Erklärung

Hiermit erkläre ich, daß ich diese schriftliche Arbeit selbstständig verfaßt und keine weiteren als die angegebenen Hilfsmittel verwendet habe.

Dresden, den 21. Juni 1995

Sebastian Maneth

Kalküle und Effizienzmaße für E–Unifikation mit kanonischen Termersetzungssystemen

Schriftliche Arbeit
von
Sebastian Maneth

Institut für Softwaretechnik I
Technische Universität Dresden

Verantwortlicher Hochschullehrer
und Betreuer: Prof. Dr. Heiko Vogler

Inhaltsverzeichnis

Kapitel 1

Einleitung

Programmiersprachen (kurz: PSn) lassen sich grob in imperative und deklarative Programmiersprachen unterteilen. In imperativen PSn spannt der Deklarationsteil eines Programms einen Zustandsraum auf, der durch die Anweisungen dieses Programms transformiert wird. In deklarativen PSn wird von einem Zustandsraum abstrahiert, indem das Programm aus mathematischen Objekten, wie etwa Funktionen oder Relationen, besteht, die dann auf bestimmte Weise ausgewertet werden.

Funktionale und logische Programmierung sind die wichtigsten deklarativen Programmierparadigmen. Prolog [Kow74, CM81] in der logischen Programmierung und ML [GMW79] oder Miranda [Tur85] (um nur einige wenige zu nennen) in der funktionalen Programmierung sind bewährte Programmiersprachen, welche in den verschiedensten Bereichen Anwendung gefunden haben. In den letzten zehn Jahren ist das Interesse an der Vereinigung dieser beiden Sprachkonzepte gestiegen (siehe [DL86] für eine Zusammenfassung). Es gibt mittlerweile eine große Anzahl von funktional–logischen PSn, die zum Teil effizient genug arbeiten, um für praktische Anwendungen eingesetzt werden zu können; z.B. SLOG [Fri85], EQULOG [GM86], FUNLOG [SY86], BABEL [RAMN88, RAMN92], K-LEAF [GLMP91], CAMEL [Mü91], JUMP [CL91, Loc92] etc.; ein guter Überblick ist in [Han94] zu finden. Für die Integration von funktionalen und logischen PSen gibt es grob gesagt zwei verschiedene Ansätze:

1. Man geht von einer funktionalen PS aus und führt logische Variablen ein, indem man pattern matching durch Unifikation ersetzt [Red85].

2. Man geht von einer logischen PS aus und erweitert diese um die Möglichkeit Funktionen zu definieren und auszuwerten (Überblick in [Han94]).

Die operationellen Semantiken der jeweils entstehenden *funktional–logischen* PSn sind sehr ähnlich zueinander. Betrachten wir zur Veranschaulichung der operationellen Semantik das folgende Programm einer fiktiven funktional–logischen PS.

$$\begin{array}{rcll} sublist(x,y) & := & (y = append(z_1, append(x, z_2))) & (1) \\ append(NIL, x) & := & x & (2) \\ append(CONS(x,y), z) & := & CONS(x, append(y,z)) & (3) \end{array}$$

Dabei definiert die erste Zeile das Prädikat *sublist* und die zweite und dritte Zeile die Funktion *append*. Für zwei Listen l_1 und l_2 gilt dann $sublist(l_1, l_2)$, wenn l_1 eine Teilliste von l_2 ist. Die Funktion *append* ist die übliche Listenkonkatenation. Offensichtlich treten auf der rechten Seite des *sublist* Prädikats die freien Variablen z_1 und z_2 auf.

Neben diesem Programm betrachten wir die Anfrage (engl. goal)

$$sublist(CONS(A, NIL), CONS(A, CONS(B, z_3)))$$

mit der freien Variable z_3 und den Konstanten A und B.

Nun legt die operationelle Semantik fest, daß die vorgelegte Anfrage dann wahr ist, wenn es Belegungen l_1 und l_2 für z_1 bzw. z_2 gibt, so daß die beiden Terme

$$\widehat{t_1} = CONS(A, CONS(B, z_3))$$

und

$$\widehat{t_2} = append(l_1, append(CONS(A, NIL), l_2))$$

E–gleich sind, wobei E die Gleichungsmenge ist, welche aus den beiden Definitionsgleichungen für *append* besteht. Die *E–Gleichheit* ist die kleinste Kongruenzrelation welche E enthält.

Als geeignete Belegungen können wir beispielsweise

$$\begin{array}{rcl} l_1 & = & NIL \\ l_2 & = & CONS(B, z_3) \end{array}$$

wählen.

Allgemein sagt man, daß zwei Terme t_1 und t_2 *E–unifizierbar* [GS87] sind, wenn es eine Belegung φ der Variablen von t_1 und t_2 gibt, so daß $\varphi(t_1)$ und $\varphi(t_2)$ E–gleich sind; φ heißt auch E–Unifikator von t_1 und t_2. Jede Implementierung einer funktional–logischen Programmiersprache muß also in der Lage sein, für eine vorgegebene Menge E von Gleichungen und zwei Terme t_1 und t_2 die Frage zu beantworten, ob t_1 und t_2 E–unifizierbar sind oder nicht und ggf. einen E–Unifikator von t_1 und t_2 zu berechnen.

Im Allgemeinen ist das E–Unifikationsproblem für beliebige Mengen von Gleichungen E *nicht* entscheidbar. Sei nämlich beispielsweise E als die Menge der Peano Axiome gewählt, so ist das E–Unifikationsproblem genau Hilbert's zehntes Problem, welches unentscheidbar ist [Mat70]. Sehr oft treten in den Untersuchungen algebraische Gesetze, wie Assoziativität, Kommutativität oder Idempotenz als Menge E der Gleichungen auf. In diesem Zusammenhang sind folgende Erkenntnisse bezüglich der Entscheidbarkeit der resultierenden E–Unifikationsprobleme gemacht worden:

- In [Mak77] wurde gezeigt, daß Unifikation unter Assoziativität entscheidbar ist. Der erste minimale Unifikationsalgorithmus wurde in [Plo72] präsentiert. Die Korrektheit, Vollständigkeit und Minimalität dieses Algorithmus wurden erstmalig in [Sie87] bewiesen. Unifikation unter Assoziativität ist die Unifikation in Halbgruppen; sie wird besonders bei Wortproblemen benötigt, also bei der Frage, ob bestimmte Zeichenketten durch Variablenersetzung gleich gemacht werden können.

- In [Sti75] wurde der erste Algorithmus für die Unifikation unter Assoziativität (A) und Kommutativität (C) einer Funktion veröffentlicht. In [Sti81] werden Korrektheit, Vollständigkeit und Minimalität dieses Algorithmus für das AC–Unifikationsproblem mit Konstanten bewiesen. Erst in [Fag84] konnte gezeigt werden, daß dieser Algorithmus auch im Falle der generellen AC–Unifikation terminiert.

- Erweiterung des Algorithmus für AC–Unifikation zur AC1–Unifikation [LSSU79], also Assoziativität, Kommutativität und Idempotenz (1). Eine gute Zusammenfassung ist in [Thi92] zu finden.

Auch wir werden uns in dieser Arbeit auf ganz bestimmte Klassen von Gleichungen beschränken, für die wir E–Unifikation betrachten wollen. Dies wird die Klasse der Gleichungen sein, die durch *kanonische Termersetzungssysteme* [DJ90, Der91] induziert werden. Allerdings ist das E–Unifikationsproblem auch hier im allgemeinen nicht entscheidbar.

Diese Beschränkung ist dadurch gerechtfertigt, daß die operationelle Semantik vieler funktionaler Programmiersprachen Termersetzung [Man95, Jon87] ist, und die Graphreduktionsmaschine (G–machine [Aug84, Joh84]) implementiert nichts anderes als ein besondere Methode der Termersetzung.

Ein Termersetzungssystem (kurz: TES) $\mathcal{R}$ besteht aus einem Rangalphabet Ω und einer Menge R von Regeln der Form $l \to r$, wobei l und r Terme über Ω und einer Menge von Variablen $\mathcal{V}$ ist. Außerdem dürfen keine Variablen in der rechten Seite einer Regel, i.e. dem Term r, auftreten, die nicht auch auf der linke Seite auftreten. In unserem Beispiel der *append*–Funktion müssen wir lediglich die Gleichheitszeichen durch Rechtspfeile ersetzen, um eine korrekte Menge R von Regeln zu erhalten. Es gilt dann $\{append^{(2)}, CONS^{(2)}, NIL^{(0)}\} \subseteq \Omega$ und R ist die folgende Menge von Regeln:

$$\begin{aligned} append(NIL, x) &\to x && (4) \\ append(CONS(x,y), z) &\to CONS(x, append(y,z)) && (5) \end{aligned}$$

Soll eine Termersetzungsregel auf einen Term t angewendet werden, so wird ein beliebiger Teilterm von t, der Instanz einer linke Regelseite von R ist, durch die entsprechende Instanz der rechten Seite dieser Regel ersetzt.

Essentiell bei der Semantikgebung für funktionale Programmiersprachen durch Termersetzung ist das Konzept der *Normalform*. Eine Normalform ist ein Zwischenergebnis, das nicht weiter ersetzt werden kann durch Anwendung einer Termersetzungsregel des entsprechenden Termersetzungssystems.

Um zu der Normalform eines Termes t zu kommen, werden so lange Regeln aus R angewendet, bis dies nicht mehr möglich ist. Damit aber jeder Term t eine *eindeutige* Normalform besitzt, müssen wir die zwei folgenden Einschränkungen an das TES $\mathcal{R}$ stellen:

1. Es gibt keine unendlichen Ersetzungssequenzen, i.e. das TES terminiert für jeden Term t.

2. Das TES ist *konfluent*, i.e. wenn zwei Terme t' und t'' aus t durch Ersetzung entstehen, dann gibt es einen Term $\hat{t}$, der sowohl aus t', als auch aus t'' durch Ersetzung ensteht.

TESe für die diese beiden Bedingungen erfüllt sind, heißen *kanonisch*.

Ein TES $\mathcal{R} = (\Omega, R)$ induziert die Gleichungsmenge $E_{\mathcal{R}} = \{l = r \mid (l \to r) \in R\}$.

In dieser Arbeit betrachten wir vier aufeinander aufbauende Methoden zum Finden von $E_{\mathcal{R}}$–Unifikatoren, wobei $\mathcal{R}$ ein TES aus der in Klammern angegebenen Klassen von TESen ist:

- Reduktion [HO80] (kanonische TESe).
- Narrowing [Sla74, Lan75] (kanonische TESe).
- Leftmost outermost Narrowing [Ech88] (ctn–trs)[1].
- Unification driven leftmost outermost Narrowing [FV94] (ctn–trs).

Alle diese Methoden werden wir als Instanzen eines abstrakten mathematischen Konzepts, des *Kalküls*, darstellen (vgl. auch [FV92]). Genau diesem Konzept wollen wir uns besonders in dieser Arbeit widmen. In einem Kalkül werden, wie der Name schon sagt, Berechnungen durchgeführt. Es gibt dazu eine Menge D von Zwischenergebnissen und eine Relation $\vdash$ über dieser Menge. Stehen zwei Zwischenergebnisse t und t' in der Relation $\vdash$, so sagt man, daß t' aus t berechnet werden kann. Darüberhinaus hat man noch die Möglichkeit, bestimmte Elemente als *Endergebnisse* auszuzeichnen, indem eine Menge $D_{\text{success}} \subseteq D$ definiert wird. Ein Kalkül ist also ein Tupel $(D, D_{\text{success}}, \vdash)$, wobei $\vdash \subseteq D \times D$ gilt.

Alle Berechnungen, die von einem Term t (im folgenden auch *Kalkulandum* genannt) ausgehen, können in einem sogenannten Berechnungsbaum dargestellt werden. Nehmen wir beispielsweise Terme über dem Rangalphabet $\{append^{(2)}, CONS^{(2)}, NIL^{(0)}, A^{(0)}, B^{(0)}\}$ als Menge der Zwischenergebnisse, $D_{\text{success}} = \emptyset$ und $\vdash$ die Reduktionsrelation des TESs mit den Regeln (4) und (5), dann ist der Baum in Abbildung 1.1 ein Berechnungsbaum des Kalkulandums $append(NIL, append(CONS(B, NIL)))$.

Es ist wünschenswert, die Möglichkeit zu haben, eindeutig einen Berechnungsbaum durch einen Kalkül darzustellen. Der oben erwähnte Kalkül reicht dazu nicht aus, denn alle Knoten auf einem Niveau des Baums stehen in der Relation $\vdash$ zum Vorgängerknoten; d.h. alle

[1] ctn–trs steht für canonical totally defined not strictly sub–unifiable term rewriting system

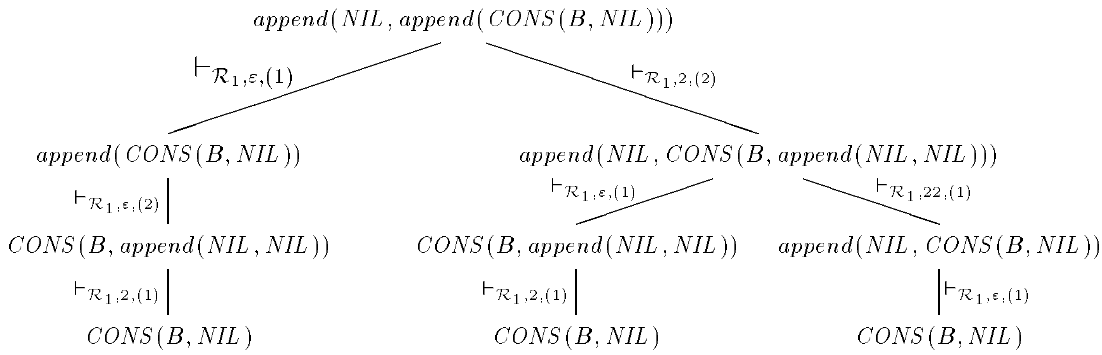

Abbildung 1.1: Berechnungsbaum von $\mathcal{R}$ und $append(NIL, append(CONS(B, NIL)))$

Permutation der Knoten innerhalb eines Niveaus sind möglich. Um eindeutig einen Baum zu bestimmen, erweitern wir den Kalkül zum *indizierten Kalkül*. Der indizierte Kalkül besitzt außer der Menge der Zwischenergebnisse D und der Menge der Endzustände D_{success} noch eine *Indexmenge* J und eine totale Ordnung $<$ über der Indexmenge. Die Relation $\vdash$ des Kalküls wird dadurch in Relationen $\vdash_j$, mit $j \in J$ partitioniert, so daß für ein Zwischenergebnis d nur höchstens ein d' in der Relation $\vdash_j$ zu d steht, für $j \in J$.

Zuletzt definieren wir den *deterministischen Kalkül* für einen indizierten Kalkül. Er beschreibt einen depth–first left–to–right tree traversal durch den Berechnungsbaum der durch den indizierten Kalkül und ein Kalkulandum beschrieben wird. Durch Veränderung der Indexmenge J und der totalen Ordnung $<$ im indizierten Kalkül können alle möglichen depth–first Baumdurchläufe durch den deterministischen Kalkül realisiert werden.

Für jeden Kalkül haben wir die Menge D_{success} der Endergebnisse definiert. Im Sinne der E–Unifikation werden Endergebnisse üblicherweise E–Unifikatoren enthalten oder Substitutionen, die leicht zu E–Unifikatoren gemacht werden können. Wie kann man nun entscheiden, welcher Kalkül effizienter E–Unifikatoren findet als ein anderer? Diese Frage ist bedeutsam für die Effizienz der Implementierung einer funktional–logischen PS, denn wir hatten ja das Auffinden von E–Unifikatoren als eine von jeder Implementation zu lösende Aufgabe herausgearbeitet. Eine generelle Antwort auf diese Frage ist nicht möglich. In dieser Arbeit soll diese Problematik formalisiert und für eine kleine Menge von Kalkülen zur E–Unifikation gelöst werden. Dazu definieren wir eine Reihe von *Effizienzmaßen*, die sich auf Kalküle anwenden lassen. In der folgenden Auflistung ist jeweils in Klammern die Klasse der Kalküle angegeben, für die das Maß definiert ist.

- *Single success Effizienz* (Kalküle)
  Mißt die Anzahl der Berechnungsschritte des zu dem indizierten Kalkül gehörenden deterministischen Kalküls, bis zum Finden des *ersten* Endergebnisses.

- *Multi success Effizienz* (Kalküle)

Mißt die Anzahl der Berechnungsschritte des zu dem indizierten Kalkül gehörenden deterministischen Kalküls, bis zum Finden der ersten k untereinander verschiedenen Endergebnisse. Die Gleichartigkeit der Zwischenergebnisse wird durch eine Äquivalenzrelation $\sim$ ausgedrückt.

- *Success density Effizienz* (Kalküle)
  Mißt die Anzahl der voneinander verschiedenen Endergebnisse auf einem bestimmten Niveau n des Berechnungsbaums.

- *Infinite Path Effizienz* (indizierte Kalküle)
  Mißt die Anzahl der unendlichen Äste im Berechnungsbaum des indizierten Kalküls und eines Kalkulandums d. Die Infinite Path Effizienz ist höher, je weniger unendliche Äste vorhanden sind.

Wir werden einige dieser Effizienzmaße auf bestimmte Narrowing Kalküle anwenden. Es soll beispielsweise Narrowing mit leftmost outermost Narrowing verglichen werden. Intuitiv stellt sich die Frage, warum eine solche Betrachtung sinnvoll ist, denn es scheint im ersten Moment trivialerweise wahr zu sein, daß leftmost outermost Narrowing effizienter ist als Narrowing, denn beim leftmost outermost Narrowing wird der Narrowingbaum verkleinert, indem nur noch ganz bestimmte Narrowing Vorkommen beim Anwenden der Narrowing Relation betrachtet werden. Kann es aber nicht auch ein Termersetzungssystem $\mathcal{R}$ und ein Kalkulandum d geben, so daß Narrowing effizienter ist als lo–Narrowing? Als Effizienz wollen wir jetzt die single succes Effizienz wählen. Betrachten wir dazu den Berechnungsbaum in Abbildung 1.2.

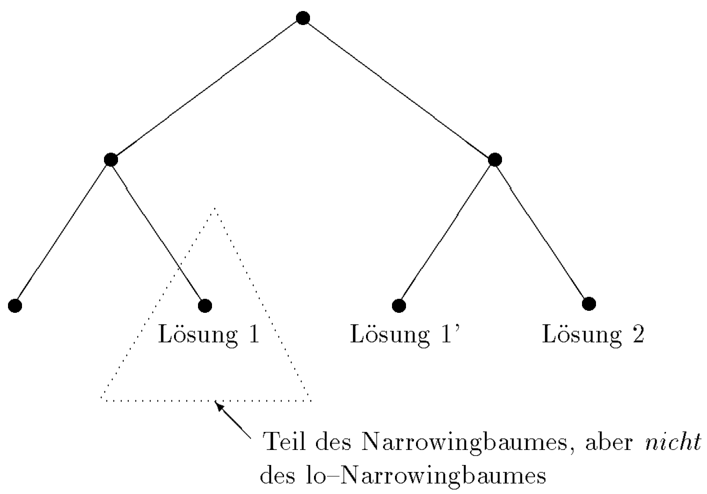

Abbildung 1.2: Narrowing- bzw. leftmost outermost Narrowingbaum

Lösung 1 ist nicht durch den lo–Narrowing Kalkül berechenbar, sondern nur durch den Narrowing Kalkül. Die Lösungen 2 und 1' sind durch beide Kalküle berechenbar und es gilt Lösung 1 äquivalent zu Lösung 1'. Kann es ein Termersetzungssystem $\mathcal{R}$ und ein Kalkulandum d geben, so daß ein lo–Narrowingbaum und ein Narrowingbaum entsprechend Abbildung 1.2 entstehen? Wäre die Antwort ja, dann gäbe es tatsächliche Fälle, in denen Narrowing effizienter ist als lo–Narrowing. Wie man sieht, ist diese Frage nicht so trivial wie sie auf den ersten Blick zu sein scheint.

Die Arbeit ist in acht Kapitel aufgeteilt. Kapitel 2 enthält grundlegende Definitionen und Schreibweisen, die wir während der gesamten Arbeit benutzen werden. In Kapitel 3 werden die verschiedenen Kalkülkonzepte, i.e. Kalkül, indizierter Kalkül und deterministischer Kalkül definiert. Im vierten Kapitel wird E–Unifikation definiert und es werden Ergebnisse über die Vollständigkeit verschiedener Methoden zum Finden von E–Unifikatoren aus der Literatur zitiert. Darüberhinaus werden die aus diesen Vollständigkeitsresultaten abgeleiteten sogenannten universellen Unifikationsalgorithmen angegeben. In Kapitel 5 wenden wir die Definitionen der Kalküle aus dem dritten Kapitel auf die vier oben genannten verschiedenen Methoden für E–Unifikation an. Dabei werden wir in drei Fällen, nämlich für Reduktion, Narrowing und leftmost outermost Narrowing, jeweils zwei Kalküle angeben; einen für das allgemeine Rechnen mit nur einem Term (also nicht für E–Unifikation) und den zweiten für E–Unifikation. In Kapitel 6 definieren wir die vier erwähnten Effizienzmaße, um sie dann im siebten Kapitel für den Vergleich zwischen Narrowing und lo–Narrowing und zwischen lo–Narrowing und ulo–Narrowing zu benutzen. Das letzte Kapitel enthält einige abschließende Bemerkungen und Fragestellungen für zukünftige Arbeiten auf diesem Gebiet.

Kapitel 2

Preliminarien

In diesem Kapitel wollen wir allgemeine Notationen und Definitionen, die während der gesamten Arbeit gebraucht und benutzt werden, aufschreiben. Wir halten uns dabei größtenteils an die Notationen aus [FV94] und [FV92].

2.1 Allgemeine Notationen

- Die *Menge der nicht–negativen ganzen Zahlen* wird durch $\mathbb{N}$ bezeichnet.
- Die *leere Menge* wird durch $\emptyset$ bezeichnet.
- Für ein $j \in \mathbb{N}$ bezeichnet $[j]$ die Menge $\{1, \ldots, j\}$; somit gilt: $[0] = \emptyset$.
- Für eine endliche Menge A bezeichnet $card(A)$ die *Kardinalität* von A.
- Für eine Menge A bezeichnet A^* die Menge $\bigcup_{n \in \mathbb{N}} \{a_1 a_2 \ldots a_n \mid \text{für alle } i \in [n] : a_i \in A\}$. Diese Menge heißt *die Menge der Wörter über A*.
- Das *leere Wort* wird durch ε bezeichnet.
- Für eine Menge M und ein Element p ist die Menge *M ohne p*, bezeichnet durch $M \setminus p$, die Menge $\{m \in M \mid m \neq p\}$.
- Für eine Menge M ist die *Potenzmenge von M*, bezeichnet durch $\mathcal{P}(M)$ die Menge $\{M' \mid M' \subseteq M\}$.
- Sind $f : A \to B$ und $g : B \to C$ Funktionen, so ist eine *Komposition* $g \circ f$ (lies: g nach f): $A \to C$ definiert durch $(g \circ f)(x) = g(f(x))$ für alle $x \in A$.

2.2 Rangalphabete, Variablen und Terme

Wir wollen Terme über einem Rangalphabet mit Variablen definieren. Dazu definieren wir zunächst den Begriff des Rangalphabets.

Definition 2.1 (Rangalphabet)
Ein Paar $(\Omega, \underline{rank}_\Omega)$ heißt *Rangalphabet*, wenn gilt:

1. Ω ist eine endliche Menge; Ω heißt *Alphabet*.
2. $\underline{rank}_\Omega : \Omega \to \mathbb{N}$ ist eine totale Funktion.

□

Es gelten folgende Konventionen:

- Für ein Symbol $f \in \Omega$ heißt $\underline{rank}_\Omega(f)$ der *Rang von f*.
- Die Menge $\Omega^{(i)} \subseteq \Omega$ besteht aus allen Symbolen aus Ω mit Rang i für $i \geq 0$.
- Es gilt: $\Omega^{(i)} \cap \Omega^{(j)} = \emptyset$ für $i \neq j$.

Ein Rangalphabet kann entweder angegeben werden durch (i) Aufzählung der nichtleeren Teilmengen $\Omega^{(i)}$, oder durch (ii) eine Menge von Symbolen, die mit ihrem eindeutigen Rang indiziert sind. Wir werden in dieser Arbeit vorwiegend die letztere Methode benutzen.

Wenn der Rang der Symbole in einem Kontext klar ist, dann lassen wir die Funktion $\underline{rank}_\Omega$ weg, und schreiben für das Rangalphabet nur noch Ω.

Beispiel:

$\Omega = \{CONS^{(2)}, NIL^{(0)}, A^{(0)}, B^{(0)}, C^{(0)}\}$ ist ein Rangalphabet.

Für den Rest dieser Ausarbeitung wählen wir als *Menge von Variablen* die Menge $\mathcal{V} = \{x, x_1, x_2, \ldots, y, y_1, y_2, \ldots, z, z_1, z_2, \ldots\}$.

Definition 2.2 (Terme)
Sei $(\Omega, \underline{rank}_\Omega)$ ein Rangalphabet und S eine beliebige Menge. Die Menge der *Terme über* Ω *indiziert durch S*, bezeichnet durch $T(\Omega, S)$ ist die kleinste Menge $T \subseteq (\Omega \cup S \cup \{(,\ ),\ ,\})^*$, so daß die beiden folgenden Aussagen gelten.

1. $S \subseteq T(\Omega, S)$.
2. Wenn $f \in \Omega^{(n)}$ mit $n \geq 0$ und $t_1, \ldots, t_n \in T(\Omega, S)$, dann $f(t_1, \ldots, t_n) \in T(\Omega, S)$.

□

<u>Beispiel:</u>

Für das Rangalphabet vom vorigen Beispiel sind dies Terme indiziert durch $\mathcal{V}$:
A, $CONS(A, x_2)$, $CONS(CONS(NIL, A))$.
Keine Terme sind $CONS(CONS, CONS(B, NIL))$, $CONS(D, NIL)$.

Im folgenden sei $(\Omega, \underline{rank}_\Omega)$ immer ein beliebiges, aber festes Rangalphabet.

Definition 2.3 (Menge der Vorkommen von t)

Für jeden Term $t \in T(\Omega, \mathcal{V})$ ist die *Menge der Vorkommen von t*, bezeichnet durch $O(t)$, eine Untermenge von $\mathbb{N}^*$ und ist wie folgt induktiv über den Aufbau von t definiert:

1. Wenn $t = x$ mit $x \in \mathcal{V}$, dann $O(t) = \{\varepsilon\}$,
2. wenn $t = f$ mit $f \in \Omega^{(0)}$, dann $O(t) = \{\varepsilon\}$, und
3. wenn $t = f(t_1, \ldots, t_n)$ mit $f \in \Omega^{(n)}$ und $n > 0$, und für alle $i \in [n] : t_i \in T(\Omega, \mathcal{V})$, dann $O(t) = \{\varepsilon\} \cup \bigcup\limits_{i \in [n]} \{iu \mid u \in O(t_i)\}$.

□

Wir werden folgende Notationen verwenden:

- Für einen Term $t \in T(\Omega, \mathcal{V})$ und ein Vorkommen u von t, bezeichnet t/u den *Teilterm von t am Vorkommen u.*
- Die *Beschriftung von t an einem Vorkommen u* wird durch $t[u]$ bezeichnet.
- Die *Menge der Variablen, die in t vorkommen*, bezeichnen wir mit $\mathcal{V}(t)$; d.h. $x \in \mathcal{V}(t)$, wenn $x \in \mathcal{V}$ und es gibt ein $u \in O(t)$, so daß $t/u = x$.
- Der Term t, in dem der Teilterm am Vorkommen u durch den Term s ersetzt wurde, wird mit $t[u \leftarrow s]$ bezeichnet.
- Die lexikographische Ordnung auf $O(t)$ wird durch $<_{lex}$ bezeichnet.
- Für eine Teilmenge S von $O(t)$ und eine totale Ordnung $\sqsubset$ wird das *minimale Element von S bezüglich* $\sqsubset$ durch $min_\sqsubset(S)$ bezeichnet.

Natürlich ist diese Art der Darstellung von Vorkommen nicht mehr eindeutig, falls ein Symbol des Rangalphabetes einen Rang > 9 hat. Dann sollte man gemäß der *Dewey–Notation* Punkte zwischen den einzelnen i's einfügen, damit die Eindeutigkeit der Darstellung erhalten bleibt.

<u>Beispiel</u>

Sei $t = CONS(A, CONS(x, NIL))$ ein Term. Die Menge der Vorkommen von t ist die Menge:

$$
\begin{aligned}
O\,(t) &= O\,(CONS(\underbrace{A}_{t_1}, \underbrace{CONS\,(x, NIL\,)}_{t_2})) \\
&= \{\varepsilon\} \cup \bigcup_{i \in \{1,2\}} \{iu \mid u \in O\,(t_i)\} \\
&= \{\varepsilon\} \cup \{1u \mid u \in \underbrace{O\,(A)}_{\{\varepsilon\}}\} \cup \{2u \mid u \in O\,(CONS\,(x, NIL\,))\} \\
&= \{\varepsilon\} \cup \{1\} \cup \{2u \mid u \in (\{\varepsilon\} \cup \{1v \mid v \in \underbrace{O\,(x)}_{\{\varepsilon\}}\} \cup \{2v \mid v \in \underbrace{O\,(NIL\,)}_{\{\varepsilon\}}\})\} \\
&= \{\varepsilon, 1\} \cup \{2u \mid u \in \{\varepsilon, 1, 2\}\} \\
&= \{\varepsilon, 1, 2, 21, 22\}
\end{aligned}
$$

Hier ist $CONS\,(A, CONS\,(x, NIL\,))/22$ gleich dem Term NIL .

Definition 2.4 (Baumfunktionen, Beschriftung)

Sei t ein Baum. Die Menge der *Knoten von* t bezeichnen wir duch $\underline{node}\,(t)$, die *Wurzel von* t wird duch $\underline{root}\,(t)$ bezeichnet. Für einen Knoten $x \in \underline{node}\,(t)$ kürzen wir die *Beschriftung von* x mit $\underline{label_t}\,(x)$ ab.

□

2.3 Substitutionen

Wir wollen nun definieren, was wir unter einer $(\mathcal{V}, \Omega)$–Substitution verstehen.

Definition 2.5 ($(\mathcal{V}, \Omega)$–Substitution)

Eine Abbildung $\varphi : \mathcal{V} \to T\,(\Omega, \mathcal{V})$ heißt $(\mathcal{V}, \Omega)$–Substitution, wenn $\{v \mid v \in \mathcal{V}$ und $\varphi\,(v) \neq v\}$ eine endliche Menge ist.

Die Menge aller möglichen $(\mathcal{V}, \Omega)$–Substitutionen bezeichnen wir duch $Sub\,(\mathcal{V}, \Omega)$.

□

Eine konkrete $(\mathcal{V}, \Omega)$–Substitution φ gibt man durch die Werte derjenigen, endlich vielen Variablen an, die sich unter φ verändern. Seien also $v_1, \ldots, v_n \in \mathcal{V}$ alle Variablen, für die $\varphi\,(v_i) \neq v_i$ gilt. Dann ist φ eindeutig durch $\varphi\,(v_1), \ldots, \varphi\,(v_n)$ bestimmt.

<u>Beispiel:</u>

Die Abbildung φ, definiert durch $\varphi\,(x_1) = NIL$ und $\varphi\,(x_2) = CONS\,(a, NIL\,)$, ist eine $(\Omega, \mathcal{V})$–Substitution (Ω wie in den vorherigen Beispielen). Die Abbildung σ, definiert durch $\sigma\,(x_i) = x_{i+1}$, ist keine $(\Omega, \mathcal{V})$–Substitution.

Im Zusammenhang mit Substitutionen werden wir noch folgende Begriffe verwenden:

- Sei $f : M \to N$ eine Funktion. Eine Funktion $g : U \to N$ heißt *Einschränkung von* f *(auf* U*)*, wenn $U \subseteq M$ und $f(x) = g(x)$ für alle $x \in U$ ist. Anstelle von g wird $f|_U$ (lies: f eingeschränkt auf U) geschrieben.

- Sei φ eine $(\mathcal{V},\Omega)$–Substitution. Die *homomorphe Fortsetzung von* φ *zu* $T(\Omega,\mathcal{V})$ ist die Funktion $\hat{\varphi}$, die induktiv wie folgt definiert ist:
  1. Für alle $v \in \mathcal{V}$ gilt: $\hat{\varphi}(v) = \varphi(v)$.
  2. Für alle $\alpha \in \Omega^{(0)}$ gilt: $\hat{\varphi}(\alpha) = \alpha$.
  3. Für alle $f(t_1,\ldots,t_n)$ mit $f \in \Omega^{(n)}$ und $n \geq 1$, und für alle $i \in [n] : t_i \in T(\Omega,\mathcal{V})$ gilt: $\hat{\varphi}(f(t_1,\ldots,t_n)) = f(\hat{\varphi}(t_1),\ldots,\hat{\varphi}(t_n))$.
- Für eine Substitution $\varphi \in \mathit{Sub}(\mathcal{V},\Omega)$ heißt die Menge $\{x \mid \varphi(x) \neq x\}$ *domain* von φ; sie wird bezeichnet durch $\mathcal{D}(\varphi)$.
- Wenn $\mathcal{D}(\varphi) = \emptyset$, dann wird φ durch $\varphi_\emptyset$ bezeichnet.
- Wir sagen daß φ *grund* ist, wenn für alle $x \in \mathcal{D}(\varphi) : \mathcal{V}(\varphi(x)) = \emptyset$. Die Menge der grund $(\mathcal{V},\Omega)$–Substitutionen wird durch $\mathit{gSub}(\mathcal{V},\Omega)$ bezeichnet.
- Die Menge $\mathcal{I}(\varphi) = \bigcup_{x\in\mathcal{D}(\varphi)} \mathcal{V}(\varphi(x))$ ist die *Menge der durch* φ *eingeführten Variablen.*

2.4 Relationen und Ordnungen

Die Begriffe Relation, Ordnung und verwandte Begriffe werden in dieser Ausarbeitung oft benutzt. Es ist deshalb angebracht sie hier genau zu definieren.

Definition 2.6 (Ordnung, etc.)

Sei M eine Menge. Eine Teilmenge $\sqsubseteq$ von $M \times M$ heißt *Ordnung*, wenn $\sqsubseteq$ reflexiv, antisymmetrisch und transitiv ist. Eine binäre Relation $R \subseteq M \times M$ heißt *totale Ordnung*, falls R asymmetrisch, transitiv und konnex ist. Eine Relation $R \subseteq M \times M$ heißt *Äquivalenzrelation*, wenn sie reflexiv, symmetrisch und transitiv ist. Die Menge aller zu $x \in M$ stehenden Elemente von M heißt *Äquivalenzklasse* $[\![x]\!]$ *über* x:

$$[\![x]\!] = \{z \mid x \in M \text{ und } z \in M \text{ und } x\,R\,z\}$$

Die Menge $M/_R = \{[\![x]\!] \mid x \in M\}$ heißt *Quotientenmenge von* M *nach* R.

Eine Äquivalenzrelation $\sim$ über $T(\Omega,\mathcal{V})$ heißt Ω*–Kongruenz über* $T(\Omega,\mathcal{V})$, wenn für alle $f \in \Omega^{(n)}$, mit $n > 0$ und für alle $t_1, s_1, \ldots t_n, s_n \in T(\Omega,\mathcal{V})$ mit $t_1 \sim s_1, \ldots t_n \sim s_n$ gilt:

$$f(t_1,\ldots,t_n) \sim f(s_1,\ldots,s_n).$$

□

Kapitel 3

Formale Kalküle

In diesem Kapitel wollen wir zunächst ganz allgemein den Begriff eines Kalküls formalisieren. Der Begriff des Kalküls (von lat. calculus: Berechnung) subsumiert die Vorstellung, daß eine Menge von Zwischenergebnissen und eine binäre Relation gegeben ist. Die binäre Relation legt fest, wie von einem Zwischenergebnis ein neues Zwischenergebnis berechnet werden kann; deshalb nennen wir diese binäre Relation hier Rechenrelation. Liegt nun ein Objekt vor, welches berechnet werden soll, so nennen wir dieses Objekt ein Kalkulandum (von lat. calculandum: das zu Berechnende); natürlich ist es auch ein Zwischenergebnis. Auf das Kalkulandum kann nun die Rechenrelation angewendet werden und es entsteht ein neues Zwischenergebnis. Auf dieses Zwischenergebnis kann wiederum die Rechenrelation angewendet werden und es entsteht wiederum ein neues Zwischenergebnis. Eine solche Sequenz von Anwendungen der Rechenrelation nennen wir eine Berechnung. Das Ziel einer Berechnung ist ein Zwischenergebnis, welches besondere Merkmale aufweist, z.B. daß die Rechenrelation nicht mehr anwendbar ist. Ein solches Zwischenergebnis nennen wir Endergebnis.

Ein Beispiel für ein Kalkül ist das Rechnen mit arithmetischen Ausdrücken. Die Menge der Zwischenergebnisse ist dann die Menge aller arithmetischen Ausdrücke. Die Rechenrelation könnte beispielsweise Umformungen gemäß der Kommutativität, Assoziativität und der Distributivität der Operatoren beinhalten.

Ein anderes Beispiel sind Kalküle der mathematischen Logik. Dort ist die Menge der Zwischenergebnisse gleich der Menge der logischen Formeln und die Rechenrelation beinhaltet die entsprechenden Deduktionsregeln des Kalküls.

3.1 Kalkül

Wir wollen jetzt eine Definition für einen Kalkül geben:

Definition 3.1 (Kalkül)
Ein *Kalkül*, bezeichnet durch $\mathcal{C}$, ist ein Tupel $(D, D_{\text{success}}, \vdash)$ mit:

1. D ist eine Menge; die Elemente von D heißen *Zwischenergebnisse*.
2. D_{success} ist eine Menge mit: $D_{\text{success}} \subseteq D$; die Elemente von D_{success} heißen *Endergebnisse*.
3. $\vdash \subseteq D \times D$ ist eine binäre Relation; $\vdash$ heißt *Rechenrelation*.

□

Wir werden folgende Bezeichnungen und Schreibweisen verwenden:

- Statt $(d, d') \in \vdash$ schreiben wir $d \vdash d'$. Wenn $d \vdash d'$, dann sagen wir: *d wird durch $\vdash$ abgeleitet zu d'*, oder: *d' wird aus d durch $\vdash$ berechnet.*
- $\vdash^+$: transitive Hülle von $\vdash$
- $\vdash^*$: transitive reflexive Hülle von $\vdash$
- $d \vdash d'$: *Berechnungsschritt von* $\vdash$
- $d \vdash^* d'$: *Berechnung durch $\vdash$ beginnend mit d*

Wir wollen nun ein Beispiel für einen Kalkül angeben. Es sollen arithmetische Ausdrücke über Addition und Multiplikation betrachtet werden, und das Assoziativgesetz und Distributivgesetz für Multiplikation als Rechenrelation benutzt werden.

Beispiel 3.2 Sei $\mathcal{C} = (D, D_{\text{success}}, \vdash)$ ein Kalkül mit:

- $D = T(\Omega_1, \emptyset)$. Mit $\Omega_1 = \{+^{(2)}, *^{(2)}, a^{(0)}, b^{(0)}, c^{(0)}\})$. Dies ist gemäß Definition 2.2 die Menge: $a, b, c \in T(\Omega_1, \emptyset)$ und für alle $t_1, t_2 \in T(\Omega_1, \emptyset)$ gilt: $+(t_1, t_2), *(t_1, t_2) \in T(\Omega_1, \emptyset)$. Wir werden hier und in den folgenden Beispielen meist Infix–Notation für Symbole wie $+$ und $*$ benutzen, d.h. wir schreiben $(a \circ b)$ anstatt $\circ(a, b)$ für $\circ \in \{+, *\}$.
- $D_{\text{success}} = \emptyset$.
- Sei $t \in T(\emptyset, \Omega_1)$, $u \in O(t)$ und $t' = t/u$, dann gilt:
  1. falls $t' = (t_1 * (t_2 * t_3))$ mit $t_1, t_2, t_3 \in T(\Omega_1, \emptyset)$, dann
  $$(t, t[u \leftarrow ((t_1 * t_2) * t_3)]) \in \vdash .$$
  2. falls $t' = (t_1 * (t_2 + t_3))$ mit $t_1, t_2, t_3 \in T(\Omega_1, \emptyset)$, dann
  $$(t, t[u \leftarrow ((t_1 * t_2) + (t_1 * t_3))]) \in \vdash .$$

Das Folgende ist eine Berechnung durch $\vdash$ beginnend mit $(a * (a * (b + c)))$:

$$\begin{array}{ll} & (a * (a * (b + c))) \\ \vdash & ((a * a) * (b + c)) \\ \vdash & (((a * a) * b) + ((a * a) * c)) \end{array}$$

Es ist offensichtlich, daß dies nicht die einzig mögliche Berechnung durch $\vdash$ beginnend mit $(a * (a * (b + c)))$ ist. Im ersten Berechnungsschritt hätte nämlich auch gemäß der Distributivität das Zwischenergebnis $(a * ((a * b) + (a * c)))$ berechnet werden können.

□

Das vorige Beispiel zeigt, daß aus einem Zwischenergebnis im allgemeinen mehrere verschiedene Zwischenergebnisse berechnet werden können.

Definition 3.3 (Menge der k–Nachfolger)

Sei $\mathcal{C} = (D, D_{\text{success}}, \vdash)$ ein Kalkül, $d \in D$ ein Zwischenergebnis und $k \geq 1$. Die *Menge der k–Nachfolger von d*, bezeichnet durch $Suc_k\,(d, \vdash)$ ist die Menge $\{d' \in D \mid d \vdash^k d'\}$. Die Menge $Suc_1\,(d, \vdash)$ heißt auch *Menge der direkten Nachfolger von d* und wird durch $Suc\,(d, \vdash)$ bezeichnet.

□

Wir werden folgende Bezeichnungen und Schreibweisen verwenden:

- Ein Zwischenergebnis $d \in D$ heißt *Normalform*, wenn $Suc\,(d, \vdash)$ leer ist.
- Eine Substitution $\varphi \in Sub\,(\mathcal{V}, \Omega)$ ist *in Normalform*, wenn für alle $x \in \mathcal{D}(\varphi) : \varphi(x)$ in Normalform ist.
- d' ist ein *Endergebnis von d*, wenn $d \vdash^* d'$ und $d' \in D_{\text{success}}$.

