution und $e \in \mathcal{L}_{exp}$, dann gilt $\theta(e) \in \mathcal{L}_\eta$ und $\theta(e)\!\downarrow \in \mathcal{L}_{exp}$.

Wenn σ und θ normalisiert sind, dann gilt $\sigma =_{\beta\eta} \theta \iff \sigma = \theta$. Wenn θ zu θ' normalisiert wird, dann gilt $\theta' =_{\beta\eta} \theta$.

Konvention
Ab jetzt nehmen wir i.a. an, daß Substitutionen normalisiert sind. Dadurch können wir Substitutionen modulo η-Äquivalenz vergleichen ($\rightarrow$ rausfaktorisieren), so daß wir z.B. statt $\leq_{\beta\eta}$ $\leq_\beta$ verwenden können.
Die Komposition zweier normalisierter Substitutionen ist nicht automatisch normalisiert, z.B. ist $[\lambda x.G(a)/F] \circ [\lambda y.y/G] = [\lambda x.((\lambda y.y)a/F, \lambda y.y/G]$, und nicht $= [\lambda x.a/F, \lambda y.y/G]$.

Definition 2.24: idempotent
Eine Substitution σ heißt *idempotent,* falls $\sigma \circ \sigma =_{\beta\eta} \sigma$.

Eine hinreichende Bedingung für Idempotenz gibt das

Lemma 2.25
Eine Substitution σ ist idempotent, wenn $I(\sigma) \cap D(\sigma) = \emptyset$.

Daß wir o.B.d.A. nur mit idempotenten Substitutionen arbeiten können, zeigt das folgende Resultat, welches zeigt, daß jede Substitution (bis auf Umbenennungen) zu einer idempotenten Substitution äquivalent ist.

Lemma 2.26
Für jede Substitution σ und eine Menge von Variablen $W \supset D(\sigma)$ gibt es eine idempotente Substitution σ', so daß $D(\sigma) = D(\sigma'), \sigma \leq_{\beta\eta} \sigma'$ und $\sigma' \leq_{\beta\eta} \sigma$ $[W]$.

Im allgemeinen vereinfacht die Annahme der Idempotenz die Problematik. In einem späteren Abschnitt werden wir spezielle Motivationen für die Benutzung idempotenter Unifikatoren geben.

Bevor wir mit der Transformationsmethode für den Fall erster Ordnung weitermachen, führen wir den Begriff der Multimenge ein.

Definition 2.27: Multimenge, Vereinigung von Multimengen
Eine *Multimenge* über einer Menge A ist eine ungeordnete Folge von Elementen von A, in der ein Element auch mehrmals auftreten darf. Formaler ist eine Multimenge über A eine Funktion $M : A \rightarrow \mathbf{N}$, so daß ein Element $a \in A$ exakt n-mal in M auftritt, falls $M(a) = n$. a gehört nicht zu M, wenn $M(a) = 0$, und wir schreiben $a \in M \iff M(a) > 0$. Die *Vereinigung* von zwei Multimengen M_1, M_2, in Zeichen $M_1 \cup M_2$, ist definiert durch $(M_1 \cup M_2)(a) := M_1(a) + M_2(a)$.

Um Verwechslung von Mengen und Multimengen zu vermeiden, werden wir immer genau angeben, wann ein Objekt eine Multimenge sein soll. Man beachte, daß Mengen-Vereinigung und Multimengen-Vereinigung unterschiedliche Begriffe sind, da z.B. für eine nichtleere Multimenge A gilt: $A \cup A \neq A$.

3. Transformationen im Fall erster Ordnung

Wir werden nun Unifikation von Termen erster Ordnung definieren und eine abstrakte Sichtweise des Unifikationsprozesses als ein System nicht-deterministischer Regeln zur Umformung eines Unifikationsproblems in eine explizite Darstellung seiner Lösung (falls lösbar) präsentieren; im nächsten Abschnitt werden wir dies auf den Fall höherer Ordnung ausdehnen. Dieser elegante Ansatz stammt aus [33], aber tauchte implizit schon in Herbrands Arbeit [25] auf.
Alle Terme in diesem Abschnitt sind Terme erster Ordnung, also gibt es keine Lambda-Abstraktionen, keine Variablen im Kopf von Termen, und für jeden Term t ist $FV(t)$ die Menge *aller* Variablen in t. Jeder Term erster Ordnung ist trivialerweise in $\mathcal{L}_{exp}$.

Unsere Darstellung der Unifikationsprobleme ist die folgende:

Definition 3.1: Term-Paar, Unifikator, Term-System
Ein *Term-Paar* (oder *Paar*) ist eine Multimenge von zwei Termen (Schreibweise $\langle s, t\rangle$). Eine Substitution θ heißt ein *Standard-Unifikator* (oder einfach *Unifikator*) des Paares $\langle s, t\rangle$, falls $\theta(s) = \theta(t)$. Ein *Term-System* (oder *System*) ist eine Multimenge solcher Paare, und eine Substitution θ ist ein Unifikator eines Systems, wenn sie jedes Paar unifiziert. Die Menge der Unifikatoren eines Systems S wird mit $U(S)$ bezeichnet, und wenn S nur aus dem Paar $\langle s, t\rangle$ besteht, bezeichnen wir die Menge seiner Unifikatoren mit $U(s, t)$.

Definition 3.2: mgu (allgemeinster Unifikator)
Eine Substitution σ ist ein *mgu (most general unifier)* oder *allgemeinster Unifikator* eines Systems S, wenn
(i) $D(\sigma) \subseteq FV(S)$;
(ii) $\sigma \in U(S)$;
(iii) Für jedes $\theta \in U(S)$ gilt $\sigma \leq \theta$.

Für unifizierbare Systeme gibt es immer mgu's, und diese sind eindeutig bis auf Umbenennungen. Wir werden daher von *dem* mgu eines Systems sprechen, in Zeichen: $mgu(S)$.

Definition 3.3: gelöst, ungelöst
Ein Paar $\langle x,t\rangle$ ist in *gelöster Form* in einem System S, und x heißt in diesem Paar eine *gelöste Variable*, wenn x eine Variable ist, die nirgendwo anders in S auftaucht; insbesondere $x \notin FV(t)$. Ein System ist in gelöster Form, wenn alle seine Paare in gelöster Form sind. Eine Variable ist *ungelöst*, wenn sie in S auftritt aber nicht gelöst ist.

Man beachte, daß ein System in gelöster Form immer eine *Menge* (keine doppelten Einträge) von Paaren ist. Die Bedeutung von Systemen in gelöster Form zeigt:

Lemma 3.4
Sei $S = \{\ \langle x_1,t_1\rangle, ..., \langle x_n,t_n\rangle\ \}$ in gelöster Form. $\sigma = [t_1/x_1, ..., t_n/x_n]$ ist ein idempotenter mgu von S. Außerdem gilt für jeden Unifikator $\theta \in U(S)$, daß $\theta = \sigma \circ \theta$.

BEWEIS: Für jeden Unifikator θ gilt nach Definition $\theta(x_i) = \theta(t_i) = \theta(\sigma(x_i)), 1 \leq i \leq n$, und für andere x gilt $\sigma(x) = x$, also ebenfalls $\theta(x) = \theta(\sigma(x))$. σ ist offensichtlich ein mgu, und da S in gelöster Form ist, gilt nach Definition $D(\sigma) \cap I(\sigma) = \emptyset$, also ist σ idempotent. $\boxed{\text{Q.E.D.}}$

Streng genommen ist die Substitution σ hier mehrdeutig für den Fall, daß mindestens ein Paar in S aus zwei gelösten Variablen besteht; aber da mgu's als eindeutig bis auf Umbenennungen betrachtet werden und solche Paare beliebig umbenannt werden können, bezeichnen wir diese Substitution mit σ_S. Als Spezialfall gilt $\sigma_\emptyset = Id$.

