Calculi and Efficiency Measures for Narrowing Techniques

Sebastian Maneth and Heiko Vogler

Grundlagen der Programmierung
Institut für Softwaretechnik I, Fakultät Informatik
Technische Universität Dresden
01062 Dresden, Germany
email: {maneth,vogler}@inf.tu-dresden.de

Abstract

Narrowing serves as operational semantics of functional–logic programming languages. Over the past two decades a vast variety of narrowing techniques have been proposed, differing from each other in the class of equational theories they are applicable to and in their efficiency to evaluate a given goal. On the latter part, i.e., the efficiency, merely vague statements can be found in the existing literature and hardly any formal investigation has been carried out comparing different narrowing techniques. In this paper we use the concept of calculi which allows us to formalize various narrowing techniques in a uniform way. Based on this concept we introduce four efficiency measures and apply two of them to two different pairs of narrowing techniques. The results make clear statements about the difference in efficiency for the narrowing techniques involved.

1 Introduction

Every implementation of a functional–logic programming language has to deal with E–unification problems. In general, the E–unification problem is undecidable, as can be seen easily by coding Post's correspondence problem as an E–unification problem [Faß95a]. However, there are semi–decision algorithms which enumerate E–unifiers for a restricted class $\mathcal{E}$ of equational theories. If, e.g., we restrict $\mathcal{E}$ to the class of equational theories which are induced by canonical, i.e., confluent and terminating, term rewriting systems, then narrowing [Sla74, Lan75] yields a complete set of E–unifiers [Hul80]. Other narrowing techniques such as leftmost outermost narrowing [Ech88], unification–driven leftmost outermost narrowing [FV94] or needed narrowing [AEH94] are complete for other classes of equational theories, see [Han94] for a survey.

In this paper we will compare the efficiencies of the following three different narrowing techniques in a formal way:

- narrowing
- leftmost outermost narrowing
- unification–driven leftmost outermost narrowing.

In the existing literature on narrowing one can mainly find the following approaches for efficiency comparisons:

- empirical run-time of the implementation of a specific narrowing technique; i.e., results in seconds or a similar unit. Such results are heavily dependent on the implementation, the machine and many other factors. They can not be verified by the reader. Moreover run-times can only be given for particular examples (see e.g. [AFJRV93], [BKW94], [Han94]);
- number of nodes of narrowing trees for particular examples or breadth of narrowing trees up to some height (see e.g. [BKW94], [Kri90], [WBK94]);
- statements on the optimality of a given narrowing technique; usually optimality in this sense means that different derivations will yield different success forms. This measure is independent of the implementation and of a specific example. It is therefore a general measure for the efficiency of narrowing methods (see [AEH94], [BKW92], [BKW94], [Han92], [WBK94]).

Let us now explain on which abstract level we will formalize our efficiency measures, and discuss briefly the formal concepts which are involved in our comparison.

Usually, the narrowing relations as well as the operational semantics of declarative programming languages are formalized by a nondeterministic binary computation relation. However, every implementation on an abstract or real machine has to be deterministic. Thus, one has to develop a deterministic version of the computation relation which can serve as basis (and as reference for correctness proofs) for the implementation on a machine. This deterministic computation relation will serve as basis for our efficiency measures. Roughly speaking, we will count the number of steps which are performed by the deterministic relation to compute an E-unifier.

In order to formalize the nondeterministic and the deterministic computation relations, we will proceed as follows. First, we use the basic concept of calculi in which the nondeterministic relation can be formalized. Second, we introduce a schematic transformation on calculi which yields deterministic versions of the given calculi together with their deterministic computation relations. Let us now explain this in more detail.