Das Zwischenergebnis $(((a * a) * b) + ((a * a) * c))$ im Beispiel 3.2 ist in Normalform, da $Suc\,((((a * a) * b) + ((a * a) * c)), \vdash)$ leer ist. Es ist aber kein Endergebnis, da die Menge D_{success} leer ist. Für D_{success} kann eine beliebige Untermenge von D gewählt werden, die für das entsprechende Problem geeignet ist.

3.2 Indizierter Kalkül und Berechnungsbaum

Betrachten wir noch einmal die Definition des Kalküls aus dem vorigen Abschnitt. Es gibt bestimmte Kalküle, in denen es für ein Zwischenergebnis $d \in D$ beliebig viele Nachfolger gibt. Für jeden dieser Nachfolger kann es wiederum beliebig viele Nachfolger geben. Wir haben bis jetzt noch keine Möglichkeit bereitgestellt, um diese verschiedenen Nachfolger eines Zwischenergebnisses zu unterscheiden und zu identifizieren.

Eine bestimmte Berechnung durch $\vdash$ können wir nur durch Auflistung aller Zwischenergebnisse $d_i \in D$, die in der Berechnung auftauchen, darstellen. Wir möchten eine Berechnung aber gerne durch Auflistung der angewandten Relation charakterisieren; deshalb müssen wir die Elemente der Menge der direkten Nachfolger von einem Zwischenergebnis noch irgendwie unterscheiden.

Wir brauchen ein Objekt, welches alle möglichen Berechnungen durch $\vdash$ beginnend mit einem Kalkulandum enthält, und es außerdem noch ermöglicht, jede Berechnung eindeutig zu charakterisieren. Für diesen Zweck bietet sich ein sogenannter Berechnungsbaum an.

Wir wollen nun die Definition des Kalküls dahingehend erweitern, daß zu jedem Kalkül $\mathcal{C}$ und jedem Kalkulandum $d \in D$ ein eindeutiger Berechnungsbaum konstruiert werden kann. Jeder Weg durch den Baum, beginnend an einem Knoten mit der Beschriftung d mit $d \in D$ und endend an einem Knoten mit der Beschriftung d' mit $d' \in D$, wird dabei genau der Berechnung durch $\vdash$, beginnend mit d und endend mit d', entsprechen. Dazu werden wir die Rechenrelation $\vdash$ partitionieren in Relationen $\vdash_j$, die für ein Zwischenergebnis $d \in D$ jeweils nur höchstens einen direkten Nachfolger besitzen. Die Indizes j werden aus einer Indexmenge gewählt, über die wir eine totale Ordnung $<$ definieren. Die Ordnung erlaubt es, für die Menge der Nachfolgeknoten mit den Beschriftungen $\{d' \mid \text{es gibt ein } j : d \vdash_j d'\}$ eines Knotens mit Beschriftung d eine eindeutige Anordnung festzulegen.

Definition 3.4 (Indizierter Kalkül)
Ein *indizierter Kalkül* $\mathcal{I}$ ist ein Tupel $(D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ mit:

1. D ist eine Menge; die Elemente von D heißen Zwischenergebnisse.
2. D_{success} ist eine Menge mit $D_{\text{success}} \subseteq D$; die Elemente von D_{success} heißen Endergebnisse.
3. J ist eine Menge; die Menge J nennen wir *Indexmenge*; die Elemente von J nennen wir *Indizes*.
4. $<$ ist eine totale Ordnung auf J.
5. $(\vdash_j \mid j \in J)$ ist eine Familie von binären Relationen $\vdash_j \subseteq D \times D$, so daß für alle $d \in D$ gilt: $\mathrm{Suc}\,(d, \vdash_j)$ enthält höchstens ein Element.

□

Definition 3.5 (Indizierter Kalkül für $\mathcal{C}$)
Sei $\mathcal{C} = (D, D_{\text{success}}, \vdash)$ ein Kalkül und sei $\mathcal{I} = (D', D'_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ ein indizierter Kalkül. $\mathcal{I}$ ist ein *indizierter Kalkül für* $\mathcal{C}$, wenn:

1. $D' = D$
2. $D'_{\text{success}} = D_{\text{success}}$
3. $(\bigcup_{j \in J} \vdash_j) = \vdash$

□

Wir greifen nun wieder das Beispiel 3.2 auf und konstruieren einen indizierten Kalkül $\mathcal{I}$ für den Kalkül $\mathcal{C}$. Wir werden dazu als Indexmenge J die Menge $\{\text{assoc}, \text{distr}\} \times \mathbb{N}^*$ wählen. Diese Wahl ist gerechtfertigt, weil die beiden Nichtdeterminismen, die in einem Berechnungsschritt $t \vdash t'$ auftreten, folgende sind:
(i) die angewandte Regel (Assoziativgesetz oder Distributivgesetz) und
(ii) das Vorkommen O(t), auf das die Regel angewendet wird.
Als totale Ordnung auf J wählen wir willkürlich (assoc, u) $<$ (distr, u) für ein Vorkommen u; falls in einem Berechnungsschritt eine Regel an verschiedenen Vorkommen angewendet werden kann, verwenden wir die lexikographische Ordnung auf den Vorkommen.

Beispiel 3.6 Sei $\mathcal{C} = (T(\{+\}, \{a, b\}), \emptyset, \vdash)$ der Kalkül aus Beispiel 3.2. Das Tupel $\mathcal{I} = (D', D'_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ ist ein indizierter Kalkül für $\mathcal{C}$ mit:

- $D' = D = T(\Omega_1, \emptyset)$
- $D'_{\text{success}} = D_{\text{success}} = \emptyset$
- $J = \{\text{assoc}, \text{distr}\} \times \mathbb{N}^*$
- $<$ ist eine totale Ordnung auf $\{\text{assoc}, \text{distr}\} \times \mathbb{N}^*$, so daß für alle $(r_1, u_2), (r_2, u_2) \in J$ gilt: $(r_1, u_1) < (r_2, u_2)$, falls eine der beiden folgenden Bedingungen erfüllt ist:
  1. $r_1 = \text{assoc}$ und $r_2 = \text{distr}$.
  2. $r_1 = r_2$ und $u_1 <_{lex} u_2$.
- $(\vdash_j \mid j \in J)$ ist eine Familie mit:
  Sei $t \in T(\Omega_1, \emptyset)$, $u \in O(t)$ und $t' = t/u$, dann gilt:
  1. falls $t' = (t_1 * (t_2 * t_3))$ mit $t_1, t_2, t_3 \in T(\Omega_1, \emptyset)$, dann
  $$(t, t[u \leftarrow ((t_1 * t_2) * t_3)]) \in \vdash_{\text{assoc}, O(t)} .$$
  2. falls $t' = (t_1 * (t_2 + t_3))$ mit $t_1, t_2, t_3 \in T(\Omega_1, \emptyset)$, dann
  $$(t, t[u \leftarrow ((t_1 * t_2) + (t_1 * t_3))]) \in \vdash_{\text{distr}, O(t)}$$

Es gilt dann:
$$\begin{array}{ll} & (a * (a * (b + c))) \\ \vdash_{\text{assoc},\varepsilon} & ((a * a) * (b + c)) \\ \vdash_{\text{distr},\varepsilon} & (((a * a) * b) + ((a * a) * c)) \end{array}$$

oder auch:
$$\begin{array}{ll} & (a * (a * (b + c))) \\ \vdash_{\text{distr},2} & ((a * ((a * b) + (a * c)) \end{array}$$

□

Dieses Beispiel zeigt, daß im indizierten Kalkül jeder mögliche Berechnungsschritt $t \vdash t'$ eines Kalküls $\mathcal{C}$ eindeutig durch einen Index $j \in J$ bestimmt ist. Darüberhinaus legt die totale Ordnung $<$ auf J eine feste Ordnung der Berechnungsschritte $\vdash_j$ fest.

Wir werden zunächst eine Menge, die wir häufig benutzen werden, definieren. Es ist die Menge der Indizes j, mit deren zugehörigen Relationen $\vdash_j$ aus einem Zwischenergebnis $d \in D$ neue Zwischenergebnisse berechnet werden können.

Definition 3.7 (Menge der Nachfolgeindizes)
Sei $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ ein indizierter Kalkül und $d \in D$ ein Zwischenergebnis. Die *Menge der Nachfolgeindizes von d*, bezeichnet durch $I(d)$, ist die Menge $\{j \in J \mid \text{es gibt ein } d' \in D : d \vdash_j d'\}$.

□

Jedes Zwischenergebnis $d \in D$ hat also genau n Nachfolger, wenn die Mächtigkeit der Menge der Nachfolgeindizes $I(d)$ gleich n ist. Wir können nun den Begriff des Berechnungsbaums definieren:

Definition 3.8 (Berechnungsbaum)
Sei $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ ein indizierter Kalkül und $d \in D$ ein Kalkulandum. Der *Berechnungsbaum für $\mathcal{I}$ und d*, bezeichnet durch $B(\mathcal{I}, d)$, ist der eindeutig bestimmte Baum, so daß gilt:

1. Jeder Knoten von $B(\mathcal{I}, d)$ ist mit einem Zwischenergebnis $d' \in D$ beschriftet.
2. Die Wurzel von $B(\mathcal{I}, d)$ ist mit d beschriftet, also $\underline{label}_{B(\mathcal{I},d)}(\underline{root}(B(\mathcal{I}, d))) = d$.
3. Sei d' die Beschriftung an einem Knoten nd von $B(\mathcal{I}, d)$, und sei die Menge der Nachfolgeindizes von d' die Menge $I(d') = \{j_1, \ldots, j_n\}$ mit $j_1 < j_2 < \ldots < j_n$. Dann besitzt der Knoten nd n Nachfolger, und für jedes $k \in [n]$ gilt: der k-te Nachfolger von nd ist beschriftet mit $\underline{label}_{B(\mathcal{I},d)}(nd.k) = \hat{d}$, wenn $d' \vdash_{j_k} \hat{d}$.

□

Der Berechnungsbaum für einen indizierten Kalkül $\mathcal{I}$ und ein Zwischenergebnis d hängt nicht von der Menge D_{success} ab. Wir haben der Menge D_{success}, außer der Bezeichnung ihrer Elemente als Endergebnisse, noch keine Bedeutung zugeteilt. Sie ist bis jetzt nur eine Menge, die optional genutzt werden kann, um ganz bestimmte Zwischenergebnisse als Endergebnisse auszuzeichnen. Für zwei indizierte Kalküle, die sich nur in der Menge D_{success} unterscheiden, sind die Berechnungsbäume für alle Zwischenergebnisse d deswegen gleich. Diese Eigenschaft wird durch folgendes Lemma beschrieben.

Lemma 3.9 Seien $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ und $\mathcal{I}' = (D, D'_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ indizierte Kalküle, dann gilt für alle $d \in D$

$$B(\mathcal{I}, d) = B(\mathcal{I}', d)$$

□

Beispiel 3.10 Sei $\mathcal{I}$ der indizierte Kalkül aus Beispiel 3.6.
Der Berechnungsbaum $B(\mathcal{I}, (a * (a * (b + c)))$ ist der folgende Baum:

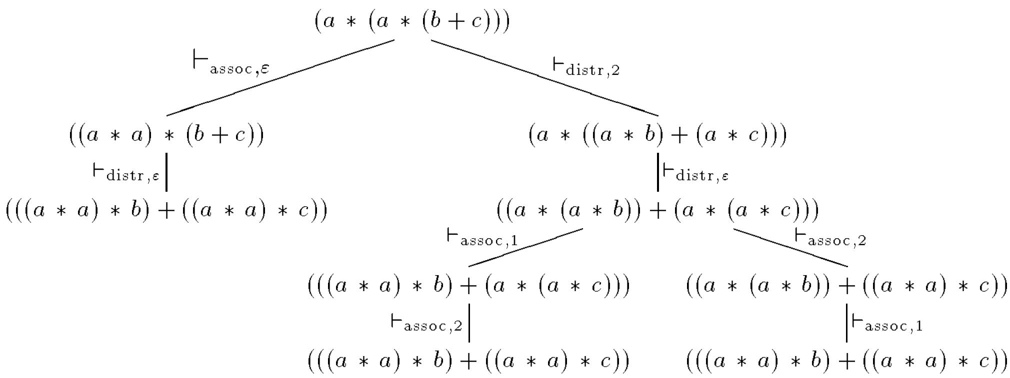

Abbildung 3.1: Berechnungsbaum für $\mathcal{I}$ und $(a * (a * (b + c)))$

$\square$

Alle Blätter des Berechnungsbaumes in der obigen Abbildung sind in Normalform; d.h. die Abbildung stellt diesen Berechnungsbaum vollständig dar und beschreibt dadurch in eindeutiger Weise alle Möglichkeiten von Berechnungen durch $\vdash$, beginnend mit dem Term $(a * (a * (b + c)))$. Wir stellen zusätzlich fest, daß alle Berechnungen zu demselben Term führen, i.e. der Term $(((a * a) * b) + ((a * a) * c))$. Diese ausgezeichnete Eigenschaft einer Rechenrelation wird später noch von besonderer Bedeutung sein.

3.3 Deterministischer Kalkül

Es sei nun ein indizierter Kalkül $\mathcal{I}$ gegeben. Wir wollen jetzt für $\mathcal{I}$ einen Kalkül definieren, der zu jedem Kalkulandum d deterministisch alle nachfolgenden Zwischenergebnisse d' mit $d \vdash^* d'$ bezüglich $\mathcal{I}$ berechnet. Im Sinne des Berechnungsbaumes wird der zu definierende Kalkül ein depth–first left–to–right tree traversal (= Baumdurchlauf) über $B(\mathcal{I}, d)$ sein. Ein Zwischenergebnis dieses Kalküls wird zusätzlich noch eine Menge M beinhalten, die angibt welche Äste des Baumes, beginnend an diesem Zwischenergebnis, noch abgearbeitet werden müssen.

Wir werden nun zwei Funktionen definieren, die wir zur Definition des deterministischen Kalküls benötigen. Es handelt sich dabei um die Funktionen *first* und *rest*, die bezüglich einer total geordneten Menge $(J, <)$ definiert werden. Die Funktion *first* liefert das minimale Element einer Menge $M \subseteq J$ bezüglich der totalen Ordnung $<$; die Funktion *rest* liefert alle Elemente aus $M \subseteq J$ ohne das minimale Element.

Definition 3.11 (*first, rest*)
Sei $(J, <)$ eine total geordnete Menge. Für alle $M \subseteq J$ gilt:

- Die Funktion *first*: $\mathcal{P}(J) \to J$ ist definiert als: $\mathit{first}(M) = \mathit{min}_{<}(M)$.
- Die Funktion *rest*: $\mathcal{P}(J) \to \mathcal{P}(J)$ ist definiert als: $\mathit{rest}(M) = M \setminus \mathit{first}(M)$.

□

Definition 3.12 (deterministischer Kalkül)
Sei $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ ein indizierter Kalkül. *Der deterministische Kalkül von* $\mathcal{I}$, bezeichnet durch $\mathit{det}(\mathcal{I})$, ist der Kalkül $(D', D'_{\text{success}}, \models)$ mit:

1. $D' = D_0^+$, mit $D_0 = \{(d, M) \mid d \in D, M \subseteq I(d)\}$
2. $D'_{\text{success}} = \{w(d, I(d)) \mid w \in D_0^*, d \in D_{\text{success}}\}$
3. $\models \subseteq D' \times D'$, so daß gilt:
   (a) für alle $w \in D'$ und $d \in D$: $w(d, \emptyset) \models w$
   (b) für alle $w \in D_0^*$ und $(d, M) \in D_0$ mit $M \neq \emptyset$,
   $$w(d, M) \models w(d, M')(\hat{d}, I(\hat{d})),$$
   falls gilt: $d \vdash_{\mathit{first}\,(\mathrm{M})} \hat{d}$ mit $M' = \mathit{rest}(M)$.

□

Sei $\mathcal{I}$ ein indizierter Kalkül, $\mathit{det}(\mathcal{I}) = (D', D'_{\text{success}}, \models)$ der deterministische Kalkül von $\mathcal{I}$ und $d \in D'$ ein Zwischenergebnis. Es ist zu bemerken, daß die Menge der k–Nachfolger mit $k \geq 1$ beim deterministischen Kalkül tatsächlich (wie gewünscht) immer nur höchstens *ein* Element enthält.

Beispiel 3.13 Sei $\mathcal{I}$ der indizierte Kalkül von Beispiel 3.13. Der deterministische Kalkül $\mathit{det}(\mathcal{I}) = (D', D'_{\text{success}}, \models)$ von $\mathcal{I}$ beschreibt alle depth–first left–to–right tree traversals durch die Berechnungsbäume von $\mathcal{I}$ und beliebigen Zwischenergebnissen.

Betrachten wir beispielsweise den Berechnungsbaum von $\mathcal{I}$ und $(a * (a * (b+c)))$, so gilt für die Berechnung durch $\models$ beginnend mit $((a * (a * (b+c))), \{(\text{assoc}, \varepsilon), (\text{distr}, 2)\})$:

$$\begin{array}{ll} & ((a * (a * (b+c))), \{(\text{assoc}, \varepsilon), (\text{distr}, 2)\}) \\ \models & ((a * (a * (b+c))), \{(\text{distr}, 2)\}) \\ & (((a * a) * (b+c)), \{(2, \varepsilon)\}) \end{array}$$

$$
\begin{array}{ll}
\models & ((a * (a * (b+c))), \{(\mathrm{distr}, 2)\}) \\
& (((a * a) * (b+c)), \emptyset) \\
& ((((a * a) * b) + ((a * a) * c)), \emptyset) \\
\models & ((a * (a * (b+c))), \{(\mathrm{distr}, 2)\}) \\
& (((a * a) * (b+c)), \emptyset) \\
\models & ((a * (a * (b+c))), \{(\mathrm{distr}, 2)\}) \\
\models & ((a * (a * (b+c))), \emptyset) \\
& ((a * ((a * b) + (a * c))), \{(\mathrm{distr}, \varepsilon)\}) \\
\models & ((a * (a * (b+c))), \emptyset) \\
& ((a * ((a * b) + (a * c))), \emptyset) \\
& (((a * (a * b)) + (a * (a * c)), \{(\mathrm{assoc}, 1), (\mathrm{assoc}, 2)\}) \\
\models & ((a * (a * (b+c))), \emptyset) \\
& ((a * ((a * b) + (a * c))), \emptyset) \\
& (((a * (a * b)) + (a * (a * c)), \{(\mathrm{assoc}, 2)\}) \\
& ((((a * a) * b) + (a * (a * c))), \{(\mathrm{assoc}, 2)\}) \\
\models & ((a * (a * (b+c))), \emptyset) \\
& ((a * ((a * b) + (a * c))), \emptyset) \\
& (((a * (a * b)) + (a * (a * c)), \{(\mathrm{assoc}, 2)\}) \\
& ((((a * a) * b) + (a * (a * c))), \emptyset) \\
& ((((a * a) * b) + ((a * a) * c)), \emptyset) \\
\models & ((a * (a * (b+c))), \emptyset) \\
& ((a * ((a * b) + (a * c))), \emptyset) \\
& (((a * (a * b)) + (a * (a * c)), \{(\mathrm{assoc}, 2)\}) \\
& ((((a * a) * b) + (a * (a * c))), \emptyset) \\
\models & ((a * (a * (b+c))), \emptyset) \\
& ((a * ((a * b) + (a * c))), \emptyset) \\
& (((a * (a * b)) + (a * (a * c)), \{(\mathrm{assoc}, 2)\}) \\
\models & ((a * (a * (b+c))), \emptyset) \\
& ((a * ((a * b) + (a * c))), \emptyset) \\
& (((a * (a * b)) + (a * (a * c)), \emptyset) \\
& (((a * (a * b)) + ((a * a) * c)), \{(\mathrm{assoc}, 1)\}) \\
\models & ((a * (a * (b+c))), \emptyset) \\
& ((a * ((a * b) + (a * c))), \emptyset) \\
& (((a * (a * b)) + (a * (a * c)), \emptyset) \\
& (((a * (a * b)) + ((a * a) * c)), \emptyset) \\
& ((((a * a) * b) + ((a * a) * c)), \emptyset) \\
\models & ((a * (a * (b+c))), \emptyset) \\
& ((a * ((a * b) + (a * c))), \emptyset) \\
& (((a * (a * b)) + (a * (a * c)), \emptyset) \\
& (((a * (a * b)) + ((a * a) * c)), \emptyset)
\end{array}
$$

$$
\begin{array}{ll}
\models & ((a * (a * (b+c))), \emptyset) \\
 & ((a * ((a * b) + (a * c))), \emptyset) \\
 & (((a * (a * b)) + (a * (a * c)), \emptyset) \\
\models & ((a * (a * (b+c))), \emptyset) \\
 & ((a * ((a * b) + (a * c))), \emptyset) \\
\models & ((a * (a * (b+c))), \emptyset)
\end{array}
$$

$\square$

Das Beispiel zeigt, daß eine Berechnung durch $\models$ in der Tat jeden Knoten des Berechnungsbaumes des zugrundeliegenden indizierten Kalküls $\mathcal{I}$ und des Zwischenergebnisses d durchläuft.

Kapitel 4

Universelle Unifikationsalgorithmen

Unifikation (lat. Vereinheitlichung) bedeutet das „Gleichmachen“ von Termen. Sind $t, s \in T(\Omega, \mathcal{V})$ Terme, so suchen wir eine Substitution $\varphi \in \mathit{Sub}(\mathcal{V}, \Omega)$, so daß $\varphi(t) = \varphi(s)$. Die Substitution φ heißt dann Unifikator von t und s. Es ist bewiesen worden, daß das Unifikationsproblem für Terme erster Ordnung entscheidbar ist [Rob65]. Algorithmen, die für zwei Terme t und s einen Unifikator finden, heißen Unifikationsalgorithmen.

4.1 E–Unifikation

Bei der E–Unifikation wird das Unifikationsproblem verallgemeinert, indem nicht die gewöhnliche Gleichheit, sondern die Gleichheit modulo einer Menge E von Gleichungen, bezeichnet durch $=_E$, betrachtet wird. Die durch E induzierte Gleichheitsrelation wird auch Gleichungstheorie genannt.

Definition 4.1 (Menge von Gleichungen)

Sei Ω ein Rangalphabet. Eine *Menge von Gleichungen über* Ω *und* $\mathcal{V}$ ist eine Menge von Gleichungen der Form $t = s$ mit $t, s \in T(\Omega, \mathcal{V})$.

□

Definition 4.2 (E–Gleichheit)

Sei E eine endliche Menge von Gleichungen über Ω und $\mathcal{V}$. Die *E–Gleichheit*, bezeichnet durch $=_E$, ist die feinste Ω–Kongruenz über $T(\Omega, \mathcal{V})$, die jedes Paar $(\psi(t), \psi(s))$ enthält, wobei $t = s \in E$ und ψ eine beliebige $(\mathcal{V}, \Omega)$–Substitution ist. Wenn $t =_E s$, dann heißen t und s *E–gleich*.

□

E–Unifikation bedeutet also für zwei Terme t und s eine Substitution $\varphi \in \mathit{Sub}(\mathcal{V}, \Omega)$ zu finden, so daß $\varphi(t) =_E \varphi(s)$. Die Substitution φ heißt dann E–Unifikator von t und s.

Definition 4.3 (Menge von E–Unifikatoren)

Sei E eine endliche Menge von Gleichungen über Ω und $\mathcal{V}$.

- Zwei Terme $t, s \in T(\Omega, \mathcal{V})$ heißen *E–unifizierbar*, wenn es eine $(\mathcal{V}, \Omega)$–Substitution φ gibt, so daß $\varphi(t) =_E \varphi(s)$.
- Die Menge $\{\varphi \mid \varphi(t) =_E \varphi(s)\}$ heißt *Menge von E–Unifikatoren von t und s*; sie wird bezeichnet durch $\mathcal{U}_E(t, s)$.

□

Betrachten wir dazu ein einfaches Beispiel, welches die übliche Konkatenation von Listen beschreibt.

Beispiel 4.4 Sei E_1 die Menge der folgenden Gleichungen über $T(\{x, y, z\}, \{append^{(2)}, CONS^{(2)}, NIL^{(0)}, A^{(0)}, B^{(0)}\})$:

$$\begin{array}{rcll} append(NIL, x) & = & x & (1) \\ append(CONS(x, y), z) & = & CONS(x, append(y, z)) & (2) \end{array}$$

Dann ist die Substitution $\varphi = [x/CONS(A, CONS(B, NIL)), y/NIL]$ ein E_1–Unifikator der Terme $t = append(x, y)$ und $s = CONS(A, CONS(B, NIL))$, weil $\varphi(t) =_{E_1} \varphi(s)$.

□

Die Entscheidbarkeit dieses Problems hängt natürlich von der Menge E von Gleichungen ab. Wenn E die leere Menge ist, haben wir es mit dem üblichen Unifikationsproblem zu tun, welches entscheidbar ist. Sei aber E zum Beispiel die Menge der Peano Axiome, dann ist das E–Unifikationsproblem genau Hilbert's zehntes Problem, welches unentscheidbar ist [Mat70].

Kapitel 5 führt vier verschiedene Kalküle ein, die für die Berechnung von E–Unifikatoren geeignet sind. In der folgenden Auflistung steht hinter dem Namen des betreffenden Kalküls die Klasse von Termersetzungssystemen, die zur Beschreibung der Gleichungstheorie benutzt wird:

- Reduktionskalkül für $E_{\mathcal{R}}$–Unifikation (kanonische Termersetzungssysteme), siehe Abschnitt 5.1.
- Narrowing Kalkül $E_{\mathcal{R}}$–Unifikation (kanonische Termersetzungssysteme), siehe Abschnitt 5.2.
- Leftmost outermost Narrowing Kalkül $E_{\mathcal{R}}$–Unifikation (canonical, totally defined, not strictly sub–unifiable term rewriting systems, kurz: ctn–trs), siehe Abschnitt 5.3.
- Unification driven leftmost outermost Narrowing Kalkül $E_{\mathcal{R}}$–Unifikation (ctn–trs), siehe Abschnitt 5.4.

Die aus diesen Kalkülen resultierenden Algorithmen zum Finden von E–Unifikatoren sind vollständig. Wir müssen jetzt definieren, was eine vollständige Menge von E–Unifikatoren ist. Für die übliche Unifikation, d.h. für $E = \emptyset$ wissen wir aus [Rob65], daß das Unifikationsproblem entscheidbar ist und daß es genau einen allgemeinsten Unifikator gibt. Wenn E nicht leer ist, dann muß es keinen einzelnen allgemeinsten Unifikator geben sondern es kann mehrere „allgemeine"geben. Wie stehen also die einzelnen Elemente von $\mathcal{U}_E$ in Beziehung zueinander? Wir definieren dazu die Instanziierungsordnung.

Definition 4.5 (Instanziierungsordnung)

Sei E eine endliche Menge von Gleichungen über Ω und $\mathcal{V}$ und $t, s \in T(\Omega, \mathcal{V})$ Terme. Die *Instanziierungsordnung für Terme*, bezeichnet durch $\preceq_E$, ist definiert als:

$$t \preceq_E s, \text{ falls es eine } (\mathcal{V}, \Omega)\text{–Substitution } \varphi \text{ gibt, so daß } \varphi(t) =_E s.$$

Sei $V \subseteq \mathcal{V}$ eine endliche Menge und $\varphi, \varphi' \in \mathit{Sub}(\mathcal{V}, \Omega)$. Die *Instanziierungsordnung* über $(\mathcal{V}, \Omega)$–Substitutionen, bezeichnet durch $\preceq_{E,V}$, ist definiert als:

$$\varphi \preceq_{E,V} \varphi', \text{ falls es eine } (\mathcal{V}, \Omega)\text{– Substitution } \psi \text{ gibt, so daß für alle } x \in V\text{:}$$
$$\psi(\varphi(x)) =_E \varphi'(x).$$

□

Mit Hilfe der Instanziierungsordnung können wir also die verschiedenen E–Unifikatoren vergleichen. Im Falle der Instanziierungsordnung für Terme und $E = \emptyset$ ist das minimale Element bezüglich $\preceq_\emptyset$ genau der allgemeinste Unifikator (häufig als MGU = most general unifier bezeichnet). Im allgemeinen gibt es bei der E–Unifikation jetzt eine Menge von Elementen von $\mathcal{U}_E(t, s)$ für zwei Terme $t, s \in T(\Omega, \mathcal{V})$, die minimal bezüglich $\preceq_{E,V}$ sind. Wir definieren dazu zunächst die sogenannte vollständige Menge von E–Unifikatoren von t und s. Die Menge der minimalen E–Unifikatoren ist dann eine Untermenge dieser Menge, nämlich genau die Menge der minimalen Elemente der vollständigen Menge der E–Unifikatoren von t und s.

Definition 4.6 (vollständige Menge von E–Unifikatoren)

Sei E eine endliche Menge von Gleichungen über Ω und $\mathcal{V}$, $t, s \in T(\Omega, \mathcal{V})$ Terme und W eine endliche Menge von Variablen, in der $V = \mathcal{V}(t) \cup \mathcal{V}(s)$ enthalten ist, d.h. $V \subseteq W$. Die Menge S ist eine *vollständige Menge von E–Unifikatoren von t und s abseits von W*, wenn diese drei Bedingungen erfüllt sind:

1. Für alle $\varphi \in S$: $\mathcal{D}(\varphi) \subseteq V$ und $\mathcal{I}(\varphi) \cap W = \emptyset$.
2. $S \subseteq \mathcal{U}_E(t, s)$
3. Für alle $\varphi \in \mathcal{U}_E(t, s)$ gibt es ein $\psi \in S$, so daß $\psi \preceq_{E,V} \varphi$.

Die Menge S heißt *minimal*, wenn zusätzlich noch gilt:

4. Für alle $\varphi, \varphi' \in S$: Wenn $\varphi \neq \varphi'$, dann $\varphi \not\leq_{E,V} \varphi'$.

□

Definition 4.7 (Universeller Unifikationsalgorithmus)

Ein Algorithmus, der als Eingabe eine Menge E von Gleichungen und zwei Terme t und s nimmt und eine vollständige Menge von E–Unifikatoren aufzählt, heißt *Universeller Unifikationsalgorithmus* (kurz: UU–Algorithmus).

□

4.2 Algorithmen

In diesem Abschnitt werden wir verschiedene bekannte UU–Algorithmen aus der Literatur zitieren. Dazu geben wir zuerst einen naiven UU–Algorithmus für beliebige Mengen E an. Aus dem naiven UU–Algorithmus werden wir sukzessive immer bessere (d.h. „effizientere“) UU–Algorithmen ableiten. Dabei zitieren wir für jeden neuen Algorithmus zunächst den Satz, der die Vollständigkeit des Verfahrens garantiert; als zweites geben wir einen nichtdeterministischen UU–Algorithmus an; schließlich als drittes zeigen wir einen deterministischen Algorithmus, der wegen der Unentscheidbarkeit des E–Unifikationsproblems im allgemeinen nicht vollständig sein kann.

Der folgende Satz impliziert einen naiven UU–Algorithmus für eine beliebige Menge E von Gleichungen.

Satz 4.8 (vgl. Theorem 4 aus [Fay79]) Sei E eine endliche Menge von Gleichungen über Ω und $\mathcal{V}$, $t, s \in T(\Omega, \mathcal{V})$ Terme und W eine endliche Menge von Variablen, in der $V = \mathcal{V}(t) \cup \mathcal{V}(s)$ enthalten ist. Sei S die Menge der $(\mathcal{V}, \Omega)$–Substitutionen, so daß φ in S ist, wenn die folgenden beiden Bedingungen erfüllt sind:

1. $\varphi(t) =_E \varphi(s)$.
2. $\mathcal{D}(\varphi) \subseteq V$ und $\mathcal{I}(\varphi) \cap W = \emptyset$.

Dann ist S eine vollständige Menge von E–Unifikatoren von t und s abseits von W.

□

Der naive UU-Algorithmus in Abbildung 4.1 besteht aus einer Endlosschleife, in der alle Substitutionen $\varphi \in Sub(\mathcal{V}, \Omega)$ auf die beiden Eigenschaften von Satz 4.8 überprüft werden. Falls beide Bedingungen erfüllt sind, gehört φ zur vollständigen Menge von E–Unifikatoren von t und s.

Eingabe: Terme $t, s \in T(\Omega, \mathcal{V})$, Menge von Gleichungen E und Menge von Variablen W mit $W \supseteq V = (\mathcal{V}(t) \cup \mathcal{V}(s))$.

Ausgabe: Vollständige Menge von E–Unifikatoren von t und s abseits von W.

Verfahren:

1. Für alle $\varphi \in \mathit{Sub}(\mathcal{V}, \Omega)$:
   - Wenn $\varphi(t) =_E \varphi(s)$, $\mathcal{D}(\varphi) \subseteq V$ und $\mathcal{I}(\varphi) \cap W = \emptyset$, dann $\mathit{output}(\varphi)$;

Abbildung 4.1: Naiver UU-Algorithmus.

4.2.1 UU–Algorithmus für Reduktionskalkül für $E_{\mathcal{R}}$–Unifikation

Bei den nun folgenden UU–Algorithmen gehen wir immer von einem Termersetzungssystem $\mathcal{R}$ aus und betrachten die dadurch induzierte Gleichungstheorie $=_{E_{\mathcal{R}}}$. Was unter einem Termersetzungssystem (im Folgenden auch TES) zu verstehen ist, erklären wir hier nur kurz und informell; im Abschnitt 5.1 wird dann eine genaue Definition des Begriffs erfolgen. Ein Termersetzungssystem $\mathcal{R}$ besteht aus einer Menge R von Regeln der Form $l \to r$, wobei $l, r \in T(\Omega, \mathcal{V})$. Rechte Regelseiten dürfen keine Variablen enthalten, die nicht auch in der linken Regelseite der entsprechenden Regel auftauchen. Die Reduktionsrelation beschreibt die Anwendung einer Termersetzungsregel auf einen Term t. Dabei wird ein beliebiger Teilterm von t, der Instanz der linken Regelseite einer Regel in R ist, durch die entsprechende rechte Regelseite dieser Regel ersetzt. Tauschen wir die Rechtspfeile der Regeln in R durch Gleichheitszeichen aus, so erhalten wir eine Menge von Gleichungen. Diese charakterisiert die durch das Termersetzungssystem $\mathcal{R}$ induzierte Gleichungstheorie $=_{E_{\mathcal{R}}}$.

Die Klasse der kanonischen Termersetzungssysteme (im Folgenden auch TES) ist eine Teilklasse der Termersetzungssysteme. Damit ein TES kanonisch ist, muß es konfluent und noethersch sein, d.h. Berechnungen, die durch die eben beschriebenen Reduktionsrelation entstehen, müssen terminieren und „zusammenfließen" (lat. confluere). Letzteres bedeutet, daß verschiedene Berechnungen, die von einem Term ausgehen, immer wieder zu einem gemeinsamen Term führen. Diese Eigenschaften garantieren, daß es zu jedem Term eine sogenannte $\mathcal{R}$–Normalform gibt; i.e. einen Term, auf den die Reduktionsrelation nicht mehr angewendet werden kann. Ein Lemma besagt (siehe 5.8), daß zwei Terme t und s genau dann $=_{E_{\mathcal{R}}}$–gleich sind, wenn ihre $\mathcal{R}$–Normalformen identisch sind. (Diese Normalform eines Terms t wird durch $\mathit{nf}_{\mathcal{R}}\,(t)$ bezeichnet)

Im Falle von Gleichungstheorien, die durch kanonische Termersetzungssysteme induziert werden, reicht es deswegen aus, die $\mathcal{R}$–Normalformen von Termen zu vergleichen, um auf $E_{\mathcal{R}}$–Gleichheit zu testen. Der folgende Satz beschreibt diese Tatsache.

Satz 4.9 (vgl. Lemma 2.1 aus [Hue80]) Sei $\mathcal{R} = (\Omega, R)$ ein kanonisches Termersetzungssystem, $t, s \in T(\Omega, \mathcal{V})$ und W eine endliche Menge von Variablen, in der $V = \mathcal{V}(t) \cup \mathcal{V}(s)$ enthalten ist. Sei S die Menge aller $(\mathcal{V}, \Omega)$–Substitutionen, so daß $\varphi \in S$ genau dann, wenn folgende Bedingungen erfüllt sind:

1. $nf_{\mathcal{R}}\,(\varphi(t)) = nf_{\mathcal{R}}\,(\varphi(s))$.
2. $\mathcal{D}(\varphi) \subseteq V$ und $\mathcal{I}(\varphi) \cap W = \emptyset$.

Dann ist S eine vollständige Menge von $E_{\mathcal{R}}$–Unifikatoren von t und s abseits von W.

$\square$

Eingabe: Terme $t, s \in T(\Omega, \mathcal{V})$, kanonisches TES $\mathcal{R}$ und Menge von Variablen W mit $W \supseteq V = (\mathcal{V}(t) \cup \mathcal{V}(s))$.

Ausgabe: Vollständige Menge von $E_{\mathcal{R}}$–Unifikatoren von t und s abseits von W.

Verfahren:
Sei $\mathrm{red}_E\mathcal{I}\,(\mathcal{R}) = (D, D_{\mathrm{success}}, \Rightarrow_{\mathrm{equ}})$ der Reduktionskalkül für $E_{\mathcal{R}}$–Unifikation von $\mathcal{R}$.

1. Für alle $\varphi \in Sub\,(\mathcal{V}, \Omega)$, mit $\mathcal{D}(\varphi) \subseteq V$ und $\mathcal{I}(\varphi) \cap W = \emptyset$:
   - Wenn es eine Berechnung $equ\,(\varphi(t), \varphi(s)) \Rightarrow^*_{\mathcal{R}_{\mathrm{equ}}} d$ gibt, mit $d \in D_{\mathrm{success}}$, dann $output\,(\varphi)$;

Abbildung 4.2: UU-Algorithmus für Reduktionskalküle für $E_{\mathcal{R}}$–Unifikation.

Der UU–Algorithmus, der durch den obigen Satz impliziert wird, ist sehr ähnlich zu dem naiven UU–Algorithmus. Es werden in einer Endlosschleife wieder alle Substitutionen $\varphi \in Sub\,(\mathcal{V}, \Omega)$ durchlaufen. Allerdings kann als Eingabe nicht mehr eine beliebige Menge E von Gleichungen angenommen werden, sondern es muß ein kanonisches TES eingegeben werden; d.h. es sind nur solche Mengen von Gleichungen zugelassen, die durch kanonische TESe induziert werden. Wir gehen also von einem TES $\mathcal{R} = (\Omega, R)$ aus, und erweitern dieses zunächst zu einem equal–Termersetzungssystem (bezeichnet durch $\mathcal{R}_{\mathrm{equ}}$). Dies geschieht, indem das Rangalphabet Ω zu einem neuen Rangalphabet Ω' wird, mit $\Omega^{(2)'} = \Omega^{(2)} \cup \{equ\}$; die Regeln verändern sich nicht (siehe Definition 5.10). In Definition 5.11 wird der Reduktionskalkül für $E_{\mathcal{R}}$–Unifikation definiert, indem D_{success} als Menge aller $equ\,(\Theta, \Theta)$ gewählt wird, mit $\Theta \in T(\Omega, \mathcal{V})$. Zum Test auf $E_{\mathcal{R}}$–Gleichheit müssen wir die $\mathcal{R}$–Normalformen der beiden Terme $\varphi(t)$ und $\varphi(s)$ berechnen. Für $equ\,(\Theta, \Theta) \in D_{\mathrm{success}}$ sind aber die Terme Θ genau diese Normalformen, falls φ ein $E_{\mathcal{R}}$– Unifikator ist. Also müssen wir nur testen, ob sich aus $equ(\varphi(t), \varphi(s))$ ein $d \in \Omega \cup \{equ^{(2)}\}$ durch $\Rightarrow_{\text{equ}}$ berechnen läßt, mit $d \in D_{\text{success}}$.

Wir können natürlich auch den deterministischen Reduktionskalkül für $E_{\mathcal{R}}$–Unifikation $det(\text{red}\mathcal{I}_E(\mathcal{R})) = (D', D'_{\text{success}}, \models_{\text{red}})$ von $\mathcal{R}$ benutzen, um zu überprüfen, ob es ein Element in der D_{success} gibt. Dann würden wir in dem Rumpf der Endlosschleife nach einer Berechnung

$$(equ(\varphi(t), \varphi(s)), I(equ(\varphi(t), \varphi(s))) \models^{*}_{\text{red}} \beta(d, \emptyset)$$

suchen, wobei $d \in D_{\text{success}}$, und $\beta \in D' \cup \{\emptyset\}$. Dies ändert natürlich nichts an dem Nichtdeterminismus des Auswählens von Substitutionen φ in der äußeren Schleife, aber der Algorithmus basierend auf dem deterministischen Reduktionskalkül enthält einen Nichtdeterminismus weniger als der Algorithmus in Abbildung 4.2; wir haben ihn deshalb in Abbildung 4.3 angegeben.

Eingabe: Terme $t, s \in T(\Omega, \mathcal{V})$, kanonisches TES $\mathcal{R}$ und Menge von Variablen W mit $W \supseteq V = (\mathcal{V}(t) \cup \mathcal{V}(s))$.

Ausgabe: Vollständige Menge von $E_{\mathcal{R}}$–Unifikatoren von t und s abseits von W.

Verfahren:
Sei $det(red_E\mathcal{I}(\mathcal{R})) = (D', D'_{\text{success}}, \models_{\text{red}})$ der deterministische Reduktionskalkül für $E_{\mathcal{R}}$–Unifikation von $\mathcal{R}$.

1. Für alle $\varphi \in Sub(\mathcal{V}, \Omega)$ mit $\mathcal{D}(\varphi) \subseteq V$ und $\mathcal{I}(\varphi) \cap W = \emptyset$:
   (a) $\Theta = (equ(\varphi(t), \varphi(s)), I(equ(\varphi(t), \varphi(s))))$
   (b) Solange es Berechnungen $\Theta \models_{\text{red}} \Theta'$ gibt mit $\Theta \notin D'_{\text{success}}$:
   - $\Theta := \Theta'$

   (c) Wenn $\Theta \in D'_{\text{success}}$, dann $output(\varphi)$;

Abbildung 4.3: Algorithmus für deterministischen Reduktionskalkül für $E_{\mathcal{R}}$–Unifikation

4.2.2 UU–Algorithmus für Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation

Einen effizienteren UU–Algorithmus impliziert der Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation (siehe Abschnitt 5.2). Beim Narrowing wird bei jedem Berechnungsschritt die „minimale" Substitution angewendet, so daß sich eine Regel des gegebenen TES anwenden läßt. Wir merken uns in jedem Schritt die Komposition der bis dahin angewandten Substitutionen $\varphi_1, \varphi_2, \ldots, \varphi_n$.