Wir zerlegen die Suche nach mgu's folgendermaßen:
Wenn $\theta(u) = \theta(v)$, dann ist entweder (i) $u = v$ und keine Unifizierung nötig, oder (ii) $u = f(u_1, ...u_n)$ und $v = f(v_1, ..., v_n)$ für ein Funktionssymbol $f \in \Sigma$, und $\theta(u_i) = \theta(v_i)$ für $1 \leq i \leq n$, oder (iii) u ist eine Variable, die nicht in $FV(v)$ ist oder umgekehrt. Wenn u eine Variable ist, die nicht in $FV(v)$ ist, dann ist $[v/u] \in U(u,v)$ und $[v/u] \leq \theta$. Dehnen wir dies auf Systeme von Paaren aus, haben wir ein Transformationssystem zur Suche nach mgu's:

Definition 3.5: System der Transformationsregeln $\mathcal{ST}$
Sei S ein beliebiges System (evtl. leer), $f \in \Sigma$ und u, v zwei Terme. Dann haben wir die folgenden Transformationen:

$$\{\ \langle u,u\rangle\ \} \cup S \implies S \tag{1}$$

$$\{\ \langle f(u_1, ..., u_n), f(v_1, ..., v_n)\rangle\ \} \cup S \implies \{\ \langle u_1,v_1\rangle, ..., \langle u_n,v_n\rangle\ \} \cup S \tag{2}$$

$$\{\ \langle x,v\rangle\ \} \cup S \implies \{\ \langle x,v\rangle\ \} \cup \sigma(S), \tag{3}$$

wo x eine Variable ist, die in S auftaucht, so daß $x \notin FV(v)$ und $\sigma = [v/x]$.
(Man sollte daran erinnern, daß Systeme Multimengen sind, also sind die hier auftretenden Vereinigungen Multimengen-Vereinigungen. Damit ist die Bedeutung, auf der linken Seite ein einzelnes Paar zu isolieren, daß transformiert werden soll. Dabei entsprechen die Regeln

(1) - (3) den Fällen (i) - (iii) aus der Vorbemerkung. Transformation (2) heißt *Termzerlegung*, und Transformation (3) heißt *Variablenelimination*. Wir schreiben $\theta \in Unify(S)$, wenn es eine Folge von Transformationen

$$S \Longrightarrow \ldots \Longrightarrow S'$$

gibt, so daß S' in gelöster Form ist und $\theta = \sigma_{S'}$. (Wenn keine der Transformationen angewandt werden kann und das System nicht bereits in gelöster Form ist, schlägt die hier angegebene Prozedur fehl.)

Wählt man $S = \{\ \langle u, v\rangle\ \}$, dann kann man mit diesem System einen Unifikator für zwei Terme u, v suchen.

Beispiel 3.6
$\langle f(x, g(a, y)),\ f(x, g(y, x))\rangle$
$\Longrightarrow_2 \langle x, x\rangle, \langle g(a, y), g(y, x)\rangle$
$\Longrightarrow_1 \langle g(a, y), g(y, x)\rangle$
$\Longrightarrow_2 \langle a, y\rangle, \langle y, x\rangle$
$\Longrightarrow_3 \langle a, y\rangle, \langle a, x\rangle.$

Diese Transformationen erhalten die logisch invarianten Eigenschaften eines Unifikationsproblems in folgendem Sinn:

Lemma 3.7
Falls S durch eine Transformation aus $\mathcal{ST}$ in S' übergeht, dann gilt $U(S) = U(S')$.

BEWEIS: Für Regeln (1) und (2) ist dies klar. Zu Regel (3): Sei $\{\ \langle x, v\rangle\ \} \cup S \Longrightarrow_3 \{\ \langle x, v\rangle\ \} \cup \sigma(S)$ mit $\sigma = [v/x]$. Für jede Substitution θ mit $\theta(x) = \theta(v)$ gilt $\theta = \sigma \circ \theta$, da sich $\sigma \circ \theta$ nur an der Stelle x von θ unterscheidet, aber $\theta(x) = \theta(v) = \theta(\sigma(x)) = \sigma \circ \theta(x)$. Daher gilt
$\theta \in U(\{\ \langle x, v\rangle\ \} \cup S)$
$\Longleftrightarrow\ \theta(x) = \theta(v)$ und $\theta \in U(S)$
$\Longleftrightarrow\ \theta(x) = \theta(v)$ und $\sigma \circ \theta \in U(S)$
$\Longleftrightarrow\ \theta(x) = \theta(v)$ und $\theta \in U(\sigma(S))$
$\Longleftrightarrow\ \theta \in U(\{\ \langle x, v\rangle\ \} \cup \sigma(S))$. $\boxed{\text{Q.E.D.}}$

Hierbei ist der entscheidende Punkt, daß die wichtigste Eigenschaft eines Unifikationsproblems, seine Lösungsmenge, unter diesen Transformationen erhalten bleibt. Also ist unsere Methode, ein solches Problem in eine triviales (gelöstes) Form zu transformieren, in der die Existenz eines mgu offensichtlich ist, korrekt.

Wir zeigen nun Korrektheit und Vollständigkeit dieser Transformationen:

Satz 3.8: Korrektheit
Falls $S \Longrightarrow^* S'$ mit S' in gelöster Form, dann gilt $\sigma_{S'} \in U(S)$.

(d.h.: Wenn das Verfahren mit einem gelösten System abbricht, dann ist der ablesbare Unifikator auch Unifikator des Ausgangssystems S.)

BEWEIS: Nach Lemma 3.7 gilt der Satz für je einen Transformationsschritt, durch Induktion also auch für eine Folge solcher Schritte. Also haben wir $U(S) = U(S')$. Daraus folgt die Behauptung. $\boxed{\text{Q.E.D.}}$

Satz 3.9: Vollständigkeit
Sei $\theta \in U(S)$. Dann muß jede Folge von Transformationen

$$S = S_0 \implies S_1 \implies S_2 \implies ...$$

nach endlich vielen Schritten mit einer gelösten Form S' abbrechen, so daß $\sigma_{S'} \leq \theta$.
(d.h. das Verfahren terminiert immer und liefert einen Unifikator, der allgemeiner als jeder beliebige Unifikator, also ein mgu ist.)

BEWEIS: Wir zeigen zunächst, daß jede Transformations-Folge terminiert. Wir definieren ein Komplexitätsmaß μ, daß jedem System S eine Komplexität $\mu(s) = < m, n >$ zuordnet, wo m die Anzahl der ungelösten Variablen des Systems und n die Summe der Größen aller Terme des Systems sind. Über die lexikographische Ordnung (d.h. $< m_1, n_1 >$ kleiner als $< m_2, n_2 >$, falls $m_1 < m_2$ oder $m_1 = m_2$ und $n_1 < n_2$) lassen sich die Systeme anordnen. Jede Transformation überführt ein System in ein neues, das echt kleineres Maß hat: (1) und (2) verkleinern offensichtlich n, da (1) einen Term entfernt und (2) einen Term durch alle seine echten Unterterme ersetzt. (3) verkleinert m, da die Variable x nach (3) gelöst ist. Da es keine unendliche monoton fallende Folge natürlicher Zahlen gibt, bricht das Verfahren also auf jeden Fall ab.
Also bricht jede der obigen Folgen $S = S_0 \implies S_1 \implies S_2 \implies ...$ mit einem System S' ab, in dem keine weiteren Transformationen mehr möglich sind. Wegen $\theta \in U(S)$ und Lemma 3.7 gilt $\theta \in U(S')$, somit kann S' keine Paare der Form $\langle f(t_1, ..., t_n),\ g(t'_1, ...t'_m)\rangle$ oder $\langle x, t\rangle$ mit $x \in FV(t)$ enthalten (die nicht unifizierbar wären). Da keine Transformation mehr anwendbar ist, müssen alle Paare in gelöster Form sein, d.h. S' ist gelöst. Schließlich folgt aus $\theta \in U(S')$ mit Lemma 3.4: $\sigma_{S'} \leq \theta$. $\boxed{\text{Q.E.D.}}$

Setzt man diese zwei Sätze zusammen, sieht man, daß $\mathcal{ST}$ immer einen mgu für ein unifizierbares System von Termen findet.

Tatsächlich haben wir mit Satz 3.9 sogar eine stärkere Aussage gezeigt: Wir haben gezeigt, daß alle Transformationsfolgen terminieren und daß *jede* Folge, die mit einem unifizierbaren System startet, schließlich in einer gelöste Form endet. Dies ist möglich, da das Problem *entscheidbar* ist. Für den Beweis der Vollständigkeit hätte es gereicht zu zeigen, daß zu unifizierbarem S *irgendeine* Folge von Transformationen existiert, die in einer gelösten Form endet, weil dann eine vollständige Suchstrategie (z.B. Breitensuche) eine gelöste Form finden könnte. Diese Form der Vollständigkeit, die man auch *nichtdeterministische Vollständigkeit* nennen kann, werden wir benötigen, um Lösungen bei der Unifikation von Systemen höherer Ordnung zu finden, wo das allgemeine Problem unentscheidbar ist.

Manchmal ist es nützlich, mit idempotenten Unifikatoren zu arbeiten, die umbenennend weg von einer Menge "geschützter" Variablen sind, aber allgemeinst über der Menge der Variablen des Ausgangssystems sind. Die folgende Definition präzisiert dies:

Definition 3.10: mgu weg von W
Es sei S ein System und W eine Menge "geschützter" Variablen. Dann heißt eine Substitution σ ein *mgu von S weg von W* (kurz: $mgu(S)[W]$), falls
(i) $D(\sigma) \subseteq FV(S)$ und $I(\sigma) \cap (W \cup D(\sigma)) = \emptyset$;
(ii) $\sigma \in U(S)$;
(iii) Für jedes $\theta \in U(S)$ gilt $\sigma \leq \theta$ $[FV(S)]$.