A calculus is a triple $(D, D_{\text{success}}, \vdash)$ where D is a set of derivation forms, $\vdash$ is a binary relation on D, called derivation relation, and D_{success} is a subset of D, the elements of which are called success forms. Apart from success forms, this notion is closely related to Huet's abstract reduction systems [Hue80] (cf. also [Klo92]). If two derivation forms d and d' are related by $\vdash$, then we say that d derives to d' by $\vdash$. The set D_{success} allows us to specify certain elements of D as success forms. All derivations starting with some derivation form d, which we call calculandum, can be incorporated into a derivation tree. To construct a derivation tree unambiguously, we need to fix an arrangement of the nodes which are on the same level. For this purpose we extend the notion of calculi to the notion of indexed calculi. An indexed calculus uses a so called index set J and a total order $<$ on J to partition the derivation relation $\vdash$ into a family $(\vdash_j \mid j \in J)$ of derivations. Then, for every derivation form d and index j there is at most one d', such that $d \vdash_j d'$ and thus, a derivation tree for an indexed calculus $\mathcal{I}$ and a calculandum d is unambiguous.

Each of the three narrowing techniques which we will investigate, will be formalized as an indexed calculus, and in particular, the success forms are the desired E-unifiers. However, indexed calculi are still nondeterministic. Therefore we define a schematic transformation which yields the deterministic calculus for the given indexed calculi: it implements a depth–first left–to–right tree traversal over derivation trees of the indexed calculi.

On the basis of indexed calculi and their deterministic versions we define four efficiency measures:

- *single success efficiency*
  Measures the number of derivation steps of the deterministic calculus for an indexed calculus, which are needed to find the *first* E-unifier.

- *multi success efficiency*
  Measures the number of derivation steps of the deterministic calculus for an indexed calculus, which are needed to find the first k pairwise different E-unifiers. The equivalence of derivation forms will be expressed by an equivalence relation $\sim$.

- *success density efficiency*
  Measures the number of pairwise different success forms up to a certain level in the derivation tree.

- *infinite path efficiency*
  Measures the number of infinite paths in the derivation tree of an indexed calculus and a calculandum d. One indexed calculus is more infinite path efficient, the less infinite paths it contains.

We will use the single success efficiency and the multi success efficiency to compare

- narrowing with leftmost outermost narrowing and

- leftmost outermost narrowing with unification–driven leftmost outermost narrowing.

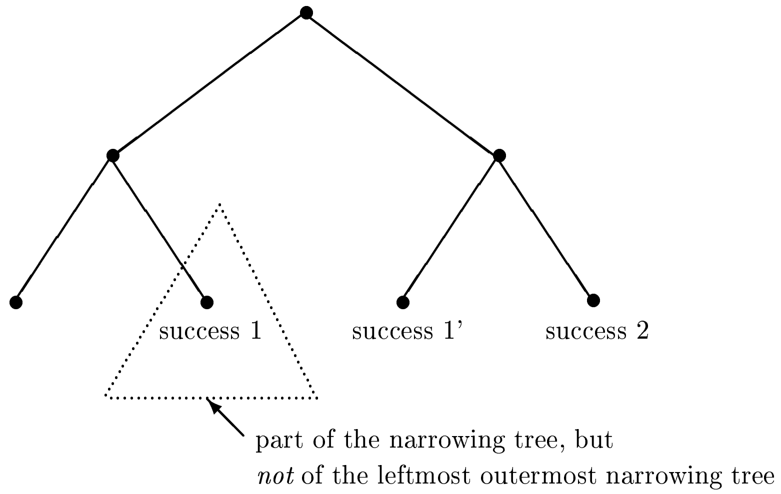

Figure 1: Narrowing and leftmost outermost narrowing trees.

Now one might ask, why such an investigation about the comparison between narrowing and leftmost outermost narrowing with respect to the single success efficiency makes sense. It seems trivially true that leftmost outermost narrowing is more efficient than narrowing, because leftmost outermost narrowing only takes specific narrowing occurrences into account and thereby, the size of the narrowing tree is reduced. But it is not that obvious. Let us take a look at Figure 1, which depicts a narrowing tree. Success 1 can not be derived by leftmost outermost narrowing, but only by narrowing. The success forms 2 and 1' are derivable by both calculi and successes 1 and 1' are equivalent. Clearly, if such a situation as depicted in Figure 1 would exist, then leftmost outermost narrowing is not more efficient than narrowing, because narrowing will find the first E-unifier earlier than leftmost outermost narrowing. However, we will prove that such a situation can not occur.