Sucht man einen E–Unifikator der Terme t und s, so beginnt man mit dem Kalkulandum $(equ\,(t,s), \varphi_\emptyset)$ und wendet solange die Narrowingrelation an, bis ein Zwischenergebnis $(equ\,(t',s'), \varphi)$ entstanden ist in dem t' und s' unifizierbar sind. Die Komposition von φ und dem allgemeinsten Unifikator von t' und s' ist dann ein E–Unifikator von t und s. Folgender Satz gibt Auskunft über die Vollständigkeit des Narrowings.

Satz 4.10 (vgl. [Hul80]) Sei $\mathcal{R} = (\Omega, R)$ ein kanonisches Termersetzungssystem, $t, s \in T(\Omega, \mathcal{V})$ Terme, und sei V die Menge $\mathcal{V}(t) \cup \mathcal{V}(s)$. Die Menge S besteht aus allen $(\mathcal{V}, \Omega)$–Substitutionen φ, so daß $\varphi \in S$, genau dann wenn es eine Berechnung durch $\rightsquigarrow_{\mathcal{R}_{\mathrm{equ}}}$ gibt:

$$\begin{aligned}(equ\,(t,s), \varphi_\emptyset) &\rightsquigarrow_{\mathcal{R}_{\mathrm{equ}}} (equ\,(t_1,s_1), \varphi_1)\\ &\rightsquigarrow_{\mathcal{R}_{\mathrm{equ}}} (equ\,(t_2,s_2), \varphi_2) \rightsquigarrow_{\mathcal{R}_{\mathrm{equ}}} \ldots \rightsquigarrow_{\mathcal{R}_{\mathrm{equ}}} (equ\,(t_n,s_n), \varphi_n),\end{aligned}$$

wobei für alle $i \in [n] : \varphi_i$ in Normalform ist, t_n und s_n sind unifizierbar mit allgemeinstem Unifikator σ, und $\varphi = (\sigma \circ \varphi_n)|_V$. Die Menge S ist dann eine vollständige Menge von $E_{\mathcal{R}}$–Unifikatoren von t und s abseits von V.

$\square$

Der UU–Algorithmus, der sich aus diesem Satz ergibt, muß jetzt nicht mehr alle Substitutionen durchlaufen, denn die Substitutionen ergeben sich aus den Narrowingschritten. Es muß lediglich für jede Berechnung durch die Narrowingrelation überprüft werden, ob die entstandenen Terme unifizierbar sind.

Natürlich wird es im allgemeinen unendlich viele Berechnungen durch $\rightsquigarrow_{\mathcal{R}_{\mathrm{equ}}}$ beginnend mit $equ\,(t,s)$ geben, so daß dieser Algorithmus auch eine Endlosschleife enthält. Außerdem schreibt der Algorithmus keine Reihenfolge der Narrowingberechnungen vor. Wir können aber den deterministischen Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation von $\mathcal{R}_{\mathrm{equ}}$ benutzen, der einen left–to–right tree traversal durch den Narrowingbaum von $(equ\,(t,s), \varphi_\emptyset)$ beschreibt (siehe Definition 5.19). Der obige Algorithmus wird dadurch verfeinert, daß die Schleife nicht mehr über alle Narrowingberechnungen läuft, sondern iterativ über Schritte der Rechenrelation $\models_{\mathrm{nar}}$ des deterministischen Narrowingkalküls. Als Kalkulandum dient das Zwischenergebnis $((equ\,(t,s), \varphi_\emptyset), I((equ\,(t,s), \varphi_\emptyset)))$. Der Algorithmus überprüft bei jedem neuen Zwischenergebnis, ob die entstandenen Terme t' und s' unifizierbar sind; da beim Backtracking Terme auftreten, die schon in vorherigen Zwischenergebnissen vorhanden waren, wird dort der Test auf Unifizierbarkeit mehrfach durchgeführt. Diese Ineffizienz läßt sich aber leicht durch entsprechendes Anpassen des Tests beheben.

Der UU-Algorithmus sieht dann wie folgt aus:

<u>Eingabe:</u> Terme $t, s \in T(\Omega, \mathcal{V})$, kanonisches TES $\mathcal{R}$.

<u>Ausgabe:</u> Vollständige Menge von $E_{\mathcal{R}}$–Unifikatoren von t und s abseits von $V = \mathcal{V}(t) \cup \mathcal{V}(s)$.

<u>Verfahren:</u>
Sei $\mathrm{nar}_E\mathcal{I}\,(\mathcal{R}) = (D, D_{\text{success}}, \rightsquigarrow_{\text{equ}})$ der Narrowingkalkül für $E_{\mathcal{R}}$–Unifikation von $\mathcal{R}$.

1. Für alle Berechnungen
   $(equ\,(t,s), \varphi_\emptyset) \rightsquigarrow_{\mathcal{R}_{\text{equ}}} (equ\,(t_1, s_1), \varphi_1) \rightsquigarrow_{\mathcal{R}_{\text{equ}}} (equ\,(t_2, s_2), \varphi_2) \rightsquigarrow_{\mathcal{R}_{\text{equ}}} \ldots \rightsquigarrow_{\mathcal{R}_{\text{equ}}} (equ\,(t_n, s_n), \varphi_n)$,
   wobei für alle $i \in [n] : \varphi_i$ in Normalform und $n \geq 0$:

   Wenn $(equ\,(t_n, s_n), \varphi_n) \in D_{\text{success}}$, und σ der allgemeinste Unifikator von t_n und s_n, dann
   $output\,((\sigma \circ \varphi_n)|_V)$;

Abbildung 4.4: UU-Algorithmus für Narrowingkalkül für $E_{\mathcal{R}}$–Unifikation

Die Abfrage in Punkt 3. des Algorithmus ist nötig, weil im Falle eines Narrowingbaums mit nur einem Knoten sonst kein Unifikator gefunden würde. Der Schleifenrumpf würde nämlich nicht abgearbeitet werden, weil es keine Berechnung für einen Baum mit einem Knoten gibt. Falls der Baum mehr als einen Knoten besitzt, wird der equ–Term an der Wurzel zweimal daraufhin überprüft, ob er in der Menge D_{success} ist.

4.2.3 UU–Algorithmus für leftmost outermost Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation

Beim leftmost outermost Narrowing Kalkül wählt man nur noch ganz bestimmte Narrowing Vorkommen aus, an denen die Narrowing Relation angewendet wird. Um unnötige Funktionsaufrufe zu vermeiden, wählt man bei jedem Narrowingschritt stets das äußerste und am weitesten links stehende Narrowingvorkommen in einem Term aus. Dies entspricht der lazy–evaluation (oder auch call–by–need) Auswertungstechnik bei funktionalen Sprachen. Im Falle der $E_{\mathcal{R}}$–Unifikation muß allerdings eine Einschränkung an das Termersetzungssystem $\mathcal{R}$ gemacht werden, damit diese Vorkommensauswahlstrategie eine vollständige Menge von $E_{\mathcal{R}}$–Unifikatoren liefert. Die Einschränkung heißt totally defined, not strictly subunifiable, und die entsprechenden TESe heißen canonical, totally–defined, not strictly subunifiable TESe, kurz ctn–trs (siehe Definition 5.28). Ctn–trs's sind bestimmte konstruktorbasierte TESe, d.h. das Rangalphabet Ω ist in eine Menge F von Funktionssymbolen und eine Menge Δ von Konstruktorsymbolen unterteilt. Der Wertebereich der Unifikatoren, die wir berechnen wollen, soll verständlicherweise keine Funktionssymbole enthalten.

Eingabe: Terme $t, s \in T(\Omega, \mathcal{V})$, kanonisches TES $\mathcal{R}$.

Ausgabe: Vollständige Menge von $E_{\mathcal{R}}$–Unifikatoren von t und s abseits von $V = \mathcal{V}(t) \cup \mathcal{V}(s)$.

Verfahren:
Sei $det(nar_E\mathcal{I}(\mathcal{R})) = (D, D_{\text{success}}, \models_{\text{nar}})$ der deterministische Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation von $\mathcal{R}$.

1. $\Theta = ((equ(t,s), \varphi_\emptyset), I((equ(t,s), \varphi_\emptyset)))$;
2. Wenn $(equ(t,s), \varphi_\emptyset) \in D_{\text{success}}$, und σ der allgemeinste Unifikator von t und s, dann $S := S \cup \{\sigma|_V\}$;
3. Solange es Berechnung $\underbrace{\beta((equ(t', s'), \varphi), M)}_{\Theta} \models_{\text{nar}} \Theta'$gibt, mit $\beta \in D \cup \{\varepsilon\}, t', s' \in T(\Omega, \mathcal{V}), \varphi \in Sub(\mathcal{V}, \Omega), M \subseteq I((equ(t', s'), \varphi))$ und φ in Normalform:
   (a) Wenn $M \neq \emptyset$ dann:
      - Wenn $(equ(t,s), \varphi) \in D_{\text{success}}$, und σ der allgemeinste Unifikator von t und s, dann $output((\sigma \circ \varphi)|_V)$;

   (b) $\Theta := \Theta'$

Abbildung 4.5: Algorithmus für deterministischen Narrowingkalkül für $E_{\mathcal{R}}$–Unifikation

Definition 4.11 ($(E_{\mathcal{R}}, \Delta)$–Unifikatoren)
Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs, $t, s \in T(F \cup \Delta, \mathcal{V})$, und sei $\varphi \in \mathcal{U}_{E_{\mathcal{R}}}$ ein $E_{\mathcal{R}}$–Unifikator von t und s.

- φ ist ein *$(E_{\mathcal{R}}, \Delta)$–Unifikator von t und s*, wenn $\varphi \in Sub(\mathcal{V}, \Delta)$.
- φ ist ein *grund $(E_{\mathcal{R}}, \Delta)$–Unifikator von t und s*, wenn $\varphi \in gSub(\mathcal{V}, \Delta)$.

Die Mengen der $(E_{\mathcal{R}}, \Delta)$–Unifikatoren und der grund $(E_{\mathcal{R}}, \Delta)$–Unifikatoren von t und s werde durch $\mathcal{U}_{(E_{\mathcal{R}}, \Delta)}(t, s)$ und durch $g\mathcal{U}_{(E_{\mathcal{R}}, \Delta)}(t, s)$ bezeichnet.

□

Ähnlich wie bei der Menge der E–Unifikatoren von zwei Termen t und s, müssen wir nicht die gesamte Menge $g\mathcal{U}_{(E_{\mathcal{R}}, \Delta)}(t, s)$ berechnen, sondern nur eine Approximation dieser Menge. Es genügt die Menge der grund–vollständigen $(E_{\mathcal{R}}, \Delta)$–Unifikatoren zu berechnen.

Definition 4.12 (grund–vollständige Menge von $(E_{\mathcal{R}}, \Delta)$–Unifikatoren)

Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs, $t, s \in T(F \cup \Delta, \mathcal{V})$, und sei W eine endliche Menge von Variablen, die die Menge $V = \mathcal{V}(t) \cup \mathcal{V}(s)$ enthält. Eine Menge S von $(\mathcal{V}, \Delta)$–Substitutionen ist eine *grund–vollständige Menge von* $(E_{\mathcal{R}}, \Delta)$*–Unifikatoren von* t *und* s *abseits von* W, wenn folgende drei Bedingungen erfüllt sind:

1. Für alle $\varphi \in S : \mathcal{D}(\varphi) \subseteq V$ und $\mathcal{I}(\varphi) \cap W = \emptyset$.
2. $S \subseteq \mathcal{U}_{(E_{\mathcal{R}}, \Delta)}(t, s)$.
3. Für alle $\varphi \in g\mathcal{U}_{(E_{\mathcal{R}}, \Delta)}(t, s)$ gibt es ein $\psi \in S$, so daß $\psi \preceq_{E_{\mathcal{R}}, V} \varphi$.

□

Der folgende Satz bezüglich der Vollständigkeit von leftmost outermost Narrowing stammt aus [Ech88].

Satz 4.13 (vgl. Theorem 3 aus [Ech88]) Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs, $t, s \in T(F \cup \Delta, \mathcal{V})$, und sei V die Menge $\mathcal{V}(t) \cup \mathcal{V}(s)$. Die Menge S besteht aus allen $(\mathcal{V}, \Delta)$– Substitutionen φ, so daß $\varphi \in S$, genau dann, wenn es eine Berechnung durch $\overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\mathrm{equ}}}$ gibt:

$$\begin{aligned}(equ(t, s), \varphi_{\emptyset}) &\overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\mathrm{equ}}} (equ(t_1, s_1), \varphi_1)\\ &\overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\mathrm{equ}}} (equ(t_2, s_2), \varphi_2) \overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\mathrm{equ}}} \ldots \overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\mathrm{equ}}} (equ(t_n, s_n), \varphi_n),\end{aligned}$$

wobei für alle $i \in [n] : \varphi_i$ in Normalform ist, t_n und s_n sind unifizierbar mit allgemeinstem Unifikator σ, und $\varphi = (\sigma \circ \varphi_n)|_V$. Die Menge S ist dann eine grund–vollständige Menge von $E_{\mathcal{R}}$–Unifikatoren von t und s abseits von V.

□

Der UU–Algorithmus, der sich aus dem leftmost outermost Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation ergibt, ist fast identisch zu dem UU– Algorithmus beim Narrowing. Es muß lediglich die Rechenrelation in der Schleife durch die entsprechende Relation des deterministischen leftmost outermost Narrowing Kalküls für $E_{\mathcal{R}}$–Unifikation ersetzt werden. Außerdem dürfen nur ctn–trs als Eingabe akzeptiert werden, und die Ausgabe ist eine grund–vollständige Menge von $(E_{\mathcal{R}}, \Delta)$–Unifikatoren von t und s abseits von V (siehe Definition 4.12).

Entsprechend ähnlich ist auch der UU–Algorithmus für den deterministischen leftmost outermost Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation, der in Abbildung 4.7 zu sehen ist.

4.2.4 UU–Algorithmus für unification driven leftmost outermost Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation

Als letzten Kalkül betrachten wir nun den unification driven leftmost outermost (kurz: ulo–) Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation. Ein ctn-trs $\mathcal{R}$ wird dabei zunächst um den

<u>Eingabe:</u> Terme $t, s \in T(\Omega, \mathcal{V})$, ctn–trs $\mathcal{R}$.

<u>Ausgabe:</u> Grund–vollständige Menge von $E_{\mathcal{R}}$–Unifikatoren von t und s abseits von $V = \mathcal{V}(t) \cup \mathcal{V}(s)$.

<u>Verfahren:</u>

Sei lo-nar$_E\mathcal{I}(\mathcal{R}) = (D, D_{\text{success}}, \overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\text{equ}}})$ der leftmost outermost Narrowingkalkül für $E_{\mathcal{R}}$–Unifikation von $\mathcal{R}$.

1. Für alle Berechnungen
   $(equ(t, s), \varphi_\emptyset) \overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\text{equ}}} (equ(t_1, s_1), \varphi_1) \overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\text{equ}}}$
   $(equ(t_2, s_2), \varphi_2) \overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\text{equ}}} \ldots \overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\text{equ}}} (equ(t_n, s_n), \varphi_n)$,
   wobei für alle $i \in [n] : \varphi_i$ in Normalform und $n \geq 0$:

   Wenn $(equ(t_n, s_n), \varphi_n) \in D_{\text{success}}$, und σ der allgemeinste Unifikator von t_n und s_n, dann
   $output((\sigma \circ \varphi_n)|_V)$;

Abbildung 4.6: UU-Algorithmus für leftmost outermost Narrowingkalkül für $E_{\mathcal{R}}$–Unifikation

decomposition–part von $\mathcal{R}$ erweitert; d.h. um Regeln, die es ermöglichen das *equ* Symbol in einem Narrowingbaum immer weiter nach unten sinken zu lassen. Das so entstehende TES wird durch $\hat{\mathcal{R}}$ bezeichnet und heißt Erweiterung von $\mathcal{R}$. Die Dekompositionsregeln werden nun in einer Berechnung immer so früh wie möglich angewendet. Ob eine Dekompositionsregel angewendet werden kann, entscheidet außerdem ein bestimmter occur check. Der folgende Satz aus [FV94] besagt, daß durch die ulo–Narrowing Relation grund-vollständige Mengen von (E_R, Δ)–Unifikatoren berechnet werden können.

Satz 4.14 Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs, $t, s \in T(F \cup \Delta, \mathcal{V})$ Termen, und sei $V = \mathcal{V}(t) \cup \mathcal{V}(s)$. Die Menge S besteht aus allen $(\mathcal{V}, \Delta)$– Substitutionen φ, so daß $\varphi \in S$, genau dann, wenn es eine Berechnung durch $\overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}s}$ gibt:

$$(equ(t, s), \varphi_\emptyset) \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}} (t_1, \varphi_1) \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}} (t_2, \varphi_2) \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}} \ldots \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}} (t_n, \varphi_n),$$

wobei für alle $i \in [n] : \varphi_i$ in Normalform, $t_n \in T(\Delta, \mathcal{V})$ und $\varphi = \varphi_n|_V$. Dann ist S eine grund-vollständige Menge von $(E_{\mathcal{R}}, \Delta)$–Unifikatoren von t und s abseits von V.

□

Der Algorithmus zum ulo–Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation ist in Abbildung 4.8 zu sehen. Er geht von der ulo–Narrowing Relation aus, und verfährt analog zu den vorhergehenden nicht–deterministischen Algorithmen.

Wir wollen nun den UU–Algorithmus betrachten, der sich aus diesem Satz und dem deterministischen ulo–Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation ergibt. Die Rechenrelation

Eingabe: Terme $t, s \in T(\Omega, \mathcal{V})$, ctn–trs $\mathcal{R}$.

Ausgabe: Vollständige Menge von $E_{\mathcal{R}}$–Unifikatoren von t und s abseits von $V = \mathcal{V}(t) \cup \mathcal{V}(s)$.

Verfahren:
Sei $det(lo - nar_E\mathcal{I}(\mathcal{R})) = (D, D_{\text{success}}, \models_{\text{lo-nar}})$ der deterministische leftmost outermost Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation von $\mathcal{R}$.

1. $\Theta = ((equ(t,s), \varphi_{\emptyset}), I((equ(t,s), \varphi_{\emptyset})))$;
2. Wenn $(equ(t,s), \varphi_{\emptyset}) \in D_{\text{success}}$ und σ der allgemeinste Unifikator von t und s, dann $S := S \cup \{\sigma|_V\}$;
3. Solange es Berechnung $\underbrace{\beta((equ(t',s'), \varphi), M)}_{\Theta} \models_{\text{lo-nar}} \Theta'$gibt, mit $\beta \in D \cup \{\varepsilon\}, t', s' \in T(\Omega, \mathcal{V}), \varphi \in Sub(\mathcal{V}, \Omega), M \subseteq I((equ(t', s'), \varphi))$ und φ in Normalform:
   (a) Wenn $M \neq \emptyset$ dann:
      - Wenn $(equ(t,s), \varphi) \in D_{\text{success}}$ und σ der allgemeinste Unifikator von t und s, dann $output((\sigma \circ \varphi)|_V)$;

   (b) $\Theta := \Theta'$

Abbildung 4.7: Algorithmus für deterministischen lo–Narrowingkalkül für $E_{\mathcal{R}}$–Unifikation

des deterministischen ulo–Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation sei dabei durch $\models_{\text{ulo–nar}}$ bezeichnet. Wir gehen dabei von dem Zwischenergebnis $((equ(t,s), \varphi_{\emptyset}), I((equ(t,s), \varphi)))$ aus und wenden iterativ die Relation $\models_{\text{ulo–nar}}$ an. Wenn ein Zwischenergebnis $((t', \varphi), M)$ auftritt, wobei t' aus der Menge $T(\Delta, \mathcal{V})$ sind, dann ist $\varphi|_V$ ein grund $(E_{\mathcal{R}}, \Delta)$–Unifikator von t und s.

<u>Eingabe:</u> Terme $t, s \in T(\Omega, \mathcal{V})$, ctn–trs $\mathcal{R}$.

<u>Ausgabe:</u> Grund–vollständige Menge von $E_{\mathcal{R}}$–Unifikatoren von t und s abseits von $V = \mathcal{V}(t) \cup \mathcal{V}(s)$.

<u>Verfahren:</u>
Sei ulo-nar$\mathcal{I}(\mathcal{R}) = (D, D_{\text{success}}, \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}})$ der unification driven leftmost outermost Narrowingkalkül für $E_{\mathcal{R}}$–Unifikation von $\mathcal{R}$.

1. Für alle Berechnungen
   $(equ(t,s), \varphi_\emptyset) \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}} (t_1, \varphi_1) \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}} \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}} \dots \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}} (t_n, \varphi_n)$
   wobei für alle $i \in [n] : \varphi_i$ in Normalform und $n \geq 0$:

   Wenn $(equ(t_n, s_n), \varphi_n) \in D_{\text{success}}$, dann
   $output(\varphi_n|_V)$;

Abbildung 4.8: UU-Algorithmus für ulo–Narrowingkalkül für $E_{\mathcal{R}}$–Unifikation

<u>Eingabe:</u> Terme $t, s \in T(\Omega, \mathcal{V})$, ctn–trs $\mathcal{R}$.

<u>Ausgabe:</u> Vollständige Menge von $E_{\mathcal{R}}$–Unifikatoren von t und s abseits von $V = \mathcal{V}(t) \cup \mathcal{V}(s)$.

<u>Verfahren:</u>
Sei $det(ulo - nar_E\mathcal{I}(\mathcal{R})) = (D, D_{\text{success}}, \models_{\text{ulo-nar}})$ der deterministische unification driven leftmost outermost Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation von $\mathcal{R}$.

1. $\Theta = ((equ(t,s), \varphi_\emptyset), I((equ(t,s), \varphi_\emptyset)))$;
2. Wenn $(equ(t,s), \varphi_\emptyset) \in D'_{\text{success}}$ und σ der allgemeinste Unifikator von t und s, dann $S := S \cup \{\sigma|_V\}$;
3. Solange es Berechnung $\underbrace{\beta((equ(t', s'), \varphi), M)}_{\Theta} \models_{\text{lo-nar}} \Theta'$gibt, mit $\beta \in D \cup \{\varepsilon\}, t', s' \in T(\Omega, \mathcal{V}), \varphi \in Sub(\mathcal{V}, \Omega), M \subseteq I((equ(t', s'), \varphi))$ und φ in Normalform:
   (a) Wenn $M \neq \emptyset$ dann:
      - Wenn $(equ(t,s), \varphi) \in D_{\text{success}}$, dann $output(\varphi|_V)$;

   (b) $\Theta := \Theta'$

Abbildung 4.9: Algorithmus für deterministischen ulo–Narrowingkalkül für $E_{\mathcal{R}}$–Unifikation

Kapitel 5

Formale Kalküle für E–Unifikation

In diesem Kapitel wollen wir verschiedene Instanzen der in Kapitel 3 formal definierten Kalküle und die eventuell dazugehörigen Berechnungsbäume untersuchen. Es soll dabei die Anwendung einiger Kalküle für E–Unifikation im Vordergrund stehen, d.h. Unifikation, die durch eine Menge E von Gleichungen induziert wird. Wir wollen uns dabei auf die Klasse von Gleichungstheorien beschränken, die durch kanonische Termersetzungssysteme induziert werden. Zur Erinnerung wiederholen wir, daß für ein Termersetzungssystem $\mathcal{R}$ die Menge der Gleichungen, die durch Ersetzen jeder Regel $l \to r$ durch die Gleichung $l = r$ entsteht, mit $E_{\mathcal{R}}$ bezeichnet wird. Es gilt, daß $=_{E_{\mathcal{R}}}$ gleich der transitiven, reflexiven, symmetrischen Hülle $\Leftrightarrow^*_{\mathcal{R}}$ der Reduktionsrelation $\Rightarrow_{\mathcal{R}}$, die zu $\mathcal{R}$ gehört, ist.

Wir werden als erstes den Reduktionskalkül untersuchen, der ein indizierter Kalkül für den Kalkül $(T(\Omega, \mathcal{V}), \Rightarrow_{\mathcal{R}})$ ist; i.e. den Kalkül mit Termen über Ω indiziert durch $\mathcal{V}$ als Zwischenergebnissen und der Reduktionsrelation $\Rightarrow_{\mathcal{R}}$ als Rechenrelation. Nachdem wir dann den Narrowing Kalkül eingeführt haben, werden wir zwei Kalküle betrachten, i.e. den leftmost outermost Narrowing Kalkül und den unification driven leftmost outermost Narrowing Kalkül, die nur noch für Gleichungstheorien, die durch ganz bestimmte Termersetzungssysteme induziert werden, gelten. Dies sind die sogenannten ctn–trs. Zu den ersten drei Kalkülen, i.e. Reduktionskalkül, Narrowing Kalkül und lo–Narrowing Kalkül gibt jeweils zwei Abschnitte. Der erste behandelt den jeweiligen Kalkül mit allgemeinen Termen als Zwischenergebnissen, der zweite behandelt eine spezielle Instanz des Kalküls aus dem jeweils ersten Abschnitt, die für $E_{\mathcal{R}}$–Unifikation geeignet ist. Für den unification driven lo–Narrowing Kalkül gibt es nur einen Abschnitt, weil er durch seine Definition ausschließlich für $E_{\mathcal{R}}$–Unifikation geeignet ist.

5.1 Reduktionskalkül

Wir werden nun den Begriff des Termersetzungssystems definieren.

Definition 5.1 (Termersetzungssystem)

Ein *Termersetzungssystem*, bezeichnet durch $\mathcal{R}$, ist ein Paar (Ω, R), wobei Ω ein Rangalphabet ist und R eine endliche Menge von Regeln der Form $l \to r$, mit $r, l \in T(\Omega, \mathcal{V})$ und $\mathcal{V}(r) \subseteq \mathcal{V}(l)$.

□

Zu jedem Termersetzungssystem $\mathcal{R} = (\Omega, R)$ kann man eine Bijektion $\pi : R \to [card(R)]$ angeben, die eine Aufzählung der Regeln von R beschreibt. Wir werden üblicherweise als π einfach die Reihenfolge, in der wir die Regeln notieren, benutzen, so wie im folgenden Beispiel.

Beispiel 5.2 Sei $\mathcal{R}_1 = (\Omega, R)$ ein Termersetzungssystem mit:

- $\Omega = \{append^{(2)}, CONS^{(2)}, NIL^{(0)}, A^{(0)}, B^{(0)}, C^{(0)}\}$ ein Rangalphabet.
- R die Menge der folgenden Regeln:

$$\begin{array}{rcll} append(NIL, x) & \to & x & (1) \\ append(CONS(x, y), z) & \to & CONS(x, append(y, z)) & (2) \end{array}$$

Die Ziffern rechts neben den Regeln geben die Aufzählung von R an, d.h. es gilt beispielsweise: $\pi\,(append(NIL, x) \to x) = 1$.

□

Wir wollen zunächst die Reduktionsrelation zu einem Termersetzungssystem $\mathcal{R}$ definieren. Dies ist eine Relation, die aus einem Term $t \in T(\Omega, \mathcal{V})$ unter Anwendung einer Regel des Termersetzungssystems $\mathcal{R}$ einen Term $t' \in T(\Omega, \mathcal{V})$ berechnet. Die Anwendung einer Regel ist dann möglich, wenn es in t einen Teilterm s gibt, der Instanz der linken Regelseite l einer Regel $l \to r$ ist. Der Term t' entsteht dann durch Ersetzen von s durch die entsprechende Instanz von r.

Abbildung 5.1 soll einen Berechnungsschritt von $\Rightarrow_{\mathcal{R}}$ veranschaulichen. Dabei ist u ein Vorkommen von t und l' und r' sind die Instanzen der linken beziehungsweise der rechten Seite der Regel $l \to r$. Also $l' = t/u = \varphi(l)$ und $r' = \varphi(r)$.

Definition 5.3 (Reduktionsrelation)

Sei $\mathcal{R}$ ein Termersetzungssystem. Die *Reduktionsrelation zu* $\mathcal{R}$, bezeichnet durch $\Rightarrow_{\mathcal{R}}$, ist die kleinste binäre Relation über $T(\Omega, \mathcal{V})$, so daß für alle $t, s \in T(\Omega, \mathcal{V})$ gilt:

1. Es gibt ein Vorkommen $u \in O(t)$ mit $t/u \notin \mathcal{V}$ und
2. es gibt $\varphi \in Sub(\mathcal{V}, \Omega)$ und $l \to r \in R$ mit $\varphi(l) = t/u$ und $s = t\,[u \leftarrow \varphi(r)]$.

□

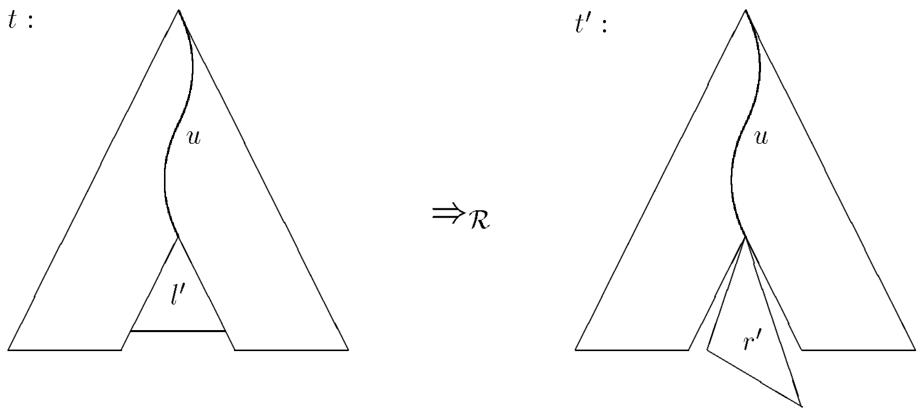

Abbildung 5.1: Beispiel für einen Berechnungsschritt von $\Rightarrow_{\mathcal{R}}$.

Wenn es klar ist, welches $\mathcal{R}$ wir in einem Kontext meinen, schreiben wir auch nur $\Rightarrow$ anstatt $\Rightarrow_{\mathcal{R}}$. Für einen bestimmten Berechnungsschritt $t \Rightarrow_{\mathcal{R}} t'$ notieren wir machmal auch das Vorkommen u von t, die Substitution φ und die Regel ξ, die für diesen Berechnungsschritt relevant sind, als Indizes von $\Rightarrow_{\mathcal{R}}$, also in der Form: $t \Rightarrow_{\mathcal{R},u,\varphi,\xi} t'$.

Beispiel 5.4 Sei $\mathcal{R}_1$ das Termersetzungssystem aus Beispiel 5.2 und sei t der folgende Term:

$$append(CONS(A, CONS(B, NIL)), CONS(C, NIL))$$

Eine Berechnung durch $\Rightarrow_{\mathcal{R}_1}$ beginnend mit t ist folgende:

$$\begin{array}{ll}
 & append(CONS(A, CONS(B, NIL)), CONS(C, NIL)) \\
\Rightarrow_{\mathcal{R}_1,\varepsilon,(2)} & CONS(A, append(CONS(B, NIL), CONS(C, NIL)) \\
\Rightarrow_{\mathcal{R}_1,2,(2)} & CONS(A, CONS(B, append(NIL, CONS(C, NIL)) \\
\Rightarrow_{\mathcal{R}_1,22,(1)} & cCONS(A, CONS(B, CONS(C, NIL)))
\end{array}$$

□

Sei $\mathcal{R} = (\Omega, R)$ ein beliebiges Termersetzungssystem und sei $\mathcal{C}_{\mathcal{R}} = (T(\Omega, \mathcal{V}), D_{\text{success}}, \Rightarrow_{\mathcal{R}})$ ein Kalkül. Wir wollen nun einen indizierten Kalkül für $\mathcal{C}_{\mathcal{R}}$ angeben. Dazu müssen wir zunächst eine geeignete Indexmenge J finden. Wir stellen fest, daß es bei der Berechnung eines Termes durch $\Rightarrow_{\mathcal{R}}$ folgende Nichtdeterminismen gibt:

- Die Wahl eines Vorkommens u von t.
- Wenn ein Vorkommen vorgegeben ist, die Wahl einer Regel in R, die dort angewendet wird.

Die Indexmenge J und die totale Ordnung $<$, die auf der Indexmenge definiert wird, sollen eine eindeutige Darstellung der Menge aller möglichen Berechnungen von Termen geben. Deshalb müssen für die Wahl dieser beiden Elemente die oben genannten Nichtdeterminismen beachtet werden. Es ist daher sinnvoll, als Indexmenge die Menge $R \times \mathbb{N}^*$ zu wählen, und eine geeignete Ordnung auf dieser festzulegen. Als Ordnung bietet sich eine Kombination aus der lexikographischen Ordnung auf $\mathbb{N}^*$ und der totalen Ordnung, die durch die Aufzählung π induziert wird, an.

Definition 5.5 (Reduktionskalkül, Reduktionsbaum)

Sei $\mathcal{R} = (\Omega, R)$ ein Termersetzungssystem mit der dazugehörigen Aufzählung π von R. Ein *Reduktionskalkül von* $\mathcal{R}$ ist ein indizierte Kalkül

$$(T(\Omega, \mathcal{V}), D_{\text{success}}, (\mathbb{N}^* \times R, <), (\Rightarrow_{\mathcal{R},u,l\to r} \mid (u, l \to r) \in \mathbb{N}^* \times R))$$

mit $D_{\text{success}} \subseteq T(\Omega, \mathcal{V})$ beliebig und $<$ ist die totale Ordnung, bei der für alle $(u, l \to r), (u', l' \to r') \in \mathbb{N}^* \times R$ gilt $(u, l \to r) < (u', l' \to l')$, wenn eine der beiden Bedingungen erfüllt ist:

1. $u <_{lex} u'$
2. $u = u'$ und $\pi\,(l \to r) < \pi\,(l' \to r')$.

Der *Reduktionsbaum von* $\mathcal{R}$ *und* $t \in T(\Omega, \mathcal{V})$ ist der Berechnungsbaum für einen Reduktionskalkül von $\mathcal{R}$ und t.

□

Wir bemerken, daß sich Reduktionskalküle von $\mathcal{R}$ nur hinsichtlich der Menge D_{success} unterscheiden können. Mit anderen Worten jede Festlegung von D_{success} bestimmt eindeutig einen Reduktionskalkül von $\mathcal{R}$.

Im folgenden Beispiel greifen wir noch einmal das Termersetzungssystem vom Beispiel 5.2 auf und geben einen Reduktionsbaum für $\mathcal{R}_1$ und einen Term t an.

Beispiel 5.6 Sei $\mathcal{R}_1$ das Termersetzungssystem aus Beispiel 5.2 und t der Term

$$append(NIL, append(CONS(B, NIL))).$$

Dies ist der Reduktionsbaum von $\mathcal{R}_1$ und t:

□

5.2 Reduktionskalkül für $E_{\mathcal{R}}$–Unifikation

Wie schon am Anfang des Kapitels erwähnt, sollen die Kalküle, die hier definiert werden, vorwiegend für E–Unifikation geeignet sein. Wie kann nun mit Hilfe des Reduktionskalküls E–Unifikation betrieben werden? Für ein Termersetzungssystem $\mathcal{R}$ ist die *Menge*

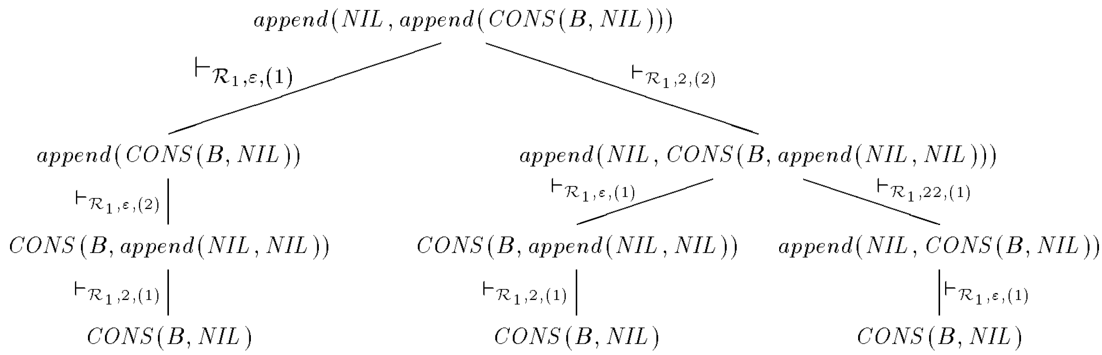

Abbildung 5.2: Reduktionsbaum von $\mathcal{R}_1$ und $append(NIL, append(CONS(B, NIL)))$

der zu $\mathcal{R}$ gehörenden Gleichungen, bezeichnet durch $E_{\mathcal{R}}$, die Menge $\{l = r \mid l \to r \in R\}$. In [HO80] wird folgende Gleichheit festgestellt: $\Leftrightarrow^*_{\mathcal{R}} \; = \; =_{E_{\mathcal{R}}}$; d.h. die Gleichheit, induziert durch das System $E_{\mathcal{R}}$ von Gleichungen, ist identisch zur transitiven, reflexiven, symmetrischen Hülle der Reduktionsrelation $\Rightarrow_{\mathcal{R}}$, die zu $\mathcal{R}$ gehört. Wir definieren nun kanonische Termersetzungssysteme und zeigen dann, wie man die Reduktionsrelation für E–Unifikation benutzen kann.

Definition 5.7 (Kanonisches Termersetzungssystem)

Sei $\mathcal{R} = (\Omega, R)$ ein Termersetzungssystem. R ist ein *kanonisches Termersetzungssystem*, wenn folgende zwei Bedingungen erfüllt sind:

1. R ist *konfluent*; d.h. für alle $s, t, t' \in T(\Omega, \mathcal{V})$ gilt: Wenn $s \Rightarrow^*_{\mathcal{R}} t$ und $s \Rightarrow^*_{\mathcal{R}} t'$, dann gibt es ein $s' \in T(\Omega, \mathcal{V})$, so daß $t \Rightarrow^*_{\mathcal{R}} s'$ und $t' \Rightarrow^*_{\mathcal{R}} s'$.
2. $\mathcal{R}$ ist *noethersch*; d.h. es gibt keine unendliche Berechnung durch $\Rightarrow_{\mathcal{R}}$.

□

Sei $\mathcal{R} = (\Omega, R)$ ein kanonisches Termersetzungssystem. Jeder Term $t \in T(\Omega, \mathcal{V})$ hat eine eindeutige Normalform, bezeichnet durch $nf_{\mathcal{R}}(t)$. Wir nennen diese Normalform auch *$\mathcal{R}$–Normalform von t*. (Diese Existenz kann durch die Eigenschaften noethersch und konfluent bewiesen werden.) In [Hue80] wird gezeigt, daß zwei Terme t und s genau dann in der Relation $\Leftrightarrow^*_{\mathcal{R}}$ zueinander stehen, wenn ihre $\mathcal{R}$–Normalformen gleich sind. Daraus folgt dieses Lemma:

Lemma 5.8 Sei $\mathcal{R} = (\Omega, R)$ ein kanonisches Termersetzungssystem und $t, s \in T(\Omega, \mathcal{V})$. Dann gilt:

$$t =_{E_{\mathcal{R}}} s, \text{ g.d.w. } nf_{\mathcal{R}}(t) = nf_{\mathcal{R}}(s).$$

□

Wir wenden nun Lemma 5.8 an, um für eine bestimmte Substitution φ zu zeigen, daß sie ein E–Unifikator für zwei Terme t und s ist.

Beispiel 5.9 Sei $\mathcal{R}_1$ das Termersetzungssystem aus Beispiel 5.2 und $t = \mathit{append}(x, y)$, $s = \mathit{CONS}(A, \mathit{CONS}(B, \mathit{NIL}))$ und $\varphi = [x/\mathit{CONS}(A, \mathit{CONS}(B, \mathit{NIL})), y/\mathit{NIL}]$. Dann gilt:

$$\begin{array}{rcl}
\varphi(t) & = & \mathit{append}(\mathit{CONS}(A, \mathit{CONS}(B, \mathit{NIL})), \mathit{NIL}) \\
 & \Rightarrow_{\mathcal{R}_1,\varepsilon,(2)} & \mathit{CONS}(A, \mathit{append}(\mathit{CONS}(B, \mathit{NIL}), \mathit{NIL})) \\
 & \Rightarrow_{\mathcal{R}_1,2,(2)} & \mathit{CONS}(A, \mathit{CONS}(B, \mathit{append}(\mathit{NIL}, \mathit{NIL})) \\
 & \Rightarrow_{\mathcal{R}_1,22,(1)} & \mathit{CONS}(A, \mathit{CONS}(B, \mathit{NIL})) \\
\varphi(s) & = & \mathit{CONS}(A, \mathit{CONS}(B, \mathit{NIL}))
\end{array}$$

Es gilt: $\mathit{nf}_{\mathcal{R}}(s) = \mathit{nf}_{\mathcal{R}}(t)$, und damit $\varphi(s) =_{E_{\mathcal{R}_1}} \varphi(t)$. Die Substitution φ ist also ein E–Unifikator der Terme s und t.

$\square$

Der Reduktionskalkül ermöglicht es zu entscheiden, ob für ein Termersetzungssystem $\mathcal{R}$ zwei Terme t und s $E_{\mathcal{R}}$–gleich sind. Dazu werden die Normalformen der Terme berechnet und verglichen; falls $\mathcal{R}$ ein kanonisches Termersetzungssystem ist, ist dies eine endlich lange Berechnung. Es wird allerdings noch kein Algorithmus bereit gestellt, der eine geeignete Substitution φ findet, die die zwei Terme $\varphi(t)$ und $\varphi(s)$ $E_{\mathcal{R}}$–gleich macht. Hier müßten wir also eine Substitution φ raten, und könnten dann prüfen, ob sie tatsächlich ein E–Unifikator ist.

Eine übersichtliche Methode, die speziell für $E_{\mathcal{R}}$–Unifikation gedacht ist, besteht darin, nicht verschiedenen Berechnungen von den Termen t und s, die $E_{\mathcal{R}}$–unifiziert werden sollen, zu starten, sondern von *einem* Term auszugehen. Wir führen dazu ein neues zweistelliges Symbol *equ* ein. Wir nennen ein derartiges Termersetzungssystem equal–Termersetzungssystem.