(d.h.: ein $mgu(S)[W]$ ist ein idempotenter $mgu(S)$, der keine der Variablen in W einführt.)

Lemma 3.11
Sei S ein unifizierbares System, und W eine geschützte Menge von Variablen. Dann existiert eine Substitution σ, die ein $mgu(S)[W]$ ist.

4. Transformationen im Fall höherer Ordnung

Unifikationsprobleme höherer Ordnung sind komplexer, da es Variablen vom Funktionstyp gibt und wir mit Scope und gebundenen Variablen arbeiten müssen. Aus dieser zusätzlichen syntaktischen Komplexität ergeben sich einige wichtige Konsequenzen: Zunächst ist die Unifikation von Termen höherer Ordnung unentscheidbar. Dann gibt es keine mgu's mehr, und wir müssen den komplexeren Begriff der *vollständigen Menge von Unifikatoren* einführen. Schließlich kann der Suchraum beim Unterproblem, zwei flexible Terme zu unifizieren, unendlich verzweigen, was eine vernünftige Implementierung unmöglich macht.

Definition 4.1: Unifikator
Die Begriffe Paar und System von Termen übertragen sich vom Fall erster Ordnung. Eine Substitution θ ist ein *Unifikator* für zwei Lambda-Terme e_1 und e_2, falls $\theta(e_1) \xleftrightarrow{*}_{\beta\eta} \theta(e_2)$. θ unifiziert ein System S, wenn θ jedes Paar aus S unifiziert. Die Menge aller Unifikatoren für S heißt $U(S)$, und wie bisher schreiben wir kurz $U(s,t)$, falls $S = \{ \langle s,t \rangle \}$.

Diese Definition ist allgemeiner, als wir sie benötigen werden, da wir unseren Ansatz in $\mathcal{L}_\eta$ entwickeln werden, um η-Konversion ausfaktorisieren zu können (vgl. Abschnitt 2). Daher sagen wir für zwei Terme $s,t \in \mathcal{L}_\eta$, daß eine normalisierte Substitution θ in $U(s,t)$ liegt, falls $\theta(s) \xleftrightarrow{*}_{\beta} \theta(t)$ bzw. falls $\theta(s){\downarrow} = \theta(t){\downarrow}$.
Ein Termpaar in einem System S ist gelöst, wenn es die Form $\langle \eta[x], t \rangle$ hat, wobei die Variable x in S nur einmal auftaucht; S ist gelöst, wenn jedes Paar in S gelöst ist. In Abweichung von der Nutzung der η-expandierten Form werden wir Paare der Form $\langle \eta[x], t \rangle$ als $\langle x, t \rangle$ darstellen, um ihre Entsprechung zu Bindungen t/x in Substitutionen (im Fall erster Ordnung) hervorzuheben.

Beispiel 4.2
Sei $u = f(a, g(\lambda x.G(\lambda y.x(b))))$ und $v = F(\lambda x.x(z))$.
Dann liegt $\theta = [\lambda x_2.f(a, g(x_2))/F, \lambda x_3.x_3(z_2)/G, b/z]$ in $U(u, v)$, da $\theta(u)\!\downarrow = \theta(v)\!\downarrow$:
$\theta(u) = f(a, g(\lambda x.G(\lambda y.x(b))))[\lambda x_2.f(a, g(x_2))/F, \lambda x_3.x_3(z_2)/G, b/z]$
$= f(a, g(\lambda x.[(\lambda x_3.x_3(z_2))(\lambda y.x(b))]))$
$\rightarrow_\beta f(a, g(\lambda x.[(\lambda y.x(b))z_2]))$
$\rightarrow_\beta f(a, g(\lambda x.x(b)))$ und
$\theta(v) = F(\lambda x.x(z))[\lambda x_2.f(a, g(x_2))/F, \lambda x_3.x_3(z_2)/G, b/z]$
$= (\lambda x_2.f(a, g(x_2)))(\lambda x.x(b))$
$\rightarrow_\beta f(a, g(\lambda x.x(b)))$

Definition 4.3: Unifikationsproblem (n-ter Ordnung)
Es sei Σ eine Menge von Funktionskonstanten. Dann ist das *Unifikationsproblem* für die von Σ erzeugte Sprache $\mathcal{L}$, für beliebige Terme $e, e' \in \mathcal{L}$ zu entscheiden, ob die Menge $U(e, e')$ nicht-leer ist. Das *Unifikationsproblem n-ter Ordnung* ist, das Unifikationsproblem für eine beliebige Sprache n-ter Ordnung zu entscheiden.

Satz 4.4: Unentscheidbarkeit
Das Unifikationsproblem zweiter Ordnung ist unentscheidbar. (ohne Beweis)

Neben der Unentscheidbarkeit ergibt sich das Problem, daß mgu's nicht mehr existieren müssen. So haben z.B. die zwei Terme $F(a)$ und a die Unifikatoren $[\lambda x.a/F]$ und $[\lambda x.x/F]$, aber es gibt keinen Unifikator, der allgemeiner als beide ist. Dies führt uns zu einer Erweiterung des Begriffs $mgu(S)[W]$ auf die Fälle höherer Ordnung, in der wir *vollständige Unifikatormengen* betrachten.

Definition 4.5: Vollständige Unifikatormenge
Zu einem System S und einer endlichen Menge "geschützter" Variablen W, heißt eine Menge U von normalisierten Substitutionen *vollständige Unifikatormenge für S weg von W* oder kurz $CSU(S)[W]$ (CSU = complete set of unifiers), falls
(i) Für alle $\sigma \in U$ gilt $D(\sigma) \subseteq FV(S)$ und $I(\sigma) \cap (W \cup D(\sigma)) = \emptyset$;
(ii) $U \subseteq U(S)$;
(iii) Für jedes normalisierte $\theta \in U(S)$ gibt es ein $\sigma \in U$, so daß $\sigma \leq_\beta \theta[FV(S)]$.
Die erste Bedingung heißt *Reinheitsbedingung*, die zweite *Kohärenzbedingung* und die letzte *Vollständigkeitsbedingung*. Besteht S nur aus dem Paar $\langle u, v\rangle$, schreiben wir kurz $CSU(u, v)$ $[W]$. Wenn es auf W nicht ankommt, lassen wir "$[W]$" weg.

Daß wir beim ausschließlichen Betrachten normalisierter Substitutionen die Allgemeinheit nicht verlieren, kann man daran sehen, daß jede Substitution zu einer normalisierten Substitution $\beta\eta$-gleich ist. Wir geben nun für diesen neuen Zusammenhang eine Version von Lemma 3.11, die zeigt, daß auch Bedingung (i) die Allgemeinheit nicht einschränkt.

Lemma 4.6
Für ein beliebiges System S, eine Substitution θ und eine Menge W geschützter Variablen gibt es zu jedem $\theta \in U(S)$ eine normalisierte Substitution σ, so daß
(i) $D(\sigma) \subseteq FV(S)$ und $I(\sigma) \cap (W \cup D(\sigma)) = \emptyset$;
(ii) $\sigma \in U(S)$;
(iii) $\sigma \leq_{\beta\eta} \theta[FV(S)]$ und $\theta \leq_{\beta\eta} \sigma[FV(S)]$.

Beweis: Falls $\sigma = \theta|_{FV(S)}$ Bedingung (i) genügt, folgt die Aussage trivial. Anderenfalls, sei für $I(\theta) = \{x_1, ..., x_n\}$ $\{y_1, ..., y_n\}$ eine Menge neuer Variablen, die disjunkt mit den Variablen in $W, I, FV(S)$ ist, so daß $\tau(x_i) = \tau(y_i)$ für $1 \leq i \leq n$. Definiere umbenennende Substitutionen $\rho_1 = [\eta(y_1)/x_1, ..., \eta(y_n)/x_n]$ und $\rho_2 = [\eta(x_1)/y_1, ..., \eta(x_n)/y_n]$, setze $\sigma' = \theta \circ \rho_1|_{FV(S)}$, und sei σ die normalisierte Version von σ'. Dann erfüllt σ sicherlich (i), und wegen $\sigma =_{\beta\eta} \theta \circ \rho_1[FV(S)]$ haben wir den zweiten Teil von (iii). Da nun $\rho_1 \circ \rho_2 =_{\beta\eta} Id[FV(S) \cup I(\theta)]$, folgt $\theta =_{\beta\eta} \theta \circ \rho_1 \circ \rho_2[FV(S) \cup I(\theta)]$. Aus $\sigma =_{\beta\eta} \theta \circ \rho_1[FV(S)]$ folgt dann $\theta =_{\beta\eta} \sigma \circ \rho_2[FV(S)]$, und damit $\sigma \leq_{\beta\eta} \theta[FV(S)]$, was den ersten Teil von (iii) zeigt. Um (ii) zu zeigen, stellen wir fest, daß für jedes Paar $\langle u, v\rangle \in S$ gilt: $\theta(u)\downarrow = \theta(v)\downarrow$, und für jeden Term t gilt $\sigma'(t) \stackrel{*}{\longleftrightarrow}_{\beta\eta} \sigma(t)$, und damit

$$\sigma(u) \stackrel{*}{\longleftrightarrow}_{\beta\eta} \sigma'(u) = \rho_1(\theta(u)) \rightarrow^*_{\beta\eta} \rho_1(\theta(u)\downarrow) = \rho_1(\theta(v)\downarrow) \leftarrow^*_{\beta\eta} \rho_1(\theta(v)) = \sigma'(v) \stackrel{*}{\longleftrightarrow}_{\beta\eta} \sigma(v),$$

was zeigt, daß $\sigma \in U(S)$. $\boxed{\text{Q.E.D.}}$

Dies zeigt, daß für beliebige S und W die Menge aller normalisierenden Unifikatoren, die Bedingungen (i) und (ii) von Definition 4.5 erfüllen, ein $CSU(S)[W]$ ist. Insbesondere verlieren wir nicht die Allgemeinheit, wenn wir im folgenden nur normalisierte idempotente Unifikatoren θ mit $D(\theta) \cap I(\theta) = \emptyset$ betrachten, was unsere Darstellung vereinfachen wird. Zum Schluß untersuchen wir die Bedeutung von System in gelöster Form in $\mathcal{L}_\eta$:

Lemma 4.7
Wenn $S = \{ \langle x_1, t_1\rangle, ..., \langle x_n, t_n\rangle \}$ ein System in gelöster Form ist, dann ist $\{\sigma_S\}$ ein $CSU(S)[W]$ für jedes W mit $W \cap FV(S) = \emptyset$.

Beweis: Die ersten beiden Bedingungen aus Definition 4.5 sind erfüllt, da σ_S ein idempotenter mgu von S ist, $W \cap FV(S) = \emptyset$ und $I(\sigma_S) \subseteq FV(S)$. Ist nun $\theta \in U(S)$, dann gilt $\theta =_\beta \sigma_S \circ \theta$, da $\theta(x_i) \stackrel{*}{\longleftrightarrow}_{\beta\eta} \theta(t_i) = \theta(\sigma_S(x_i))$ für $1 \leq i \leq n$, und $\theta(x) = \theta(\sigma_S(x))$ für andere x. Also $\sigma_S \leq_\beta \theta$, und damit auch $\sigma_S \leq_\beta \theta[FV(S)]$. $\boxed{\text{Q.E.D.}}$

Wir kommen nun zum entscheidenden Teil dieser Ausarbeitung, in dem wir das Transformationssystem $\mathcal{ST}$ aus Kapitel 3 für die Fälle höherer Ordnung verallgemeinern.

4.1 Transformationen für die Unifikation höherer Ordnung

Wir werden den Prozeß der Unifikation höherer Ordnung wie folgt untersuchen. Ohne Einschränkung der Allgemeinheit wollen wir annehmen, daß u und v zwei Lambda-Terme in $\mathcal{L}_{exp}$ sind und daß θ ein idempotenter normalisierter Unifikator für u und v ist. Also gibt es eine Folge von Reduktionen in β-Normalform: $\theta(u) \rightarrow_\beta^* w \leftarrow_\beta^* \theta(v)$. (Man beachte, daß diese Folge trivial ist, wenn alle von dieser Substitution erzeugten Instanzen von erster Ordnung sind, da es dann keine β-Reduktion gibt.) Wir werden diese Folge *top-down* untersuchen, wobei wir die folgenden fünf Fälle unterscheiden:

(A) $u = v$: keine Unifizierung nötig (in den übrigen Fällen gelte also $u \neq v$.)

(B) In beiden Termen verändert keine Substitution den Kopf:
Dann gilt $Head(u) = Head(v)$, und wegen $u \neq v$ haben wir $|u|, |v| > 0$. Es seien $u = \lambda\overline{x_k}.a(\overline{u_n})$, $w = \lambda\overline{x_k}.a(\overline{w_n})$ und $v = \lambda\overline{x_k}.a(\overline{v_n})$, wobei $n > 0$ und entweder $a \in \Sigma$, $a = x_i$ für ein $i : 1 \leq i \leq k$, oder a ist eine freie Variable mit $a \notin D(\theta)$. Es muß dann gelten: $\theta(\lambda\overline{x_k}.u_i) \rightarrow_\beta^* \lambda\overline{x_k}.w_i \leftarrow_\beta^* \theta(\lambda\overline{x_k}.v_i)$ für $1 \leq i \leq n$, d.h. die Unterterme von u und v werden paarweise von θ unifiziert.

(C) Die Terme haben die Form $u = \lambda\overline{x_k}.F(\overline{x_k})$, $v = \lambda\overline{x_k}.v'$, für eine Variable F und einen Term v', wobei $F \notin FV(v)$:
In diesem Fall muß gelten $\theta(\lambda\overline{x_k}.F(\overline{x_k})) \stackrel{*}{\longleftrightarrow}_\beta \theta(\lambda\overline{x_k}.v')$ mit $F \notin FV(v)$, und wenn $\theta = [\lambda\overline{y_k}.t/F] \cup \theta'$, dann gilt wegen $\theta(F) \leftarrow_\beta \theta(\lambda\overline{x_k}.F(\overline{x_k}))$, daß $\theta(F) \stackrel{*}{\longleftrightarrow}_\beta \theta(\lambda\overline{x_k}.v')$, wobei F nicht in v' auftritt. Somit können wir dieselbe Argumentation wie im Fall erster Ordnung anwenden: Setzen wir $\sigma = [\lambda\overline{x_k}.v'/F]$, dann gilt $\theta =_\beta \sigma \circ \theta$, da σ und $\sigma \circ \theta$ sich nur bei F unterscheiden, aber $\theta(F) \stackrel{*}{\longleftrightarrow}_\beta \theta(\lambda\overline{x_k}.v') = \sigma \circ \theta(F)$.
Dies zeigt, daß ein Termpaar dieser Form einen mgu besitzt. (z.B. werden $\lambda x.F(x)$ und $\lambda x.f(x,z)$ durch $\theta = [\lambda y.f(y,a)/F,\ a/z]$ unifiziert, aber $\sigma = [\lambda y.f(y,z)/F]$ ist ein mgu.) Dies ist eine Verallgemeinerung der Variablenelimination für den Fall höherer Ordnung, da u (bis auf η-Äquivalenz) einfache eine Variable ist, die nicht in $FV(v)$ auftaucht.

(D) Im Kopf nur eines der Terme findet eine Substitution statt: Dieser Term sei o.B.d.A. u (d.h. $Head(w) = Head(v)$). Dann sei $u = \lambda\overline{x_k}.F(\overline{u_n})$, $v = \lambda\overline{x_k}.a(\overline{v_m})$ für ein Atom $a \neq F$, welches entweder eine Funktionskonstante, eine gebundene Variable oder eine freie Variable $\notin D(\theta)$ ist. Um die beiden Terme zu unifizieren, muß an einer Stelle der β-Reduktionsfolge von $\theta(u)$ nach w der Kopf von u zu a werden. Dafür gibt es zwei Möglichkeiten: Entweder Imitation des Kopfes von v durch Substitution eines Termes für F, dessen Kopf a ist, oder Substitution eines Termes für F, der auf einen Unterterm von v projiziert (d.h. es gibt in v einen Unterterm, dessen Kopf dann gematcht werden kann). Projektion ist nur möglich, wenn F vom Typ höherer Ordnung ist.) Wir betrachten im folgenden diese beiden Fälle.

Imitation: Die Substitution für F matcht das Kopfsymbol von v durch Imitieren des Kopfsymbols a, wo $a \in \Sigma$ oder a eine freie Variable $\notin D(\theta)$ ist (wie in Beispiel 4.2: $u = f(a, g(\lambda x.G(\lambda y.x(b))))$, $v = F(\lambda x.x(z))$, $\theta = [\lambda x_2.f(a, g(x_2))/F, \lambda x_3.x_3(z_2)/G, b/z]$). Also gilt $\theta(F) = \lambda\overline{z_n}.a(\overline{r_m})$ für Terme $\overline{r_m}$, und wir haben eine Reduktionsfolge der Form

$$\theta(u) = \theta(\lambda\overline{x_k}.((\lambda\overline{z_n}.a(\overline{r_m}))\overline{u_n})) \rightarrow_\beta \theta(\lambda\overline{x_k}.a(\overline{r'_m}) \xleftrightarrow{*}_\beta \theta(\lambda\overline{x_k}.a(\overline{v_m})),$$

wo $r'_i = r_i[u_1/z_1, ..., u_n/z_n]$ für $1 \leq i \leq m$. (Wegen der Idempotenz von θ instanziieren wir in dieser Folge zur Anschauung den Term u nur teilweise mit der Bindung für den Kopf F.)