This paper is organized as follows: Section 2 contains important notations which we use throughout the paper. Section 3 introduced the notions of calculi, indexed calculi, derivation trees and deterministic calculi. In Section 4 we recall important notions concerning E–unification. Section 5 contains four instances of indexed calculi, namely the reduction calculus for $E_{\mathcal{R}}$–unification, the narrowing calculus for $E_{\mathcal{R}}$–unification, the leftmost outermost narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification, and the unification–driven leftmost outermost narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification. In Section 6 we define the four different efficiency measures, two of which we will use in Section 7 to compare narrowing with leftmost outermost narrowing and leftmost outermost narrowing with unification–driven leftmost outermost narrowing. Section 8 draws some conclusions.

2 Preliminaries

In this section we want to fix some general notations and definitions that will be used throughout the rest of this paper.

2.1 General Notations

The *set of non-negative integers* is denoted by $\mathbb{N}$. The *empty set* is denoted by $\emptyset$. For some $j \in \mathbb{N}$, $[j]$ denotes the set $\{1, \ldots, j\}$; therefore $[0] = \emptyset$. For a finite set A, the *cardinality* of A will be denoted by $card(A)$. For a set A, the *set of words over* A is denoted by A^*. The empty word is denoted by ε. For a set M and an element $p \in M$, the set M *without* p, denoted by $M \setminus p$, is the set $\{m \in M \mid m \neq p\}$, and similarly, for a subset $N \subseteq M$, $M \setminus N$ denotes the set $\{m \in M \mid m \notin N\}$.

Let $f : A \to B$ and $g : B \to C$ be functions. The *composition* $g \circ f : A \to C$ is defined by $(g \circ f)(x) = g(f(x))$ for all $x \in A$.

Let $f : M \to N$ be a function. A function $g : U \to N$ is called *limitation of* f *(to* U*)*, if $U \subseteq M$ and $f(x) = g(x)$ for all $x \in U$. In place of g we write $f|_U$ (in words: f limited to U).

2.2 Ranked Alphabets, Variables and Terms

A pair $(\Omega, \underline{rank}_\Omega)$ is called *ranked alphabet* if Ω is a finite set and $\underline{rank}_\Omega : \Omega \to \mathbb{N}$ is a total function. For every symbol $f \in \Omega$, the natural number $\underline{rank}_\Omega(f)$ is called the *rank of f*. For every $i \geq 0$, the set $\Omega^{(i)} \subseteq \Omega$ consists of all symbols in Ω that are of rank i. A ranked alphabet can be defined by either (i) enumerating the finitely many non–empty subsets $\Omega^{(i)}$, or by (ii) a set of symbols which are indexed by their unambiguous rank. In this paper we will mostly use the latter of these methods. In the following let $(\Omega, \underline{rank}_\Omega)$ be an arbitrary but fixed ranked alphabet.

For the rest of this paper we choose the *set of variables* to be the set $\mathcal{V} = \{x, x_1, x_2, \ldots, y, y_1, y_2, \ldots, z, z_1, z_2, \ldots\}$.

Let S be an arbitrary set. The *set of terms over* Ω *indexed by* S, denoted by $T(\Omega, S)$, is the smallest set $T \subseteq (\Omega \cup S \cup \{(,),,\})^*$, such that the following holds: $S \subseteq T$ and if $f \in \Omega^{(n)}$ with $n \geq 0$ and $t_1, \ldots, t_n \in T$, then $f(t_1, \ldots, t_n) \in T$.

For every term $t \in T(\Omega, \mathcal{V})$, the *set of occurrences of* t, denoted by $O(t)$, is a subset of $\mathbb{N}^*$ which is inductively defined as follows: (i) if $t = x$ with $x \in \mathcal{V}$, then $O(t) = \{\varepsilon\}$, (ii) if $t = f$ with $f \in \Omega^{(0)}$, then $O(t) = \{\varepsilon\}$, and (iii) if $t = f(t_1, \ldots, t_n)$ with $f \in \Omega^{(n)}$ and $n > 0$, and for all $i \in [n] : t_i \in T(\Omega, \mathcal{V})$, then $O(t) = \{\varepsilon\} \cup \bigcup_{i \in [n]} \{iu \mid u \in O(t_i)\}$. Obviously this representation becomes ambiguous if the ranked alphabet contains a symbol with rank larger than 9. Then one should be adapting *Dewey's notation* by placing dots between the i's.