Definition 5.10 (equal–Termersetzungssysteme)

Sei $\mathcal{R} = (\Omega, R)$ ein Termersetzungssystem. Das *zu* $\mathcal{R}$ *gehörende equal–Termersetzungssystem*, bezeichnet durch $\mathcal{R}_{\text{equ}}$ ist folgendes TES (Ω', R): $\Omega' = \Omega \cup \{\mathit{equ}^{(2)}\}$.

$\square$

Der Reduktionskalkül, der für $E_{\mathcal{R}}$–Unifikation geeignet ist, ist nun lediglich ein Reduktionskalkül für das zu $\mathcal{R}$ gehörende equal–Termersetzungssystem. Außerdem ist nun die Erfolgsmenge D_{success} nicht mehr beliebig, sondern enthält alle Terme der Form $\mathit{equ}(\Theta, \Theta)$, wobei $\Theta \in T(\Omega, \mathcal{V})$.

Definition 5.11 (Reduktionskalkül für $E_{\mathcal{R}}$– Unifikation)

Sei $\mathcal{R} = (\Omega, R)$ ein kanonisches Termersetzungssystem und $\mathcal{R}_{\text{equ}} = (\Omega', R')$ das zu $\mathcal{R}$ gehörende equal–Termersetzungssystem. Der *Reduktionskalkül für $E_{\mathcal{R}}$–Unifikation von* $\mathcal{R}$, bezeichnet durch $\text{red}_E\mathcal{I}\,(\mathcal{R})$, ist der indizierte Kalkül

$$(\text{Equ}\,(T(\Omega, \mathcal{V})), D_{\text{success}}, (\mathbb{N}^* \times R, <), (\Rightarrow_{\mathcal{R},u,l\to r} \mid (u, l \to r) \in \mathbb{N}^* \times R))$$

mit

- $\text{Equ}\,(T(\Omega, \mathcal{V})) = \{equ\,(t, s) \mid t, s \in T(\Omega, \mathcal{V})\}$.
- $D_{\text{success}} = \{equ\,(\Theta, \Theta) \mid \Theta \in T(\Omega, \mathcal{V}),\ equ\,(\Theta, \Theta)$ ist Normalform bzgl. $\Rightarrow_{\mathcal{R}_{eq}}\}$.
- $<$ ist die totale Ordnung, bei der für alle $(u, l \to r), (u', l' \to r') \in \mathbb{N}^* \times R$ gilt $(u, l \to r) < (u', l' \to l')$, wenn eine der beiden Bedingungen erfüllt ist:

  1. $u <_{lex} u'$
  2. $u = u'$ und $\pi\,(l \to r) < \pi\,(l' \to r')$.

□

Definition 5.12 (Deterministischer Reduktionskalkül für $E_{\mathcal{R}}$–Unifikation.)

Sei $\mathcal{R}$ ein kanonisches Termersetzungssystem. Der *deterministische Reduktionsalkül für $E_{\mathcal{R}}$–Unifikation von* $\mathcal{R}$ ist der deterministische Kalkül $det\,(\text{red}_E\mathcal{I}\,(\mathcal{R}))$ von $\text{red}_E\mathcal{I}\,(\mathcal{R})$.

□

Da das neue Symbol *equ* nicht in R auftritt, gilt, daß die $E_{\mathcal{R}}$–Gleichheit $=_{E_{\mathcal{R}}}$ identisch ist mit der $E_{\mathcal{R}_{\text{equ}}}$–Gleichheit eingeschränkt auf $T(\Omega, \mathcal{V})$. Es gilt nämlich für Terme Θ und Θ': $nf_{\mathcal{R}_{\text{equ}}}(equ\,(\Theta, \Theta')) = equ\,(nf_{\mathcal{R}}(\Theta), nf_{\mathcal{R}}(\Theta'))$. Wenn für ein Kalkulandum $equ\,(t, s)$ die $\mathcal{R}_{equ}$–Normalform in D_{success} ist, dann sind die Terme t und s $E_{\mathcal{R}}$–gleich. Das folgende Lemma beschreibt diese Eigenschaft.

Lemma 5.13 Sei $\mathcal{R} = (\Omega, R)$ ein kanonisches Termersetzungssystem und $\text{red}_E\mathcal{I}(\mathcal{R})$ der Reduktionskalkül für $E_{\mathcal{R}}$–Unifikation von $\mathcal{R}$. Für ein Kalkulandum $equ\,(t, s)$ mit t, s in $T(\Omega, \mathcal{V})$ gilt:

$$t =_{E_{\mathcal{R}}} s, \text{ g.d.w. } nf_{\mathcal{R}_{equ}}(equ\,(t, s)) \in D_{\text{success}}$$

□

Betrachten wir die Berechnung im Beispiel 5.9, dann ist die entsprechende Berechnung im Reduktionskalkül für $E_{\mathcal{R}_1}$ die folgende (das equal-TES von $\mathcal{R}_1$ sei durch $\mathcal{R}$ bezeichnet):

$$\begin{array}{ll} & equ(append(CONS(A, CONS(B, NIL)), NIL), CONS(A, CONS(B, NIL))) \\ \Rightarrow_{\mathcal{R},1,(2)} & equ(CONS(A, append(CONS(B, NIL), NIL)), CONS(A, CONS(B, NIL))) \\ \Rightarrow_{\mathcal{R}_1,12,(2)} & equ(CONS(A, CONS(B, append(NIL, NIL)), CONS(A, CONS(B, NIL))) \\ \Rightarrow_{\mathcal{R}_1,22,(1)} & equ(CONS(A, CONS(B, NIL)), CONS(A, CONS(B, NIL))) \\ = & equ(\Theta, \Theta) \end{array}$$

und $\Theta = CONS(A, CONS(B, NIL)) \in T(\Omega, \mathcal{V})$, also ist φ aus Beispiel 5.9 ein $E_{\mathcal{R}_1}$–Unifikator der von $append(x, y)$ und $CONS(A, CONS(B, NIL))$.

In den nächsten Abschnitten werden wir Kalküle definieren, die effizientere Möglichkeiten bereitstellen, um E–Unifikatoren zu finden.

5.3 Narrowing Kalkül

Eine Verfeinerung des Reduktionskalküls ist der Narrowing Kalkül. Die Idee beim Narrowing besteht darin, in jedem Berechnungsschritt $t \rightsquigarrow t'$ die minimale Substitution φ zu finden, die es ermöglicht, einen Teilterm von t mit einer linken Regelseite zu unifizieren; dieser Teilterm wird dann, wie bei der Reduktionsrelation, durch die entsprechende Instanz der rechten Regelseite ersetzt. Wir merken uns bei jedem solchen Schritt die bis dahin angewandten Substitutionen $\varphi_1, \varphi_2, \ldots, \varphi_n$. Narrowing wurde von Slagle [Sla74] und Lankford [Lan75] definiert.

Für den Reduktionskalkül aus dem vorigen Abschnitt galt folgendes: Ein Term t' kann aus t durch die Reduktionsrelation berechnet werden, wenn es eine Regel $l \to r$, eine Substitution φ und ein Vorkommen u gibt, so daß $t/u = \varphi(l)$ ist. Da wir bei der Narrowing Relation die einzelnen Substitutionen komponieren wollen, werden wir die Variablenvorkommen auf den linken Seiten der Regeln umbenennen, so daß dort keine Variablen benutzt werden, die auch in t vorkommen, d.h. $\mathcal{V}(\rho(l)) \cap \mathcal{V}(t)$, wobei ρ eine Variablenumbenennung ist. In der Narrowing Relation wenden wir den allgemeinsten Unifikator φ von t/u und $\rho(l)$ an, so daß gilt: $\varphi(\rho(l)) = \varphi(t/u)$. Die Substitution φ muß dann bei jedem Schritt auf den gesamten Term t angewendet werden.

Um die Substitutionen der einzelnen Schritte verfolgen zu können, verwenden wir als Zwischenergebnisse im Narrowing Kalkül Paare der Form (t, ψ), wobei t ein Term ist, und ψ die Komposition der Substitutionen aller vorhergehenden Berechnungsschritte. Natürlich sind wir bei den Substitutionen nicht an den Variablenbindungen der jeweiligen linken Regelseiten interessiert, sondern nur an den Variablen, die in t auftauchen. Deshalb gilt: wenn (t', ψ') aus (t, ψ) durch die Narrowing Relation berechnet wird, dann ist ψ' die Komposition von ψ und φ, wobei φ der allgemeinste Unifikator von $\rho(l)$ und t/u eingeschränkt auf die Variablen in t ist.

Definition 5.14 (Narrowing Relation)

Sei $\mathcal{R} = (\Omega, R)$ ein Termersetzungssystem. Die *Narrowing Relation von* $\mathcal{R}$, bezeichnet durch $\rightsquigarrow_{\mathcal{R}}$ ist die folgende binäre Relation: Für alle $t, s \in T(\Omega, \mathcal{V})$ und $\psi, \psi' \in Sub(\mathcal{V}, \Omega)$ gilt $(t, \psi) \rightsquigarrow_{\mathcal{R}} (s, \psi')$, wenn die drei folgenden Bedingungen erfüllt sind:

1. Es gibt ein Vorkommen $u \in O(t)$ mit $t/u \notin \mathcal{V}$, eine Substitution $\varphi \in Sub(\mathcal{V}, \Omega)$,eine Regel $(l \to r) \in R$ und eine Umbenennung ρ der Variablen in l, so daß gilt:
   - $\mathcal{V}(\rho(l)) \cap \mathcal{V}(t)$ und
   - φ ist der allgemeinste Unifikator von $\rho(l)$ und t/u.
2. $s = \varphi(t[u \leftarrow \rho(r)])$
3. $\psi' = (\hat{\varphi}|_{\mathcal{V}(t)}) \circ \psi$, wobei $\hat{\varphi}$ die homomorphe Fortsetzung von φ zu $T(\Omega, \mathcal{V})$ ist. □

Für einen bestimmten Berechnungsschritt $t \rightsquigarrow_{\mathcal{R}} s$ notieren wir manchmal auch das Vorkommen u, die Substitution φ eingeschränkt auf $\mathcal{V}(t)$ und die Regel ξ, die für diesen Berechnungsschritt relevant sind, als Indizes von $\rightsquigarrow_{\mathcal{R}}$, also in der Form: $t \rightsquigarrow_{\mathcal{R},u,\varphi|_{\mathcal{V}(t)},\xi} s$.

Beispiel 5.15 Wir verwenden wieder das Termersetzungssystem $\mathcal{R}_1$ aus Beispiel 5.2. Sei t der Term

$$append(x, y).$$

Das Folgende ist eine Berechnung durch $\rightsquigarrow_{\mathcal{R}_1}$ beginnend mit $(t, \varphi_\emptyset)$:

$$
\begin{array}{l}
\qquad (append(x, y), \varphi_\emptyset) \\
\rightsquigarrow_{\mathcal{R}_1, \varepsilon, [x/CONS(x_1,x_2)], (2)} \\
\qquad (CONS(x_1, append(x_2, y)), [x/CONS(x_1, x_2)]) \\
\rightsquigarrow_{\mathcal{R}_1, 2, [x_2/CONS(x,x_3)], (2)} \\
\qquad (CONS(x_1, CONS(x, append(x_3, y))), [x/CONS(x_1, CONS(x, x_3))]) \\
\rightsquigarrow_{\mathcal{R}_1, 22, [x_3/NIL], (2)} \\
\qquad (CONS(x_1, CONS(x_3, y)), [x/CONS(x_1, CONS(x, NIL))])
\end{array}
$$

Diese Berechnung zeigt, daß wir den Term $(CONS(x_1, CONS(x_3, y))$ aus dem Term $append(x, y)$ durch mehrfaches Anwenden der Narrowing Relation $\rightsquigarrow_{\mathcal{R}_1}$ berechnen können. Dazu muß die Variable x durch den Term $CONS(x_1, CONS(x, NIL))$ ersetzt werden.

In allen Berechnungsschritten der obigen Rechnung, wurden Variablenumbenennung ρ_i mit $1 \leq i \leq 3$ durchgeführt. Es sei ρ_i die zum i–ten Berechnungsschritt gehörige Variablenumbenennung. Im ersten Berechnungsschritt treten die Variablen x und y sowohl im Kalkulandum $append(x, y)$, als auch auf der linken Seite der angewandten Regel (2) des Termersetzungssystems $\mathcal{R}_1$ auf. Es wurde $\rho_1 : \mathcal{V} \to \mathcal{V}$ mit $\rho_1(x) = x_1$ und $\rho_2(y) = x_2$ gewählt; für $\rho_2, \rho_3 : \mathcal{V} \to \mathcal{V}$ gilt $\rho_1(y) = x_3$, $\rho_2(x) = x_4$. (Wir haben hier gemäß der Definition der $(\mathcal{V}, \Omega)$– Substitution nur die Werte aufgezählt, für die $\rho_i(v) \neq v$ gilt).

□

Sei $\mathcal{R} = (\Omega, R)$ ein beliebiges Termersetzungssystem und sei $\mathcal{C}_{\mathcal{R}} = (T(\Omega, \mathcal{V}) \times Sub(\mathcal{V}, \Omega), D_{\text{success}}, \rightsquigarrow_{\mathcal{R}})$ ein Kalkül. Wir wollen nun einen indizierten Kalkül für $\mathcal{C}_{\mathcal{R}}$ angeben. Bei einem Berechnungsschritt von $\rightsquigarrow_{\mathcal{R}}$ treten wieder die beiden Nichtdeterminismen wie bei der Reduktionsrelation auf, i.e. (i) die Wahl eines Vorkommens u von t und (ii) wenn ein Vorkommen vorgegeben ist, die Wahl einer Regel in R, die dort angewendet wird.

Wir wählen deswegen als die Indexmenge J wieder die Menge $R \times \mathbb{N}^*$; und als Ordnung eine Kombination aus der lexikographischen Ordnung auf $\mathbb{N}^*$ und der totalen Ordnung, die durch die Aufzählung π induziert wird.

Definition 5.16 (Narrowing Kalkül)

Sei $\mathcal{R}$ ein Termersetzungssystem mit der dazugehörigen Aufzählung π von R. Ein *Narrowing Kalkül von* $\mathcal{R}$ ist ein indizierter Kalkül

$$(T(\Omega, \mathcal{V}) \times Sub(\mathcal{V}, \Omega), D_{\text{success}}, (\mathbb{N}^* \times R, <), (\rightsquigarrow_{\mathcal{R},u,l\to r} \mid (u, l \to r) \in \mathbb{N}^* \times R))$$

mit $D_{\text{success}} \subseteq T(\Omega, \mathcal{V}) \times Sub(\mathcal{V}, \Omega)$ beliebig und $<$ ist die totale Ordnung, definiert als: für alle $(u, l \to r), (u', l' \to r') \in \mathbb{N}^* \times R$ gilt $(u, l \to r) < (u', l' \to l')$, wenn eine der beiden Bedingungen erfüllt ist:

1. $u <_{lex} u'$
2. $u = u'$ und $\pi(l \to r) < \pi(l' \to r')$.

Der *Narrowingbaum von* $\mathcal{R}$ *und* $t \in T(\Omega, \mathcal{V}) \times Sub(\mathcal{V}, \Omega)$ ist der Berechnungsbaum für einen Narrowingkalkül von $\mathcal{R}$ und t.

□

Wir greifen abermals das Termersetzungssystem aus Beispiel 5.2 auf, und geben einen Ausschnitt eines Narrowingbaums an.

Beispiel 5.17 Sei $\mathcal{R}_1$ das Termersetzungssystem aus Beispiel 5.2 und t der Term

$$append(z_1, append(z_2, NIL)).$$

In der gesamten Arbeit wird aus Platzgründen in den Narrowingbäumen immer nur die erste Komponente eines Zwischenergebnisses an den Knoten dargestellt. An dem Ast zu einem Zwischenergebnis steht die Substitution, die für diesen Schritt relevant ist; d.h. die Substitution φ, die zu jedem Knoten gehört, ergibt sich aus der Komposition der Substitutionen, die an den Ästen zu diesem Knoten stehen.

Abbildung 5.3 zeigt den Narrowingbaum von $\mathcal{R}_1$ und t (die skizzenhaft dargestellten Teilbäume enthalten unendliche Äste): □

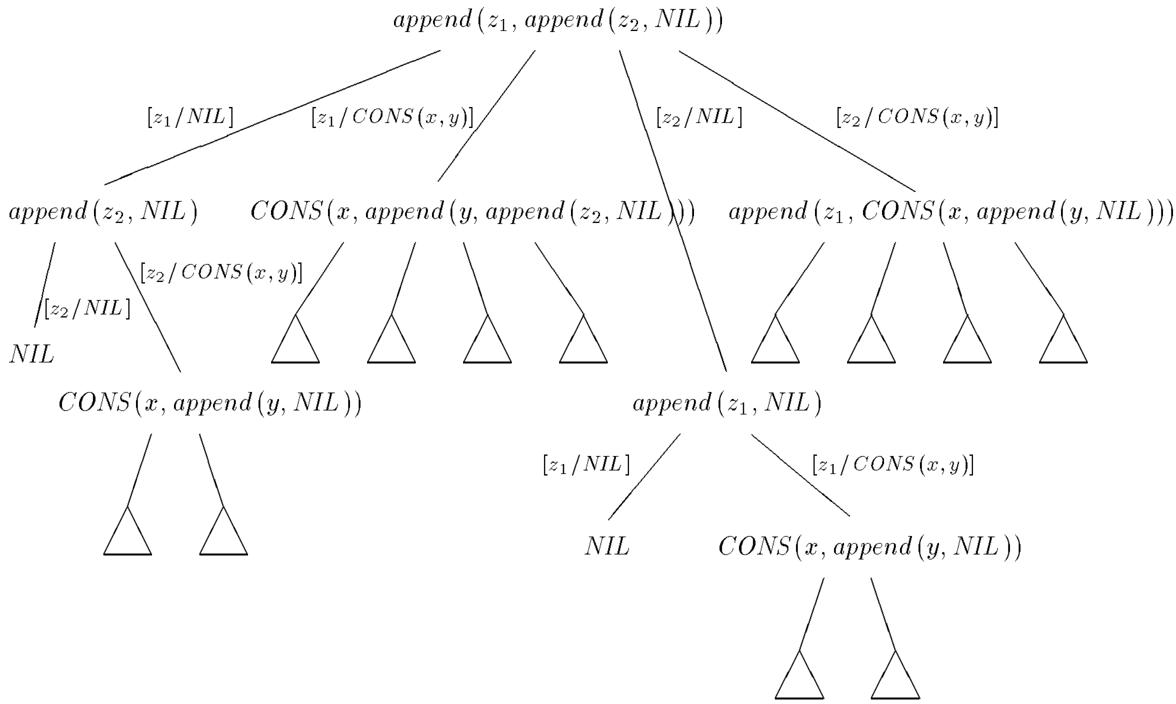

Abbildung 5.3: Narrowingbäume von $\mathcal{R}_1$ und $append(z_1, append(z_2, A))$

Wie man sieht, enthält der Narrowingbaum von t in dem Beispiel unendlich lange Äste. Für die Anwendung von Narrowing zur E–Unifikation bedeutet dies, daß beliebig lange Listen (Folgen von *CONS* Anwendungen) mit Variablen erzeugt werden können, die mit einem *NIL* enden.

Wird ein E–Unifikator von t und s gesucht, so muß bei jedem Schritt versucht werden, den Term t' eines Zwischenergebnisses (t', ψ') mit s zu unifizieren. Gibt es eine Substitution μ, die diese Unifikation bewirkt, dann ist $\psi' \circ \mu$ ein E–Unifikator von t und s.

Wir haben also jetzt die Effizienz des Algorithmus zum Finden von E–Unifikatoren im Reduktionskalkül erhöht; die Substitution φ muß jetzt nicht mehr direkt am Anfang geraten werden, sondern kann durch den Narrowing Kalkül und zusätzlicher Unifikation berechnet werden (siehe dazu Hullot's Satz in [Hul80]). Natürlich ist der Narrowingbaum im allgemeinen unendlich, deswegen ist nicht garantiert, daß diese Methode einen E–Unifikator zu finden, terminiert. Außerdem muß das Termersetzungssystem $\mathcal{R}$ kanonisch sein, damit überhaupt garantiert ist, daß jeder Term eine eindeutige Normalform hat.

5.4 Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation

Wie im vorhergehenden Abschnitt wollen wir jetzt zunächst einen Narrowing Kalkül definieren, der ganz speziell für $E_{\mathcal{R}}$–Unifikation geeignet ist und der von einem equal–TES

$\mathcal{R}_{\text{equ}}$ eines TESs $\mathcal{R}$ ausgeht. Dazu muß als Erfolgsmenge D_{success} die Menge der Terme $(equ(\Theta_1, \Theta_2), \varphi)$ gewählt werden, so daß Θ_1 und Θ_2 unifizierbar sind.

Definition 5.18 (Narrowing Kalkül für $E_{\mathcal{R}}$– Unifikation)
Sei $\mathcal{R} = (\Omega, R)$ ein kanonisches Termersetzungssystem, und $\mathcal{R}_{\text{equ}} = (\Omega', R')$ das zu $\mathcal{R}$ gehörende equal–Termersetzungssystem. Der *Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation von* $\mathcal{R}$, bezeichnet durch $\text{nar}_E\mathcal{I}(\mathcal{R})$, ist der indizierte Kalkül

$$(\text{Equ}(T(\Omega, \mathcal{V})) \times Sub(\mathcal{V}, \Omega), D_{\text{success}}, (\mathbb{N}^* \times R, <), (\rightsquigarrow_{\mathcal{R},u,l\to r} \mid (u, l \to r) \in \mathbb{N}^* \times R))$$

mit

- $\text{Equ}(T(\Omega, \mathcal{V})) = \{equ(t, s) \mid t, s \in T(\Omega, \mathcal{V})\}$
- $D_{\text{success}} = \{(equ(\Theta_1, \Theta_2), \varphi) \mid \Theta_1, \Theta_2 \in T(\Omega, \mathcal{V}),\ \Theta_1$ und Θ_2 sind unifizierbar, $\varphi \in Sub(\mathcal{V}, \Omega)\}$
- $<$ ist definiert wie beim Narrowing Kalkül (siehe Definition 5.16). □

Definition 5.19 (Deterministischer Narrowing Kalkül für $R_{\mathcal{R}}$–Unifikation.)
Sei $\mathcal{R}$ ein kanonisches Termersetzungssystem. Der *deterministische Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation von* $\mathcal{R}$ ist der deterministische Kalkül $det(\text{nar}_E\mathcal{I}(\mathcal{R}))$ von $\text{nar}_E\mathcal{I}(\mathcal{R})$. □

Sei $d = (equ(t, s), \varphi_\emptyset)$ ein Kalkulandum. Wenn $(equ(\Theta_1, \Theta_2, \varphi) \in D_{\text{success}}$, und der allgemeinste Unifikator von Θ_1 und Θ_2 ist σ, dann ist $\sigma \circ \varphi$ ein $E_{\mathcal{R}}$–Unifikator von t und s (siehe Satz 4.9). Hier ein Beispiel dazu:

Beispiel 5.20 Sei $\mathcal{R}$ das TES aus Beispiel 5.2. Wir möchten nun mit Hilfe des Narrowing Kalküls für $E_{\mathcal{R}}$–Unifikation Substitutionen finden, die $E_{\mathcal{R}}$–Unifikatoren der Terme $t = append(x, y)$ und $s = CONS(A, CONS(B, NIL))$ sind. Im Beispiel 5.9 haben wir durch den Reduktionskalkül gezeigt, daß die Substitution $\varphi = [x/CONS(A, CONS(B, NIL)), y/NIL]$ ein $E_{\mathcal{R}}$– Unifikator von t und s ist. Wie berechnet sich diese Substitution nun durch Narrowing? Betrachten wir dazu den Narrowingbaum von $\mathcal{R}_{\text{equ}}$ und $equ(t, s)$:

Alle drei Zwischenergebnisse an den Blättern des Narrowingbaums von Abbildung 5.4, sind in der Menge D_{success}. Nämlich:

1. $(equ(y, s), \varphi_1) \in D_{\text{success}}$, weil es den allgemeinsten Unifikator $\sigma = [y/s]$ gibt, also ist

$$\sigma \circ \varphi_1 = [x/NIL, y/CONS(A, CONS(B, NIL))]$$

   ein $E_{\mathcal{R}}$–Unifikator von t und s.

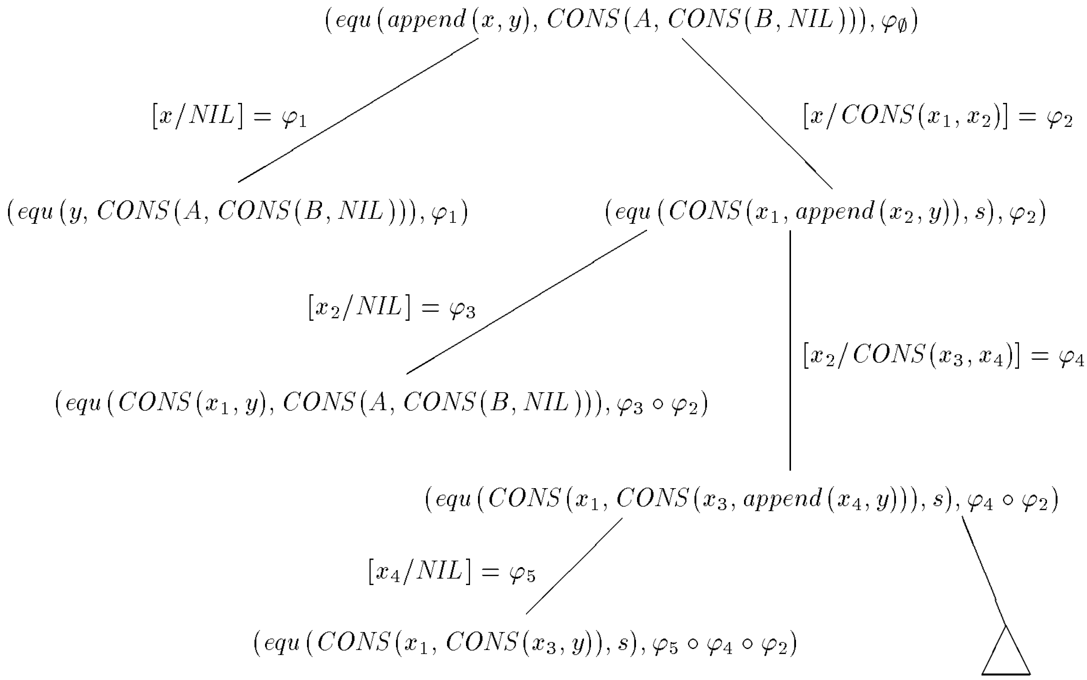

Abbildung 5.4: Narrowingbaum von $equ(t, s)$

2. $(equ(CONS(x_1, y), CONS(A, CONS(B, NIL)), \varphi_3 \circ \varphi_2) \in D_{\text{success}}$, weil es den allgemeinsten Unifikator $\sigma_2 = [x_1/A, y/CONS(B, NIL)]$, also ist

$$\sigma_2 \circ \varphi_3 \circ \varphi_2 = [x/CONS(A, NIL), y/CONS(B, NIL)]$$

   ein $E_{\mathcal{R}}$–Unifikator von t und s.

3. $(equ(CONS(x_1, CONS(x_3, y)), CONS(A, CONS(B, NIL))), \varphi_5 \circ \varphi_4 \circ \varphi_2) \in D_{\text{success}}$, weil es den allgemeinsten Unifikator $\sigma_3 = [x_1/A, x_3/B, y/NIL]$ gibt, also ist

$$\sigma_3 \circ \varphi_5 \circ \varphi_4 \circ \varphi_2 = [x/CONS(A, CONS(B, NIL)), y/NIL]$$

   ein $E_{\mathcal{R}}$–Unifikator von t und s.

Es ist offensichtlich, daß es nur diese drei $E_{\mathcal{R}}$–Unifikatoren geben kann, d.h. der undenliche Teilbaum, der rechts unten in der Abbildung zu sehen ist, enthält keine Lösungen mehr. Würden wir also den deterministischen Narrowingkalkül von $\mathcal{R}_{equ}$ benutzen, der einen depth–first left–to–right tree traversal durch den Narrowingbaum beschreibt, so wären nach sieben Schritten alle Lösungen gefunden; die Suche nach einer weiteren Lösung würde aber nicht terminieren.

□

In den folgenden beiden Abschnitten werden wir uns weiterhin mit Narrowing beschäftigen. Wir werden aber nicht mehr den kompletten Berechnungsbaum betrachten, sondern nur noch einzelne Äste des Baumes. Zur Auswahl der Äste gibt es verschiedene Strategien. Wir werden die Wahl der Regeln nicht einschränken, da sonst die Vollständigkeit einer solchen Strategie im allgemeinen sofort verletzt wäre. Eine Strategie wird also in unserem Sinne lediglich die Auswahl von Vorkommen betreffen. Eine bestimmte Weise der Wahl von Vorkommen ist das leftmost outermost Wählen, welches wir jetzt betrachten wollen.

5.5 Leftmost Outermost Narrowing Kalkül

Wir wollen nun die Effizienz, mit der E–Unifikation durch das eben beschriebene Narrowing betrieben werden kann, noch weiter erhöhen, indem wir eine Strategie festlegen. Wir wollen zuerst ganz allgemein definieren, was wir im Kontext des Narrowings unter einer Strategie verstehen wollen. Die leftmost outermost Narrowing Strategie ist dann lediglich eine Instanz dieser Definition.

Definition 5.21 (Strategie)

Eine *(Narrowing Vorkommen Auswahl–) Strategie*, bezeichnet durch ST, ist eine partielle Funktion von Termen nach Vorkommen. $ST : T(\Omega, \mathcal{V}) \to \mathbb{N}^*$.

□

Die Strategie, die wir für das Narrowing festlegen, erlaubt das Narrowing nur an „leftmost outermost“ Vorkommen. Das leftmost outermost Vorkommen ist das in einem Term am weitesten links vorkommende und innerhalb von Funktionsverschachtelungen am weitesten außen gelegene Vorkommen.

Sei also zum Beispiel ein Termersetzungssystem gegeben, in welchem eine Regel mit linker Seite $g(x, y)$ auftritt, und sei t der Term $f(g(g(A, B), B), g(C, D), D)$. Es gibt zwei leftmost Vorkommen von g in t, nämlich 11 und 1. Das letztere ist offensichtlich das äußere und damit ein leftmost *outermost* Vorkommen von g in t. Es ist einfach zu sehen, daß das leftmost outermost Vorkommen lexikographisch minimal Vorkommen sind. Das Vorkommen 11 ist in diesem Fall ein leftmost innermost Vorkommen von t. Es gibt auch Strategien, die innermost Vorkommen des Narrowingbaums auswählen, siehe dazu z.B. [Fri85] und [Hö87].

Durch die leftmost outermost Strategie kann die Breite des Narrowingbaumes verringert werden. Zusätzlich kann auch die Höhe des Baumes verringert werden, weil Vorkommen, die durch Regelanwendung am lefmost outermost Vorkommen gelöscht werden, nicht mehr ausgewertet werden müssen.

Definition 5.22 (Menge der Narrowing Vorkommen)

Sei $\mathcal{R} = (\Omega, R)$ ein Termersetzungssystem und $t \in T(\Omega, \mathcal{V})$ ein Term. Die *Menge der Narrowing Vorkommen für $\mathcal{R}$ und t*, bezeichnet durch $\mathrm{nar}O(\mathcal{R}, t)$ ist die Menge aller

Vorkommen u mit $t/u \notin \mathcal{V}$ und: Es gibt eine Substitution $\varphi \in Sub(\mathcal{V}, \Omega)$, eine Regel $(l \to r) \in R$ und eine Umbenennung ρ der Variablen in l, so daß gilt:

1. $\mathcal{V}(\rho(l)) \cap \mathcal{V}(t)$ und
2. φ ist der allgemeinste Unifikator von $\rho(l)$ und t/u.

□

Definition 5.23 (leftmost outermost Strategie)
Sei $\mathcal{R} = (\Omega, R)$ ein Termersetzungssystem und $t \in T(\Omega, \mathcal{V})$ ein Term, auf den die Narrowing Relation $\rightsquigarrow_{\mathcal{R}}$ angewendet werden kann, d.h. nar$O(\mathcal{R}, t) \neq \emptyset$. Die *leftmost outermost* Strategie, $ST_{lo} : T(\Omega, \mathcal{V}) \to \mathbb{N}^*$ ist definiert als:

$$ST_{lo}(t) = \min_{<_{\text{lex}}}(\text{nar}O(\mathcal{R}, t))$$

□

Die leftmost outermost Narrowing Relation ist nun einfach definiert als Narrowing unter der leftmost outermost Strategie, d.h. ein Term t wird nicht an einem beliebigen Vorkommen genarrowt, sondern nur am leftmost outermost Vorkommen; vorausgesetzt, daß überhaupt ein Zwischenergebnis aus t durch die Narrowing Relation berechnet werden kann.

Wir definieren jetzt die leftmost outermost Narrowing Relation und den dazugehörigen Kalkül. Dann zeigen wir, wie er benutzt werden kann, um E–Unifikatoren zu finden.

Definition 5.24 (leftmost outermost Narrowing Relation)
Sei $\mathcal{R}$ ein Termersetzungssystem. Die *leftmost outermost Narrowing Relation von* $\mathcal{R}$, bezeichnet durch $\overset{lo}{\rightsquigarrow}_{\mathcal{R}}$, ist die folgende binäre Relation: Für alle $t, s \in T(\Omega, \mathcal{V})$ und $\psi, \psi' \in Sub(\mathcal{V}, \Omega)$ gilt $(t, \psi) \rightsquigarrow_{\mathcal{R}} (s, \psi')$, wenn die drei Bedingungen erfüllt sind:

1. Es gibt ein Vorkommen $u \in O(t)$ mit $t/u \notin \mathcal{V}$, mit $u = ST_{lo}(t)$, eine Substitution $\varphi \in Sub(\mathcal{V}, \Omega)$, eine Regel $(l \to r) \in R$ und eine Umbenennung ρ der Variablen in l, so daß gilt:
   - $\mathcal{V}(\rho(l)) \cap \mathcal{V}(t) = \emptyset$ und
   - φ ist der allgemeinste Unifikator von $\rho(l)$ und t/u.
2. $s = \varphi(t[u \leftarrow \rho(r)])$
3. $\psi' = (\hat{\varphi}|_{\mathcal{V}(t)}) \circ \psi$, wobei $\hat{\varphi}$ die homomorphe Fortsetzung von φ zu $T(\Omega, \mathcal{V})$ ist.

□

Es ist offensichtlich, daß die leftmost outermost Narrowing Relation eine Untermenge der Narrowing Relation ist. Außerdem haben wir durch Wahl einer Strategie zur Auswahl eines bestimmten Vorkommens einen Nichtdeterminismus der Reduktionsrelation beseitigt. Es gibt demnach für die leftmost outermost Narrowing Relation nur noch einen Nichtdeterminismus, nämlich die Wahl einer Regel.

Entsprechend der Reduktionsrelation und der Narrowing Relation definieren wir jetzt einen indizierten Kalkül für die leftmost outermost Narrowing Relation. Als Indexmenge J wählen wir die Menge der Regeln R des zugehörigen Termersetzungssystems. Die totale Ordnung auf J ist die Aufzählung π der Regeln.

Definition 5.25 (leftmost outermost Narrowing Kalkül)

Sei $\mathcal{R}$ ein Termersetzungssystem mit der dazugehörigen Aufzählung π von R. Ein *leftmost outermost Narrowing Kalkül von* $\mathcal{R}$, bezeichnet durch lo-nar$\mathcal{I}(\mathcal{R})$, ist ein indizierte Kalkül

$$(T(\Omega, \mathcal{V}) \times Sub(\mathcal{V}, \Omega), D_{\text{success}}, (R, <), (\overset{lo}{\rightsquigarrow}_{\mathcal{R}, l \to r} \mid l \to r \in R))$$

mit $D_{\text{success}} \subseteq T(\Omega, \mathcal{V}) \times Sub(\mathcal{V}, \Omega)$ beliebig und $<$ ist die totale Ordnung, definiert als: für alle $(l \to r), (l' \to r') \in R$ gilt $(l \to r) < (l' \to l')$, wenn $\pi(l \to r) < \pi(l' \to r')$.

Der *leftmost outermost Narrowingbaum von* $\mathcal{R}$ *und* $t \in T(\Omega, \mathcal{V}) \times Sub(\mathcal{V}, \Omega)$ ist der Berechnungsbaum für einen leftmost outermost Narrowing Kalkül von $\mathcal{R}$. Im folgenden werden wir auch machmal die Abkürzung *lo–Narrowingbaum* verwenden.

□

Ein lo–Narrowingbaum kann natürlich aus einem Narrowingbaum konstruiert werden, indem man einfach alle die Äste löscht, die nicht zu Berechnungen am leftmost outermost Vorkommen gehören. Das folgende Beispiel illustriert diese Gegebenheit.

Beispiel 5.26 Wir betrachten das Termersetzungssystem aus Beispiel 5.2 und wieder den Term $append(z_1, append(z_2, NIL))$. Um einen lo–Narrowingbaum zu konstruieren, übernehmen wir einfach den Narrowingbaum aus Beispiel 5.17 und erhalten nur noch die Äste, die zu Berechnungen an leftmost outermost Vorkommen gehören. Der entstehende leftmost outermost Narrowingbaum ist in Abbildung 5.5 gezeigt.

□

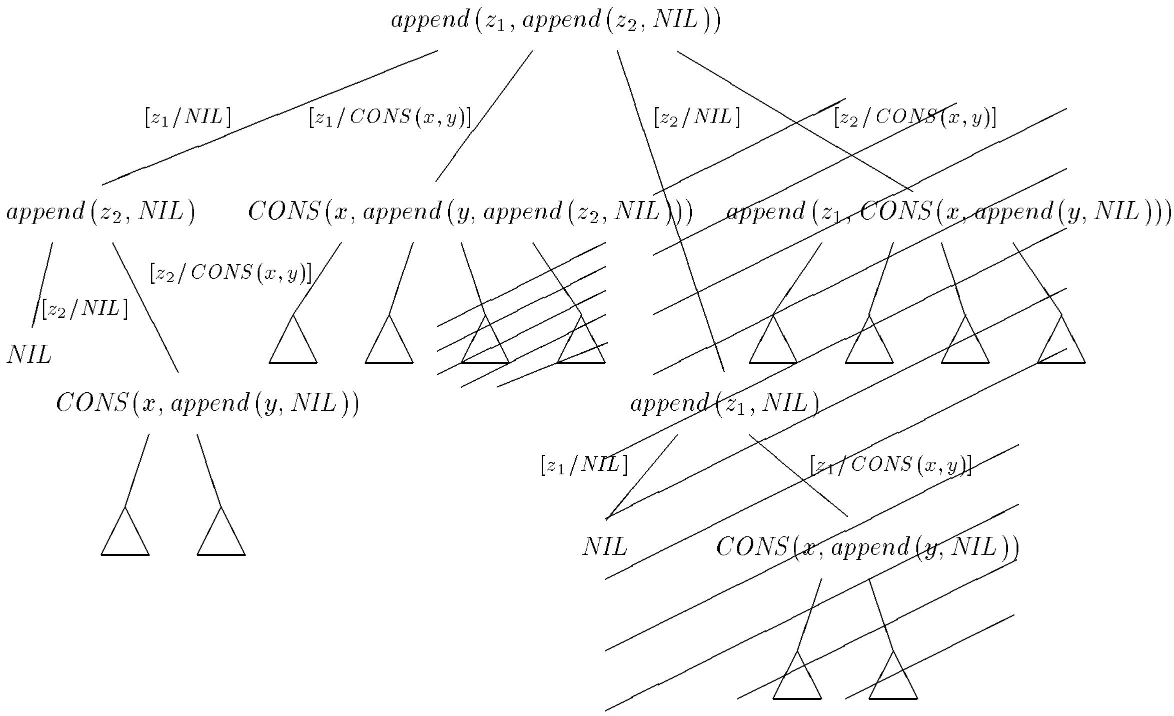

Abbildung 5.5: lo–Narrowingbaum von $\mathcal{R}_1$ und $append(z_1, append(z_2, A))$.

5.6 Leftmost Outermost Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation

Wir wollen nun untersuchen, wie der leftmost outermost Narrowing Kalkül für E–Unifikation benutzt werden kann. Natürlich stellt sich, in Anbetracht der Tatsache, daß der lo–Narrowingbaum nur noch einen Teil der Äste des Narrowingbaums enthält, die Frage, ob der lo-Narrowingbaum überhaupt vollständig ist für ein beliebiges Termersetzungssystem $\mathcal{R}$. Das folgende Beispiel zeigt, daß dies *nicht* der Fall ist; d.h. es gibt Termersetzungssysteme, für die lo-Narrowing nicht vollständig ist.

Beispiel 5.27 (vergleiche Beispiel 1 in [Ech88]) Sei $\mathcal{R} = (\Omega, R)$ ein Termersetzungssystem mit $\Omega = \{f^{(2)}, \gamma^{(1)}\}$ und R die Menge der folgenden Regeln:

$$\begin{array}{rcll} f(\alpha,\alpha) & \to & \alpha & (1) \\ f(\gamma(x),\alpha) & \to & \gamma(\alpha) & (2) \\ f(x,\gamma(y)) & \to & \gamma(\gamma(\alpha)) & (3) \end{array}$$

Sei t der Term $f(f(z_1,z_2),z_3)$. Die einzige Berechnung durch $\overset{lo}{\rightsquigarrow}_{\mathcal{R}}$ beginnend mit $(t, \varphi_\emptyset)$ ist:

$$(f(f(z_1,z_2),z_3),\varphi_\emptyset) \overset{lo}{\rightsquigarrow}_{\mathcal{R},\varepsilon,[x/f(z_1,z_2),z_3/\gamma(y)],(3)} (\gamma(\gamma(\alpha)),[z_3/\gamma(y)]).$$

Aber es gibt die folgende Berechnung durch $\rightsquigarrow_{\mathcal{R}}$, beginnend mit t:

$$\begin{array}{lll} (f(f(z_1,z_2),z_3),\varphi_\emptyset) & \rightsquigarrow_{\mathcal{R},1,[z_1/\alpha,z_2/\alpha],(1)} & (f(\alpha,z_3),[z_1/\alpha,z_2/\alpha]) \\ & \rightsquigarrow_{\mathcal{R},\varepsilon,[z_3/\alpha],(1)} & (\alpha,[z_1/\alpha,z_2/\alpha,z_3/\alpha]) \end{array}$$

Die Substitution $[z_1/\alpha, z_2/\alpha, z_3/\alpha]$ ist daher E–Unifikator der Terme t und α. Mit lo–Narrowing ist es aber nicht möglich diese Substitution zu berechnen.