Projektion: Die Substitution für F versucht, das Kopfsymbol a von v durch Projektion auf einen Unterterm von u zu matchen. Es gibt drei Arten, dies zu tun, in Abhängigkeit vom Kopfsymbol des Terms, auf den projiziert wird.
Zunächst könnte ein Unterterm von u den Kopf a besitzen und das Matchen erlauben: z.B. werden $F(\lambda x.f(x,c))$ und $f(b,c)$ durch die Substitution $[\lambda y.y(b)/F]$ (hier ist also f das Kopfsymbol) unifiziert.
Die zweite Möglichkeit für eine Projektion ist, daß vielleicht ein Unterterm von u flexibel ist (d.h. sein Kopf ist eine freie Variable) – dann können wir versuchen, diesen Kopf mit v zu matchen. Z.B. werden $F(\lambda x.G(x,a))$ und b durch $[\lambda y.y(b)/F, \lambda x_1 x_2.x_1/G]$ unifiziert.
Die dritte Möglichkeit ist, daß der Unterterm selbst eine Projektion ist, und nach einer Reihe von Reduktionsschritten entsteht ein Term, der entweder flexibel ist (dann weitermachen) oder dessen Kopf a ist, so daß wir matchen können. Z.B. unifiziert $\theta = [\lambda y_1.y_1(\lambda y_2.y_2(a))/F]$ die beiden Terme $u = F(\lambda x_1.x_1(\lambda x_2.f(x_2)))$ und $v = f(a)$ auf folgende Art:
$\theta(u) = [\lambda y_1.y_1(\lambda y_2.y_2(a))]\lambda x_1.x_1(\lambda x_2.f(x_2))$
$\rightarrow_\beta [\lambda x_1.x_1(\lambda x_2.f(x_2))]\lambda y_2.y_2(a)$
$\rightarrow_\beta (\lambda y_2.y_2(a))\lambda x_2.f(x_2)$
$\rightarrow_\beta (\lambda x_2.f(x_2))a$
$\rightarrow_\beta f(a) = \theta(f(a))$.
Wenn wir den Kopf eines flexiblen Terms $u = \lambda\overline{x_k}.F(\overline{u_n})$ durch eine Projektion ersetzen, sind wir durch den Typ von F darauf beschränkt, auf einen Unterterm u_i zu projizieren, der den Typ von u erhält. Insbesondere – da wir F nur durch einen Term vom gleichen Typ ersetzen können, und da Unifikation nur für Terme des gleichen Typs definiert ist – gilt: Ist $\tau(u) = \tau(v) = \alpha_1, ..., \alpha_k \rightarrow \beta$, dann muß $\tau(u_i)$ ein Typ der Form $\gamma_1, ..., \gamma_m \rightarrow \beta$ sein, damit das Ergebnis der Projektion den Typ von u erhält. Also muß der Typ der Matrix (in $t = \lambda\overline{x_k}.e$ ist e die Matrix von t) von u_i gleich dem von u sein, und die Substitution muß für alle Variablen im Lambda-Binder von u_i Argumente zur Verfügung stellen. Wenn also $\theta(F) = \lambda\overline{z_n}.z_i(\overline{r_{m'}})$ für ein $i : 1 \leq i \leq n$ ist, dann muß u_i die Form $u_i = \lambda\overline{y_{m'}}.u'_i$ haben, wobei die Matrix von u'_i den selben Typ hat wie die Matrizen von u und v. In diesem Fall kann der Kopf a von u eine Funktionskonstante, eine freie Variable oder eine gebundene Variable (d.h. eines der x_i) sein, und damit haben wir eine Reduktionsfolge der Form

$$\theta(u) = \theta(\lambda\overline{x_k}.[\lambda\overline{z_n}.z_i(\overline{r_{m'}})\overline{u_n}]) \rightarrow_\beta \theta(\lambda\overline{x_k}.[(\lambda\overline{y_{m'}}.u'_i)\overline{r'_{m'}}])$$

$$\rightarrow^*_\beta \theta(\lambda\overline{x_k}.a'(\overline{t_p})) \xleftrightarrow{*}_\beta \theta(\lambda\overline{x_k}.a(\overline{v_m})),$$

wo $r'_i = r_i[u_1/z_1, ..., u_n/z_n]$ für $1 \leq i \leq m'$, $\lambda\overline{x_k}.a'(\overline{t_p}) = (\lambda\overline{x_k}.[(\lambda\overline{y_{m'}}.u'_i)\overline{r'_{m'}}])\!\downarrow$ und entweder $a' = a$ oder a ist eine freie Variable aus $D(\theta)$.

(E) In den Köpfen beider Terme treten Substitutionen auf.
Dann sei $u = \lambda\overline{x_k}.F(\overline{u_n})$ und $v = \lambda\overline{x_k}.G(\overline{u_m})$ mit $F, G \in D(\theta)$. Hier müssen wir eventuell die beiden Köpfe F und G matchen, aber dafür haben wir viele Möglichkeiten. Um unsere Analyse zu vereinfachen, versuchen wir, diesen Fall (falls möglich) auf den vorangegangenen Fall zu reduzieren. O.B.d.A. konzentrieren wir uns auf die Bindung, die für die Variable F gemacht wurde. Dann gibt es zwei Unterfälle.
(i) θ ersetzt F durch einen Nicht-Projektionsterm, z.B. $\theta(F) = \lambda\overline{z_n}.a(\overline{s_p})$, wo $a \neq G$ keine gebundene Variable ist (und wegen Idempotenz nicht in $D(\theta)$ ist), und dann (eventuell) eine β-Reduktion verursacht, wonach wir das Ergebnis gemäß Fall (D) untersuchen können.
(ii) θ ersetzt F durch einen Projektionsterm (der den oben diskutierten Typerhaltungsbedingungen genügt), z.B. $\theta(F) = \lambda\overline{z_n}.z_j(\overline{t_q})$. Wenn dann nach Reduktion in Normalform das Kopfsymbol entweder eine Funktionskonstante, eine gebundene Variable oder eine Variable $\notin D(\theta)$ ist, können wir das Ergebnis mit Fall (D) untersuchen. Ist der Kopf eine Variable in $D(\theta)$, können wir (rekursiv) Fall (E) auf diese neuen Terme anwenden.

Durch rekursives Anwenden dieser Analyse auf die erzeugten Unterprobleme können wir jede Bindung und jede β-Reduktion der Originalsequenz bestimmen. Dies bildet die Grundlage für die folgende Menge von Transformationsregeln, welche Unifikatoren durch den "inkrementellen" Aufbau von Bindungen findet, wobei sie partielle Bindungen benutzt. In obigem Fall (D) bedeutet dies, daß es nur eine endliche Anzahl von Wahlmöglichkeiten für eine partielle Bindung gibt, weil es nur eine mögliche Imitation und nur eine endliche Anzahl möglicher Projektionen gibt. In Fall (E) ist dies leider falsch. In [29] wird gezeigt, daß zwei flexible Terme nicht notwendig einen endlichen CSU besitzen müssen, und tatsächlich kann es eine unendliche Menge von unabhängigen Unifikatoren geben, welche flexible Terme als Bindungen enthalten, so daß es selbst, wenn wir nur versuchen, das oberste Funktionssymbol der Bindung zu finden, eine unendliche Anzahl von Wahlmöglichkeiten geben kann, da es für jeden Typ immer eine unendliche Menge von Funktionsvariablen gibt. Sogar wenn es nur eine endliche Menge von Funktionskonstanten in der Sprache gibt, ist es nicht möglich, den Nichtdeterminusmus dieses Falles allgemein auf eine endliche Zahl von Wahlmöglichkeiten für partielle Bindungen zu reduzieren, also muß der Suchbaum unendlich verzweigen.