For every term $t \in T(\Omega, \mathcal{V})$ and every occurrence u of t, the *subterm of* t *at occurrence* u is denoted by t/u. The *label of* t *at occurrence* u is denoted by $t[u]$. The *set of variables occurring in* t is denoted by $\mathcal{V}(t)$; that means $v \in \mathcal{V}(t)$, if $v \in \mathcal{V}$ and there is a $u \in O(t)$, such that $t/u = v$. The term t with the subterm at occurrence u replaced by the term s is denoted by $t[u \leftarrow s]$. The lexicographical order on $O(t)$ is denoted by $<_{lex}$.

Let t be a tree. The set of *nodes of* t is denoted by $\underline{node}(t)$ and the *root of* t is denoted by $\underline{root}(t)$. For every node $nd \in \underline{node}(t)$, we will denote its label by $\underline{label}_t(nd)$.

2.3 Substitutions

A mapping $\varphi : \mathcal{V} \to T(\Omega, \mathcal{V})$ is called $(\mathcal{V}, \Omega)$–*substitution*, if $\{v \mid v \in \mathcal{V} \text{ and } \varphi(v) \neq v\}$ is a finite set. The set of all $(\mathcal{V}, \Omega)$–substitutions is denoted by $Sub(\mathcal{V}, \Omega)$.

Every $(\mathcal{V}, \Omega)$–substitution φ is characterized in terms of those (finitely many) variables that change under φ. Let $v_1, \ldots, v_n \in \mathcal{V}$ be the complete list of variables, such that $\varphi(v_i) \neq v_i$. Then φ is unambiguously defined by $\varphi(v_1), \ldots, \varphi(v_n)$.

Let φ be a $(\mathcal{V}, \Omega)$–substitution. The *homomorphical extension of* φ *to* $T(\Omega, \mathcal{V})$ is a function $\hat{\varphi} : T(\Omega, \mathcal{V}) \to T(\Omega, \mathcal{V})$ defined inductively as follows: (i) for every $v \in \mathcal{V}$: $\hat{\varphi}(v) = \varphi(v)$, (ii) for every $\alpha \in \Omega^{(0)}$: $\hat{\varphi}(\alpha) = \alpha$, and (iii) for every $f(t_1, \ldots, t_n)$ with $f \in \Omega^{(n)}$ and $n \geq 1$, and for all $i \in [n] : t_i \in T(\Omega, \mathcal{V})$: $\hat{\varphi}(f(t_1, \ldots, t_n)) = f(\hat{\varphi}(t_1), \ldots, \hat{\varphi}(t_n))$.

For every substitution $\varphi \in Sub(\mathcal{V}, \Omega)$, the set $\{v \mid v \in \mathcal{V} \text{ and } \varphi(v) \neq v\}$ is called the *domain* of φ; it is denoted by $\mathcal{D}(\varphi)$. If $\mathcal{D}(\varphi) = \emptyset$, then φ is denoted by $\varphi_\emptyset$. We say that φ is *ground*, if for every $v \in \mathcal{D}(\varphi) : \mathcal{V}(\varphi(v)) = \emptyset$. The set of ground $(\mathcal{V}, \Omega)$–substitutions is denoted by $gSub(\mathcal{V}, \Omega)$. The *set of variables introduced by* φ is the set $\mathcal{I}(\varphi) = \bigcup\limits_{v \in \mathcal{D}(\varphi)} \mathcal{V}(\varphi(v))$.

2.4 Relations and Orders

Let M be a set. A subset $\sqsubseteq$ of $M \times M$ is called a *partial order* if $\sqsubseteq$ is reflexive, antisymmetric and transitive. A partial order $\sqsubseteq$ is called a *total order* if for every $x, y \in M$: $x \sqsubseteq y$ or $y \sqsubseteq x$. A partial order $\sqsubseteq$ on M is called *well–founded* if every non-empty subset S of M has a minimal element (w.r.t. $\sqsubseteq$); this element is denoted by $min_{\sqsubseteq}(S)$.