□

In [Ech88] zeigt Echahed, daß lo–Narrowing nur noch für ganz bestimmte kanonische Termersetzungssysteme vollständig ist, nämlich für solche, welche die *property of free strategies* besitzen. Wir nennen diese Termersetzungssysteme *canonical, totally defined, not strictly sub–unifiable term rewriting systems*, oder kurz: *ctn–trs*.

Definition 5.28 (ctn–trs)

Sei $\mathcal{R} = (\Omega, R)$ ein Termersetzungssystem. $\mathcal{R}$ ist ein *canonical, totally defined, not strictly sub–unifiable term rewriting system*, kurz *ctn–trs* wenn folgende Bedingungen erfüllt sind:

1. $\mathcal{R}$ ist kanonisch.
2. $\Omega = F \cup \Delta$ und $F \cap \Delta = \emptyset$. F heißt *Menge der Funktionssymbole* und Δ heißt *Menge der Konstruktoren.*
3. Jede linke Regelseite ist linear in $\mathcal{V}$.
4. Jede linke Regelseite hat die Form $f(t_1,\ldots,t_n)$ mit $f \in F$ und für alle $i \in [n] : t_i \in T(\Delta,\mathcal{V})$.
5. Jedes Funktionssymbol in F ist total definiert; d.h. für alle $t \in T(\Omega,\mathcal{V}) : nf_{\mathcal{R}}\,(t) \in T(\Delta,\mathcal{V})$.
6. Die linken Regelseiten von R sind paarweise *not strictly sub–unifiable*; zwei linke Regelseiten l und l' sind strictly sub–unifiable, wenn es ein Vorkommen $u \in O(l) \cap O(l')$, so daß gilt:
   - Sei ρ eine Variablenumbenennung, so daß gilt: $\mathcal{V}(l/u) \cap \mathcal{V}(l'/u) = \emptyset$. l/u und $\rho(l'/u)$ sind unifizierbar mit dem allgemeinsten Unifikator σ_u, und σ_u ist weder eine Variablenumbennung, noch die leere Substitution.
   - Für alle Vorkommen w mit $w < u$ gilt:
     (a) Wenn $l/w = f(t_1,\ldots t_n)$, $l'/w = f'(t'_1,\ldots,t'_n)$ mit $f,f' \in \Omega^{(n)}$, $n \geq 1$, für $i \in [n] : t_i, t'_i \in T(\Omega,\mathcal{V})$, dann ist $f = f'$.
     (b) Wenn $l/w, l'/w \in \Omega^{(0)} \cup \mathcal{V}$, dann ist $l/w = l'/w$.

Im Folgenden werden wir für ctn–trs auch die Schreibweise $\mathcal{R} = (F, \Delta, R)$ benutzen.

□

Das Termersetzungssystem aus Beispiel 5.27 ist *kein* ctn–trs. Betrachten wir beispielsweise die linken Regelseiten der Regeln (1) und (3). Die Teilterme am Vorkommen 1, i.e. die Terme α und x sind strictly subunifiable. Dagegen ist das Termersetzungssystem aus Beispiel 5.2 ein ctn–trs, wenn wir als Menge der Funktionssymbole $\{append\}$ wählen und als Menge von Konstruktoren die Menge $\{CONS^{(2)}, NIL^{(0)}, A^{(0)}, B^{(0)}, C^{(0)}\}$ wählen.

Es ist außerdem zu bemerken, daß jeder modular tree transducer [EV91] ein ctn–trs ist; die Klasse der modular tree transducers charakterisiert die Klasse der primitiv rekursiven Baumfunktionen [Hup78].

Ganz analog zu der Definition des equal-Termersetzungssystems für beliebige Termersetzungssysteme definieren wir ein equal–ctn–trs. Es wird die gleiche Bezeichnung, i.e. $\mathcal{R}_{\text{equ}}$ verwendet, da aus dem Kontext klar ist, ob es sich um ein allgemeines TES oder um ein ctn–trs handelt.

Ausgehend von einem equal–ctn–trs können wir nun den lo–Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation definieren. Die Definition ist fast identisch zur Definition des Narrowing Kalküls für $E_{\mathcal{R}}$–Unifikation. Wieder wählen wir als Erfolgsmenge D_{success} die Menge der Terme $(equ(\Theta_1, \Theta_2), \varphi)$, so daß Θ_1 und Θ_2 unifizierbar sind.

Definition 5.29 (lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation)

Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs. Der *lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation*, bezeichnet durch lo-nar$_E\mathcal{I}(\mathcal{R})$, ist der indizierte Kalkül

$$(\text{Equ}(T(\Omega, \mathcal{V})) \times Sub(\mathcal{V}, \Omega), D_{\text{success}}, (R, <), (\overset{lo}{\leadsto}_{\mathcal{R}, l \to r} \mid (l \to r) \in R))$$

mit

- $\text{Equ}(T(\Omega, \mathcal{V})) = \{equ(t, s) \mid t, s \in T(\Omega, \mathcal{V})\}$
- $D_{\text{success}} = \{(equ(\Theta_1, \Theta_2), \varphi) \mid \Theta_1, \Theta_2 \in T(\Omega, \mathcal{V}),\ \Theta_1$ und Θ_2 sind unifizierbar, $\varphi \in Sub(\mathcal{V}, \Delta)\}$
- $<$ ist definiert wie beim lo–Narrowing Kalkül (siehe Definition 5.25).

□

Definition 5.30 (Deterministischer lo–Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation)

Sei $\mathcal{R}$ ein ctn–trs und lo-nar$_E\mathcal{I}(\mathcal{R})$ der lo–Narrowing Kalkül von $\mathcal{R}$. Der *deterministische lo–Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation von $\mathcal{R}$* ist der deterministische Kalkül det(lo-nar$_E\mathcal{I}(\mathcal{R})$) von lo-nar$_E\mathcal{I}(\mathcal{R})$.

□

Das Beispiel 5.20 ist natürlich auch ein Beispiel für den lo–Narrowingkalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation, weil es dort nur jeweils ein Narrowing–Vorkommen gibt. Es soll aber auch ein Beispiel im Sinne von Beispiel 5.26 gegeben werden, in dem Äste durch die Einschränkung auf das leftmost outermost Vorkommen abgeschnitten werden.

Beispiel 5.31 Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs mit (dies ist das gleiche TES, wie in 5.2, nur *append* ist durch *ap* ersetzt worden):

- $F = \{ap^{(2)}\}$
- $\Delta = \{CONS^{(2)}, NIL^{(0)}, A^{(0)}, B^{(0)}, C^{(0)}\}$
- R enthält folgende Regeln:

$$\begin{array}{rcll} ap\,(NIL\,, x) & \to & x & (1) \\ ap\,(CONS\,(x, y), z) & \to & CONS\,(x, ap\,(y, z)) & (2) \end{array}$$

Wir wollen die beiden Terme $s = ap\,(z_1, ap\,(z_2, NIL\,))$ und $t = CONS\,(B, CONS\,(A, NIL\,))$ unter der durch das obige ctn–trs induzierten Gleichheit unifizieren. Betrachten wir dazu den leftmost–outermost Narrowingbaum in Abbildung 5.6.

Die Substitutionen in den Zwischenergebnissen sowie an den Ästen sind der Übersichtlichkeit halber weggelassen worden. Wir erhalten folgende Lösungen (in der Abbildung unterstrichen; MGU steht für most general unifier):

- $\varphi_1 = \underbrace{[x/B, z_3/A]}_{\text{MGU}} \circ [z_4/NIL\,] \circ [y/CONS\,(z_3, z_4)] \circ [z_2/CONS\,(x, y)] \circ [z_1/NIL\,] = [z_1/NIL\,, z_2/CONS\,(B, CONS\,(A, NIL\,))]$
- $\varphi_2 = \underbrace{[x/B, z_3/A]}_{\text{MGU}} \circ [z_4/NIL\,] \circ [z_2/CONS\,(z_3, z_4)] \circ [y/NIL\,] \circ [z_1/CONS\,(x, y)] = [z_1/CONS\,(B, NIL\,), z_2/CONS\,(A, NIL\,)]$
- $\varphi_3 = \underbrace{[x/B, z_3/A]}_{\text{MGU}} \circ [z_2/NIL\,] \circ [z_4/NIL\,] \circ [y/CONS\,(z_3, z_4)] \circ [z_1/CONS\,(x, y)] = [z_1/CONS\,(B, CONS\,(A, NIL\,)), z_2/NIL\,]$

Es ist zu bemerken, daß die Teilbäume direkt neben den Knoten, die mit einer Lösung beschriftet sind, unendliche Äste enthalten. Dies bedeutet für den deterministischen Kalkül, daß nur die erste Lösung gefunden wird, i.e. φ_1. Die anderen beiden Lösungen können nicht gefunden werden, weil der deterministische Kalkül nicht mehr aus dem unendlichen Teilbaum neben dem Blatt der ersten Lösung zurückkehrt.

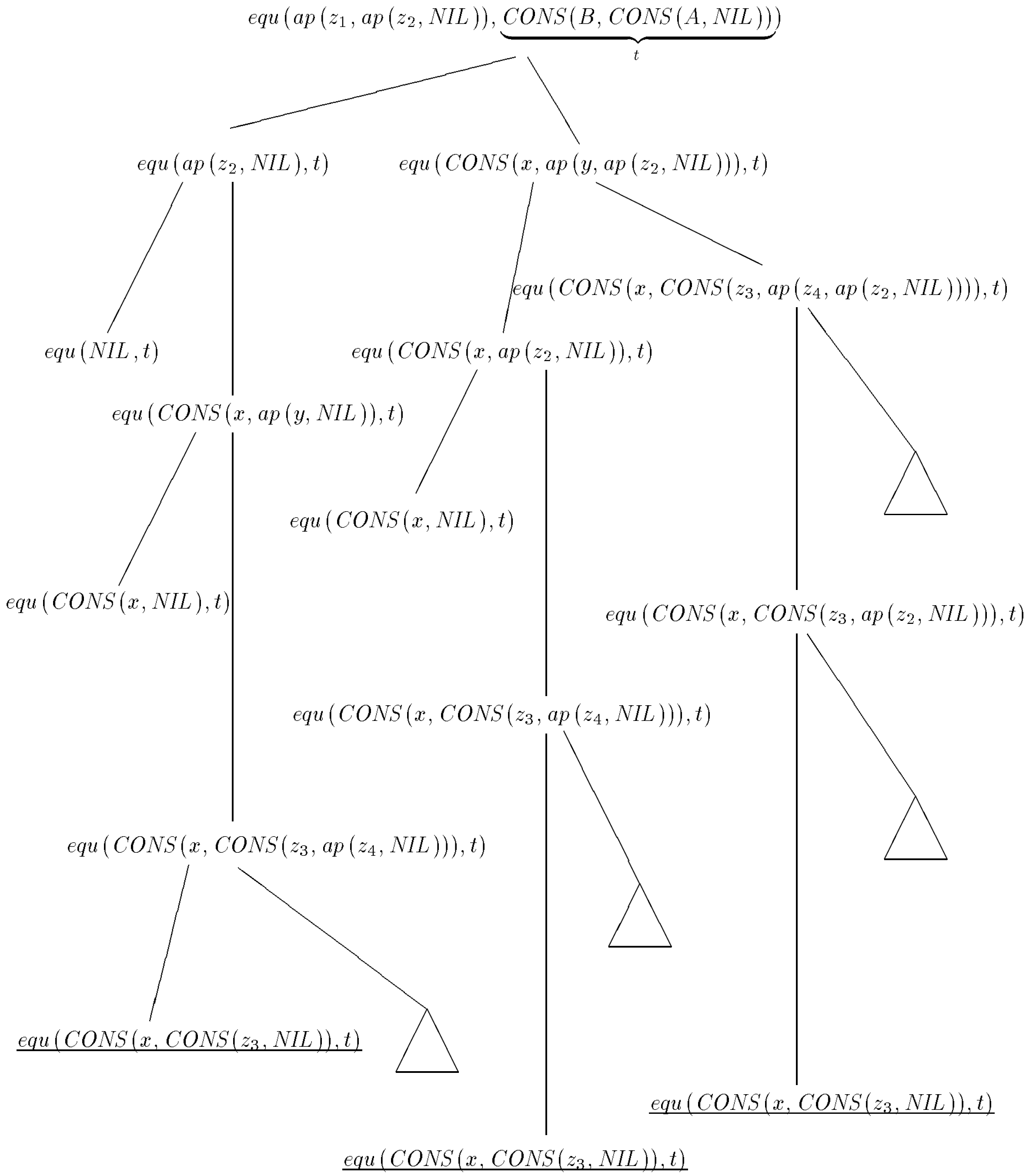

Abbildung 5.6: lo–Narrowingbaum für $E_{\mathcal{R}}$–Unifikation der Terme s und t

$\square$

Wir haben jetzt die Effektivität, mit der E–Unifikation betrieben werden kann, gegenüber dem Narrowing erhöht, indem wir den Berechnungsbaum verkleinert haben. Dies geschah durch Wahl einer Strategie zur Auswahl eines bestimmten Vorkommens, an dem genarrowt wird. Damit aber die lo–Narrowingrelation vollständig bleibt, mußten wir eine Einschränkung machen, indem wir nicht mehr beliebige kanonische Termersetzungssysteme zulassen, sondern nur noch ctn–trs.

5.7 Unification Driven LO-Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation

Im letzten Kalkül den wir hier vorstellen, wird die Effizienz, mit der E–Unifikation betrieben werden kann, gegenüber dem leftmost outermost Narrowing vom letzten Abschnitt, noch weiter erhöht. Die Unifikation, die an einem Blatt durchgeführt wird, soll nun als eine Sequenz von Dekompositionsschritten angesehen werden [MM82]. Außerdem wollen wir diese Dekompositionsschritte so früh wie möglich in einem Narrowingbaum anwenden und die Anwendung von einem occur check abhängig machen. Wir führen dazu neue Regeln ein, die wir Dekompositionsregeln nennen. Die Vereinigung der Regeln R eines TESes $\mathcal{R}$ und den Dekompositionsregeln nennen wir Erweiterung von $\mathcal{R}$. Definieren wir zunächst diese Begriffe.

Definition 5.32 (Dekompositionsregeln, Erweiterung)
Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs.

- Sei $\alpha \in \Delta^{(k)}$, mit $k \geq 0$. Die *Dekompositionsregel für* α ist der Form:

$$equ\,(\alpha(x_1, \ldots, x_k), \alpha(x_{k+1}, \ldots, x_{2k})) \rightarrow \alpha(equ\,(x_1, x_{k+1}), \ldots, equ\,(x_k, x_{2k})).$$

- Der *Dekompositionsteil von* $\mathcal{R}$, bezeichnet durch $\mathcal{R}(\Delta)$, ist das Tupel $(\hat{F}, \Delta, R(\Delta))$, wobei $\hat{F} = F \cup \{equ\}$ und $R(\Delta)$ ist die Menge aller Dekompositionsregeln für die Elemente von Δ.
- Die *Erweiterung von* $\mathcal{R}$, bezeichnet durch $\hat{\mathcal{R}}$, ist das ctn–trs $(\hat{F}, \Delta, \hat{R})$, wobei $\hat{R}$ die Menge $R \cup R(\Delta)$ ist.

□

Die unification driven lo–Narrowing Relation (kurz: ulo–Narrowing Relation) unterscheidet sich in folgenden drei Punkten von der lo–Narrowing Relation:

1. <u>*constructor consistency check:*</u>
Betrachten wir den Term $t = equ\,(NIL, CONS\,(append\,(z_1, z_2), append\,(z_3, z_4)))$. Das leftmost outermost Narrowing Vorkommen von t ist 21. Es ist aber auf den ersten Blick ersichtlich, daß es sinnlos ist, dort eine Regel anzuwenden, weil ein Term beginnend mit $CONS$ niemals gleich NIL werden kann. Verallgemeinert bedeutet das, daß die Rechnung abgebrochen werden kann, sobald die Wurzelsymbole der beiden Terme innerhalb eines equ Terms beide aus Δ und verschieden sind.

In der ulo–Narrowing Relation wird dies erziehlt, indem Narrowing nur am leftmost outermost Vorkommen von equ erlaubt wird. Dieses Vorkommen nennen wir *wichtiges Vorkommen* und bezeichnen es durch $impO(t)$.

2. occur check: Wenn $t/impO(t) = equ(z_i, t')$ oder $t/impO(t) = equ(t', z_i)$, wobei die Wurzel von t' mit einem Konstruktorsymbol α beschriftet ist, dann kann die Dekompositionsregel für α angewendet werden. Analog zur 'Variableneliminationsregel' bei der gewöhnlichen Unifikation, müssen wir hier einen occur check durchführen. Es darf aber lediglich überprüft werden, ob z_i im $(\Delta \cup \mathcal{V})$– Skelett von t auftritt, nicht im gesamten Term. Das $(\Delta \cup \mathcal{V})$– Skelett eine Terms beinhaltet nur Konstruktorterme.

3. Falls $t/impO(t) = equ(x, y)$, für zwei Variablen x, y, dann würde die leftmost outermost Narrowing Relation von $(\hat{\mathcal{R}})$, angewendet auf dem Term (t, φ) den Term $(\varphi_s(t[impO(t) \leftarrow s]), \varphi_s \circ \varphi)$ für alle $s \in T(\Delta)$ mit $\varphi_s[x/s, y/s]$ errechnen. Dies bedeutet, daß φ_s als MGU von x und y berechnet werden würde. Das ist natürlich falsch, denn der MGU von x und y ist $[x/z_k, y/z_k]$. Wir werden also die ulo– Narrowing Relation so definieren, daß dieser Fall berücksichtigt wird und der MGU korrekt berechnet wird.

Bevor wir die ulo–Narrowing Relation einführen, definieren wir einige zusätzliche Begriffe.

Definition 5.33 (wichtiges Vorkommen, Skelett, etc.) Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs, und sei $t \in T(\hat{F} \cup \Delta, \mathcal{V})$.

- Das *wichtige Vorkommen von t*, bezeichnet durch $impO(t)$, ist das Vorkommen

$$min_{lex}(\{u \in O(t) \mid t[u] = equ\}).$$

- Der Term t ist im *Bindemodus*, wenn $t[impO(t)1], t[impO(t)2] \in \mathcal{V}$.

- Das $(\Delta \cup \mathcal{V})$*-Skelett von t* ist die Menge

$$\{u \in O(t) \mid \text{es gibt kein } v \in O(t), v \leq u \text{ und } t[v] \in F\}$$

- Der *occur check von t ist erfolgreich*, wenn folgende Bedingungen erfüllt sind:
  1. t ist nicht im Bindemodus.
  2. Sei $t/impO(t) = equ(s, s')$. Entweder
     - $s \in \mathcal{V}, s' \notin \mathcal{V}$ und es gibt ein Vorkommen u in s', so daß $s'/u = s$, oder
     - $s \notin \mathcal{V}, s' \in \mathcal{V}$ und es gibt ein Vorkommen u in s, so daß $s/u = s'$.

□

Mit Hilfe dieser Begriffe können wir nun die ulo–Narrowing Relation definieren.

Definition 5.34 (ulo–Narrowing Relation)

Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs. Die *unification driven leftmost outermost Narrowing Relation von* $\hat{\mathcal{R}}$, bezeichnet durch $\overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}}$, ist wie folgt definiert: Für alle $t, s \in T(\hat{F} \cup \Delta, \mathcal{V})$ und $\psi, \psi' \in Sub(\mathcal{V}, \Delta) : (t, \psi) \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}} (s, \psi')$, wenn $t/impO(t) = equ(t_1, t_2)$, wobei $t_1, t_2 \in T(F \cup \Delta, \mathcal{V})$ und eine der folgenden fünf Bedingungen ist erfüllt:

1. $(t_1[\varepsilon], t_2[\varepsilon] \in \Delta$ und $t_1[\varepsilon] = t_2[\varepsilon])$ oder $(((t_1[\varepsilon] \in \Delta$ und $t_2[\varepsilon] \in \mathcal{V})$ oder $(t_1[\varepsilon] \in \mathcal{V}$ und $t_2[\varepsilon] \in \Delta))$ und der occur check von t ist nicht erfolgreich) und diese drei Bedingungen sind erfüllt:
   (a) $(equ(t_1, t_2), \varphi_\emptyset) \overset{lo}{\rightsquigarrow}_{\mathcal{R}(\Delta)} (t', \varphi')$.
   (b) $s = \varphi'(t[impO(t) \leftarrow t'])$.
   (c) $\psi' = \varphi' \circ \psi$.
2. $t_1, t_2 \in \mathcal{V}, t_1 \neq t_2$, und folgende drei Bedingungen sind erfüllt:
   Es sei $k = min\{i \mid z_i \in \mathcal{V}(\mathcal{V}(t) \cup \mathcal{D}(\psi) \cup \mathcal{I}(\psi))\}$:
   (a) $\varphi' = [t_1/z_k, t_2/z_k]$.
   (b) $s = \varphi'(t[impO(t) \leftarrow z_k])$.
   (c) $\psi' = \varphi' \circ \psi$.
3. $t_1, t_2 \in \mathcal{V}, t_1 = t_2$, und folgende zwei Bedingungen sind erfüllt:
   (a) $s = t[impO(t) \leftarrow t_1]$.
   (b) $\psi' = \varphi' \circ \psi$.
4. $t_1[\varepsilon] \in F$ und folgende drei Bedingungen sind erfüllt:
   (a) $(t_1, \varphi_\emptyset) \overset{lo}{\rightsquigarrow}_{\mathcal{R}} (t', \varphi')$.
   (b) $s = \varphi'(t[impO(t)1 \leftarrow t'])$.
   (c) $\psi' = \varphi' \circ \psi$.
5. $t_1[\varepsilon] \notin F$ und $t_2[\varepsilon] \in F$ und folgende drei Bedingungen sind erfüllt:
   (a) $(t_2, \varphi_\emptyset) \overset{lo}{\rightsquigarrow}_{\mathcal{R}} (t', \varphi')$.
   (b) $s = \varphi'(t[impO(t)2 \leftarrow t'])$.
   (c) $\psi' = \varphi' \circ \psi$.

□

Wenn eine Regeln $(l \rightarrow r) \in \hat{R}$ angewendet wird, i.e. in den Fällen 1, 4 und 5, dann schreiben wir $\overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}, l \rightarrow r}$. In den Fällen 2 und 3 schreiben wir $\overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}, bm}$, um anzuzeigen, daß der augenblickliche Term im Bindemodus ist. Das folgende Beispiel zeigt die Anwendung der ulo–Narrowing Relation. Ausgehend von der ulo–Narrowing Relation können wir nun den zugehörigen indizierten Kalkül angeben. Als Indexmenge wählen wir die Menge $\hat{R} \cup \{bm\}$, wobei das Symbol bm einen Berechnungsschritt der Form $\overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}},bm}$. Die Aufzählung $\hat{\pi}$ wird als totale Ordnung benutzt, wobei wir das Symbol bm als minimales Element der Indexmenge definieren.

Definition 5.35 (unification driven leftmost outermost Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation)

Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs mit Aufzählung π.

- Der *unification driven leftmost outermost Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation*, bezeichnet durch ulo-nar$_E\mathcal{I}\,(\mathcal{R})$, ist der indizierte Kalkül

  $$(T(\hat{F} \cup \Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Delta), T(\Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Delta), (\hat{R} \cup \{bm\}, <), (\overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}},\xi} \;\mid \xi \in \hat{R} \cup \{bm\})).$$

  wobei $<$ die totale Ordnung auf $\hat{R} \cup \{bm\}$ ist und wie folgt definiert wird: Für alle $\xi, \xi' \in \hat{R} \cup \{bm\}, \xi < \xi'$, wenn eine dieser beiden Bedingungen erfüllt ist:

  1. $\xi = bm$ und $\xi' \neq bm$.
  2. $\xi \neq bm, \xi' \neq bm$ und $\hat{\pi}(\xi) < \hat{\pi}(\xi')$.

- Der *unification driven leftmost outermost Narrowingbaum von $\mathcal{R}$ und $t \in T(\Omega, \mathcal{V}) \times Sub(\mathcal{V}, \Omega)$* (kurz: ulo–Narrowingbaum) ist der Berechnungsbaum

  $$B(\text{ulo-nar}_E\mathcal{I}\,(\mathcal{R}), t).$$

$\square$

Ganz analog zu den vorhergehenden Kalkülen für Narrowing, definieren wir hier den zugehörigen deterministischen ulo–Narrowingkalkül für $R_{\mathcal{R}}$–Unifikation.

Definition 5.36 (Deterministischer ulo–Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation.)

Sei $\mathcal{R}$ ein ctn–trs. Der *deterministische ulo–Narrowing Kalkül von $\mathcal{R}$* ist der deterministische Kalkül $det\,(\text{ulo-nar}_E\mathcal{I}\,(\mathcal{R}))$ von ulo-nar$_E\mathcal{I}\,(\mathcal{R})$.

$\square$

Beispiel 5.37 Sei $\mathcal{R}$ das ctn–trs aus Beispiel 5.31. Die Dekompositionsregeln der Konstruktorsymbole sind dann:

$$\begin{array}{rcl}
equ\,(A, A) & \to & A \\
equ\,(B, B) & \to & B \\
equ\,(C, C) & \to & C \\
equ\,(NIL, NIL) & \to & NIL \\
equ\,(CONS\,(z_1, z_2), CONS\,(z_3, z_4)) & \to & CONS\,(equ\,(z_1, z_3), equ\,(z_2, z_4))
\end{array}$$

Die Abbildungen 5.7 und 5.8 zeigt den ulo–Narrowingbaum von $\mathcal{R}$ und den Term $equ(s,t)$ mit $s = ap(z_1, ap(z_2, NIL))$ und $t = CONS(B, CONS(A, NIL))$. Es gilt dabei:

$$\begin{array}{lcl} p & = & CONS(B, CONS(A, equ(CONS(z_5, ap(z_6, ap(z_2, NIL)), NIL))) \\ q & = & CONS(B, CONS(A, equ(CONS(z_5, ap(z_6, NIL), NIL))) \end{array}$$

An den Termen p und q hört die Rechnung aufgrund des constructor consistency check's auf.

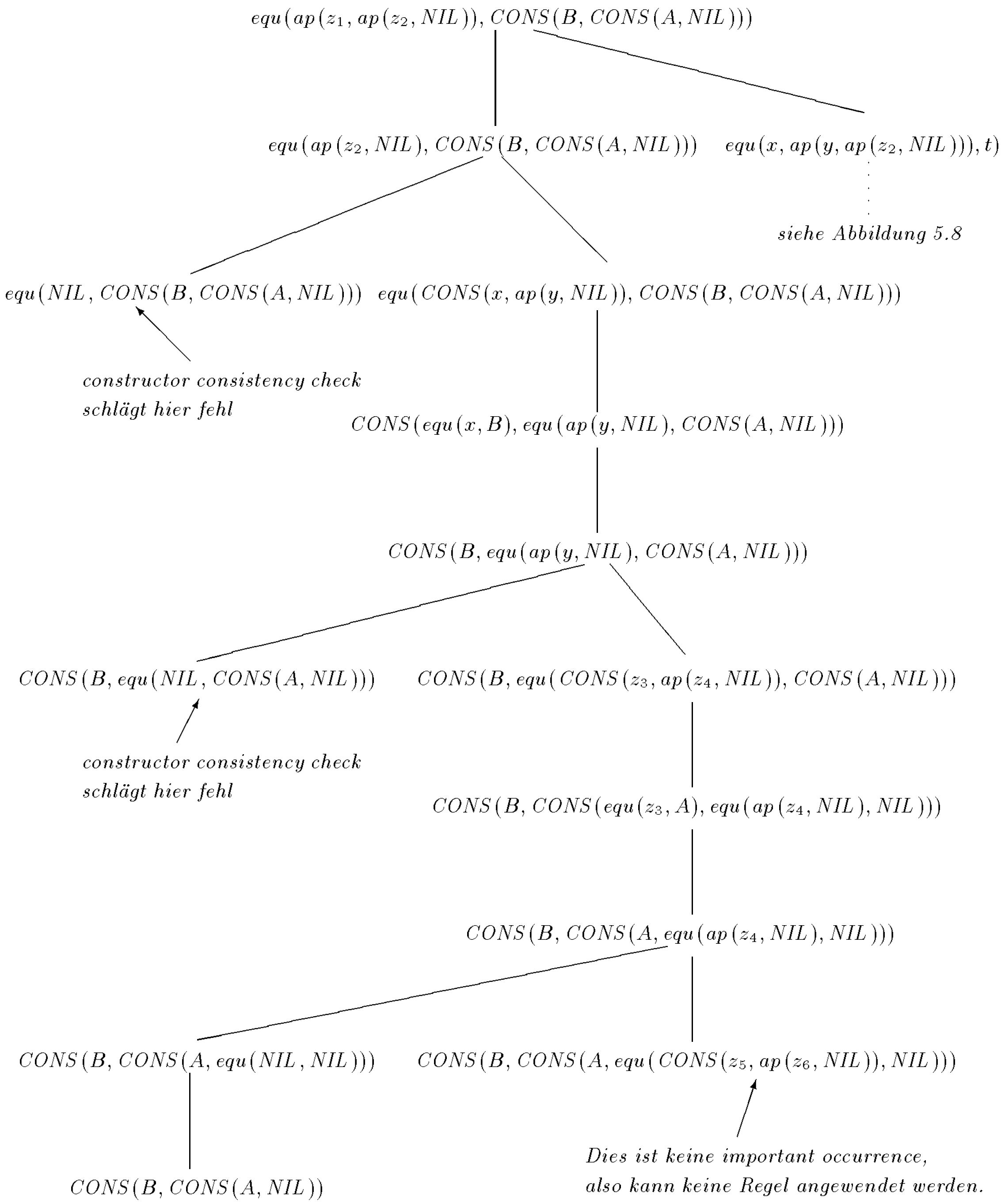

Abbildung 5.7: ulo–Narrowingbaum für $equ(ap(z_1, ap(z_2, NIL)), t)$

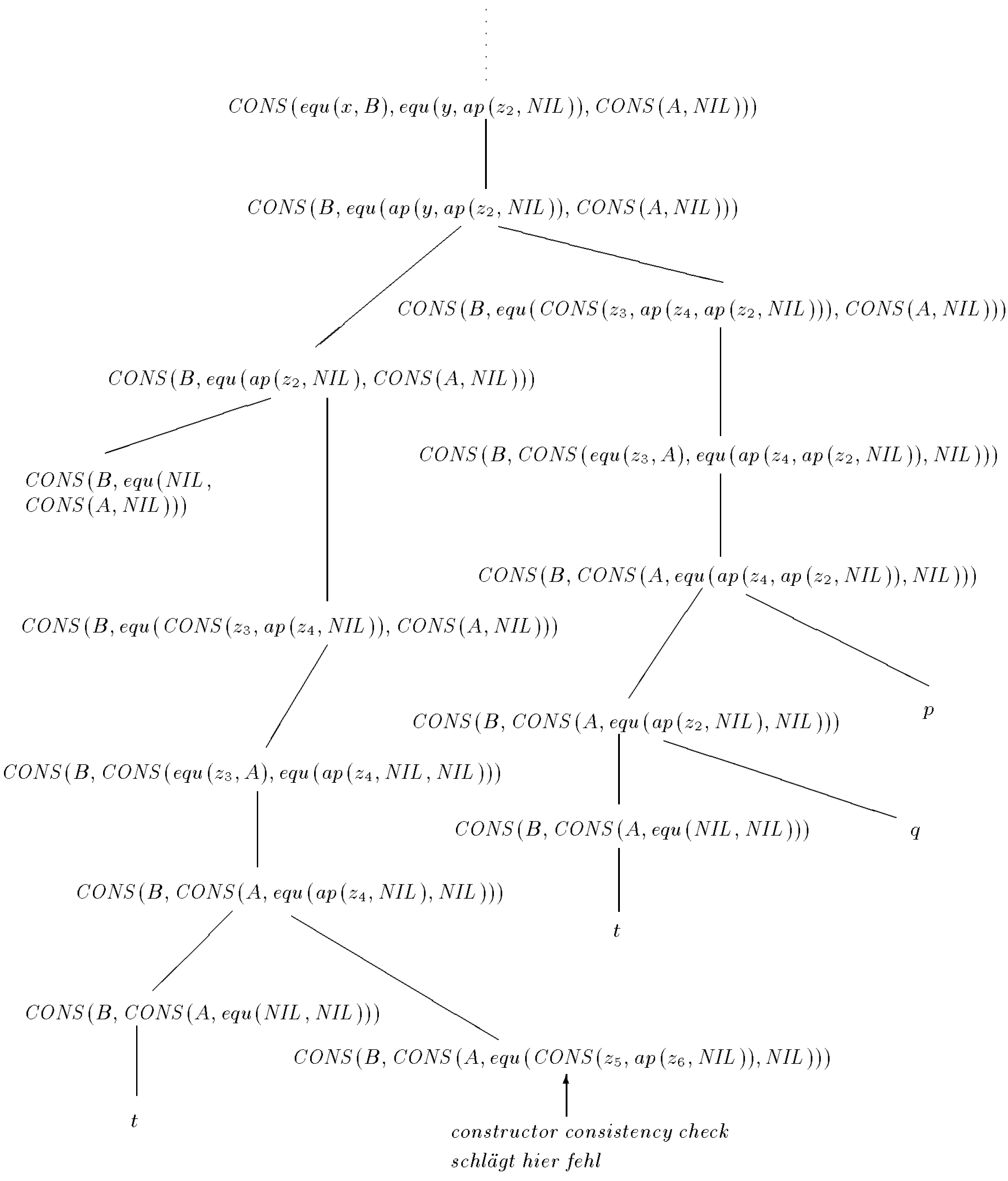

Abbildung 5.8: rechter Ast des ulo–Narrowingbaums für $equ(ap(z_1, ap(z_2, NIL)), t)$

□

Im vorhergenden Beispiel konnten wir sehen, daß der unendliche lo–Narrowingbaum aus Beispiel 5.31 im unification driven lo–Narrowing Kalkül zu einem endlichen Baum wird. Allgemein ist dies natürlich nicht immer der Fall, sonst wäre das $E_{\mathcal{R}}$–Unifikationsproblem für ctn–trs entscheidbar. Wir wollen dazu ein letztes Beispiel in diesem Kapitel betrachten, welches veranschaulicht, daß nur minimale Änderung in der Aufzählungsfunktion π, bzw. $\hat{\pi}$ den ulo–Narrowingbaum unendlich werden lassen.

Beispiel 5.38 Sei $\mathcal{R}$ das ctn–trs aus Beispiel 5.31. Die Aufzählung $\hat{\pi}$ sei wie folgt definiert (wir werden nur die Dekompositionsregel für *CONS* und *NIL* benutzen, deshalb zählen wir hier nur diese auf):

$$\begin{array}{rcll} ap\,(NIL\,, x) & \to & x & (1)\\ ap\,(CONS\,(x, y), z) & \to & CONS\,(x, ap\,(y, z)) & (2)\\ equ\,(CONS\,(z_1, z_2), CONS\,(z_3, z_4)) & \to & CONS\,(equ\,(z_1, z_3), equ\,(z_2, z_4)) & (3)\\ equ\,(NIL\,, NIL\,) & \to & NIL & (4) \end{array}$$

Es sollen nun $E_{\mathcal{R}}$–Unifikatoren für die beiden Terme $t = ap\,(x, NIL\,)$ und $s = ap\,(y, x)$ berechnet werden. Eine Lösung wird durch folgende ulo–Berechnung beschrieben:

$$\begin{array}{ll} (equ\,(ap\,(x, NIL\,), ap\,(y, x)), \varphi_{\emptyset}) & \overset{u}{\leadsto}_{\hat{\mathcal{R}},1,(2)}\\ (equ\,(NIL\,, ap\,(y, NIL\,)), \varphi_1 = [x/NIL\,]) & \overset{u}{\leadsto}_{\hat{\mathcal{R}},2,(2)}\\ (equ\,(NIL\,, NIL\,), \varphi_2 = [y/NIL\,] \circ \varphi_1) & \overset{u}{\leadsto}_{\hat{\mathcal{R}},\varepsilon,(4)}\\ (NIL\,, \varphi_2) & \end{array}$$

Diese Lösung liegt allerdings auf dem rightmost path im ulo–Narrowingbaum. Betrachten wir andererseits den leftmost path und damit den path, den der deterministische ulo-Narrowingkalkül bei seinem depth–first left–to–right tree traversal durchgeht:

$$\begin{array}{ll} (equ\,(ap\,(x, NIL\,), ap\,(y, x)), \psi_{\emptyset}) & \overset{u}{\leadsto}_{\hat{\mathcal{R}},1,(1)}\\ (equ\,(CONS\,(z_1, ap\,(z_2, NIL\,)), ap\,(y, CONS\,(z_1, z_2))), \psi_1 = [x/CONS\,(z_1, z_2)]) & \overset{u}{\leadsto}_{\hat{\mathcal{R}},2,(1)}\\ (equ\,(CONS\,(z_1, ap\,(z_2, NIL\,)), & \\ \quad CONS\,(z_3, ap\,(z_4, CONS\,(z_1, z_2)))), \psi_2 = [y/CONS\,(z_3, z_4)]) & \overset{u}{\leadsto}_{\hat{\mathcal{R}},\varepsilon,(3)}\\ (CONS\,(equ\,(z_1, z_3), equ\,(ap\,(z_2, NIL\,), ap\,(z_4, CONS\,(z_1, z_2)))), \psi_2) & \overset{u}{\leadsto}_{\hat{\mathcal{R}},1,bm}\\ (CONS\,(z_5, equ\,(ap\,(z_2, NIL\,), ap\,(z_4, CONS\,(z_5, z_2)))), \psi_3 = [z_1/z_5, z_3/z_5] \circ \psi_2) & \overset{u}{\leadsto}_{\hat{\mathcal{R}},21,(1)}\\ (CONS\,(z_5, equ\,(CONS\,(z_6, ap\,(z_7, NIL\,)), & \\ \quad ap\,(z_4, CONS\,(z_5, CONS\,(z_6, z_7))))), \psi_4 = [z_2/CONS\,(z_6, z_7] \circ \psi_3) & \overset{u}{\leadsto}_{\hat{\mathcal{R}},22,(1)}\\ (CONS\,(z_5, equ\,(CONS\,z_6, ap\,(z_7, NIL\,)), & \\ \quad CONS\,(z_8, ap\,(z_9, CONS\,(z_5, CONS\,(z_6, z_7)))))), \psi_5 = [z_4/CONS\,(z_8, z_9] \circ \psi_4) & \overset{u}{\leadsto}_{\hat{\mathcal{R}},2,(3)}\\ (CONS\,(equ\,(z_6, z_8), equ\,(ap\,(z_7, NIL\,), ap\,(z_9, CONS\,(z_5, CONS\,(z_6, z_7))))), \psi_5) & \overset{u}{\leadsto}_{\hat{\mathcal{R}},21,bm}\\ (CONS\,(z_5, CONS\,(z_{10}, equ\,(ap\,(z_7, NIL\,), & \\ \quad ap\,(z_9, CONS\,(z_5, CONS\,(z_6, z_7))))), \psi_6 = [z_6/z_{10}, z_8/z_{10}] \circ \psi_5) & \\ \vdots & \end{array}$$

Es ist offentsichtlich, daß es sich hier um einen unendlichen Ast im ulo–Narrowingbaum handelt. An allen Knoten, an denen die Regel (1) angewendet werden kann, kann natürlich auch Regel (2) angewendet werden. Nach gegebenerfalls mehrfacher Anwendung von Regel (2) würden dann $E_{\mathcal{R}}$–Unifikatoren der Form $CONS(z_i, CONS(z_j, \ldots))$ gefunden werden. Die Substitution ψ_5 in der oben aufgezeichneten Berechnung lautet:

$$\psi_5 = [x/CONS(z_5, CONS(z_{10}, z_7)), y/CONS(z_5, CONS(z_{10}, z_9))]$$

Es ist zu bemerken, daß allgemein nach jedem vierten Berechnungsschritt eine Substitution $\psi = [x/CONS(z_5, CONS(z_{10}, CONS(\ldots, CONS(z_{5n}, z_{5n-2})))), [y/CONS(z_5, CONS(z_{10}, CONS(\ldots, CONS(z_{5n}, z_{5n-1}))))$ erhalten wird. Dieser Ast bricht also nie ab; der UU–Algorithmus für den deterministischen ulo–Narrowingkalkül findet keine Lösung und terminiert nicht.

□

Kapitel 6

Effizienzmaße

In Kapitel 3 wurden formale Kalküle definiert. Ein Kalkül ist ein Tupel bestehend aus einer Menge D von Zwischenergebnissen, der Menge D_{success} der Endergebnisse und der Rechenrelation $\vdash$, die über D definiert ist. In diesem Kapitel wollen wir Effizienzmaße definieren, die es erlauben zwei Kalküle $\mathcal{C}$ und $\mathcal{C}'$ bezüglich ihrer Effizienz zum Berechnen von Endergebnissen zu vergleichen. Die Effizienz eines Kalküls soll dabei stets durch eine natürliche Zahl dargestellt werden, bzw. durch das Symbol ∞, falls kein Endergebnis errechnet werden kann; der Vergleich von Effizienzen beruht auf der üblichen totalen Ordnung „kleiner als" auf natürlichen Zahlen. Anstelle dieses relativ einfachen Maßbereichs $(\mathbb{N}, <)$ ließen sich auch allgemeinere Maßbereiche $(N, \leq)$ betrachten, wobei N eine beliebige Menge und $\leq$ eine *partielle* Ordnung auf N ist. Diese Betrachtung konnte hier aus Zeitgründen nicht vorgenommen werden.

Allgemein läßt sich der Maßbereich als eine *abstrakte Interpretation* auffassen; hier in dem Sinne, daß ein indizierter Kalkül $\mathcal{I}$ und ein Zwischenergebnis d über einem abstrakten Grundbereich, nämlich der Menge der natürlichen Zahlen, interpretiert wird. Der Begriff der Abstrakten Interpretation geht auf [CC79] zurück, die Programme über einem abstrakten Grundbereich interpretiert haben.