Eine vollständige Unifikations-Prozedur muß zu einem beliebigen System S von Termen aus $\mathcal{L}_{exp}$ und einer normalisierten Substitution $\theta \in U(S)$ stets eine Substitution σ finden, so daß $\sigma \in U(S)$ und $\sigma \leq_\beta \theta[FV(S)]$. Die grundlegende Idee der Transformationsmethode ist, zu gegebenem $\theta \in U(S)$, "Stücke" von θ zu finden, indem wir gelöste Paare $\langle x, t\rangle$ finden, für die $\theta(x) \stackrel{*}{\longleftrightarrow}_\beta \theta(t)$ gilt; in diesem Fall können wir mit einer Argumentation, die der aus Lemma 4.7 ähnelt, schließen, daß $\theta =_\beta [t/x] \circ \theta$, und wenn wir ausreichend viele solcher Paare finden, erreichen wir schließlich $\sigma =_\beta [t_1/x_1] \circ ... \circ [t_n/x_n]$, wobei σ ein Unifikator für S ist, der allgemeiner als (oder äquivalent zu) θ ist. In anderen Worten: wir approximieren θ sukzessive, bis die Substitution genügend aufgebaut ist, so daß sie S unifiziert. Dies tun wir durch "Auflösen" von Variablen (wie im Fall (C)) oder durch Benutzen von Approximationen für individuelle Bindungen, die wir *partielle Bindungen* nennen (Methode von Huet):

Definition 4.8: partielle Bindung, allgemeiner flexibler Term
Eine *partielle Bindung vom Typ* $\alpha_1, ..., \alpha_n \to \beta$ (für einen Grundtyp β) ist ein Term der Form

$$\lambda\overline{y_n}.a\Big(\lambda\overline{z^1_{p_1}}.H_1(\overline{y_n},\overline{z^1_{p_1}}),\ \ldots\ ,\lambda\overline{z^m_{p_m}}.H_m(\overline{y_n},\overline{z^m_{p_m}})\Big)$$

für ein Atom a, wobei
(1) $\tau(y_i) = \alpha_i$ für $1 \leq i \leq n$,
(2) $\tau(a) = \gamma_1, ..., \gamma_m \to \beta$, wobei $\gamma_i = \varphi^i_1, ..., \varphi^i_{p_i} \to \gamma'_i$ für $1 \leq i \leq m$,
(3) $\tau(z^i_j) = \varphi^i_j$ für $1 \leq i \leq m$ und $1 \leq j \leq p_i$,
(4) $\tau(H_i) = \alpha_1, ..., \alpha_n, \varphi^i_1, ..., \varphi^i_{p_i} \to \gamma'_i$ für $1 \leq i \leq m$,
wobei $\gamma'_1, ..., \gamma'_m$ Grundtypen sind. Die unmittelbaren Unterterme der partiellen Bindung (d.h. die Argumente des Atoms a) heißen *allgemeine flexible Terme*.
Man beachte, daß diese partiellen Bindungen durch ihren Typ und ihr Kopfsymbol a eindeutig (bis auf Umbenennung der freien Variablen) festgelegt sind.

Definition 4.9: Imitationsbindung, Projektionsbindung, Variante, passend
Eine partielle Bindung gemäß Definition 4.8 heißt *Imitationsbindung für a*, falls a eine Funktionskonstante oder eine freie Variable ist. Sie heißt eine *i-te Projektionsbindung*, falls a eine gebundene Variable y_i für ein $i : 1 \leq i \leq n$ ist. Eine *Variante* einer partiellen Bindung t ist ein Term $\rho(t)\!\downarrow$, wobei ρ eine Umbenennung der Menge $H_1, ..., H_m$ der freien Variablen in den Köpfen der allgemeinen flexiblen Terme in t ist, die weg von allen Variablen im Kontext, in dem t benutzt wird, ist. Für eine beliebige Variable F heißt eine partielle Bindung t *passend zu F*, falls $\tau(t) = \tau(F)$. Eine Imitationsbindung ist *passend zu* $\lambda\overline{x_k}.F(\overline{u_n})$, falls sie passend zu F ist.

Um die auftretenden Ausdrücke klein zu halten, erweitern wir unsere vektorartige Schreibweise auf partielle Bindungen in der Form

$$\lambda\overline{y_n}.a(\overline{\lambda\overline{z_{p_m}}.H_m(\overline{y_n},\overline{z_{p_m}})}) := \lambda\overline{y_n}.a\Big(\lambda\overline{z^1_{p_1}}.H_1(\overline{y_n},\overline{z^1_{p_1}}),\ \ldots\ ,\lambda\overline{z^m_{p_m}}.H_m(\overline{y_n},\overline{z^m_{p_m}})\Big)$$

Entsprechend unserer oben gegebenen Analyse der Unifikation höherer Ordnung haben wir die folgende Menge von Transformationen:

Definition 4.10: System der Transformationsregeln $\mathcal{HT}$
Sei S ein System von Lambda-Termen (evtl. leer). Dann haben wir die folgenden Transformationen:

– Weglassen unifizierter Paare:

$$\{\ \langle u,u\rangle\ \} \cup S \Longrightarrow\ S \qquad (1)$$

– Termzerlegung:

$$\{\ \langle\lambda\overline{x_k}.a(\overline{u_n}),\lambda\overline{x_k}.a(\overline{v_n})\rangle\ \} \cup S \Longrightarrow \bigcup_{1\leq i\leq n}\Big\{\langle\lambda\overline{x_k}.u_i,\lambda\overline{x_k}.v_i\rangle\Big\} \cup S, \qquad (2)$$

wo a ein beliebiges Atom ist.

– Variablenelimination
Falls $u = \lambda\overline{x_k}.F(\overline{x_k})$ und $v = \lambda\overline{x_k}.v'$ für ein k, eine Variable F und einen Term v' mit $F \notin FV(v)$, dann gilt

$$\{\ \langle u, v\rangle\ \} \cup S \Longrightarrow\ \{\ \langle F, \lambda\overline{x_k}.v'\rangle\ \} \cup \sigma(S)\downarrow, \qquad (3)$$

wobei $\sigma = [\lambda\overline{x_k}.v'/F]$.

Diese drei Transformationen sind analog zum System $\mathcal{ST}$. Um auch Funktionsvariablen verarbeiten zu können, brauchen wir eine weitere Transformation, die in drei Fälle unterteilt ist:

– Imitation

$$\{\ \langle\lambda\overline{x_k}.F(\overline{u_n}), \lambda\overline{x_k}.a(\overline{v_m})\rangle\ \} \cup S \Longrightarrow\ \{\ \langle F, t\rangle, \langle\lambda\overline{x_k}.F(\overline{u_n}), \lambda\overline{x_k}.a(\overline{v_m})\rangle\ \} \cup S, \qquad (4a)$$

wenn a eine Funktionskonstante oder eine freie Variable $\neq F$ ist und t eine Variante einer Imitationsbindung für a ist, die passend zu F ist, z.B. $t = \lambda\overline{y_n}.a(\overline{\lambda\overline{z_{p_m}}.H_m(\overline{y_n}, \overline{z_{p_m}})})$.

– Projektion

$$\{\ \langle\lambda\overline{x_k}.F(\overline{u_n}), \lambda\overline{x_k}.a(\overline{v_m})\rangle\ \} \cup S \Longrightarrow\ \{\ \langle F, t\rangle, \langle\lambda\overline{x_k}.F(\overline{u_n}), \lambda\overline{x_k}.a(\overline{v_m})\rangle\ \} \cup S, \qquad (4b)$$

wenn a ein beliebiges Atom (eventuell gebunden) ist und t eine Variante einer i-ten Projektionsbindung (für ein $i : 1 \leq i \leq n$) ist, die passend zum Term $\lambda\overline{x_k}.F(\overline{u_n})$ ist, d.h. $t = \lambda\overline{y_n}.y_i(\overline{\lambda\overline{z_{p_q}}.H_q(\overline{y_n}, \overline{z_{p_q}})})$, so daß, falls $Head(u_i)$ eine Funktionskonstante ist, gilt: $Head(u_i) = a$.

– partielle Bindung

$$\{\ \langle\lambda\overline{x_k}.F(\overline{u_n}), \lambda\overline{x_k}.G(\overline{v_m})\rangle\ \} \cup S \Longrightarrow\ \{\ \langle F, t\rangle, \langle\lambda\overline{x_k}.F(\overline{u_n}), \lambda\overline{x_k}.G(\overline{v_m})\rangle\ \} \cup S, \qquad (4c)$$

wenn $t = \lambda\overline{y_n}.a(\overline{\lambda\overline{z_{p_m}}.H_m(\overline{y_n}, \overline{z_{p_m}})})$ eine Variante einer beliebigen partiellen Bindung ist, die passend zum Term $\lambda\overline{x_k}.F(\overline{u_n})$ ist, so daß $a \neq F$ und $a \neq G$.

Als Teil der Transformationen (4a)-(4c) wenden wir unmittelbar Transformation (3) auf das neue Paar $\langle F, t\rangle$ an, was sich als Anwendung der Substitution $[t/F]$ auf den Rest des Systems auswirkt. Wie beim System $\mathcal{ST}$ sind auch hier die Vereinigungen wieder Vereinigungen von *Multimengen*, nicht von Mengen!

Im folgenden sagen wir $\theta \in Unify(S)$ genau dann, wenn es eine Folge von Transformationen $S \Longrightarrow^* S_n$ gibt, so daß S_n in gelöster Form ist und $\theta = \sigma_{S_n}|_{FV(S)}$.