A relation $R \subseteq M \times M$ is called an *equivalence relation* if R is reflexive, symmetric, and transitive. The set of all elements related to $x \in M$ by R is called *equivalence class* $[\![x]\!]$ *over* x: $[\![x]\!] = \{z \mid x \in M \text{ and } z \in M \text{ and } x \, R \, z\}$. The set $M/_R = \{[\![x]\!] \mid x \in M\}$ is called *quotient set* of M to R.

An equivalence relation $\sim$ over $T(\Omega, \mathcal{V})$ is called Ω–*congruence over* $T(\Omega, \mathcal{V})$ if for every $f \in \Omega^{(n)}$, with $n > 0$ and for every $t_1, s_1, \ldots t_n, s_n \in T(\Omega, \mathcal{V})$ with $t_1 \sim s_1, \ldots t_n \sim s_n$: $f(t_1, \ldots, t_n) \sim f(s_1, \ldots, s_n)$.

2.5 Term Rewriting Systems

A *term rewriting system*, denoted by $\mathcal{R}$, is a pair (Ω, R) where Ω is a ranked alphabet and R is a finite set of rules; every rule has the form $l \to r$ with $r, l \in T(\Omega, \mathcal{V})$ and $\mathcal{V}(r) \subseteq \mathcal{V}(l)$. With every term rewriting system $\mathcal{R} = (\Omega, R)$ a bijection $\pi : R \to [card(R)]$ is associated implicitly that describes an enumeration of R.

3 Formal Calculi

In this section we will formalize the general notion of a calculus. The notion of a calculus subsumes the idea that a set of derivation forms and a binary relation, called derivation relation are given where the derivation relation transforms one derivation form into another one. The initial derivation form, i.e., the "object" which is to be calculated, is called a calculandum. A sequence of applications of the derivation relation is called a derivation. The aim of a derivation is to arrive at a derivation form with certain characteristics. Such derivation forms are called success forms.

One example of a calculus is the computation of arithmetic terms. In that case the set of derivation forms consists of all arithmetic terms. The derivation relation could for instance be based on the algebraic laws of commutativity, associativity, and distributivity for addition and multiplication.

Other examples are logic calculi. The set of derivation forms is then made up of the set of logical formulae and the derivation relation will contain the corresponding deduction rules.

3.1 Calculus

We will now define the notion of a calculus.

Definition 3.1 (calculus)
A *calculus*, denoted by $\mathcal{C}$, is a tuple $(D, D_{\text{success}}, \vdash)$ where:

1. D is a set; the elements of D are called *derivation forms*.
2. D_{success} is a set where: $D_{\text{success}} \subseteq D$; elements of D_{success} are called *success forms*.
3. $\vdash \subseteq D \times D$ is a binary relation; $\vdash$ is called *derivation relation*. □

We adopt the following notations:

- Instead of $(d, d') \in \vdash$ we write $d \vdash d'$. If $d \vdash d'$ then we say d derives to d' by $\vdash$, or d' is derived from d by $\vdash$.
- $\vdash^+$: transitive closure of $\vdash$
- $\vdash^*$: transitive reflexive closure of $\vdash$
- $d \vdash d'$: derivation step by $\vdash$
- $d \vdash^* d'$: derivation by $\vdash$ starting with d.

We will now give an example of a calculus in which the algebraic laws of associativity and distributivity for arithmetic expressions over addition and multiplication are formalized as a derivation relation.

Example 3.2 Let $\mathcal{C} = (D, D_{\text{success}}, \vdash)$ be the calculus where:

- $D = T(\Omega_1, \emptyset)$ with $\Omega_1 = \{+^{(2)}, *^{(2)}, a^{(0)}, b^{(0)}, c^{(0)}\}$. According to the definition of terms, this set contains the terms a, b, c and, for every $t_1, t_2 \in T(\Omega_1, \emptyset)$, the terms $+(t_1, t_2)$ and $*(t_1, t_2)$. Here and in the following examples we will use infix notation for the symbols $+$ and $*$; i.e., we write $(a \circ b)$ in place of $\circ(a, b)$, for $\circ \in \{+, *\}$.