Der indizierte Kalkül partitioniert die Rechenrelation $\vdash$ in die Relationen $\vdash_j$ mit $j \in J$. Es wird dadurch eine eindeutige Anordnung der Knoten im Berechnungsbaum festgelegt. Der jeweils zugehörige deterministische Kalkül definiert einen depth–first left–to–right tree traversal durch diesen Baum. Ein mögliches Effizienzmaß ist daher die Anzahl der Berechnungsschritte des deterministischen Kalküls, die benötigt wird, um die erste Lösung zu finden; wir nennen dieses Maß *single–success Effizienz.* Als Erweiterung dazu kann man die Schrittanzahl bis zum Finden der n–ten Lösung berechnen (*multi–success Effizienz*). Ein anderes Maß entsteht, wenn die Anzahl der Lösungen in Abhängigkeit von der Höhe im Baum betrachtet wird. Wir nennen diese Effizienzmaß *success density Effizienz.*

Der Vergleich der eben beschriebenen Effizienzmaße ist für Kalküle, die Berechnungsbäume mit vielen unendlichen Ästen erzeugen, natürlich problematisch. Der deterministische Kalkül erzeugt keine weiteren Lösungen, wenn er in einen unendlichen Ast gerät. Es ist dadurch kein Vergleich von Anzahlen von Berechnungsschritten mehr möglich.

Ein anderes Effizienzmaß könnte deshalb von der Anzahl der unendlichen Äste in einem Berechnungsbaum abhängen. Dieses Maß nennen wir *infinite path Effizienz*.

6.1 Single Success Effizienz

Die single success Effizienz soll ein Maß dafür sein, wie früh, innerhalb einer Rechnung eines bestimmten Kalküls, das *erste* Endergebnis berechnet wird. Endergebnisse sind laut der Definition eines Kalküls die Elemente der Menge D_{success}. Wir wollen also die Zahl der Berechnungsschritte eines bestimmten Kalküls $\mathcal{C}$, ausgehend von einem Kalkulandum d, bis zum ersten Erreichen eines Zwischenergebnisses $d' \in D_{\text{success}}$ zählen.

Bei der single success Effizienz handelt es sich um eine Funktion, die jedem Kalkulandum d dieses Kalküls eine natürliche Zahl zuordnet. Falls kein Endergebnis erreicht werden kann, weil die Berechnung des Kalküls in einen unendlichen Ast geraten ist oder kein Endergebnis existiert, ordnen wir der Effizienzfunktion das Symbol ∞ zu.

Definition 6.1 (single success Effizienz)
Sei $\mathcal{C} = (D, D_{\text{success}}, \vdash)$ ein Kalkül und für alle $d \in D$ sei $E(d)$ die Menge

$$\{k \mid k \geq 0 \text{ und } Suc_k\,(d, \vdash) \cap D_{\text{success}} \neq \emptyset\}.$$

der Schrittzahlen k, die auf ein Endergebnis stoßen.

Die *single success Effizienz von* $\mathcal{C}$, bezeichnet durch $\eta(\mathcal{C})$, ist die Funktion $\eta(\mathcal{C}) : D \to (\mathbb{N} \cup \{\infty\})$, so daß für alle $d \in D$ gilt:

$$(\eta(\mathcal{C}))\,(d) = \begin{cases} \min\,(E(d)), & \text{falls } E(d) \neq \emptyset \\ \infty, & \text{sonst} \end{cases}$$

□

Die Definition der single success Effizienz benutzt die Menge $Suc_k\,(d, \vdash)$, welche die Zwischenergebnisse, die sich in k Berechnungsschritten aus d durch $\vdash$ berechnen lassen, enthält. Im Allgemeinen wird diese Menge mehr als ein Endergebnis enthalten; angesichts des Berechnungsbaumes von $\mathcal{C}$ und d, enthält diese Menge nämlich für ein k jeweils die Zwischenergebnisse, die auf dem k-ten Niveau dieses Berechnungsbaumes stehen. Allen Endergebnissen auf einem bestimmten Niveau des Baumes ist demnach die gleiche single success Effizienz zugeordnet. In den meisten Fällen wird dies natürlich ein viel zu grobes Maß sein, denn es muß bei der Auswertung des Berechnungsbaumes ein bestimmter Baumdurchlauf gewählt werden, der bei der single success Effizienz nicht berücksichtigt wird. Wir werden deshalb sowohl in dem folgenden Beispiel, als auch in den in Kapitel 7 folgenden Vergleichen, die single success Effizienz für einen *deterministischen Kalkül* eines indizierten Kalküls berechnen. Man beachte, daß Definition 6.1 anwendbar ist, weil der deterministische Kalkül eines indizierten Kalküls selbst per definitionem ein Kalkül ist.

Es sei hier noch einmal darauf hingewiesen, daß die Wahl des depth–first left–to–right tree traversals im deterministischen Kalkül keine Einschränkung der Allgemeinheit eines Baumdurchlaufs ist, denn die Anordnung der Knoten des Berechnungsbaumes eines indizierten Kalküls $\mathcal{I}$ hängt allein von der Indexmenge J und der auf ihr definierten totalen Ordnung $<$ ab. Wollte man beispielsweise einen depth–first right–to–left tree traversal festlegen, so müßte man lediglich die totale Ordnung $<$ umkehren und könnte dann den deterministischen Kalkül für diesen Kalkül benutzen. Allerdings ist die Wahl des depth–first Vorgehens durch die Definition des deterministischen Kalküls festgelegt, so daß beispielsweise ein breadth–first Vorgehen nicht möglich ist.

Beispiel 6.2 Sei $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ der indizierte Kalkül aus Beispiel 3.6 und wählen wir eine neue Endergebnismenge $\widehat{D}_{\text{success}}$ wie folgt:

$$\widehat{D}_{\text{success}} = \{d \in D \mid \mathit{Suc}\,(d) = \emptyset\}$$

$\widehat{D}_{\text{success}}$ enthält demnach alle Zwischenergebnisse, die in Normalform sind.
Es sei

- $\mathcal{I}' = (D, \widehat{D}_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ und
- $d = (a * (a * (b + c)))$ ein Kalkulandum.

Der Berechnungsbaum $B(\mathcal{I}', d)$ von $\mathcal{I}'$ und d ist nach Lemma 3.9 identisch mit dem in Abbildung 3.1 zu sehenden Baum. Sei $\mathcal{C} = \mathit{det}\,(\mathcal{I}')$ der deterministische Kalkül von $\mathcal{I}'$. Es gilt dann:

$$(\eta(\mathcal{C}))\,(d) = 2$$

denn

$$\begin{array}{ll} & ((a * (a * (b + c))), \{(\text{assoc}, \varepsilon), (\text{distr}, 2)\}) \\ \models & ((a * (a * (b + c))), \{(\text{distr}, 2)\}) \\ & (((a * a) * (b + c)), \{(2, \varepsilon)\}) \\ \models & ((a * (a * (b + c))), \{(\text{distr}, 2)\}) \\ & (((a * a) * (b + c)), \emptyset) \\ & (\underbrace{(((a * a) * b) + ((a * a) * c))}_{\in\, \widehat{D}_{\text{success}}}, \emptyset) \end{array}$$

□

Es stellt sich nun die Frage, welche Bedingungen an zwei Kalküle $\mathcal{C}$ und $\mathcal{C}'$ gestellt werden müssen, um sicherzustellen, daß ihre single success Effizienzen vergleichbar sind. Würde man beispielsweise fordern, daß die Mengen der Zwischenergebnisse der beiden Kalküle identisch sind, so könnte man später nicht den Narrowing Kalkül und den ulo–Narrowing Kalkül miteinander vergleichen. Im Narrowing Kalkül gilt nämlich $D = \text{Equ}\,(T(\Omega, \mathcal{V})) \times \mathit{Sub}\,(\mathcal{V}, \Omega)$ und im ulo–Narrowing Kalkül $D' = T(\hat{F} \cup \Delta, \mathcal{V}) \times \mathit{Sub}\,(\mathcal{V}, \Omega)$. Die hier definierten Effizienzmaße sollen aber für genau solche Vergleiche benutzt werden, so daß die

$$
\begin{tikzcd}
\mathcal{C}: & t \arrow[r, phantom, "\vdash"] & \bar{t} \\
\mathcal{C}': & t' \arrow[u, "h"] \arrow[r, phantom, "\vdash'^{*}"] & \bar{t}' \arrow[u, "h"']
\end{tikzcd}
$$

Abbildung 6.1: $\mathcal{C}$ ist η–vergleichbar durch h mit $\mathcal{C}'$.

Forderung auf Gleichheit der Mengen der Zwischenergebnisse zu stark ist. Um eine Aussage über die Vergleichbarkeit bezüglich der Effizienz machen zu können, muß demnach die Semantik der Kalküle betrachtet werden.

Damit es möglich ist, die Kalkulandi der beiden Kalküle zu vergleichen, genügt das Vorhandensein einer Abbildung $h : D' \to D$, wobei D und D' die Mengen der Zwischenergebnisse der beiden Kalküle sind. Diese Abbildung beschreibt die semantische Beziehung zwischen D' und D, und stellt sicher, daß zur Berechnung der Effizienz von jedem Kalkulandum $d \in D'$ ausgegangen werden kann, weil es ein entsprechendes Zwischenergebnis $h(d) \in D$ gibt. Da die Effizienz vom Finden von Endergebnissen abhängt, muß außerdem sichergestellt werden, daß alle Endergebnisse von $\mathcal{C}'$ auch Endergebnisse von $\mathcal{C}$ unter h sind; d.h. $\hat{h}(D'_{\text{success}}) \subseteq D_{\text{success}}$, wobei $\hat{h}$ die natürliche Erweiterung von h auf Mengen ist. Wenn zwei Kalküle diese Eigenschaften erfüllen, dann nennen wir sie η–vergleichbar. Da die oben genannten Bedingungen nur von D, D_{success} und der Abbildung h abhängen, kann der Begriff der η–Vergleichbarkeit sowohl für Kalküle, als auch für indizierte Kalküle angewendet werden.

Definition 6.3 (η–vergleichbar)

Es seien $\mathcal{C}, \mathcal{C}'$ [indizierte][1] Kalküle mit Zwischenergebnismengen D bzw. D', und Endergebnismengen D_{success} bzw. D'_{success}. Sei $h : D' \to D$ eine Abbildung. Der [indizierte] Kalkül $\mathcal{C}$ *ist η–vergleichbar durch h mit dem [indizierten] Kalkül* $\mathcal{C}'$, falls gilt: $\hat{h}(D'_{\text{success}}) \subseteq D_{\text{success}}$; wobei $\hat{h}$ die Fortsetzung von h auf Mengen ist.

□

Es seien $t, \bar{t} \in D$ und $t', \bar{t}' \in D'$ wobei D und D' die Zwischenergebnismengen zweier Kalküle $\mathcal{C}$ und $\mathcal{C}'$ sind. Betrachten wir die Bedeutung des Begriffes η–vergleichbar durch h anhand der Abbildung 6.1.

Insbesondere ist zu bemerken, daß h *nicht* injektiv sein muß, denn es kann durchaus $t' = \bar{t}'$ gelten. Unsere Definition der η–Vergleichbarkeit zweier Kalküle ist ein Versuch möglichst schwache Bedingungen an die Kalküle zu stellen und gleichzeitig zu garantieren, daß die Semantik der Kalküle ähnlich ist. Von der Abbildung h lediglich zu fordern,

[1] die eckigen Klammern sind hier als Option zu verstehen. Wird die Option in einer Definition gewählt, müssen allerdings *alle* Vorkommen von Optionen in dieser Definition gewählt werden

daß $\hat{h}(D_{\text{success}}) \subseteq D'_{\text{success}}$ gilt, ist gerechtfertigt, weil dadurch sichergestellt wird, daß zumindest die Endergebnisse der Kalküle in einer gewissen Beziehung zueinander stehen; dies bedeutet mit anderen Worten, daß die Kalküle etwas Vergleichbares berechnen. Es ist durchaus denkbar viel strengere Bedingungen an die Abbildung h zu stellen, beispielsweise zu fordern, daß $\hat{h}(D) = D$; i.e. h ist surjektiv. Trotzdem ist die Definition der η–Vergleichbarkeit mit Vorsicht zu genießen, denn es läßt sich beispielsweise ein simpler Kalül $\mathcal{C} = (\mathbb{N}, \mathbb{N}, \emptyset)$ definieren, der in 0 Schritten jede natürliche Zahl als Endergebnis berechnet. Man könnte dann zu jedem Kalül $\mathcal{C}' = (D', D'_{\text{success}}, \vdash')$ mit abzählbarer Endergebnismenge D'_{success} eine Funktion $h : D' \to D$ angeben, so daß $\mathcal{C}$ η–vergleichbar durch h mit $\mathcal{C}'$ ist. Der völlig nutzlose Kalkül $\mathcal{C}$ wäre dann immer effizienter als $\mathcal{C}'$, sobald $\mathcal{C}'$ nur einen einzigen Schritt zur Berechnung eines Endergebnisses benötigt. Intuitiv könnte man daher folgende informelle Zusatzbedingung für die Funktion h angeben: h soll lediglich Umformungen *syntaktischer* Art durchführen und sollte keine „Berechnungen“ druchführen.

In Kapitel 7 wollen wir zum Beispiel den lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation mit dem ulo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation bezüglich der single success Effizienz vergleichen. Wir werden dann also eine Abbildung h benötigen, welche die Menge der Zwischenergebnisse des lo–Narrowing Kalküls auf Menge der Zwischenergebnisse des ulo–Narrowing Kalküls abbildet. In der Tat wäre aber eine surjektive Abbildung h wesentlich komplizierter und aufwendiger, als ein h gemäß unserer Definition der η–Vergleichbarkeit. Intuitiv ist jedoch klar, daß ein Effizienzvergleich zwischen lo–Narrowing und ulo–Narrowing gerechtfertigt ist, die beiden Kalküle also η–vergleichbar sind.

Definition 6.4 (single success effizienter)
Es seien

- $\mathcal{C} = (D, D_{\text{success}}, \vdash)$.
- $\mathcal{C}' = (D', D'_{\text{success}}, \vdash')$.
- $h : D' \to D$ eine Abbildung;

Der Kalkül $\mathcal{C}$ ist *single success effizienter* als der Kalkül $\mathcal{C}'$, wenn die folgenden beiden Bedingungen gelten:

1. $\mathcal{C}$ ist η–vergleichbar durch h mit $\mathcal{C}'$;
2. Es gibt ein $d \in D'$, so daß
$$(\eta(\mathcal{C}))\,(h(d)) <_{\mathbb{N}} (\eta(\mathcal{C}'))\,(d)$$
und es gibt kein $d' \in D'$, so daß
$$(\eta(\mathcal{C}))\,(h(d')) >_{\mathbb{N}} (\eta(\mathcal{C}'))\,(d').$$

□

Beispiele für die Anwendung dieses Begriffes werden wir im nächsten Kapitel kennenlernen, wenn wir den

- Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation mit dem
- lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation

vergleichen, oder den

- lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation mit dem
- ulo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation

vergleichen.

6.2 Multi Success Effizienz

Die Menge D_{success} der Endergebnisse eines Kalküls $\mathcal{C}$ wird im allgemeinen mehr als ein Element enthalten, so daß die single success Effizienz, die nur die Anzahl der Berechnungsschritte bis zum Finden des ersten Endergebnisses zählt, nicht ausreicht, um eine repräsentative Aussage über die Effizienz zweier Kalküle zu machen. Ein deterministischer Kalkül $\mathcal{C}_1$ kann zum Beispiel nach sehr wenigen Berechnungsschritten ein erstes Endergebnis berechnen, danach aber in einen unendlichen Ast geraten und überhaupt keine Lösung mehr berechnen. Dieser Kalkül wäre aber single success effizienter als ein anderer deterministischer Kalkül $\mathcal{C}_2$, der zwar zur Berechnung des ersten Endergebnisses mehr Schritte als $\mathcal{C}_1$ benötigt, aber noch mehr Endergebnisse findet. Wenn also das Finden *mehrerer* Endergebnisse (oder: successes) entscheidend ist, dann benötigen wir ein allgemeineres Effizienzmaß. Die *k–success Effizienz* ist ein solches Maß. Sie hängt — ausgehend von einem Kalkulandum d — von der Anzahl der Berechnungsschritte eines Kalküls $\mathcal{C}$ ab, die benötigt wird um die *ersten k verschiedenen* Endergebnisse zu berechnen.

Was bedeutet aber „verschieden“ beispielsweise im Kontext des Narrowing Kalküls? Falls einfache syntaktische Verschiedenheit gemeint ist, dann wären dort zwei Zwischenergebnisse $d = (CONS(z_1, NIL), \underbrace{[x/CONS(z_1, NIL)]}_{\varphi})$ und $d' = (CONS(z_7, NIL), \underbrace{[x/CONS(z_7, NIL)]}_{\varphi'})$ verschieden, obwohl die Substitution φ' lediglich durch Variablenumbenennung aus φ hervorgeht und damit ihre Bedeutung für das Narrowing identisch ist. Auch für andere Kalküle kann es durchaus der Fall sein, daß syntaktische Verschiedenheit *nicht* semantische Verschiedenheit in dem betreffenden Kalkül bedeutet. Es ist daher sinnvoll in die Definition der k–success Effizienz von einem deterministischen Kalkül $\mathcal{C}$ noch eine Äquivalenzrelation $\sim$ mit aufzunehmen, die über der Menge der Zwischenergebnisse D des Kalküls $\mathcal{C}$ definiert ist. Wenn für zwei Zwischenergebnisse d und d' gilt $d \not\sim d'$, dann sind sie als verschieden zu verstehen.

Sei $\mathcal{C}$ ein Kalkül und d ein Kalkulandum. Bevor wir die k–success Effizienz definieren, wollen wir jetzt eine Menge definieren, die alle Endergebnisse von $\mathcal{C}$ enthält, die sich in höchstens j Schritten aus d berechnen lassen und die in Bezug auf die Äquivalenzrelation $\sim$ verschieden sind.

Definition 6.5 (Menge der j–berechenbaren Endergebnisse)

Es sei $\mathcal{C} = (D, D_{\text{success}}, \vdash)$ ein Kalkül, $d \in D$ ein Kalkulandum, $j \in \mathbb{N}$ und $\sim \subseteq D \times D$ eine Äquivalenzrelation. Die *Menge der aus d j-berechenbaren und $\sim$–verschiedenen Endergebnisse*, bezeichnet durch $E(j, d, \mathcal{C}, \sim)$ ist die Menge

$$S(j, d, \mathcal{C})/_{\sim}, \text{ wobei } S(j, d, \mathcal{C}) = \bigcup_{i \in [j]} (Suc_i\,(d, \vdash) \cap D'_{\text{success}})$$

□

Definition 6.6 (k–success Effizienz)

Sei $\mathcal{C} = (D, D_{\text{success}}, \vdash)$ ein Kalkül, $\sim$ eine Äquivalenzrelation auf D, d.h. $\sim \subseteq D \times D$ und k eine ganze Zahl mit $k \geq 2$.

Die *k–success Effizienz von $\mathcal{C}$ und $\sim$*, bezeichnet durch $\eta_k(\mathcal{C}, \sim)$, ist eine Funktion $\eta_k(\mathcal{C}, \sim) : D \to (\mathbb{N} \cup \{\infty\})$, so daß für alle $d \in D$ gilt:

$$(\eta_k(\mathcal{C}, \sim))\,(d) = \begin{cases} \min\{j \mid |E(j, d, \mathcal{C}, \sim)| = k\}, & \text{falls ein } j \text{ existiert mit } |E(j, d, \mathcal{C}, \sim)| = k \\ \infty, & \text{sonst} \end{cases}$$

□

Bei dem nun folgenden Beispiel, sowie bei den Vergleichen die in Kapitel 7 folgen werden, wird die multi–success Effizienz für deterministische Kalküle verwendet. Ein deterministischer Kalkül ist laut unserer Definition 3.12 zu einem indizierten Kalül definiert. Es reicht aber vollkommen aus, eine Äquivalenzrelation $\sim'$ über der Menge der Zwischenergebnisse des zugrundeliegenden indizierten Kalküls zu betrachten. Durch die folgende, einfache Konstruktion kann aus $\sim'$ eine Äquivalenzrelation $\sim$ über der Menge der Zwischenergebnissen des deterministischen Kalküls zu $\mathcal{I}$ erhalten werden.

Sei $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ ein indizierter Kalkül, $\sim \subseteq D \times D$ eine Äquivalenzrelation und $d \in D$ ein Kalkulandum. Wir definieren eine neue Äuqivalenzrelation $\sim'$ über der Menge D_0^+, mit $D_0 = \{(d, M) \mid d \in D, M \subseteq I(d)\}$, wie folgt

Für alle $\Theta, \Theta' \in D$ mit $\Theta \sim \Theta'$ definiere

$$w(\Theta, I(\Theta)) \sim' w'(\Theta', I(\Theta')),$$

wobei $w, w' \in D_0^*$ und es natürliche Zahlen i und j gibt mit $w(\Theta, I(\Theta)) \in Suk_i\,(d, I(d))$ und $w'(\Theta', I(\Theta')) \in Suk_j\,(d, I(d))$.

Beispiel 6.7 (für k–success Effizienz)

Als Kalkül $\mathcal{C}$ betrachten wir den deterministischen Kalkül zum indizierten Kalkül $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$, den wir nur informell durch Skizzierung eines Berechnungsbaumes von $\mathcal{I}$ und dem Kalkulandum $d = (append(x, append(y, NIL)), \varphi_\emptyset)$ angeben wollen. Dieser Berechnungsbaum ist in Abbildung 6.2 zu sehen.

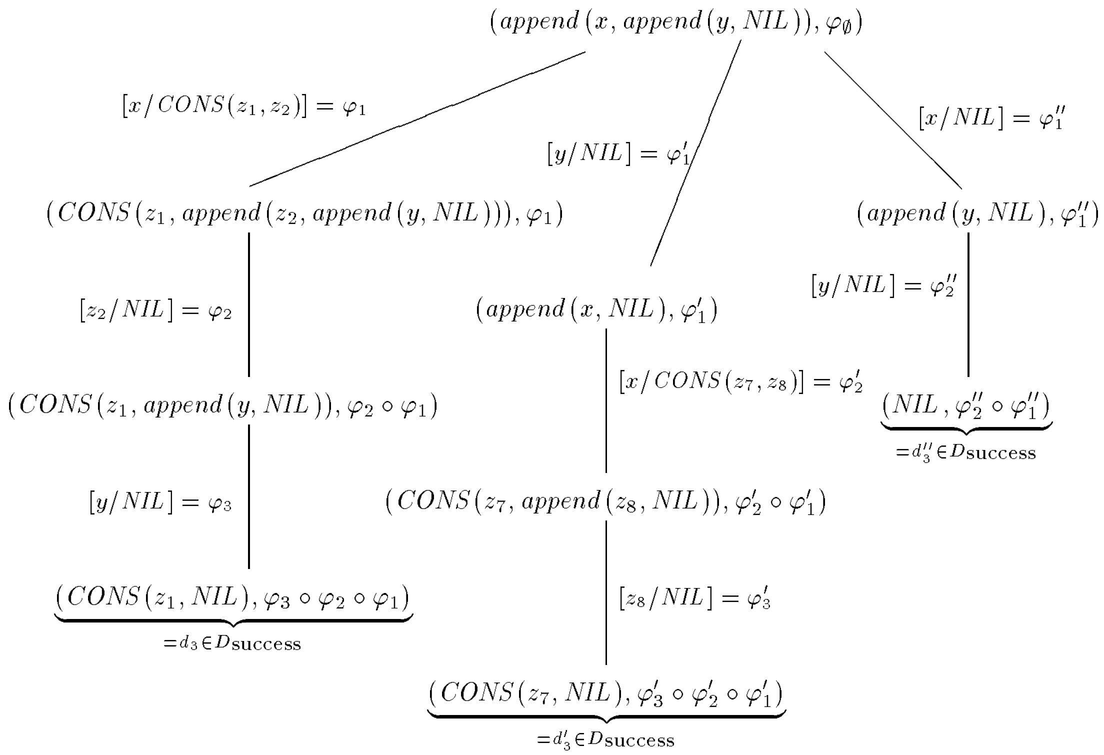

Abbildung 6.2: Berechnungsbaum von $\mathcal{I}$ und Kalkulandum d

Die Zwischenergebnisse auf dem am weitesten links stehenden Ast in dem Berechnungsbaum bezeichnen wir mit d_i, wobei i das Niveau im Baum angibt. Also beispielsweise $d_0 = d$ und $d_3 = (CONS(z_1, NIL), \varphi_3 \circ \varphi_2 \circ \varphi_1)$. Für den mittleren Ast wählen wir d'_i und für den rechten Ast d''_i als Bezeichnung der Zwischenergebnisse in der gleichen Art (s.o.).

Wir definieren eine Äquivalenzrelation $\sim$ wie folgt:
Für alle $d, d' \in D$ mit $d = (t, [v_1/t_1, \ldots v_n/t_n])$ gilt $d \sim d'$, g.d.w. es eine Variablenumbenennung ρ gibt, so daß $d' = (\hat{\rho}(t), [\rho(v_1)/\hat{\rho}(t_1), \ldots \rho(v_n)/\hat{\rho}(t_n)])$; wobei $\hat{\rho}$ die homomorphe Fortsetzung von $\rho : \mathcal{V} \to \mathcal{V}$ auf Terme $T(\Omega, \mathcal{V})$ ist.

Es soll nun die 2–success Effizienz für $\mathcal{C}$ und $\sim$ für das Zwischenergebnis d berechnet werden. Die beiden Zwischenergebnisse $d_3 = (t, [x/t_1, y/t_2])$ und d'_3 stehen in der Relation $\sim$ zueinander, weil es eine Variablenumbenennung $\rho = [z_1/z_7]$ gibt, so daß $d'_3 = (\hat{\rho}(t), [x/\hat{\rho}(t_1), y/\hat{\rho}(t_2)])$. Entsprechend der vor diesem Beispiel erwähnten Konstruktion stehen damit $w(d_3, \emptyset)$ und $w'(d'_3, \emptyset)$ in der Relation $\sim'$ zueinander, wobei $\sim'$

eine Äquivalenzrelation über der Zwischenergebnismenge des deterministischen Kalküls von $\mathcal{I}$ ist.

Es ergibt sich:

j	$E(j,d,\mathcal{C},\sim)$	$\lvert E(j,d,\mathcal{C},\sim)\rvert$
3	$\{\{w(d_3,\emptyset)\}\}$	1
9	$\{\{w(d_3,\emptyset), w'(d'_3,\emptyset)\}\}$	1
13	$\{\{w(d_3,\emptyset), w'(d'_3,\emptyset)\}\}$	1
14	$\{\{w(d_3,\emptyset), w'(d'_3,\emptyset)\}, \{w''(d''_3,\emptyset)\}\}$	2

wobei
$w = (d, \{d'_1, d''_1\})(d_1,\emptyset)(d_2,\emptyset)$,
$w' = (d, \{d''_1\})(d'_1,\emptyset)(d'_2,\emptyset)$,
$w'' = (d,\emptyset)(d''_1,\emptyset)(d''_2,\emptyset)$.

und damit:

$$(\eta_2(\mathcal{C},\sim))\,(d) = 14$$

□

Ganz analog zum vorhergehenden Abschnitt definieren wir nun, wann ein Kalkül $\mathcal{C}$ k–success effizienter ist als ein anderer Kalkül $\mathcal{C}'$. Die Bedingungen, welche an die beiden Kalküle gestellt werden, damit sie bezüglich der Effizienz vergleichbar sind, sind dieselben wie in Definition 6.4.

Definition 6.8 (k–success effizienter)

Es seien

- $\mathcal{C} = (D, D_{\text{success}}, \vdash)$, $\mathcal{C}' = (D', D'_{\text{success}}, \vdash')$
- $h : D' \to D$ eine Abbildung
- $\sim\, \subseteq D \times D$ und $\sim' \subseteq D' \times D'$
- $k \in \mathbb{N}$ mit $k \geq 2$

Der Kalkül $\mathcal{C}$ ist *k–success effizienter* als der Kalkül $\mathcal{C}'$, wenn die folgenden beiden Bedingungen gelten:

1. $\mathcal{C}$ ist η–vergleichbar durch h mit $\mathcal{C}'$.
2. Es gibt ein $d \in D'$, so daß

$$(\eta_k(\mathcal{C},\sim)\,(h(d)) <_{\mathbb{N}} (\eta_k(\mathcal{C}',\sim')\,(d)$$

es gibt kein $d' \in D$, so daß

$$(\eta_k(\mathcal{C},\sim)\,(h(d)) >_{\mathbb{N}} (\eta_k(\mathcal{C}',\sim')\,(d)$$

□

Ebenfalls sind die entsprechenden Beispiele für die Anwendung dieses Begriffes im nächsten Kapitel zu finden.

6.3 Weitere Effizienzmaße

Die beiden Effizienzmaße, die wir bis jetzt definiert haben, werden wir auf den deterministischen Kalkül angewendet. Dieser wiederum beschreibt einen depht–first left–to–right tree traversal durch den entsprechenden Berechnungsbaum. Ist ein Ast des Berechnungsbaumes, den der deterministische Kalkül vor Finden eines Enderbebnisses durchlaufen muß, unendlich, dann liefern die beiden Maße das Symbol ∞ als Ergebnis zurück. Ist dies bei zwei Kalkülen der Fall, so ist ihre single success Effizienz und auch die k–success Effizienz gleich. Trotzdem kann es aber sein, daß ein Kalkül wesentlich effizienter Endergebnisse berechnet, als der andere. Enthalten die Berechnungsbäume der zu vergleichenden Kalküle viele unendliche Äste, sind demnach die single success Effizienz und die multi success Effizienz angwendet auf den deterministischen Kalkül keine praktikablen Maße.

Wir möchten deswegen weitere Effizienzmaße definieren, die unabhängig vom deterministischen Kalkül Aussagen über die Effizienz eines Kalküls geben können. Die *success density Effizienz* und die *infinite path* Effizienz, die in diesem Abschnitt definiert werden, sind solche Maße.

6.3.1 Success Density Effizienz

Als drittes Effizienzmaß wollen wir eine Funktion definieren, welche die Anzahl der verschiedenen Endzustände auf einem bestimmten Niveau n des Berechnungsbaums eines Kalküls $\mathcal{C}$ und eines Kalkulandums d berechnet. Dabei ist der Begriff Niveau so zu verstehen, daß alle Zwischenergebnisse, die sich in n Berechnungsschritten aus d berechnen lassen, auf dem Niveau n liegen. Es handelt sich demnach um eine Funktion, welche die Dichte (auf Englisch: *density*) der Lösungen in Abhängigkeit von der Entfernung von der Wurzel beschreibt. Die Anordnung der Knoten innerhalb eines Niveaus ist zur Berechnung dieser Funktion nicht ausschlaggebend; man kann deshalb von einem allgemeinen Kalkül $\mathcal{C}$ ausgehen und muß nicht von einem indizierten Kalkül ausgehen. Da die Effizienzmaße aber vornehmlich auf die Narrowing Kalküle aus Kapitel 5 angewendet werden sollen, und diese als indizierte Kalküle dargestellt sind, werden wir neben der Definition der success density Effizienz für allgemeine Kalküle auch eine spezielle Definition für indizierte Kalküle angeben.

Da wieder nur voneinander verschiedene Endzustände gezählt werden sollen, nehmen wir in die Definition eine Äquivalenzrelation $\sim$ über D auf, welche festlegt wann zwei Zwischenergebnisse d und d' als gleich anzusehen sind. Es ist einleuchtend, daß ein Kalkül um so effizienter Endergebnisse berechnet, je näher an der Wurzel eines entsprechenden Berechnungsbaumes diese Endergebnisse auftreten. Wie in den Abschnitten 6.1 und 6.2 wollen wir einem bestimmten Berechnungsbaum eine natürliche Zahls als Effizienz zuordnen. Dies ist im Falle der success density Effizienz eine Summe, bestehend aus den gewichteten Anzahlen der Endergebnisse pro Niveau. Je näher an der Wurzel des Baumes sich ein Endergebnis berechnet, desto höher soll die Effizienz sein; i.e. die Gewichtung nimmt von der Wurzel ausgehen proportional zum Niveau ab. In der nun folgenden Definition der success density Effizienz drückt sich dies in dem Multiplikant $n + 1 - \nu$ aus.

Definition 6.9 (success density Effizienz)

Sei $\mathcal{C} = (D, D_{\text{success}}, \vdash)$ ein Kalkül, $\sim$ eine Äquivalenzrelation $\sim \subseteq D \times D$, $d \in D$ ein Kalkulandum und n eine natürliche Zahl. Die *success density Effizienz von $\mathcal{C}$ und $\sim$ bis Niveau n*, bezeichnet durch $\mathrm{d}\eta_n(\mathcal{C}, \sim)$, ist eine Funktion $\mathrm{d}\eta_n(\mathcal{C}, \sim) : D \to \mathbb{N}$, so daß für alle $d \in D$ gilt:

$$(\mathrm{d}\eta_n(\mathcal{C}, \sim))\,(d) = \sum_{\nu=0}^{n} (n + 1 - \nu) \cdot \left|(Suc_n\,(d, \vdash) \cap D_{\text{success}})/_{\sim}\right|$$

□

Beispiel 6.10 Betrachten wir noch einmal Abbildung 6.2. Sei $\mathcal{C}$ der zugrundeliegende Kalkül für den dort abgebildeten Berechnungsbaum. Die Anordnung der Knoten auf den einzelnen Niveaus ist nun beliebig, weil $\mathcal{C}$ ein allgemeiner Kalkül ist und kein indizierter Kalkül. Die Äquivalenzrelation $\sim$ sei wieder wie in Beispiel 6.7 definiert und d sei das Zwischenergebnis $(append(x, append(y, NIL)), \varphi_\emptyset)$.

Wir wollen nun die success density Effizienz von $\mathcal{C}$ und $\sim$ bis Niveau 3 berechnen:

ν	$(Suc_n\,(d, \vdash) \cap D_{\text{success}})/_{\sim}$	$(n + 1 - \nu) \cdot \left|(Suc_n\,(d, \vdash) \cap D_{\text{success}})/_{\sim}\right|$
0	$\emptyset$	$0 \cdot 4$
1	$\emptyset$	$0 \cdot 3$
2	$\{\{d_3''\}\}$	$1 \cdot 2$
3	$\{\{d_3, d_3'\}\}$	$1 \cdot 1$

Die success density Effizienz bildet sich nun aus der Summe der rechten Spalte und es gilt damit: $(\mathrm{d}\eta_n(\mathcal{C}, \sim))\,(d) = 3$.

□

Definition 6.11 (success density Effizienz für indizierte Kalküle)

Sei $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ ein indizierter Kalkül, $\sim$ eine Äquivalenzrelation $\sim \subseteq D \times D$, $d \in D$ ein Kalkulandum und n eine natürliche Zahl. Es sei:

- für alle $M \subseteq D : s(M) = \bigcup_{m \in M, i \in I(m)} Suc\,(d, \vdash_i)$
- $s^k(M)$ ist eine Abkürzung für: $\underbrace{(s \circ \ldots \circ s)}_{k \text{ mal}}\,(M)$

Die *success density Effizienz für indizierte Kalküle von $\mathcal{I}$ und $\sim$ bis Niveau n*, bezeichnet durch $\mathrm{d}\eta_n(\mathcal{I}, \sim)$, ist eine Funktion $\mathrm{d}\eta_n(\mathcal{I}, \sim) : D \to \mathbb{N}$, so daß für alle $d \in D$ gilt:

$$\mathrm{d}\eta_n(\mathcal{I}, \sim)\,(d) = \sum_{\nu=0}^{n} (n + 1 - \nu) \cdot \left|(s^n(\{d\}) \cap D_{\text{success}})/_{\sim}\right|$$

□

Die in der Definition der success density Effizienz eingeführte Funktion s ist natürlich nichts anderes als die Menge $Suc_n\,(d, \vdash)$ des zugrundeliegenden allgemeinen Kalküls. Da wir aber für indizierte Kalküle nur die Menge der Nachfolgeindizes $I(d)$ definiert haben, muß eine solche Funktion eingeführt werden.

Als Beispiel für die success density Effizienz für indizierte Kalküle kann das Beispiel 6.10 betrachtet werden. Es müssen dazu folgende Beziehungen beachtet werden:

- Nach Definition des indizierten Kalküls für einen Kalkül $\mathcal{C}$ gilt: $\bigcup_{j \in J} \vdash_j = \vdash$.
- Das n–malige Anwenden der Funktion s in der Definition der success density Effizienz für indizierte Kalküle ist identisch mit der Menge $Suc_n\,(d, \vdash)$ in Definition 6.9.

Definition 6.12 (success density effizienter)

Es seien $\mathcal{C}$ und $\mathcal{C}'$ [indizierte] Kalküle mit Zwischenergebnismengen D bzw. D', $h : D \to D'$ eine injektive Abbildung, $\sim \subseteq D \times D$ und $\sim \subseteq D' \times D'$..

Der [indizierte] Kalkül $\mathcal{C}$ ist *success density effizienter* als der [indizierte] Kalkül $\mathcal{C}'$, wenn es ein $n \in \mathbb{N}$ gibt, so daß folgende zwei Bedingungen erfüllt sind:

1. $\mathcal{C}$ ist η–vergleichbar mit $\mathcal{C}'$ durch h.
2. es gibt ein $d \in D$, so daß $(\mathrm{d}\eta_n(\mathcal{C}, \sim))\,(d) <_{\mathbb{N}} (\mathrm{d}\eta_n(\mathcal{C}', \sim')\,(h(d))$ und es gibt kein $d' \in D$, so daß $(\mathrm{d}\eta_n(\mathcal{C}, \sim))\,(d') >_{\mathbb{N}} (\mathrm{d}\eta_n(\mathcal{C}', \sim')\,(h(d'))$

□

6.3.2 Infinite Path Effizienz

Die single- und k–success Effizienzen benutzen den deterministischen Kalkül eines indizierten Kalküls, um die Schrittzahlen bis zum Finden von Endergebnissen zu berechnen. Sie sind dadurch, wie schon oben erwähnt an ein depth–first Vorgehen durch den Berechnungsbaum gebunden. Wollen wir aber die success–Effizienzen zweier Kalküle miteinander vergleichen, und es geraten die Berechnungen *beider* Kalküle in unendliche Äste, so schlägt der Vergleich fehl. Die success Effizienz beträgt für beide Kalküle ∞. Tatsächlich kann aber bei einer anderen depth–first Strategie ein ganz anderes Verhalten vorhanden sein, so daß ein Kalkül effizienter ist als der andere. Wir möchten ein Maß definieren, welches in Bezug auf unendliche Äste *nicht* von der Wahl der Strategie abhängt, sondern eine allgemein Aussage über die Anzahl der unendlichen Äste liefert. Die infinite path Effizienz ist ein solches Maß. Es berechnet die Anzahl der unendlichen Äste eines indizierten Kalküls $\mathcal{I}$ für ein Kalkulandum d. Definieren wir zunächst was unter einem unendlichen Ast (auf Englisch *infinite path*) zu verstehen ist.

Definition 6.13 (infinite path)

Sie $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ ein indizierter Kalkül, $d \in D$ ein Kalkulandum und $i_0, i_1, \ldots \in J$. Die unendliche Folge w mit $w = (i_0 i_1 \ldots)$ beschreibt einen *infinite path von d*, wenn es kein $k \in \mathbb{N}$ gibt, so daß:

$$d \vdash_{i_0} d_0 \vdash_{i_1} d_1 \vdash_{i_2} \ldots \vdash_{i_k} d_k \text{ und } \bigcup_{j \in J} Suc\,(d_k, \vdash_j) = \emptyset.$$

Die Menge aller infinite paths von d bezeichnen wir durch $IP\,(\mathcal{I}, d)$.

□

Beispiel 6.14 Betrachten wir einen Kalkül, der Berechnungen nach dem Kommutativgesetz für arithmetische Terme über dem Rangalphabet $\{+^{(2)}\}$ mit Variablen $\{a, b\}$ beschreibt, d.h.

$$\mathcal{I} = (\underbrace{T(\{+^{(2)}\}, \{a^{(0)}, b^{(0)}\})}_{D}, \emptyset, (\mathbb{N}^*, <_{lex}), (\vdash_j \mid j \in \mathbb{N}^*)).$$

wobei $(\vdash_j \mid j \in \mathbb{N}^*)$ eine Familie ist mit folgender Eigenschaft: Sei $t \in D$, $u \in O(t)$ und $t' = t/u$, dann gilt:
falls $t' = (t_1 + t_2)$ mit $t_1, t_2 \in T(\{+^{(2)}\}, \{a^{(0)}, b^{(0)}\})$, dann $(t, t[u \leftarrow (t_2 + t_1)]) \in \vdash_u$.
Das Wort $w = (\varepsilon\varepsilon\varepsilon\ldots)$ ist ein infinite path von $d = (a + b)$, denn es gibt kein d' in der Berechnung

$$(a + b) \vdash_\varepsilon (b + a) \vdash_\varepsilon (a + b) \ldots \vdash_\varepsilon d'$$

mit $\bigcup_{j \in J} Suc\,(d_k, \vdash_j) = \emptyset$.

□

Sei $\mathcal{I}$ ein indizierter Kalkül und d ein Kalkulandum. Die infinite path Effizienz hängt von der Anzahl der infinite paths von d in dem Kalkül $\mathcal{I}$ ab. Da aber der Berechnungsbaum eines Kalküls, der nur wenige unendliche Äste enthält, mit großer Wahrscheinlichkeit effizienter durchlaufen werden kann, als ein anderer Baum mit vielen unendlichen Ästen, ist ein indizierter Kalkül um so effizienter, je *weniger* infinite paths er enthält. Um dies auszudrücken, definieren wir die infinite path Effizienz eines indizierten Kalküls $\mathcal{I}$ als eine Funktion $\mathrm{i}\eta(\mathcal{I}) : \mathcal{D} \to \mathbb{N}$.

Definition 6.15 (infinite path Effizienz)

Sei $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ ein indizierter Kalkül. Die *infinite path Effizienz von $\mathcal{I}$*, bezeichnet durch $\mathrm{i}\eta(\mathcal{I})$ ist eine Funktion $\mathrm{i}\eta(\mathcal{I}) : D \to \mathbb{N}$, so daß für alle $d \in D$ gilt:

$$(\mathrm{i}\eta(\mathcal{I}))\,(d) = |IP\,(\mathcal{I}, d)|$$

□

In Beispiel 6.14 gibt es nur eine Berechnung, die von dem Term $(a+b)$ ausgeht. Offensichtlich gilt dann für die infinite path Effizienz von $\mathcal{I}$ für das Kalkulandum $(a+b)$ folgendes:

$$(\mathrm{i}\eta(\mathcal{I})\ ((a+b)) = |IP(\mathcal{I}, (a+b))| = |\{(\varepsilon\varepsilon\ldots)\}| = 1.$$

Ganz analog zu den vorhergehenden Abschnitten, ist ein indizierter Kalkül $\mathcal{I}$ infinite path effizienter als ein indizierter Kalkül $\mathcal{I}'$, wenn die beiden Kalküle vergleichbar sind, und die infinite path Effizienz von $\mathcal{I}$ kleiner ist als die von $\mathcal{I}'$.