Beispiel 4.11
Die folgende Transformationsfolge führt zu einem System in gelöster Form. (Um die Wirkung des β-Reduktionsschrittes zu zeigen, der auf die Substitution in Regel (3) folgt, benutzen wir die Schreibweise $\frac{\theta(e)}{\theta(e)\downarrow}$. Im "Zähler" steht also das Ergebnis nach der Substitution und im

"Nenner" das Endergebnis nach β-Reduktion.)
$\langle F(f(a)), f(F(a))\rangle$
$\Longrightarrow_{4a} \langle F, \lambda x.f(Y(x))\rangle, \langle \frac{\lambda x.f(Y(x)))f(a)}{f(Y(f(a)))}, f(\frac{\lambda x.f(Y(x)))a}{f(Y(a))})\rangle$
$\Longrightarrow_{2} \langle F, \lambda x.f(Y(x))\rangle, \langle Y(f(a)), f(Y(a))\rangle$
$\Longrightarrow_{4b} \langle F, \lambda x.f(\frac{(\lambda x.x)x}{x})\rangle, \langle Y, \lambda x.x\rangle, \langle \frac{(\lambda x.x)f(a)}{f(a)}, f(\frac{(\lambda x.x)a}{a})\rangle$
$\Longrightarrow_{1} \langle F, \lambda x.f(x)\rangle, \langle Y, \lambda x.x\rangle$
Also ist $[\lambda x.f(x)/F] \in Unify(F(f(a)), f(F(a)))$.

4.2 Korrektheit der Transformationen

Die folgenden Lemmata erlauben uns, die Korrektheit dieses Systems von Transformationen zu beweisen.

Lemma 4.12
Falls $S \Longrightarrow_1 S'$ oder $S \Longrightarrow_3 S'$, dann gilt $U(S) = U(S')$.

Beweis: Wie im Fall erster Ordnung liegt die einzige Schwierigkeit in Transformation (3). Wir müssen zeigen, daß $U(\{\langle x, v\rangle\} \cup S) = U(\{\langle x, v\rangle\} \cup \sigma(S)\downarrow)$, wobei $\sigma = [v/x]$ und $x \notin FV(v)$. Für jede Substitution θ gilt: aus $\theta(x) \overset{*}{\longleftrightarrow}_\beta \theta(v)$ folgt $\theta =_\beta \sigma \circ \theta$, da sich $\sigma \circ \theta$ und θ nur bei x unterscheiden und $\theta(x) \overset{*}{\longleftrightarrow}_\beta \theta(v) = \theta(\sigma(x)) = \sigma \circ \theta(x)$. Mit Lemma 2.21 kann man dann zeigen: $\theta \in U(S) \iff \sigma \circ \theta \in U(S)$. Da für jeden Term u: $\sigma \circ \theta(u) = \theta(\sigma(u)) \rightarrow_\beta^* \theta(\sigma(u)\downarrow)$ gelten muß, kann man außerdem einfach zeigen, daß $\sigma \circ \theta \in U(S) \iff \theta \in U(\sigma(S)\downarrow)$. Also:
$\theta \in U(\{\langle x, v\rangle\} \cup S)$
$\iff \theta(x) \overset{*}{\longleftrightarrow}_\beta \theta(v)$ und $\theta \in U(S)$
$\iff \theta(x) \overset{*}{\longleftrightarrow}_\beta \theta(v)$ und $\sigma \circ \theta \in U(S)$
$\iff \theta(x) \overset{*}{\longleftrightarrow}_\beta \theta(v)$ und $\theta \in U(\sigma(S)\downarrow)$
$\iff \theta \in U(\{\langle x, v\rangle\} \cup \sigma(S)\downarrow)$. $\boxed{\text{Q.E.D.}}$

Dieses Lemma zeigt, daß die Transformationen (1) und (3) die invarianten Eigenschaften eines Problems genauso festlassen wie im Fall erster Ordnung.

Lemma 4.13
Es sei $S \Longrightarrow_2 S'$, und es sei $\langle \lambda\overline{x_k}.a(\overline{u_n}), \lambda\overline{x_k}.a(\overline{v_n})\rangle$ das Paar aus S, das transformiert wird. Für jede Substitution θ gelten:
(i) Wenn a eine Konstante, eine gebundene Variable oder eine *nicht* in $D(\theta)$ enthaltene freie Variable ist, dann gilt $\theta \in U(S) \iff \theta \in U(S')$.
(ii) Wenn $a \in D(\theta)$, dann gilt $\theta \in U(S') \Longrightarrow \theta \in U(S)$.

Beweis: Falls $\theta(\lambda\overline{x_k}.u_i) \overset{*}{\longleftrightarrow}_\beta \theta(\lambda\overline{x_k}.v_i)$ für $1 \leq i \leq n$, dann muß gelten:

$$\theta(\lambda\overline{x_k}.a(\overline{u_n})) = \lambda\overline{x_k}.\theta(a)(\theta(u_1), ..., \theta(u_n)) \overset{*}{\longleftrightarrow}_\beta \lambda\overline{x_k}.\theta(a)(\theta(v_1), ..., \theta(v_n)) = \theta(\lambda\overline{x_k}.a(\overline{v_n})),$$

und damit gilt für jedes Atom a: $\theta \in U(S') \Longrightarrow \theta \in U(S)$. Wenn a eine Funktionskonstante, eine gebundene Variable oder eine nicht in $D(\theta)$ auftretende Variable ist, dann gilt $\theta(a) = a$, und damit auch $\theta \in U(S) \Longrightarrow \theta \in U(S')$. Q.E.D.

Lemma 4.14
Falls $S \Longrightarrow_2 S'$ oder $S \Longrightarrow_4 S'$, dann gilt $U(S) \supseteq U(S')$.

BEWEIS: Für Regel (2) folgt dies aus dem vorangegangenen Lemma. Transformation (4) besteht aus zwei Schritten: zunächst wird dem System S das neue Paar $\langle F, t\rangle$ hinzugefügt, dann wird Transformation (3) auf dieses neue Paar angewandt. Wegen $S \subseteq \{\langle F, t\rangle\} \cup S$ gilt $U(S) \supseteq U(\{\langle F, t\rangle\} \cup S)$ (je mehr Paare unifiziert werden müssen, desto weniger Unifikatoren gibt es). Daß die darauf folgende Anwendung von Regel (3) auf das neue Paar korrekt ist, wurde bereits in Lemma 4.12 gezeigt. Q.E.D.

Da wir in Transformation (4) eine spezielle Approximation einer Lösung wählen, ist die Inklusion $U(S) \subseteq U(S')$ im allgemeinen echt. In Transformation (2) kann die Termzerlegung flexibler Paare Unifikatoren eliminieren: z.B. hat $\langle F(a, b), F(c, d)\rangle$ unendlich viele Unifikatoren, aber das System $\langle a, b\rangle$, $\langle c, d\rangle$ hat keine. Diese Ergebnisse zeigen uns, daß im Fall höherer Ordnung die Lösungsmenge nur unter den Transformationen (1) und (3) invariant ist; unter Transformation (2) gilt diese Invarianz nur für starre Terme (d.h. Terme, deren Kopf eine Funktionskonstante oder eine gebundene Variable ist).

Zusammenfassend ergibt sich aus diesen Lemmata

Satz 4.15: Korrektheit
Es sei $S \Longrightarrow^* S'$ mit S' in gelöster Form. Dann gilt $\sigma_{S'}|_{FV(S)} \in U(S)$.

BEWEIS: Durch Induktion über die Länge der Transformationsfolge und mit Hilfe der gerade bewiesenen Lemmata können wir zeigen, daß $\sigma_{S'} \in U(S)$. Die Einschränkung auf $FV(S)$ ändert nicht den Effekt der Substitution auf Termen in S, so daß $\sigma_{S'}|_{FV(S)} \in U(S)$. Q.E.D.

4.3 Vollständigkeit der Transformationen

Die Vollständigkeit des Transformationssystems $\mathcal{HT}$ zeigen wir ähnlich wie bei $\mathcal{ST}$ im Fall erster Ordnung, mit dem Unterschied, daß nun die Transformationsrelation im allgemeinen nicht terminierend ist. Daher werden wir nur die *nichtdeterministische Vollständigkeit* des Systems zeigen, d.h. wir zeigen, daß für jedes System S und jeden Unifikator $\theta \in U(S)$ *eine* Transformationsfolge existiert, die einen Unifikator σ findet, so daß $\sigma \leq_\beta \theta[FV(S)]$. (Im Fall erster Ordnung hatten wir das "schönere" Ergebnis, daß *jede* Folge von Transformationen terminieren muß und dann einen Unifikator liefert.)

Zunächst zeigen wir, wie partielle Bindungen als Approximationen für Bindungen in Substitutionen betrachtet werden können.

Lemma 4.16
Sei s ein beliebiger Term der Form $s = \lambda\overline{x_n}.a(\overline{s_m})$. Dann gibt es eine Variante einer partiellen Bindung t und eine Substitution η, so daß $\eta(t) \to_\beta^* s$.