- $D_{\text{success}} = \emptyset$.

- Let $t \in T(\Omega_1, \emptyset), u \in O(t)$, and $t' = t/u$.

  1. If $t' = (t_1 * (t_2 * t_3))$ where $t_1, t_2, t_3 \in T(\Omega_1, \emptyset)$, then
  $$(t, t[u \leftarrow ((t_1 * t_2) * t_3)]) \in \vdash .$$

  2. If $t' = (t_1 * (t_2 + t_3))$ where $t_1, t_2, t_3 \in T(\Omega_1, \emptyset)$, then
  $$(t, t[u \leftarrow ((t_1 * t_2) + (t_1 * t_3))]) \in \vdash .$$

Next we show a derivation by $\vdash$ starting with the term $(a * (a * (b + c)))$:

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

Obviously this is not the only possible derivation by $\vdash$ starting with $(a * (a * (b+c)))$. In the first derivation step the derivation form $(a * ((a * b) + (a * c)))$ could have been derived by means of distributivity. □

This example shows that in general different derivation forms can be derived from one calculandum. For a given number of derivation steps, these derivation forms are collected into one set.

Definition 3.3 (set of k–successors)

Let $\mathcal{C} = (D, D_{\text{success}}, \vdash)$ be a calculus, $d \in D$ a derivation form, and $k \geq 1$ an integer. The *set of k–successors of d*, denoted by $Suc_k(d, \vdash)$ is the set $\{d' \in D \mid d \vdash^k d'\}$. The set $Suc_1(d, \vdash)$ is called *set of direct successors of d* and it will be denoted by $Suc(d, \vdash)$. □

We use the following notations:

- A derivation form $d \in D$ is called *in normal form* if $Suc(d, \vdash)$ is empty.

- d' is a *success form of d* if $d \vdash^* d'$ and $d' \in D_{\text{success}}$.

The derivation form $(((a * a) * b) + ((a * a) * c))$ of Example 3.2 is in normal form since $Suc((((a * a) * b) + ((a * a) * c)), \vdash)$ is empty. However it is *not* a success form because the set D_{success} of success forms is empty. The set $D_{\text{success}} \subseteq D$ can be chosen according to the specific current problem.

3.2 Indexed Calculus and Derivation Tree

Let us reconsider the definition of a calculus from the previous section. There are certain calculi, such that a derivation form $d \in D$ has arbitrarily many successors. For each of these successors there might again be arbitrarily many successors. Until now we do not have a way to distinguish between the different successors of one derivation form; i.e., we can not identify one specific successor.

In order to solve this identification problem, we introduce the notion of a *derivation tree*. Such a tree incorporates all possible derivations by $\vdash$ starting with some calculandum and thus, it also allows to unambiguously identify one specific derivation and the derivation forms involved. More precisely, every path through the tree, starting at a node labeled with d, where $d \in D$, and ending at a node labeled with d', where $d' \in D$, will correspond to a derivation by $\vdash$ starting with d and ending with d'. To prepare the definition of derivation trees we will introduce the concept of indexed calculi. In an indexed calculus, the derivation relation $\vdash$ is partitioned into relations $\vdash_j$, such that, for every derivation form d, each $\vdash_j$ leads to at most one direct successor of d. The indices j will be chosen from an index set J on which we define a total order $<$. This order $<$ allows us to fix, for a node labeled by d, one specific arrangement for the set of successor nodes labeled by $\{d' \mid \text{there is a } j : d \vdash_j d'\}$. Note that we will also use the symbol "$<$" as the usual less–than order on natural numbers. It will always be clear from the context which order we refer to.