Definition 6.16 (infinite path effizienter)

Sein $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ und $\mathcal{I}' = (D', D'_{\text{success}}, (J', <'), (\vdash_j \mid j \in J'))$ indizierte Kalküle und $h : D \to D'$ eine Abbildung. Der indizierte Kalkül $\mathcal{I}$ ist *infinite path effizienter* als der indizierte Kalkül $\mathcal{I}'$, wenn folgende beiden Bedingungen erfüllt sind:

1. $\mathcal{I}$ ist vergleichbar mit $\mathcal{I}'$ durch h.

2. es gibt ein $d \in D$, so daß $(\mathrm{i}\eta(\mathcal{I}))\ (d) <_{\mathbb{N}} (\mathrm{i}\eta(\mathcal{I}'))\ (h(d))$ und es gibt kein $d' \in D$, so daß $(\mathrm{i}\eta(\mathcal{I}))\ (d') >_{\mathbb{N}} (\mathrm{i}\eta(\mathcal{I}'))\ (h(d'))$.

□

Kapitel 7

Vergleich von Effizienzen

In Kapitel 5 wurden vier verschiedene Kalküle definiert, mit denen $E_{\mathcal{R}}$–Unifikation bzw. $(E_{\mathcal{R}}, \Delta)$–Unifikation betrieben werden kann. Der Reduktionskalkül und der Narrowing Kalkül für Gleichungstheorien, die durch kanonische Termersetzungssysteme induziert werden, der lo–Narrowing Kalkül und der ulo–Narrowing Kalkül für Gleichungstheorien, die durch ctn–trs induziert werden. Jetzt wollen wir einige der Maße, welche wir in Kapitel 6 definiert haben, benutzen, um die Effizienz, mit der durch diese Kalküle $E_{\mathcal{R}}$– bzw. $(E_{\mathcal{R}}, \Delta)$–unifiziert wird, zu messen. In der bestehenden Literatur über Narrowing findet man zu diesem Thema hauptsächlich folgende Ansatzpunkte für Effizienzvergleiche:

- empirisch ermittelte Laufzeit der Implementation eines Narrowing Verfahrens; d.h. Angaben in Sekunden oder einer ähnlichen Einheit. Eine solche Angabe ist natürlich vom Rechner und der Implementation, sowie von vielen anderen Faktoren abhängig. Sie kann vom Leser kaum auf seine Richtigkeit hin überprüft werden. Darüber hinaus sind Laufzeitangaben immer nur für konkrete Beispiele möglich. Siehe e.g. [AFJRV93], [BKW94], [Han94].

- Anzahl der Knoten in Narrowingbäumen konkreter Beispiele oder auch Breite des Narrowingbaums eines konkreten Beispiels bis zu einer bestimmten Tiefe. Siehe e.g. [BKW94], [Kri90], [WBK94].

- Aussagen über die Optimalität der gewählten Narrowing Methode; dabei ist unter Optimalität üblicherweise zu verstehen, daß zwei verschiedene Berechnungen verschiedene Ergebnisse erzeugen. Dieses Maß ist weder von der Implementation noch von einem bestimmten Beispiel abhängig, sondern beschreibt eine Eigenschaft des Narrowing Verfahrens. Es ist damit ein allgemeines Maß für die Effizienz mit der E–unifiziert werden kann. Siehe [AEH94], [BKW92], [BKW94], [Han92], [WBK94].

Allgemein ist festzustellen, daß bei allen Effizienzmaßen eine allgemeine Aussage der Form: Kalkül A ist weniger effizient als Kalkül B, bezüglich des Effizienzmaßes α für alle TESe $\mathcal{R}$ und alle Kalkulandi d, nur sehr schwer möglich ist. Die Ursache dafür liegt in der

Tatsache, daß alle hier erörterten Effizienzmaße von bestimmten Kalkulandi und Kalkülen ausgehen. In diesem Kapitel wollen wir dennoch versuchen, derartige Aussagen für die in dieser Arbeit vorgestellten Kalküle zur $E_{\mathcal{R}}$–Unifikation herzuleiten — oder zumindest zu erörtern.

Da Reduktion nur sehr bedingt für $E_{\mathcal{R}}$–Unifikation benutzt werden kann, nämlich nur mit vorherigem Raten einer passenden Substitution φ, werden wir den Reduktionskalkül für $E_{\mathcal{R}}$–Unifikation nicht in die folgenden Vergleiche aufnehmen. Es verbleiben demnach der Narrowing Kalkül, der lo–Narrowing Kalkül und der ulo–Narrowing Kalkül. Im ersten Abschnitt soll Narrowing und lo–Narrowing auf die im vorhergehenden Kapitel definierten Effizienzmaße single–success Effizienz und multi–success Effizienz hin untersucht werden, im zweiten Abschnitt dann lo–Narrowing mit ulo–Narrowing für diese beiden Maße.

Bei den Vergleichen werden wir uns auf die kleinste Klasse von Gleichungstheorien beschränken, für die alle drei Kalkül vollständig sind. Wie in Kapitel 5 gezeigt, ist dies die Klasse der durch ctn–trs induzierten Gleichungstheorien. Darüberhinaus beschränken wir uns bei der Wahl der Endergebnismenge auf grund–vollständige Mengen von $(E_{\mathcal{R}}, \Delta)$–Unifikatoren. Diese Beschränkung gilt nur für den Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation, welcher gemäß Satz 4.10 eine vollständige Menge von E–Unifikatoren berechnet; i.e. eine Obermenge der grund–vollständigen $(E_{\mathcal{R}}, \Delta)$–Unifikatoren. Der lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation und der ulo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation liefern eine grund–vollständige Menge von $(E_{\mathcal{R}}, \Delta)$–Unifikatoren.

7.1 Narrowing vs. lo–Narrowing

In diesem Abschnitt sollen der Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation (vgl. Definition 5.18) und der lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation (vgl. Definition 5.29) miteinander bezüglich der Effizienzmaße aus Kapitel 6 verglichen werden. Wir wiederholen kurz die Kalküle.

Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs, $\Omega = F \cup \Delta$ und $\mathcal{R}' = (\Omega, R)$ ein TES.

- Narrowing Kalkül für $E_{\mathcal{R}}$–Unifikation
  $\mathrm{nar}_E\mathcal{I}\,(\mathcal{R}') = (D, D_{\mathrm{success}}, (J, <), (\vdash_j \;\mid j \in J))$ mit
  - $D = \mathrm{Equ}\,(T(\Omega, \mathcal{V})) \times \mathit{Sub}\,(\mathcal{V}, \Omega)$
  - $D_{\mathrm{success}} = \{(\mathit{equ}\,(\Theta_1, \Theta_2), \varphi) \mid \Theta_1, \Theta_2 \in T(\Omega, \mathcal{V}),\ \Theta_1$ und Θ_2 sind unifizierbar, $\varphi \in \mathit{Sub}\,(\mathcal{V}, \Delta)\}$
  - $J = \mathbb{N}^* \times R$
  - $<$ ist definiert wie beim Narrowing Kalkül (siehe Definition 5.16).
  - für $j = (u, l \to r) \in J$, $\vdash_j = \rightsquigarrow_{\mathcal{R}', u, l \to r}$ (siehe Definition 5.14)
- lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation
  $\text{lo–nar}_E\mathcal{I}\,(\mathcal{R}) = (D', D'_{\mathrm{success}}, (J', <'), (\vdash'_j \;\mid j \in J'))$ mit

- $D' = \text{Equ}\,(T(\Omega, \mathcal{V})) \times Sub\,(\mathcal{V}, \Omega)$
- $D'_{\text{success}} = \{(equ\,(\Theta_1, \Theta_2), \varphi) \mid \Theta_1, \Theta_2 \in T(\Omega, \mathcal{V}),\ \Theta_1$ und Θ_2 sind unifizierbar, $\varphi \in Sub\,(\mathcal{V}, \Delta)\}$
- $J' = R$
- $<'$ ist definiert wie beim lo–Narrowing Kalkül (siehe Definition 5.25).
- für $j = (l \to r) \in J$, $\vdash'_j = \overset{lo}{\rightsquigarrow}_{\mathcal{R},l\to r}$ (siehe Definition 5.24)

Da der lo–Narrowing Kalkül nur eine Einschränkung des Narrowing Kalküls ist, nämlich die Einschränkung der Narrowing Vorkommen auf die leftmost–outermost Vorkommen, sind die Bestandteil der beiden Kalküle sehr ähnlich. Es gilt nämlich $D = D'$ und $D_{\text{success}} = D'_{\text{success}}$ (beachte, daß wir bei der Menge D_{success} nur noch $(E_{\mathcal{R}}, \Delta)$–Unifikatoren betrachten).

Die beiden Kalküle $\text{nar}_E\mathcal{I}\,(\mathcal{R}')$ und $\text{lo–nar}_E\mathcal{I}\,(\mathcal{R})$ sind demnach η–vergleichbar durch h, wobei h die Identitätsfunktion ist.

7.1.1 Single success Effizienz

Betrachtet man die Berechnungsbäume für $\text{nar}_E\mathcal{I}\,(\mathcal{R}')$ und $\text{lo–nar}_E\mathcal{I}\,(\mathcal{R})$ und ein Kalkulandum d so fällt folgendes auf: Der Berechnungsbaum des Narrowing Kalküls für $E_{\mathcal{R}}$–Unifikation ist entweder

- identisch mit dem Berechnungsbaum von $\text{lo–nar}_E\mathcal{I}\,(\mathcal{R})$, oder
- er enthält *mehr* Äste als der Berechnungsbaum von $\text{lo–nar}_E\mathcal{I}\,(\mathcal{R})$.

Rein intuitiv ist klar, daß die single success Effizienz von $det\,(\text{lo–nar}_E\mathcal{I}\,(\mathcal{R}))$, also die Anzahl der Berechnungsschritte des deterministischen lo–Narrowing Kalküls für $E_{\mathcal{R}}$–Unifikation bis zum Finden des ersten Endergebnisses, *kleiner oder gleich* der single success Effizienz von $det\,(\text{nar}_E\mathcal{I}\,(\mathcal{R}'))$, ist. Um zu sehen wann genau der erste Fall zutrifft, also für ein Kalkulandum d

$$(\eta(det\,(\text{lo–nar}_E\mathcal{I}\,(\mathcal{R})))\,(d) < (\eta(det\,(\text{nar}_E\mathcal{I}\,(\mathcal{R}')))\,(d)$$

gilt, betrachten wir zunächst das folgende Beispiel.

Beispiel 7.1 Sei $\mathcal{R}$ ein ctn–trs mit:

- $F = \{f^{(1)}, g^{(1)}\}$
- $\Delta = \{A^{(0)}, B^{(0)}, G^{(2)}\}$
- R enthält die folgenden Regeln:

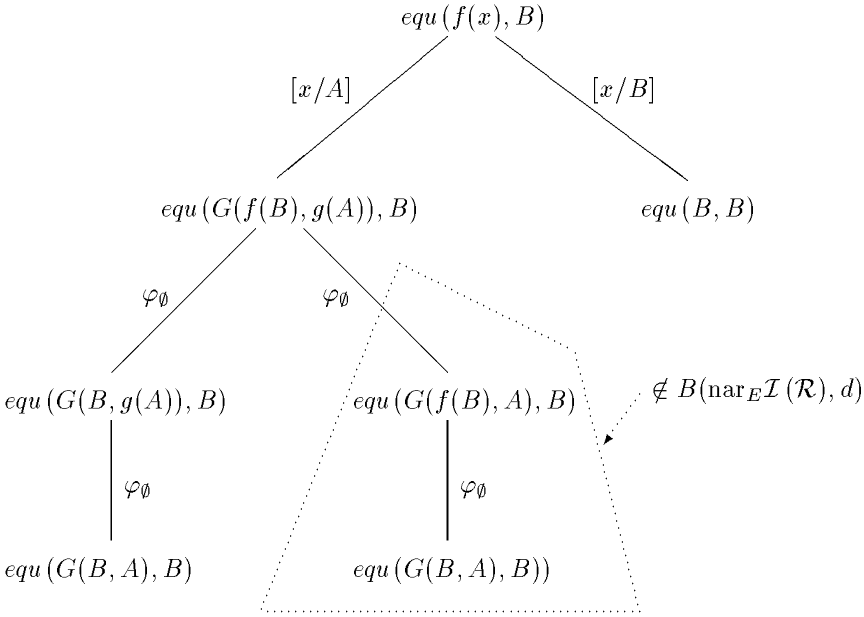

Abbildung 7.1: Narrowing- und lo–Narrowingbaum für Kalkulandum $equ(f(x), B)$

$$\begin{array}{rcll} f(A) & \to & G(f(B), g(A)) & (1)\\ f(B) & \to & B & (2)\\ f(G(x,y)) & \to & G(f(x), f(y)) & (3)\\ g(x) & \to & A & (4) \end{array}$$

Die Regeln (3) und (4) werden wir in diesem Beispiel nicht brauchen, sie sind nur der Vollständigkeit halber angegeben, damit das Funktionssymbol f total definiert ist. Der Narrowing Kalkül ist nicht für ctn–trs, sondern für TESe definiert. Wir definieren deshalb das zu $\mathcal{R}$ entsprechende TES $\mathcal{R}' = (F \cup \Delta, R)$. Betrachten wir die Berechnungsbäume von $\mathrm{nar}_E\mathcal{I}(\mathcal{R}')$ und lo–$\mathrm{nar}_E\mathcal{I}(\mathcal{R})$ für das Kalkulandum $equ(f(x), B)$ in Abbildung 7.1. Der eingekreiste Ast ist nicht Teil des leftmost outermost Narrowingbaumes, sondern nur Teil des Narrowingbaumes von d. Offensichtlich gilt

$$\begin{array}{rcl} (\eta(det(\text{lo–nar}_E\mathcal{I}(\mathcal{R}))))(d) & = & 7.\\ (\eta(det(\mathrm{nar}_E\mathcal{I}(\mathcal{R}'))))(d) & = & 11. \end{array}$$

Damit ist der lo–Narrowingkalkül für $(E_\mathcal{R}, \Delta)$–Unifikation für dieses Beispiel single success effizienter als der Narrowingkalkül für $E_\mathcal{R}$–Unifikation.

□

Kann es aber nicht auch ein TES $\mathcal{R}$ und ein Kalkulandum d geben, so daß Narrowing single success effizienter ist als lo–Narrowing? Betrachten wir dazu das Szenario in Abbildung

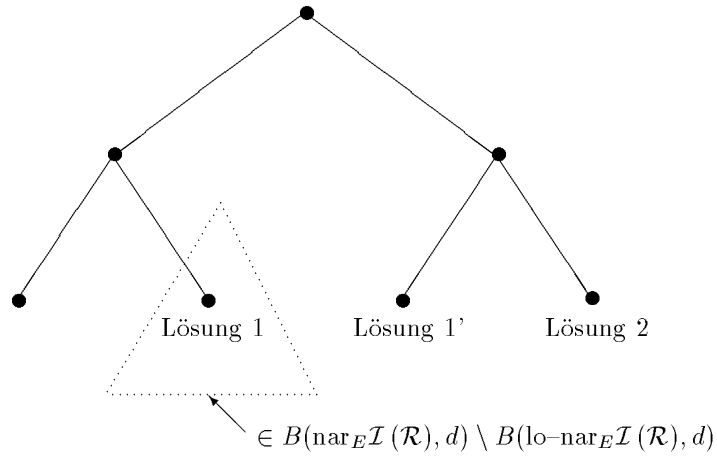

Abbildung 7.2: Narrowing- bzw. lo–Narrowingbaum mit $(\eta(\mathit{det}(\text{lo–nar}_E\mathcal{I}\,(\mathcal{R}))))\,(d) > (\eta(\mathit{det}\,(\text{nar}_E\mathcal{I}\,(\mathcal{R}'))))\,(d)$.

7.2. Dabei ist Lösung 1 nicht durch den lo–Narrowing Kalkül berechenbar, sondern nur durch den Narrowing Kalkül. Die Lösungen 2 und 1' sind durch beide Kalküle berechenbar und es gilt, daß Lösung 1 und Lösung 1' zwei bezüglich Variablenumbenennung äquivalente Lösungen sind. Kann es ein ctn–trs $\mathcal{R} = (F, \Delta, R)$, TES $\mathcal{R}' = (F \cup \Delta, R)$ und ein Kalkulandum d geben, so daß Berechnungsbäume $B(\text{nar}_E\mathcal{I}\,(\mathcal{R}'), d)$ und $B(\text{lo–nar}_E\mathcal{I}(\mathcal{R}), d)$ entsprechend Abbildung 7.2 entstehen? Wäre die Antwort ja, so würde sich eine größere single success Effizienz für lo–Narrowing als für Narrowing ergeben, also $\eta(\mathit{det}(\text{lo–nar}_E\mathcal{I}\,(\mathcal{R})))\,(d) > (\eta(\mathit{det}(\text{nar}_E\mathcal{I}\,(\mathcal{R}')))\,(d)$. Das folgende Lemma zeigt, daß dies nicht der Fall sein kann.

Wir definieren zunächst die Menge *left* (x), die für einen Knoten x eines Berechnungsbaumes die Menge der Knoten *links* von dem Knoten x enthält. Es soll dabei der Pfad zu dem Knoten x selber *nicht* in diese Menge aufgenommen werden.

Definition 7.2 *(left)*

Sei $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ ein indizierter Kalkül, $d \in D$ ein Kalkulandum und $x \in \underline{\mathit{node}}\,(B(\mathcal{I}, d))$ ein Knoten, wobei x als Folge der Indizes aus J von $\underline{\mathit{root}}\,(B(\mathcal{I}, d))$ bis zu dem Knoten x dargestellt wird; also $x = (j_1 \ldots j_n)$ mit $j_i \in J$ für $1 \leq i \leq n$. Die Mengen *prefix* und *left* sind wie folgt definiert:

- $\mathit{prefix}\,(x, \mathcal{I}, d) = \{(j_1 \ldots j_m) \mid 1 \leq m \leq n\}$.
- $\mathit{left}\,(x, \mathcal{I}, d) = \{x' \mid x' \in \underline{\mathit{node}}\,(B(\mathcal{I}, d)) \text{ mit } x' <_{\text{lex}} x\} \setminus \mathit{prefix}\,(x, \mathcal{I}, d)$.

□

Lemma 7.3 Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs, $\mathcal{R}' = (F \cup \Delta, R)$ ein TES und d ein Kalkulandum.
Wenn $x \in \underline{\mathit{node}}\,(B(\mathrm{nar}_E\mathcal{I}\,(\mathcal{R}'), d)) \setminus \underline{\mathit{node}}\,(B(\text{lo–nar}_E\mathcal{I}\,(\mathcal{R}), d))$ und
$\underline{\mathit{label}}_{B(\mathrm{nar}_E\mathcal{I}\,(\mathcal{R}'),d)}\,(x) \in D_{\mathrm{success}}$,
dann gibt es ein $x' \in \mathit{left}\,(x, \mathrm{nar}_E\mathcal{I}\,(\mathcal{R}'), d) \cap \underline{\mathit{node}}\,(B(\text{lo–nar}_E\mathcal{I}\,(\mathcal{R}), d))$ mit $\underline{\mathit{label}}_{B(\text{lo–nar}_E\mathcal{I}\,(\mathcal{R}),d)}\,(x')$ geht durch Variablenumbenennung aus $\underline{\mathit{label}}_{B(\mathrm{nar}_E\mathcal{I}\,(\mathcal{R}'),d)}\,(x)$ hervor.

□

Dieses Lemma besagt also folgendes: Wenn es einen Knoten x im Narrowingbaum gibt, dessen Beschriftung ein Endergebnis d_1 ist und der nicht im lo–Narrowingbaum enthalten ist, dann gibt es einen Knoten x' links von x, der in lo–Narrowingbaum enthalten ist, und dessen Beschriftung d_2 ein äquivalentes Endergebnis ist; wobei unter Äquivalenz hier Variablenumbenennung zu verstehen ist. Hiermit ist bewiesen, daß es kein TES $\mathcal{R}$ und Kalkulandum d gibt, so daß $(\eta(\mathit{det}(\text{lo–nar}_E\mathcal{I}\,(\mathcal{R})))\,(d) > (\eta(\mathit{det}(\mathrm{nar}_E\mathcal{I}\,(\mathcal{R}')))\,(d)$ gilt.

<u>Beweis zu Lemma 7.3</u>

Sei d ein Kalkulandum und $x \in \underline{\mathit{node}}\,(B(\mathrm{nar}_E\mathcal{I}\,(\mathcal{R}'), d)) \setminus \underline{\mathit{node}}\,(B(\text{lo–nar}_E\mathcal{I}\,(\mathcal{R}), d))$ und $\underline{\mathit{label}}_{B(\mathrm{nar}_E\mathcal{I}\,(\mathcal{R}'),d)}\,(x) \in D_{\mathrm{success}}$. Da wir die Menge D_{success} auf $(E_{\mathcal{R}}, \Delta)$–Unifikatoren beschränkt haben, ist x ein Blatt des Baumes. Dann gibt es einen Knoten $y \in \mathit{prefix}\,(x, \mathrm{nar}_E\mathcal{I}\,(\mathcal{R}'), d)$ mit mindestens zwei Narrowingvorkommen. Wir wählen den untersten solchen Knoten y', der noch Teil des lo–Narrowingbaumes ist, also

$$\max_{\mathrm{lex}} \{\hat{x} \mid \hat{x} \in \mathit{prefix}(x, \mathrm{nar}_E\mathcal{I}\,(\mathcal{R}'), d) \cap \text{lo–nar}_E\mathcal{I}\,(\mathcal{R}) \text{ und } \underline{\mathit{label}}_{B(\text{lo–nar}_E\mathcal{I}\,(\mathcal{R}),d)}\,(\hat{x}) \text{ enthält zwei Narrowingvorkommen}\}.$$

Die Menge F der Funktionssymbole des ctn–trs $\mathcal{R}$ enthalte die Symbole f und g. Die Beschriftung des Knotens y' sei (t, φ). Für die Verschachtelung der beiden Narrowingvorkommen in t gibt es folgende Möglichkeiten:

1. $t = \ldots f(\ldots) \ldots g(\ldots) \ldots$; d.h. g tritt *nicht* als Argument von f auf.
2. $t = \ldots f(\ldots g(\ldots) \ldots) \ldots$; d.h. g tritt als Argument von f auf.

Die in $\ldots$ enthaltenen Symbole sollen nur aus $\Delta \cup \mathcal{V}$ sein. Dies ist keine Einschränkung, denn die Verschachtelung der Funktionssymbole kann dort auch nur eine der beiden oben erwähnten Formen annehmen, so daß die folgende Erklärung dort in gleicher Weise gilt.

<u>Fall 1:</u> $t = \ldots f(\ldots) \ldots g(\ldots) \ldots$

Es seien $u_f, u_g \in \mathbb{N}^*$ mit $t/u_f = f(\ldots)$ und $t/u_g = g(\ldots)$. Dann ist u_f ein lo–Narrowingvorkommen und u_g ein Narrowingvorkommen. Es ist zu zeigen, daß es für jedes Endergebnis, welches durch eine Narrowingberechnung, i.e. Regelanwendungen an beliebigen Narrowingvorkommen, berechnet werden kann, auch ein modulo Variablenumbenennung gleiches Endergebnis im lo–Narrowingbaum links von y' gibt. Der Knoten y' ist der unterste Knoten auf dem Pfad zu x, der noch Teil des lo–Narrowingbaumes ist; d.h. um zu x zu kommen, wird am Vorkommen u_g ein Narrowingschritt angewendet. Wir unterscheiden zwei Arten von Regeln:

(a) Regeln, deren rechte Seiten jeweils mehr als ein Funktionssymbol enthalten, die also die Anzahl der Funktionssymbole in t erhöhen.

(b) Regeln, deren rechte Seiten jeweils höchstens ein Funktionssymbol enthahlten.

O.B.d.A betrachten wir nur Regeln der Form (b); diese Einschränkung ist gerechfertigt, weil für die zusätzlich dazukommenden Narrowingvorkommen jeweils wieder Fall 1 oder Fall 2 dieses Beweises gilt. Die Narrowingableitung ausgehend von t sieht wie folgt aus:

$$t \rightsquigarrow_{\mathcal{R}',\delta_1} \cdots \rightsquigarrow_{\mathcal{R}',\delta_n} \ldots \underbrace{\sigma_f(\ldots)}_{T(\Delta,\mathcal{V})} \ldots \underbrace{\sigma_g(\ldots)}_{T(\Delta,\mathcal{V})} \ldots = \bar{t}$$

wobei für $1 \leq i \leq n : \delta_i = (u_i, \varphi_i, \xi_i)$ mit $u_i \in \mathbb{N}^*$, $\varphi_i \in Sub(\mathcal{V}, \Omega)$, $\xi_i \in R$ und die Regeln ξ_i sind von der Form (b) von oben. Sei Γ die Sequenz $\delta_1, \ldots, \delta_n$ der Narrowing–Tripel. Wir definieren dann zwei Sequenzen Γ_f und Γ_g wie folgt:

$\Gamma_f = \{(u, \varphi, \xi) \in \Gamma \mid u = u_f\} = \delta_{f_1}, \ldots, \delta_{f_k}$ mit $f_i < f_j$ für $i < j$.

$\Gamma_g = \{(u, \varphi, \xi) \in \Gamma \mid u = u_g\} = \delta_{g_1}, \ldots, \delta_{g_l}$ mit $g_i < g_j$ für $i < j$.

Die Sequenz Γ_f bezieht sich nur auf Narrowingschritte am Vorkommen u_f, und die Sequenz Γ_g bezieht sich nur auf Narrowingschritte am Vorkommen u_g. Es gibt demnach folgende lo–Narrowingberechnung ausgehend von t:

$$t \overset{lo}{\rightsquigarrow}_{\mathcal{R},\varphi_{f_1},\xi_{f_1}} \cdots \overset{lo}{\rightsquigarrow}_{\mathcal{R},\varphi_{f_k},\xi_{f_k}} \ldots \sigma'_f(\ldots) \ldots g(\ldots) \ldots \overset{lo}{\rightsquigarrow}_{\mathcal{R},\varphi_{g_1},\xi_{g_1}} \cdots \overset{lo}{\rightsquigarrow}_{\mathcal{R},\varphi_{g_l},\xi_{g_l}}$$
$$\ldots \underbrace{\sigma'_f(\ldots)}_{T(\Delta,\mathcal{V})} \ldots \underbrace{\sigma'_g(\ldots)}_{T(\Delta,\mathcal{V})} \ldots = \bar{t}'$$

Der Term $\bar{t}'$ geht durch Variablenumbenennung aus $\bar{t}$ hervor.

<u>Fall 2:</u> $t = \ldots f(\ldots g(\ldots) \ldots) \ldots$

Für das Narrowing am Vorkommen u_f gibt es nur *eine* mögliche Regel ζ in R, weil $\mathcal{R}$ totally defined und not strictly subunifiable ist. Sei die Narrowingberechnung, die von der Beschriftung des Knotens y' zur Beschriftung des Knotens x führt wieder folgender Gestalt

$$t \rightsquigarrow_{\mathcal{R}',\delta_1} \cdots \rightsquigarrow_{\mathcal{R}',\delta_n} \ldots \underbrace{\sigma_f(\ldots \sigma_g(\ldots) \ldots)}_{T(\Delta,\mathcal{V})} \ldots = \bar{t}$$

wobei für $1 \leq i \leq n : \delta_i = (u_i, \varphi_i, \xi_i)$ mit $u_i \in \mathbb{N}^*$, $\varphi_i \in Sub(\mathcal{V}, \Omega)$, $\xi_i \in R$ und die Regeln ξ_i sind von der Form (b) von oben. Sei Γ die Sequenz $\delta_1, \ldots, \delta_n$ der Narrowing–Tripel. Selektieren wir wieder alle Narrowingschritte, die sich auf das Vorkommen u_f, bzw. u_g beziehen,

$\Gamma_f = \{(u, \varphi, \xi) \in \Gamma \mid u = u_f\} = \delta_{f_1}, \ldots, \delta_{f_k}$ mit $f_i < f_j$ für $i < j$.

$\Gamma_g = \{(u, \varphi, \xi) \in \Gamma \mid u = u_g\} = \delta_{g_1}, \ldots, \delta_{g_l}$ mit $g_i < g_j$ für $i < j$.

dann gilt für $1 \leq i \leq k$: $\xi_{f_i} = \zeta$. Wir können deshalb erst alle Regeln am Vorkommen u_f anwenden, und dann die Regeln am Vorkommen u_g. Es entsteht dann eine lo–Berechnung

$$t \overset{lo}{\leadsto}_{\mathcal{R},\varphi_{f_1},\zeta} \cdots \overset{lo}{\leadsto}_{\mathcal{R},\varphi_{f_k},\zeta} \ldots \sigma'_f(\ldots g(\ldots)\ldots)\ldots \overset{lo}{\leadsto}_{\mathcal{R},\varphi_{g_1},\xi_{g_1}} \cdots \overset{lo}{\leadsto}_{\mathcal{R},\varphi_{g_l},\xi_{g_l}}$$
$$\ldots \underbrace{\sigma_f(\ldots\sigma_g(\ldots)\ldots)}_{T(\Delta,\mathcal{V})} \ldots = \bar{t}$$

Es gibt also auch im Fall 2 eine lo–Narrowing Berechnung die das gleiche Endergebnis wie die entsprechende Narrowing Berechnung erzeugt; damit ist Lemma 7.3 bewiesen.

□

Unterschiede in der single success Effizienz können nur dann auftreten, wenn der deterministische Narrowingkalkül für $E_{\mathcal{R}}$–Unifikation mindestens einen Ast mehr durchläuft, als der lo–Narrowingkalkül für $E_{\mathcal{R}}$–Unifikation bevor das erste Endergebnis gefunden wird.

Satz 7.4 Es gibt ein ctn–trs $\mathcal{R} = (F, \Delta, R)$ und ein TES $\mathcal{R}' = (F \cup \Delta, R)$, so daß gilt

$$det(\text{lo–nar}\mathcal{I}_E(\mathcal{R})) \text{ ist single–success effizienter als } det(\text{nar}\mathcal{I}_E(\mathcal{R}')),$$

und es gibt kein ctn–trs $\bar{\mathcal{R}} = (\bar{F}, \bar{\Delta}, \bar{R})$ und TES $\bar{\mathcal{R}}' = (\bar{F} \cup \bar{\Delta}, \bar{R})$, so daß gilt

$$det(\text{nar}\mathcal{I}_E(\bar{\mathcal{R}}')) \text{ ist single–success effizienter als } det(\text{lo–nar}\mathcal{I}_E(\bar{\mathcal{R}})).$$

□

Beweis zu Satz 7.4

Es ist folgendes zu zeigen:

1. Es gibt ein h : Equ$(T(\Omega, \mathcal{V}))\times$ $Sub(\mathcal{V}, \Omega) \to$ Equ$(T(\Omega, \mathcal{V}))\times$ $Sub(\mathcal{V}, \Omega)$, so daß $det(\text{lo-nar}_E\mathcal{I}(\mathcal{R}))$ η–vergleichbar durch h mit $det(\text{nar}_E\mathcal{I}(\mathcal{R}'))$ ist.

2. Es gibt ein ctn–trs $\mathcal{R} = (F, \Delta, R)$, TES $\mathcal{R}' = (F \cup \Delta, R)$ und Kalkulandum $d \in$ Equ$(T(\Omega, \mathcal{V}))\times$ $Sub(\mathcal{V}, \Omega)$, so daß $det(\text{lo-nar}_E\mathcal{I}(\mathcal{R}))$ single-success effizienter ist als $det(\text{nar}_E\mathcal{I}(\mathcal{R}'))$.

3. Es gibt kein ctn–trs $\mathcal{R} = (F, \Delta, R)$, TES $\mathcal{R}' = (F \cup \Delta, R)$ und Kalkulandum $d \in$ Equ$(T(\Omega, \mathcal{V}))\times$ $Sub(\mathcal{V}, \Omega)$, so daß $det(\text{nar}_E\mathcal{I}(\mathcal{R}'))$ single-success effizienter ist als $det(\text{lo-nar}_E\mathcal{I}(\mathcal{R}))$.

1. Da wir uns auf $(E_{\mathcal{R}}, \Delta)$–Unifikatoren beschränken, sind die success-Mengen von $\text{nar}_E\mathcal{I}(\mathcal{R}')$ und $\text{lo-nar}_E\mathcal{I}(\mathcal{R})$ gleich und $det(\text{lo-nar}_E\mathcal{I}(\mathcal{R}))$ ist η–vergleichbar durch die Identitätsfunktion mit $det(\text{nar}_E\mathcal{I}(\mathcal{R}'))$.

2. Für das ctn–trs $\mathcal{R}$ aus Beispiel 7.1 gilt: $det(\text{lo-nar}_E\mathcal{I}(\mathcal{R}))$ ist single success effizienter als $det(\text{nar}_E\mathcal{I}(\mathcal{R}'))$. In dem Beispiel wurde zwar nicht gezeigt, daß es kein Kalkulandum

d gibt, so daß $(\eta(\,det(\text{nar}_E\mathcal{I}\,(\mathcal{R}'))))\,(d) < (\eta(\,det(\text{lo-nar}_E\mathcal{I}\,(\mathcal{R}))))\,(d)$, aber in Punkt 3 werde wir zeigen, daß es kein ctn–trs $\bar{\mathcal{R}}$ mit dieser Eigenschaft gibt.

3. Wir gehen von der Annahme aus, es gäbe ein ctn–trs $\mathcal{R} = (F, \Delta, R)$, TES $\mathcal{R}' = (F \cup \Delta, R)$ und Kalkulandum $d \in \text{Equ}\,(T(\Omega, \mathcal{V})) \times Sub\,(\mathcal{V}, \Omega)$, so daß

$$(\eta(det(\text{nar}_E\mathcal{I}\,(\mathcal{R}'))))\,(d) < (\eta(det(\text{lo-nar}_E\mathcal{I}\,(\mathcal{R}))))\,(d)$$

gilt. Das bedeutet, daß der deterministische Narrowing Kalkül das erste Endergebnis früher findet, als der deterministische lo–Narrowing Kalkül. Es gibt demnach einen Knoten $x \in \underline{node}\,(B(\text{nar}_E\mathcal{I}\,(\mathcal{R}'), d)) \setminus \underline{node}(B(\text{lo-nar}_E\mathcal{I}\,(\mathcal{R})))$ und $\alpha = \underline{label}_{B(\text{nar}_E\mathcal{I}\,(\mathcal{R}'),d)}\,(x)$ ist ein Endergebnis des Narrowing Kalküls. Außerdem gibt es *kein* bezüglich Variablenumbenennung zu α äquivalentes Endergebnis α', welches einen Knoten des lo–Narrowingbaumes beschriftet und links von x liegt, denn sonst wäre $(\eta(det(\text{lo-nar}_E\mathcal{I}\,(\mathcal{R}))))\,(d) < (\eta(det\,(\text{nar}_E\mathcal{I}\,(\mathcal{R}'))))\,(d)$. Lemma 7.3 besagt aber, daß es einen solchen Fall nicht gibt, weil es zu jedem Knoten y, der nur im Narrowingbaum liegt und dessen Beschriftung ein Endergebnis ist, immer ein bezüglich Variablenumbenennung äquivalentes Endergebnis im lo–Narrowingbaum gibt, welches einen Knoten des lo–Narrowingbaumes links von y beschriftet. Dies ist ein Widerspruch zu der obigen Annahme.

□

7.1.2 Multi Success Effizienz

In diesem Abschnitt wollen wir die Ergebnisse von Abschnitt 7.1.1 auf die multi success Effizienz erweitern. Der Vergleich von Effizienzen soll also nicht mehr lediglich für das Finden eines Endergebnisses stattfinden, sondern für das Finden der ersten k Endergebnisse, die $\sim$–verschieden zueinander sind. Sinnvollerweise wählen wir die Äquivalenzrelation $\sim$ so, daß Zwischenergebnisse, die lediglich durch Variablenumbenennung auseinander entstehen, in der Relation $\sim$ zueinander stehen (siehe Beispiel 6.7).

Satz 7.5 Es gibt ein ctn–trs $\mathcal{R} = (F, \Delta, R)$ und TES $\mathcal{R}' = (F \cup \Delta, R)$, so daß für ein $k \geq 2$ gilt:

$$det(\text{lo-nar}_E\mathcal{I}\,(\mathcal{R})) \text{ ist } k\text{–success effizienter als } det(\text{nar}_E\mathcal{I}\,(\mathcal{R}')),$$

es gibt kein ctn–trs $\bar{\mathcal{R}} = (\bar{F}, \bar{\Delta}, \bar{R})$ und TES $\bar{\mathcal{R}}' = (\bar{F} \cup \bar{\Delta}, \bar{R})$, so daß für ein $k \geq 2$ gilt:

$$det(\text{nar}_E\mathcal{I}\,(\bar{\mathcal{R}}')) \text{ ist } k\text{–success effizienter als } det(\text{lo-nar}_E\mathcal{I}\,(\bar{\mathcal{R}}));$$

wobei die Äquivalenzrelationen $\sim$ wie folgt definiert ist:
Für alle $d, d' \in D$ mit $d = (t, [v_1/t_1, \ldots v_n/t_n])$ gilt $d \sim d'$, g.d.w. es eine Variablenumbenennung ρ gibt, so daß $d' = (\hat{\rho}(t), [\rho(v_1)/\hat{\rho}(t_1), \ldots \rho(v_n)/\hat{\rho}(t_n)])$; wobei $\hat{\rho}$ die homomorphe Fortsetzung von $\rho : \mathcal{V} \to \mathcal{V}$ auf Terme $T(\Omega, \mathcal{V})$ ist.

□

<u>Beweis zu Satz 7.5:</u> Wir müssen wieder die drei Bedingunge wie im Beweis von Satz 7.4 zeigen.

1. $det\,(\text{lo-nar}_E\mathcal{I}\,(\mathcal{R}))$ ist η–vergleichbar durch die Identitätsfunktion mit $det\,(\text{nar}_E\mathcal{I}\,(\mathcal{R}'))$.

2. Wir erweitern das ctn–trs aus Beispiel 7.1 um eine fünfte Regel:

$$f(C) \quad \to \quad B \quad (5)$$

Betrachten wir nun die 2–success Effizienz für das Kalkulandum $d = equ\,(f(x), B)$, dann gilt:

$$\begin{aligned} (\eta(det\,(\text{lo–nar}_E\mathcal{I}\,(\mathcal{R}))))\,(d) &= 9. \\ (\eta(det\,(\text{nar}_E\mathcal{I}\,(\mathcal{R}'))))\,(d) &= 13. \end{aligned}$$

3. Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs, $\mathcal{R}' = (F \cup \Delta, R)$ ein TES und $d \in \text{Equ}\,(T(\Omega, \mathcal{V})) \times Sub\,(\mathcal{V}, \Omega)$ ein Kalkulandum, so daß für ein $k \geq 2$ gilt:

$$(\eta_k(det\,(\text{nar}_E\mathcal{I}\,(\mathcal{R}')), \sim))\,(d) < (\eta_k(det\,(\text{lo-nar}_E\mathcal{I}\,(\mathcal{R})), \sim))\,(d).$$

Dies bedeutet, daß das k–te $\sim$–verschiedene Enderergebnis vom deterministischen Narrowing Kalkül früher gefunden wird als vom deterministischen lo–Narrowing Kalkül. Es gibt also einen Knoten $x \in \underline{node}\,(B(\text{nar}_E\mathcal{I}\,(\mathcal{R}'), d) \backslash \underline{node}\,(B(\text{lo-nar}_E\mathcal{I}\,(\mathcal{R}), d))$, der mit einem Endergebnis beschriftet ist. Nach Lemma 7.3 gibt es dann aber einen Knoten y im lo–Narrowingbaum, der links von x liegt und der mit einem bezüglich Variablenumbenennung zu $\underline{label}_{\text{nar}_E\mathcal{I}\,(\mathcal{R}')}\,(x)$ äquivalenten Endergebnis beschriftet ist. Das Endergebnis am Knoten y wird demnach vom deterministischen lo–Narrowing Kalkül früher gefunden als das Endergebnis am Knoten x und es gilt nicht $(\eta_k(det\,(\text{nar}_E\mathcal{I}\,(\mathcal{R}')), \sim))\,(d) < (\eta_k(det\,(\text{lo-nar}_E\mathcal{I}\,(\mathcal{R})), \sim))\,(d)$. Dies ist eine Widerspruch zu der Annahmen, von der wir ausgegangen sind.

$\square$

7.2 LO–Narrowing vs. ULO–Narrowing

In diesem Abschnitt wollen wir den lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation (vgl. Definition 5.29) mit dem ulo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation (vgl. Definition 5.35) bezüglich der single und multi success Effizienz vergleichen. Wie in Kapitel 5 definiert unterscheidet sich der ulo-Narrowing Kalkül vom lo–Narrowing Kalkül darin, daß er innerhalb einer Berechnung Narrowingschritte und Unifikationsschritten durchführt und dann die in einem Endergebnis enthaltene Substitution φ bereits ein E–Unifikator ist. Außerdem sorgt der constructor consistency check dafür, daß ulo–Narrowing Berechnungen vorzeitig abbrechen, falls die beiden äußersten und am weitesten links vorkommenden Konstruktorsymbole eines equ–Terms sich unterscheiden.

Will man den lo–Narrowing Kalkül benutzen, um $E_{\mathcal{R}}$–Unifikation durchzuführen, so muß man für ein Endergebnis $(equ\,(t, s), \varphi)$ noch den most general unifier ψ von t und s finden, um diesen mit φ zu einem E–Unifikator zu komponieren. Diese Aufgabe wird im ulo–Narrowing Kalkül „automatisch“ während einer Berechnung durchgeführt. Vergleichen wir die Anzahl der Berechnungsschritte im lo–Narrowing Kalkül mit der im ulo–Narrowing Kalkül, ist es daher nicht fair, die Schritte des ulo–Narrowing Kalküls mitzuzählen, die für die Unifikation zuständig sind. Wir definieren daher einen ulo–ähnlichen Kalkül, den wir ulo′–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation nennen. Hierbei wird eine gewöhnlicher lo–Narrowingschritt gefolgt von den ulo–Narrowingschritten, die für Unifikation zuständig sind, zu einem ulo′–Narrowingschritt zusammengefaßt.