Beweis: Falls $m = 0$, also $s = \lambda\overline{x_n}.a$, sei $t = s$ und $\eta = Id$ (dann gilt $\eta(t) = \eta(s) = s \to_\beta^* s$). Anderenfalls ($m > 0$) sei $t = \lambda\overline{x_n}.a(\overline{\lambda\overline{z_{p_m}}.H_m(\overline{x_n}, \overline{z_{p_m}})})$ und $\eta = [\lambda\overline{x_n}.s_1/H_1, ..., \lambda\overline{x_n}.s_m/H_m]$. Dann muß wegen des Typs vom Kopf a der i-te Unterterm s_i die Form $\lambda z_{p_i}.s_i'$ haben, so daß

$$\eta(\lambda\overline{z_{p_i}}.H_i(\overline{x_n}, \overline{z_{p_i}})) \to_\beta s_i,$$

für alle $i : 1 \leq i \leq m$. Also $\eta(t) \to_\beta^* s$. $\boxed{\text{Q.E.D.}}$

Lemma 4.17
Sei $\theta = [s/F] \cup \theta'$. Dann gibt es eine Variante einer partiellen Bindung t, die passend zu F ist, sowie eine Substitution η, so daß

$$\theta = [s/F] \cup \eta \cup \theta'[D(\theta)] =_\beta [t/F] \circ \eta \cup \theta'[D(\theta)].$$

Außerdem: Falls $D(\theta) \cap I(\theta) = \emptyset$, dann ist $\theta'' := [s/F] \cup \eta \cup \theta'$ ein Unifikator für das Paar $\langle F, t\rangle$ mit $D(\theta'') \cap I(\theta'') = \emptyset$.

Im folgenden definieren wir ein System von Transformationen auf Paaren (θ, S), welches zeigt, wie der Aufbau einer Substitution θ eine geeignete Folge von Transformationen festlegt.

Definition 4.18: Transformationssystem $\mathcal{CT}$
Sei θ eine normalisierte Substitution und S ein beliebiges System. Die ersten drei Transformationen entsprechen denen aus $\mathcal{HT}$:

$$\theta, S \Longrightarrow_i \theta, S'$$

für $i = 1, 2, 3$, falls $S \Longrightarrow_i S'$ in $\mathcal{HT}$, mit der Einschränkung, daß Regel (2) nur auf ein Paar $\langle u, v\rangle$ angewandt werden darf, bei dem das vorderste Funktionssymbol in u und v *keine* freie Variable in $D(\theta)$ ist. Außerdem haben wir die Regel

$$[s/F] \cup \theta, \{\langle\lambda\overline{x_k}.F(\overline{u_n}), \lambda\overline{x_k}.v\rangle\} \cup S \Longrightarrow_4 [s/F] \cup \eta \cup \theta, \{\langle F, t\rangle, \langle\lambda\overline{x_k}.F(\overline{u_n}), \lambda\overline{x_k}.v\rangle\} \cup S,$$

wo F auf der linken Seite noch nicht gelöst ist, s ein Term der Form $\lambda\overline{y_n}.a(\overline{s_m})$ ist,

$$t = \lambda\overline{y_n}.a(\overline{\lambda\overline{z_{p_m}}.H_m(\overline{y_n}, \overline{z_{p_m}})})$$

eine zu F passende partielle Bindung mit dem (bis auf α-Konversion) gleichen Kopf wie s ist, und

$$\eta = [\lambda\overline{y_n}.s_1/H_1, ..., \lambda\overline{y_n}.s_m/H_m].$$

(Falls $m = 0$, wird η weggelassen.) Wie in $\mathcal{HT}$ wird im Anschluß an Regel (4) unmittelbar Regel (3) auf das neue Paar $\langle F, t\rangle$ angewandt.

Beispiel 4.19
Wir ergänzen das Paar in Beispiel 4.11 um die Substitution $\theta = [\lambda x.f(x)/F]$. Dann ergibt sich diese $\mathcal{CT}$-Transformationsfolge:
$[\lambda x.f(x)/F], \{\langle F(f(a)), f(F(a))\rangle\}$
$\Longrightarrow_4 [\lambda x.f(x)/F, \lambda x.x/Y], \{\langle F, \lambda x.f(Y(x))\rangle, \langle \frac{\lambda x.f(Y(x)))f(a)}{f(Y(f(a)))}, f(\frac{\lambda x.f(Y(x)))a}{f(Y(a))})\rangle\}$
$\Longrightarrow_2 [\lambda x.f(x)/F, \lambda x.x/Y], \{\langle F, \lambda x.f(Y(x))\rangle, \langle Y(f(a)), f(Y(a))\rangle\}$
$\Longrightarrow_4 [\lambda x.f(x)/F, \lambda x.x/Y], \{\langle F, \lambda x.f(\frac{(\lambda x.x)x}{x})\rangle, \langle Y, \lambda x.x\rangle, \langle \frac{(\lambda x.x)f(a)}{f(a)}, f(\frac{(\lambda x.x)a}{a})\rangle\}$
$\Longrightarrow_1 [\lambda x.f(x)/F, \lambda x.x/Y], \{\langle F, \lambda x.f(x)\rangle, \langle Y, \lambda x.x\rangle\}$

Lemma 4.20
Es sei S ein System in ungelöster Form, $\theta \in U(S)$ und W eine Menge von Variablen. Dann gibt es eine $\mathcal{CT}$-Transformation $\theta, S \Longrightarrow \theta', S'$, so daß
(i) $\theta = \theta'[W]$;
(ii) Falls $D(\theta) \cap I(\theta) = \emptyset$, dann gilt $\theta' \in U(S')$ und $D(\theta') \cap I(\theta') = \emptyset$;
(iii) $S \Longrightarrow_{\mathcal{HT}} S'$.

Korollar 4.21
Falls $\theta \in U(S)$ und keine der Regeln aus $\mathcal{HT}$ auf θ, S angewandt werden kann, ist S in gelöster Form.

Satz 4.22: Vollständigkeit von $\mathcal{HT}$
Es sei S ein System und $\theta \in U(S)$. Dann gibt es eine Transformationsfolge

$$S = S_0 \Longrightarrow S_1 \Longrightarrow S_2 \Longrightarrow \ldots \Longrightarrow S_n,$$

wo S_n in gelöster Form ist und $\sigma_{S_n} \leq_\beta \theta[FV(S)]$.

Man beachte, daß diese Aussage sich von der Vollständigkeitsaussage für das System $\mathcal{ST}$ dadurch unterscheidet, daß wir hier nur die Existenz *einer* terminierenden Transformationsfolge erhalten, wogegen im Fall erster Ordnung *alle* Transformationsfolgen terminierten.

Die Aussagen über Korrektheit und Vollständigkeit fassen wir im folgenden, letzten Satz zusammen:

Satz 4.23
Für ein beliebiges System S ist die Menge

$$\{\sigma_{S'}|_{FV(S)} : S \Longrightarrow^*_{\mathcal{HT}} S' \text{ und } S' \text{ in gelöster Form}\}$$

ein $CSU(S)$.

5. Zusammenfassung

Ausgehend von Unifikationsproblemen erster Ordnung haben wir ein System von Transformationen entwickelt, unter denen die Lösungen des Problems invariant bleiben: wir haben Korrektheit und Vollständigkeit nachgewiesen und außerdem bemerkt, daß das Verfahren terminierend ist. Diese Transformationen waren im wesentlichen Termzerlegung und Variablenelimination. Anschließend haben wir unsere Transformationen verallgemeinert und so das Verfahren auf Lambda-Terme höherer Ordnung ausgedehnt: hierzu mußten wir unser Transformationssystem um Regeln zur Imitation und Projektion ergänzen, was die Komplexität stark vergrößerte: während jede der Transformationen im Fall erster Ordnung die Terme "verkleinerte" (in einem von uns definierten Sinn) und somit schließlich keine Transformation mehr anwendbar ist, gilt dies im Fall höherer Ordnung nicht mehr: Dadurch können Suchbäume ins Unendliche wachsen, und das Verfahren ist nicht mehr terminierend. Somit erhielten wir bei der Vollständigkeit die Einschränkung, daß das Verfahren nur nichtdeterministisch vollständig ist, d.h. es muß nicht jede Transformationsfolge terminieren, aber es gibt in jedem Fall eine, die terminiert.
Außerdem haben wir beim Übergang zum Fall höherer Ordnung die Einfachheit der Existenz eines allgemeinsten Unifikators (mgu) verloren; hier gibt es im allgemeinen nur noch eine vollständige Menge von Unifikatoren, die zu jedem Unifikator einen allgemeineren Unifikator enthält.

Literatur

[Ω] Wayne Snyder, Jean Gallier, "Higher-Order Unification Revisited: Complete Sets of Transformations" in: Journal of Symbolic Computation (1989) **8**, 101-140)
[25] Herbrand, "Sur la Théorie de la Démonstration" in: Logical Writings, W. Goldfarb ed., Cambridge, 1971
[33] Martelli, Montanari, "An Efficient Unification Algorithm", ACM Transactions on Programming Languages and Systems 4:2 (1982) 258-282
[29] Huet, G., *Résolution d'Equations dans Langages d'Ordre 1,2,...,ω*, Thèse d'Etat, Université de Paris VII (1976)

Index