Wie auch im vorhergehenden Abschnitt, wiederholen wir hier noch einmal kurz die beiden Kalküle, die wir vergleichen wollen. Wir wiederholen zunächst den ulo–Narrowing Kalkül, und modifizieren ihn dann zum ulo′–Narrowing Kalkül, welcher dann tatsächlich bei den Vergleichen benutzt werden wird.

- lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation

  lo–nar$_E\mathcal{I}\,(\mathcal{R}) = (D, D_{\text{success}}, (J, <), (\vdash_j \ \mid j \in J))$ mit
  - $D = \text{Equ}\,(T(\Omega, \mathcal{V})) \times Sub\,(\mathcal{V}, \Omega)$
  - $D_{\text{success}} = \{(equ\,(\Theta_1, \Theta_2), \varphi) \mid \Theta_1, \Theta_2 \in T(\Omega, \mathcal{V}),\ \Theta_1$ und Θ_2 sind unifizierbar, $\varphi \in Sub\,(\mathcal{V}, \Delta)\}$
  - $J = R$
  - $<$ ist definiert wie beim lo–Narrowing Kalkül (siehe Definition 5.25).
  - für $j = (l \to r) \in J$, $\vdash_j = \overset{lo}{\rightsquigarrow}_{\mathcal{R}, l \to r}$ (siehe Definition 5.24)

- ulo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation

  ulo–nar$_E\mathcal{I}\,(\mathcal{R}) = (D', D'_{\text{success}}, (J', <'), (\vdash'_j \ \mid j \in J'))$ mit
  - $D' = T(\hat{F} \cup \Delta, \mathcal{V}) \times Sub\,(\mathcal{V}, \Delta)$
  - $D'_{\text{success}} = T(\Delta, \mathcal{V}) \times Sub\,(\mathcal{V}, \Delta)$
  - $J' = \hat{R} \cup \{bm\}$

- $<'$ ist definiert wie in Definition 5.35
- für $\xi \in J'$, $\vdash'_j = \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}},\xi}$

Wir definieren jetzt die ulo'–Narrowing Relation:

Definition 7.6 (ulo'–Narrowing Relation)
Sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs. Die *ulo'–Narrowing Relation von* $\hat{\mathcal{R}}$, bezeichnet durch $\rightsquigarrow_{\hat{\mathcal{R}}}$, ist wie folgt definiert: Für alle $t, t', s \in T(\hat{F} \cup \Delta, \mathcal{V})$ und $\psi, \varphi, \psi' \in Sub(\mathcal{V}, \Delta)$: $(t, \psi) \rightsquigarrow_{\hat{\mathcal{R}}} (s, \psi')$, wenn $t/impO(t) = equ(t_1, t_2)$, wobei $t_1, t_2 \in T(F \cup \Delta, \mathcal{V})$ und entweder $t_1[\varepsilon] \in F$ oder ($t_1[\varepsilon] \notin F$ und $t_2[\varepsilon] \in F$) und gilt

$$(t, \psi) \overset{lo}{\rightsquigarrow}_{\mathcal{R}} (t', \varphi) \overset{u}{\rightsquigarrow}{}^{*}_{\mathcal{R}(\Delta)} (s, \psi')$$

wobei es kein $d' \in T(\hat{F} \cup \Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Delta)$ mit $(s, \psi') \overset{u}{\rightsquigarrow}_{\mathcal{R}(\Delta)} d'$ gibt; d.h. (s, ψ') ist in Normalform bezüglich $\overset{u}{\rightsquigarrow}_{\mathcal{R}(\Delta)}$.

□

Wir schreiben $\rightsquigarrow_{\hat{\mathcal{R}},l\to r}$ wenn die Regel $(l \to r)$ der $\overset{lo}{\rightsquigarrow}_{\mathcal{R}}$–Relation angewendet wird.

Der ulo'–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation, bezeichnet durch ulo'–nar$_E\mathcal{I}$ $(\mathcal{R})$, ist nun der indizierte Kalkül

$$(D', D'_{\text{success}}, (J, <), (\vdash''_j \mid j \in J))$$

wobei für $j \in J$, $\vdash''_j = \rightsquigarrow_{\hat{\mathcal{R}},j}$.

7.2.1 Single Success Effizienz

Um zu entscheiden, ob ein Kalkül $\mathcal{C} = (D, D_{\text{success}}, \vdash)$ single success effizienter ist als ein Kalkül $\mathcal{C}' = (D', D'_{\text{success}}, \vdash')$, benötigen wir gemäß der Definition 6.4 eine Abbildung $h : D \to D'$, so daß gilt: $\hat{h}(D_{\text{success}}) \subseteq D'_{\text{success}}$. Wir möchten im folgenden zeigen, daß der ulo'–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation single success effizienter ist als der lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation. Wir suchen also eine Abbildung h, der Form:

- h : Equ $(T(\Omega, \mathcal{V})) \times Sub(\mathcal{V}, \Omega) \to T(\hat{F} \cup \Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Omega)$ so daß
- $\hat{h}(D_{\text{success}}) \subseteq D'_{\text{success}}$ mit $D_{\text{success}} = \{(equ(\Theta_1, \Theta_2), \varphi) \mid \Theta_1, \Theta_2 \in T(\Omega, \mathcal{V})$, Θ_1 und Θ_2 sind unifizierbar, $\varphi \in Sub(\mathcal{V}, \Omega)\}$ und $D'_{\text{success}} = T(\Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Delta)$.

Es ist zu bemerken, daß wir der Einfachheit halber ein h konstruieren, so daß der lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation η–vergleichbar durch h mit dem ulo'–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation ist. Genau genommen hätten wir aber die Eigenschaft der η–Vergleichbarkeit für die deterministischen Kalküle der eben genannten Kalküle zeigen müssen. Das folgende Lemma rechtfertigt diese Vereinfachung.

Lemma 7.7 Es seien $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ und $\mathcal{I}' = (D', D'_{\text{success}}, (J', <'), (\vdash'_j \mid j \in J'))$ indizierte Kalküle und $h : D \to D'$ eine Abbildung.

> Wenn $\mathcal{I}$ η–vergleichbar durch h mit $\mathcal{I}'$ ist, dann existiert ein $h' : \bar{D} \to \bar{D}'$ so daß $det(\mathcal{I})$ η–vergleichbar durch h' mit $det(\mathcal{I}')$ ist; wobei $\bar{D} = \{(d, M) \mid d \in D, M \subseteq I(d)\}^+$ und $\bar{D}' = \{(d', M) \mid d' \in D', M \subseteq I(d')\}^+$.

□

<u>Beweis:</u>

Es seien $det(\mathcal{I}) = (\bar{D}, \bar{D}_{\text{success}}, \models)$, $det(\mathcal{I}') = (\bar{D}', \bar{D}'_{\text{success}}, \models')$ und $d \in D$ ein Kalkulandum. Wir konstruieren h' aus h wie folgt:

Für alle $\Theta \in D_{\text{success}}$ mit $(d, I(d)) \models^* w(\Theta, I(\Theta))$ mit $w \in \{(d, M) \mid d \in D, M \subseteq I(d)\}^*$ definiere:

$$h'(w(\Theta, I(\Theta))) = w'(h(\Theta), I(h(\Theta)))$$

mit w' derart, daß es eine Berechnung durch $\models'$ gibt mit

$$(h(d), I(h(d))) \models'^* w'(h(\Theta), I(h(\Theta))).$$

Für alle anderen $\Theta \in \bar{D} \setminus \bar{D}_{\text{success}}$ definiere willkürlich $h'(\Theta) = (h(d), I(h(d)))$.

Es gilt dann: $h' : \bar{D} \to \bar{D}'$ und $\hat{h}'(\bar{D}_{\text{success}}) \subseteq \bar{D}_{\text{success}}{}'$, folglich ist $det(\mathcal{I})$ η–vergleichbar durch h' mit $det(\mathcal{I}')$.

□

Wie sieht nun eine solche Abbildung h für die beiden indizierten Kalküle lo–Narrowing für $(E_{\mathcal{R}}, \Delta)$–Unifikation und ulo′–Narrowing für $(E_{\mathcal{R}}, \Delta)$–Unifikation aus? Betrachten wir zwei Äste innerhalb der Berechnungsbäume dieser beiden Kalküle. In Abbildung 7.3 sehen wir den Zusammenhang zwischen lo–, ulo– und ulo′–Narrowing und der Abbildung h für ein Beispiel der ap–Funktion, wie sie in Beispiel 5.31 definiert worden ist.

Wie man sieht, entspricht die Abbildung h der Identitätsfunktion falls die ulo–Berechnung einer lo–Berechnung entspricht und es nur ein equ–Symbol an der root–Position des Terms t gibt (oberer Fall in Abbildung 7.3). Andererseits bewirkt h im unteren Fall der Abbildung, daß die Dekompositionsregeln der ulo–Narrowing Relation angewendet werden. Dies entspricht in Definition 5.34 den Fällen 1. bis 3. Gemäß unserer Notation können wir die Abbildung h dann wie folgt definieren.

Für alle $d \in \text{Equ}(T(\Omega, \mathcal{V})) \times Sub(\mathcal{V}, \Omega)$ ist $h(d) = \Theta$ wobei $d \overset{u}{\rightsquigarrow}{}^*_{\mathcal{R}(\Delta)} \Theta$ und Θ ist Normalform bezüglich $\overset{u}{\rightsquigarrow}_{\mathcal{R}(\Delta)}$.

Es ist offensichtlich, daß $h(d) \in T(\hat{F} \cup \Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Omega)$ für alle $d \in \text{Equ}(T(\Omega, \mathcal{V})) \times Sub(\mathcal{V}, \Omega)$. Nach Definition des lo–Narrowing Kalküls für $E_{\mathcal{R}}$–Unifikation hat ein Endergebnis d' die Form $(equ(t, s), \varphi)$ wobei t und s mit most general unifier ψ unifizierbar sind. Es soll $\hat{h}(d') \in T(\Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Omega)$ für alle Endergebnisse d' gelten, denn $T(\Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Omega)$ ist die Menge der Endergebnisse für den ulo'–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation. Das bedeutet, daß aus einem Zwischenergebnis $(equ(t, s), \varphi)$ durch

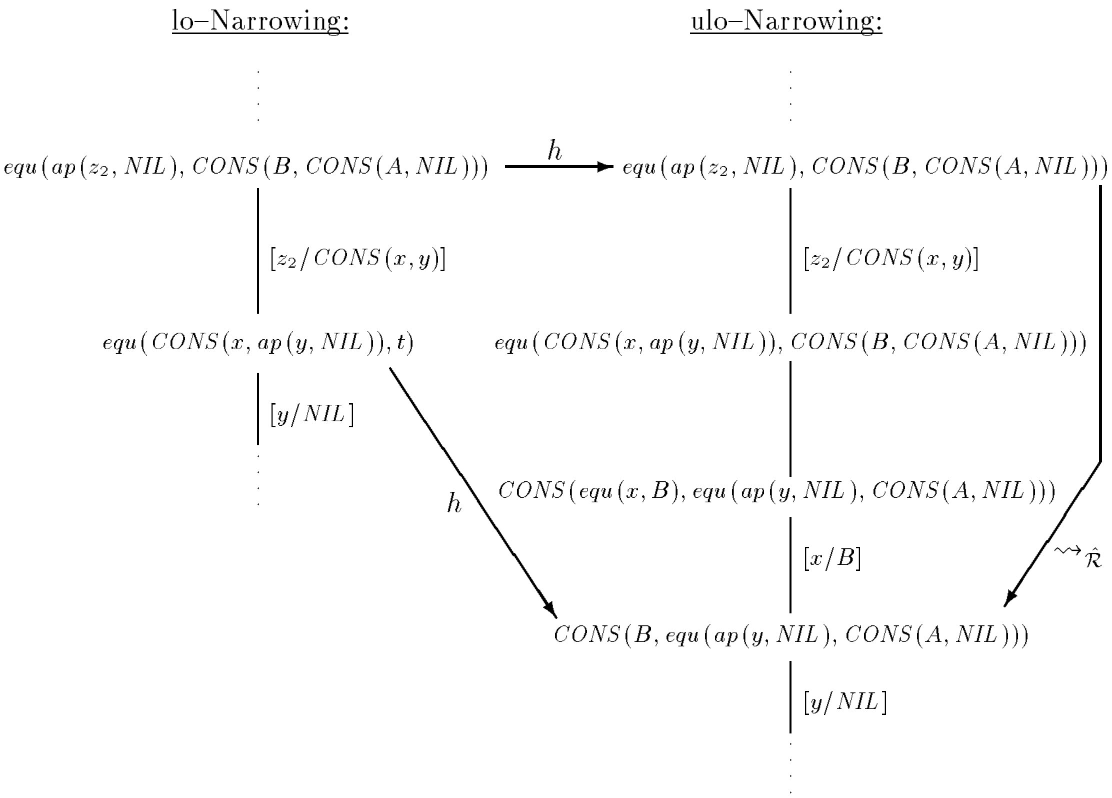

Abbildung 7.3: Äste aus lo– und ulo–Narrowingbaum

eine Berechnung durch $\overset{u}{\leadsto}{}^{*}_{\mathcal{R}(\Delta)}$ ein Term (Θ, ψ) werden muß, wobei $\Theta \in T(\hat{F} \cup \Delta, \mathcal{V})$ und ψ durch Komposition von φ und dem MGU von t und s entsteht. Mit anderen Worten, $\overset{u}{\leadsto}{}^{*}_{\mathcal{R}(\Delta)}$ muß die Unifikation von t und s durchführen. Genau dies ist aber nach Lemma 4.5 aus [FV94] der Fall. Demnach ist lo-nar$_E\mathcal{I}$ $(\mathcal{R})$ η–vergleichbar durch h mit ulo'-nar$_E\mathcal{I}$ $(\mathcal{R})$.

Intuitiv ist klar, daß die single success Effizienz des ulo'–Narrowing Kalküls für $E_{\mathcal{R}}$–Unifikation kleiner oder gleich der single success Effizienz des lo–Narrowing Kalküls für $E_{\mathcal{R}}$–Unifikation sein wird, denn es werden beim ulo'–Narrowing durch den constructor consistency check Teilbäume des lo–Narrowingbaumes eliminiert. Der folgende Satz beschreibt diese Eigenschaft.

Satz 7.8 Sei h die Abbildung wie oben definiert. Es gibt ein ctn–trs $\mathcal{R} = (F, \Delta, R)$, so daß gilt:

$$det(\text{ulo'-nar}_E\mathcal{I}(\mathcal{R})) \text{ ist single success effizienter als } det(\text{lo-nar}_E\mathcal{I}(\mathcal{R})),$$

und es gibt kein ctn–trs $\bar{R} = (\bar{F}, \bar{\Delta}, \bar{R})$, so daß gilt:

$$det(\text{lo-nar}_E\mathcal{I}(\mathcal{R})) \text{ ist single success effizienter als } det(\text{ulo'-nar}_E\mathcal{I}(\mathcal{R})),$$

□

Beweis:

Es sind wieder die drei gleichen Bedingungen wie im Beweis von Satz 7.4 zu zeigen.

1. Wie oben erklärt erfüllt unsere Abbildung h die Bedingung 1 aus Definition 6.4, i.e. lo–nar$_E\mathcal{I}(\mathcal{R})$ ist η–vergleichbar durch h mit ulo'-nar$_E\mathcal{I}(\mathcal{R})$.

2. Betrachte folgendes ctn–trs: $\mathcal{R} = (\{ap^{(2)}\}, \{A^{(0)}, NIL^{(0)}, CONS^{(2)}\}, R)$ mit R:

$$\begin{array}{rcll} ap(CONS(x,y),z) & \to & CONS(x, ap(y,z)) & (1) \\ ap(NIL, x) & \to & x & (2) \end{array}$$

und Kalkulandum $d = (equ(ap(x, NIL), CONS(A, NIL)), \varphi_\emptyset)$.

Eine Berechnung des deterministischen ulo'–Narrowing Kalküls benötigt 4 Schritte um das erste Endergebnis zu berechnen. Der determinsitsiche lo–Narrowing Kalkül dagegen kann kein Endergebnis berechnen, da der leftmost Ast im lo–Narrowingbaum unendlich ist. Es gilt also:

$$\begin{array}{rcl} (\eta(det(\text{ulo'-nar}_E\mathcal{I}(\mathcal{R}))))(d) & = & 4 \\ (\eta(det(\text{lo-nar}_E\mathcal{I}(\mathcal{R}))))(d) & = & \infty \end{array}$$

und damit gibt es ein ctn–trs $\mathcal{R}$, so daß der deterministische ulo'–Narrowing Kalkül single success effizienter ist als der deterministische lo–Narrowing Kalkül. Wir haben zwar wieder nicht gezeigt, daß es kein Kalkulandum d' gibt, so daß $(\eta(det(\text{lo-nar}_E\mathcal{I}(\mathcal{R}))))(d) < (\eta(det(\text{ulo'-nar}_E\mathcal{I}(\mathcal{R}))))(d)$, aber Teil 3 dieses Beweises zeigt, daß es kein ctn–trs mit dieser Eigenschaft gibt.

3. Es sei $\mathcal{I} = \text{lo-nar}_E\mathcal{I}(\mathcal{R})$, $\mathcal{I}' = \text{ulo'-nar}_E\mathcal{I}(\mathcal{R})$ und $x \in \underline{node}(B(\mathcal{I}, d))$, $x' \in \underline{node}(B(\mathcal{I}', d))$ jeweils die ersten Knoten, die der deterministische lo–Narrowing bzw. ulo'–Narrowing Kalkül für $(E_\mathcal{R}, \Delta)$–Unifikation findet und die jeweils mit einem Endergebnis beschriftet sind.

Fall 1: Es existiert weder ein solcher Knoten x noch ein Knoten x' für den das oben gesagt gilt; d.h. die deterministischen lo– und ulo'–Narrowing Kalküle für $E_\mathcal{R}$–Unifikation finden keine Lösung. Offensichtlich gilt dann

$$(\eta(det(\text{ulo'-nar}_E\mathcal{I}(\mathcal{R}))))(d) = (\eta(det(\text{lo-nar}_E\mathcal{I}(\mathcal{R}))))(h(d)) = \infty.$$

<u>Fall 2:</u> Es gibt keinen Knoten x für den das oben gesagt gilt, wohl aber einen entsprechenden Knoten x'; d.h. deterministisches lo–Narrowing findet kein Endergebnis, wohingegen deterministisches ulo'–Narrowing ein Endergebnis finde. Es ist dann

$$(\eta(\mathit{det}(\text{ulo'-nar}_E\mathcal{I}\,(\mathcal{R}))))\,(d) < (\eta(\mathit{det}(\text{lo-nar}_E\mathcal{I}\,(\mathcal{R}))))\,(h(d)) = \infty.$$

Den umgekehrten Fall kann es nicht geben, weil die Endergebnisse in den beiden Bäumen an entsprechenden Knoten zu finden sind, und der ulo'–Narrowingbaum weniger oder gleich viele unendliche Äste enthält wie der lo–Narrowingbaum.

<u>Fall 3:</u> Es gibt Knoten x, x' wie oben beschrieben, d.h. deterministisches lo–Narrowing und deterministisches ulo'–Narrowing finden jeweils ein Endergebnis. Betrachten wir die beiden Pfade, die zu den Knoten x bzw. x', also zu mit Endergebnissen beschrifteten Knoten führen. Nach Claim 1 im Beweis von Theorem 4.7 in [FV94] läßt sich eine Berechnung der Form

$$(\mathit{equ}\,(t,s), \varphi_\emptyset) \overset{lo}{\leadsto}{}^{k}_{\mathcal{R}}\ (\mathit{equ}\,(\zeta_t,\zeta_s), \varphi) \overset{u}{\leadsto}{}^{\text{eqpos}\,(\zeta_t,\zeta_s)}_{\mathcal{R}(\Delta)}\ (\zeta,\psi)$$

auch als Berechnung durch die ulo–Narrowing Relation der Form

$$(\mathit{equ}\,(t,s), \varphi_\emptyset) \overset{u}{\leadsto}{}^{k+\text{eqpos}\,(\zeta_t,\zeta_s)}_{\hat{\mathcal{R}}}\ (\zeta,\psi)$$

darstellen. Die Funktion $\mathit{eqpos}\,(\zeta_t,\zeta_s)$ liefert die maximale Anzahl der Berechnugsschritte, die durch $\overset{u}{\leadsto}_{\mathcal{R}(\Delta)}$ ausgeführt werden können (für genaue Definition siehe [FV94]). Es gelte nun $\underline{\mathit{label}}_{B(\mathcal{I},d)}\,(x) = (\mathit{equ}\,(\zeta_t,\zeta_s), \varphi)$ und $\underline{\mathit{label}}_{B(\mathcal{I}',h(d))}\,(x') = (\zeta,\psi)$. Die $\mathit{eqpos}\,(\zeta_t,\zeta_s)$ Schritte, welche der ulo–Narrowing Pfad länger ist als der lo–Narrowing Pfad, sind genau solche Schritte, die im ulo'–Narrowing Kalkül zu einem ulo'–Narrowingschritt zusammengefaßt sind. Eine Berechnung der obigen Form hat dort also diese Form

$$(\mathit{equ}\,(t,s), \varphi_\emptyset) \leadsto^{k}_{\hat{\mathcal{R}}} (\zeta,\psi)$$

Die beiden Pfade zu den Endergebnissen sind also jeweils k Schritte lang. Da der deterministische Kalkül einen depth–first left–to–right tree traversal durch einen Berechnungsbaum durchführt, kann ein Unterschied in der single success Effizienz also nur links von dem jeweiligen Pfad im Berechnungsbaum entstanden sein. Wir greifen dazu wieder auf die Definition 7.2 zurück. Ein Unterschied in der single success Effizienz zwischen $\mathcal{I}$ und $\mathcal{I}'$ kann nur dadurch entstehen, daß beim ulo'–Narrowingbaum Teilbäume des entsprechenden lo–Narrowingbaumes auf Grund des constructor consistency checks wegfallen. Wir definieren eine Menge $\mathit{Diff}\,(x)$, die genau solche Knoten enthält, an denen ein constructor consistency check dafür sorgt, daß die Berechnung des ulo'–Narrowing Kalküls dort abbricht.

$$\begin{aligned}\mathit{Diff}\,(x) = \{y \in \underline{\mathit{node}}\,(B(\mathcal{I}',d)) \mid \underline{\mathit{label}}_{B(\mathcal{I}',d)}\,(y) = (\mathit{equ}\,(t_1,t_2),\varphi) \text{ mit } t_1[\varepsilon], t_2[\varepsilon] \in \Delta \text{ und}\\ t_1[\varepsilon] \neq t_2[\varepsilon] \text{ und es gibt ein } (\Theta,\psi) \text{ mit } \underline{\mathit{label}}_{B(\mathcal{I}',d)}\,(y) \overset{lo}{\leadsto}_R (\Theta,\psi), \}\end{aligned}$$

Es gibt folgende beiden Fälle:

<u>Fall a:</u> $\mathit{Diff}\,(x') \cap \mathit{left}\,(x',\mathcal{I}',h(d)) = \emptyset$; d.h. es gibt keinen Knoten links von den Pfad, der zum ersten Endergebnis führt, den der deterministische ulo'–Narrowing Kalkül für

$E_{\mathcal{R}}$–Unifikation findet, an dem ein constructor consistency check fehl schlägt. Damit sind die beiden Bäume bis auf die Unifikationsschritte, die wir nicht zählen, identisch. Es ist damit:

$$(\eta(\mathit{det}(\text{ulo'-nar}_E\mathcal{I}\ (\mathcal{R}))))\ (d) = (\eta(\mathit{det}(\text{lo-nar}_E\mathcal{I}\ (\mathcal{R}))))\ (h(d))$$

<u>Fall b:</u> $\mathit{Diff}(x') \cap \mathit{left}(x', \mathcal{I}', h(d)) \neq \emptyset$. Es gibt Teilbäume des lo–Narrowingbaumes, die im ulo'–Narrowingbaum nicht mehr auftreten, weil der constructor consistency check für den Abbruch der ulo'–Narrowing Rechnung an solchen Stellen gesorgt hat. Die Rechnung des deterministischen ulo'–Narrowing ist damit kürzer als die entsprechende Rechnung des deterministischen lo–Narrowing und es gilt somit:

$$(\eta(\mathit{det}(\text{ulo'-nar}_E\mathcal{I}\ (\mathcal{R}))))\ (d) < (\eta(\mathit{det}(\text{lo-nar}_E\mathcal{I}\ (\mathcal{R}))))\ (h(d))$$

□

Wie auch schon im vorhergehenden Vergleich zwischen Narrowing und lo–Narrowing, werden wir im nächsten Abschnitt unser Resultat für die single success Effizienz auf die multi success Effizienz erweitern.

7.2.2 Multi Success Effizienz

Wie leicht einzusehen ist, gilt das Ergebnis für die single success Effizienz, i.e. single success Effizienz des ulo'–Narrowing Kalküls für $E_{\mathcal{R}}$–Unifikation ist kleiner gleich der single success Effizienz des lo–Narrowing Kalküls für $E_{\mathcal{R}}$–Unifikation, auch für die multi success Effizienz.

Satz 7.9 Es sei $\mathcal{R} = (F, \Delta, R)$ ein ctn–trs, $d = (\mathit{equ}(t, s), \varphi_\emptyset)$ mit $t, s \in T(F \cup \Delta, \mathcal{V})$ ein Kalkulandum und $k \geq 1$ eine ganze Zahl. Es gilt:

$$(\eta_k(\mathit{det}(\text{ulo'-nar}_E\mathcal{I}\ (\mathcal{R})), v))\ (d) \leq (\eta_k(\mathit{det}(\text{lo-nar}_E\mathcal{I}\ (\mathcal{R})), v))\ (h(d))$$

wobei die Äquivalenzrelation v wie folgt definiert ist:
Für alle $d, d' \in D$ mit $d = (t, [v_1/t_1, \ldots v_n/t_n])$ gilt $d \sim d'$, g.d.w. es eine Variablenumbenennung ρ gibt, so daß $d' = (\hat{\rho}(t), [\rho(v_1)/\hat{\rho}(t_1), \ldots \rho(v_n)/\hat{\rho}(t_n)])$; wobei $\hat{\rho}$ die homomorphe Fortsetzung von $\rho : \mathcal{V} \to \mathcal{V}$ auf Terme $T(\Omega, \mathcal{V})$ ist.

□

Der Beweis erfolgt ganz analog zum Beweis von Lemma 7.8. Die Knoten x und x' sind lediglich durch die jeweils k–ten Endergebnisse, welche der deterministische lo–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation bzw. der ulo'–Narrowing Kalkül für $(E_{\mathcal{R}}, \Delta)$–Unifikation findet, zu ersetzen.

Kapitel 8

Abschließende Bemerkungen

Wir haben in dieser Arbeit eine formale Definition eines Kalküls gegeben. Als speziellen Kalkül haben wir den indizierten Kalkül definiert, für den bei gegebenem Kalkulandum ein Berechnungsbaum eindeutig festgelegt wird. Der deterministische Kalkül beschreibt einen depth-first left–to–right tree traversal durch diesen Baum. Er kann wegen seines Determinismus als Grundlage für eine abstrakte Maschine und diese wiederum als Grundlage für die Implementation einer funktional–logischen Programmiersprache auf einer Rechenmaschine, benutzt werden. Es ist aber durchaus denkbar, einen *breadth–first* left–to–right tree traversal durch den Berechnungsbaum als deterministischen Kalkül zu definieren. Wie einfach dies auf der Ebene der Kalkül möglich ist, betrachten wir anhand des folgenden Beispiels:

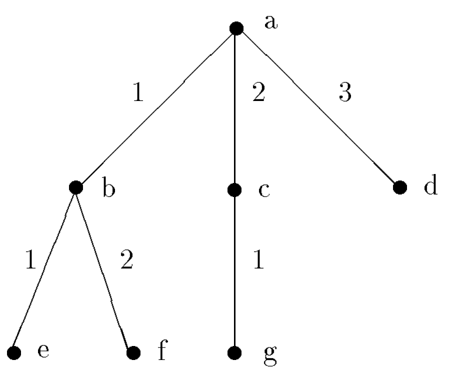

Abbildung 8.1: Berechnungsbaum eines indizierten Kalküls

$$
\begin{array}{ll}
 & (a, \{1,2,3\}) \\
\models_{\mathrm{bf}} & (a, \{2,3\})(b, \{1,2\}) \\
\models_{\mathrm{bf}} & (a, \{3\})(b, \{1,2\})(c, \{1\}) \\
\models_{\mathrm{bf}} & (a, \emptyset)(b, \{1,2\})(c, \{1\})(d, \emptyset) \\
\models_{\mathrm{bf}} & (b, \{1,2\})(c, \{1\})(d, \emptyset) \\
\models_{\mathrm{bf}} & (b, \{2\})(c, \{1\})(d, \emptyset)(e, \emptyset) \\
\models_{\mathrm{bf}} & (b, \emptyset)(c, \{1\})(d, \emptyset)(e, \emptyset)(f, \emptyset) \\
\models_{\mathrm{bf}} & (c, \{1\})(d, \emptyset)(e, \emptyset)(f, \emptyset) \\
\models_{\mathrm{bf}} & (c, \emptyset)(d, \emptyset)(e, \emptyset)(f, \emptyset)(g, \emptyset) \\
\models_{\mathrm{bf}} & (d, \emptyset)(e, \emptyset)(f, \emptyset)(g, \emptyset) \\
\models_{\mathrm{bf}} & (e, \emptyset)(f, \emptyset)(g, \emptyset) \\
\models_{\mathrm{bf}} & (f, \emptyset)(g, \emptyset) \\
\models_{\mathrm{bf}} & (g, \emptyset)
\end{array}
$$

Tatsächlich ist es dann möglich alle Endergebnisse eines Berechnungsbaumes zu finden; d.h. wenn man von vornherein weiß, daß es k Endergebnisse in dem gegebenen Berechnungsbaum gibt, so wird der breadth–first left–to–right tree traversal diese k Endergebnisse finden und dann abbrechen. Im Allgemeinen kennt man aber ein entsprechendes k nicht, so daß im Falle einen unendlichen Berechnungsbaumes der breadth–first left–to–right Baumlauf nicht terminiert. Der Vorteil der depth–first Variante gegenüber liegt offensichtlich darin, daß man nicht *vor* Finden jeglicher Endergebnisse in einen unendlichen Teilbaum geraten kann und damit überhaupt kein Endergebnis berechnen kann. Das Problem des breadth–first liegt in der Implementation, die sehr speicherintensiv den Berechnungsbaum abarbeitet. Es wäre also interessant, eine abstrakte Maschine zu entwickeln, die mit minimaler Speicherbelegung einen solchen Baumlauf realisiert. Die universellen Unifikationsalgorithmen aus Kapitel 4 könnten dann auf ihre „Laufzeit" bezüglich einer depth–first und einer breadth–first abstrakten Maschine hin untersucht werden. In der Tat existiert eine solche abstrakte Maschine für den depth–first Fall schon (die sogenannte „twin unification machine", siehe [FVW94, Faß95]).

In Kapitel 5 haben wir unsere formalen Kalküle durch verschiedene Methoden zum Finden von $E_{\mathcal{R}}$–Unifikatoren instanziiert. Es wäre interessant, andere Narrowing–Methoden, beispielsweise LSE–Narrowing [WBK94] als indizierte Kalküle darzustellen, um dann die in Kapitel 6 definierten Effizienzmaße darauf anzuwenden und Ergebnissen für lo– bzw. ulo–Narrowing zu vergleichen. Außerdem sollten bei solchen Vergleichen weitere Effizienzmaße in Betracht gezogen werden; Beispielsweise die success density Effizienz (Abschnitt 6.3.1), oder die infinite path Effizienz (Abschnitt 6.3.2) oder auch neue, andere Effizienzmaße. Wie schon am Anfang von Kapitel 7 erwähnt, ist die Optimalität, i.e. daß zwei verschiedene Berechnungen auch zwei verschiedene Ergebnisse liefern, eine repräsentatives Effizienzmaß. Es sollte daher auch als Maß für unsere Kalküle definiert werden.

Literaturverzeichnis

[AEH94] S. Antoy, R. Echahed, and M. Hanus. A needed narrowing strategy. In *Proceedings 21st ACM Symposium on Principles of Programming Languages, Portland*, pages 268–279, 1994.

[AFJRV93] M. Alpuente, M. Falaschi, M. Jose-Ramis, and G. Vidal. Narrowing approximations as an optimization for equational logic programs. In *Proceedings of the 5th International Symposium on Programming Language Implementation and Logic Programming, LNCS 714*, pages 391–409. Springer-Verlag, 1993.

[Aug84] L. Augustsson. A compiler for lazy ML. In *Proceedings of the ACM Symposium on Lisp and Functional Programming, Austin*, pages 218–227, August 1984.

[BKW92] A. Bockmayer, S. Krischer, and A. Werner. An optimal narrowing strategy for general canonical systems. In *Proceedings of the 3rd International Workshop on Conditional Term Rewriting Systems, LNCS 656*, pages 483–497. Springer-Verlag, 1992.

[BKW94] A. Bockmayr, S. Krischer, and A. Werner. Narrowing strategies for arbitrary canonical systems. *Fundamenta Informaticae*, 1994. To appear.

[CC79] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. *ACM Symposium on Principles of Programming Languages, San Antonio*, 1979.

[CL91] M. M. T. Chakravarty and H. C. R. Lock. The implementation of lazy narrowing. In *Proceedings of the 3th International Symposium on Programming Language Implementation and Logic Programming, LNCS 528*, pages 123–134. Springer-Verlag, 1991.

[CM81] W.F. Clocksin and C.S. Mellish. *Programming in PROLOG*. Springer-Verlag-Verlag, 1981.

[Der91] N. Dershowitz. A taste of rewrite systems. In P. E. Lauer, editor, *Functional Programming, Concurrency, Simulation and Automated Reasoning, LNCS 693*, pages 199–228. Springer-Verlag, 1991.

[DJ90] N. Dershowitz and J.P. Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B*, chapter 6, pages 243–320. Elsevier, 1990.

[DL86] D. DeGroot and G. Lindstrom. *Logic Programming: Functions, Relations and Equations.* Prentice Hall, 1986.

[Ech88] R. Echahed. On completeness of narrowing strategies. In *Proceedings of the 13th Conference on Trees in Algebra and Programming, LNCS 299*, pages 89–101. Springer-Verlag, 1988.

[EV91] J. Engelfriet and H. Vogler. Modular tree transducers. *Theoret. Comput. Sci.*, 78:267–304, 1991.

[Fag84] F. Fages. Associative-commutative unification. In *Proceedings of the 7th International Conference on Automated Deduction (CADE)*, pages 194–208, Napa, California, may 1984.

[Faß95] H. Faßbender. *E–Unifikation für primitiv–rekursive Baumfuktionen.* PhD thesis, University of Ulm, March 1995. Abteilung Theoretische Informatik.

[Fay79] M. Fay. First-order unification in an equational theory. In *Proceeding of the 4th workshop on automated deduction, Austin*, pages 161–167. Academic Press, 1979.

[Fri85] L. Fribourg. SLOG: A logic programming language interpreter based on clausual superposition and rewriting. In *Proceedings of the IEEE International Symposium on logic programming*, pages 172–184. IEEE Computer Society Press, 1985.

[FV92] H. Faßbender and H. Vogler. A universal unification algorithm based on unification-driven leftmost outermost narrowing. Technical report, University of Ulm, 1992. Ulmer Informatik-Berichte Nr. 92-07.

[FV94] H. Faßbender and H. Vogler. A universal unification algorithm based on unification-driven leftmost outermost narrowing. *Acta Cybernetica*, 11:139–167, 1994.

[FVW94] H. Faßbender, H. Vogler, and A. Wedel. Implementation of a deterministic partial E-unification algorithm for macro tree transducers. Technical report, University of Ulm, 1994. Ulmer Informatik-Berichte Nr. 94-04.

[GLMP91] E. Giovannetti, G. Levi, C. Moiso, and C. Palamidessi. Kernel-LEAF: A logic plus functional language. *J. Comput. Syst. Sci.*, 42:139–185, 1991.

[GM86] J.A. Goguen and J. Meseguer. Equality, types and generic modules for logic programming. In D. DeGroot and G. Lindstrom, editors, *Logic Programming: Functions, Relations and Equations*, pages 295–363. Prentice Hall, 1986.

[GMW79] M.J. Gordon, R. Milner, and C.P. Wadsworth. *Edinburgh LCF*, volume 78 of *LNCS*. Springer-Verlag, 1979.

[GS87] J.H. Gallier and W. Snyder. A general complete E-unification procedure. In P. Lescanne, editor, *Rewriting techniques and applications RTA 87, LNCS 256*, pages 216–227. Springer-Verlag, 1987.

[Han92] M. Hanus. Incremental rewriting in narrowing derivations. In *Proceedings of the 3rd International Conference on Algebraic and Logic Programming, LNCS 632*, pages 228–243. Springer-Verlag, 1992.

[Han94] M. Hanus. The integration of functions into logic programming: from theory to practice. *The Journal of Logic Programming*, pages 583–628, 1994.

[HO80] G. Huet and D.C. Oppen. Equations and rewrite rules: a survey. In R. Book, editor, *Formal Language Theory: Perspectives and Open Problems*. Academic Press, New York, 1980.

[Hö87] S. Hölldobler. *Foundations of Equational Logic Programming, LNCS 353*. Springer-Verlag, 1987.

[Hue80] G. Huet. Confluent reductions: abstract properties and applications to term rewriting systems. *J. Assoc. Comput. Mach.*, 27:797–821, 1980.

[Hul80] J.M. Hullot. Canonical forms and unification. In *Proceedings of the 5th conference on automated deduction, LNCS 87*, pages 318–334. Springer-Verlag, 1980.

[Hup78] U. Hupbach. Rekursive Funktionen in mehrsortigen Algebren. *Elektron. Informationsverarb. Kybernetik*, 15:491–506, 1978.

[Joh84] T. Johnsson. Efficient compilation of lazy evaluation. In *Proceedings of the 11th ACM Conference on Compiler Construction, Montreal*, pages 58–69, June 1984.

[Jon87] S.L. Peyton Jones. *The Implementation of Functional Programming Languages*. Series in Computer Science. Prentice-Hall, 1987.

[Kow74] R. Kowalski. Predicate logic as a programming language. *Information Processing*, 74:569–574, 1974.

[Kri90] S. Krischer. Vergleich und Optimierung von Narrowing–Strategien. Master's thesis, University of Karlsruhe, 1990. Diplomarbeit, University of Karlsruhe.

[Lan75] D. S. Lankford. Canonical inference. Technical report, Department of Mathematics, Southwestern University, Georgetown, Texas, 1975.

[Loc92] H. C. R. Lock. *The Implementation of Functional Logic Programming Languages*. PhD thesis, Technical University of Berlin, 1992. Also available as GMD Report 208, Oldenbourg Verlag, München.

[LSSU79] M. Livesey, J. Siekmann, P. Szabo, and E. Unvericht. Unification problems for combinations of associativity, commutativity, distributivity and idempotence axioms. In *4th Workshop on Automated Deduction*, pages 161–167, Austin, Texas, feb 1979.

[Mak77] G. S. Makanin. The problem of solvability of equations in a free semigroup. *Math. USSR Sbornik*, 32(2):129–198, 1977.

[Man95] S. Maneth. Implementation einer einfachen funktionalen Programmiersprache durch Termersetzung. Programmierbeleg zur Vorlesung Objektorientierte Programmierung, Technische Universität Dresden, 1995.

[Mat70] Y. Matiyasevich. Diophantine representation of recursively enumerable predicates. In *Proceedings of the Second Scandinavian Logic Symposium*. North-Holland, 1970.

[MM82] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages Systems*, 4:258–282, 1982.

[Mü91] A. Mück. CAMEL: An extension of the categorical abstract machine to compile functional/logic programs. In *Proceedings of the 4th International Symposium on Programming Language Implementation and Logic Programming, LNCS 631*, pages 341–354. Springer-Verlag, 1991.

[Plo72] G.D. Plotkin. Building-in equational theories. *Machine Intelligence*, 7:73–90, 1972.

[RAMN88] M. Rodriguez-Artalejo and J.J. Moreno-Navarro. BABEL: a functional and logic programming language based on constructor discipline and narrowing. In *Conference on algebraic and logic programming*. Springer-Verlag, 1988. LNCS 343.

[RAMN92] M. Rodriguez-Artalejo and J.J. Moreno-Navarro. Logic programming with functions and predicates: the language BABEL. *J. Logic Programming*, 12:189–223, 1992.

[Red85] U. S. Reddy. Narrowing as the operational semantics of functional languages. In *Proceedings of the IEEE International Symposium on Logic Programming*, pages 138–151, Boston. IEEE Computer Society Press, 1985.

[Rob65] J.A. Robinson. A machine-oriented logic based on the resolution principle. *J. Assoc. Comput. Mach.*, 20:23–41, 1965.

[Sie87] J.H. Siekmann. *Unification and Matching Problems*. PhD thesis, Essex University, 1987.

[Sla74] J. R. Slagle. Automatic theorem proving in theories with simplifiers, commutativity and associativity. *J. Assoc. Comput. Mach.*, 21:622–642, 1974.

[Sti75] M. E. Stickel. A complete unification algorithm for associative-commutative functions. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 71–76, Tbilisi, 1975.

[Sti81] M. Stickel. A unification algorithm for associative-commutative functions. *Journal of the Assoc. Comput. Sci.*, 28:423–434, 1981.

[SY86] P.A. Subrahmanyam and J.H. You. FUNLOG:a computational model integrating logic programming and functional programming. In D. DeGroot and G. Lindstrom, editors, *Logic Programming: Functions, Relations and Equations*, pages 157–198. Prentice Hall, 1986.

[Thi92] M. Thielscher. AC1–Unifikation in der linearen logischen Programmierung. Diploma thesis, TH Darmstadt, September 1992.

[Tur85] D.A. Turner. Miranda - a non-strict functional language with polymorphic types. In J.P. Jouannaud, editor, *Conference on Functional Programming Languages and Computer Architecture, Nancy*, pages 1–16. Springer-Verlag, 1985. LNCS 201.

[WBK94] A. Werner, A. Bockmayr, and S. Krischer. How to realize LSE narrowing. In *Proceedings of the 4th Conference on Algebraic and Logic Programming, Madrid, LNCS 850*, pages 59–76. Springer-Verlag, 1994.