Definition 3.4 (indexed calculus)
An *indexed calculus* $\mathcal{I}$ is a tuple $(D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ where:

1. D is a set; elements of D are called *derivation forms*.
2. D_{success} is a set with $D_{\text{success}} \subseteq D$; elements of D_{success} are called *success forms*.
3. J is a well–founded set; the set J is called *index set*; elements of J are called *indices*.
4. $<$ is a total order on J.
5. $(\vdash_j \mid j \in J)$ is a family of binary relations $\vdash_j \subseteq D \times D$, such that for every $d \in D$ and $j \in J$: $Suc(d, \vdash_j)$ contains at most one element. □

Definition 3.5 (indexed calculus for $\mathcal{C}$)
Let $\mathcal{C} = (D, D_{\text{success}}, \vdash)$ be a calculus and let $\mathcal{I} = (D', D'_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ be an indexed calculus. $\mathcal{I}$ is an *indexed calculus for* $\mathcal{C}$ if:

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

Let us go back to Example 3.2 and construct an indexed calculus $\mathcal{I}$ for the calculus $\mathcal{C}$. We will use the set $\{\text{assoc}, \text{distr}\} \times \mathbb{N}^*$ as index set. This choice is justified by the fact that there are two non-determinisms which occur in a derivation step $d \vdash d'$:

(i) the choice which rule (associativity or distributivity) is applied and

(ii) the choice at which occurrence $O(t)$ the rule is applied.

As total order on J we will arbitrarily choose $(\text{assoc}, u) < (\text{distr}, u)$ for every occurrence u; if one rule can be applied at different occurrences, then we use the lexicographical order on the occurrences.

Example 3.6 Let $\mathcal{C} = (T(\{+^{(2)}, *^{(2)}, a^{(0)}, b^{(0)}, c^{(0)}\}, \emptyset), \emptyset, \vdash)$ be the calculus of Example 3.2. The tuple $\mathcal{I} = (D', D'_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ is an indexed calculus for $\mathcal{C}$ where:

- $D' = D = T(\Omega_1, \emptyset)$
- $D'_{\text{success}} = D_{\text{success}} = \emptyset$
- $J = \{\text{assoc}, \text{distr}\} \times \mathbb{N}^*$
- $<$ is a total order on $\{\text{assoc}, \text{distr}\} \times \mathbb{N}^*$, such that for every $(r_1, u_2), (r_2, u_2) \in J$: $(r_1, u_1) < (r_2, u_2)$, if one of the following two conditions hold:
  1. $r_1 = \text{assoc}$ and $r_2 = \text{distr}$.
  2. $r_1 = r_2$ and $u_1 <_{lex} u_2$.
- $(\vdash_j \mid j \in J)$ is a family defined as follows:
  Let $t \in T(\Omega_1, \emptyset), u \in O(t)$ and $t' = t/u$, then:
  1. if $t' = (t_1 * (t_2 * t_3))$ where $t_1, t_2, t_3 \in T(\Omega_1, \emptyset)$, then
  $$(t, t[u \leftarrow ((t_1 * t_2) * t_3)]) \in \vdash_{(\text{assoc},u)}$$
  2. it $t' = (t_1 * (t_2 + t_3))$ where $t_1, t_2, t_3 \in T(\Omega_1, \emptyset)$, then
  $$(t, t[u \leftarrow ((t_1 * t_2) + (t_1 * t_3))]) \in \vdash_{(\text{distr},u)} .$$

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

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

This example shows that, for an indexed calculus for $\mathcal{C}$, every derivation step $t \vdash t'$ of the calculus $\mathcal{C}$ is unambiguously characterized by an index $j \in J$. Moreover, the total order $<$ on J defines one fixed order of the derivation steps $\vdash_j$.

We now define a set that we will frequently need later on. It is the set of indices j, such that, for a given $d \in D$, there is a d' with $d \vdash_j d'$.

Definition 3.7 (set of successor indices)

Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ be an indexed calculus and let $d \in D$ be a derivation form. The *set of successor indices of* d, denoted by $I(d)$, is the set

$$\{j \in J \mid \text{there is a } d' \in D : d \vdash_j d'\}.$$
□

We are now ready to define the notion of a derivation tree.

Definition 3.8 (derivation tree)

Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ be an indexed calculus and let $d \in D$ be a calculandum. The *derivation tree for* $\mathcal{I}$ *and* d, denoted by $B(\mathcal{I}, d)$, is the tree defined as follows.

1. Every node of $B(\mathcal{I}, d)$ is labeled by a derivation form $d' \in D$.
2. The root of $B(\mathcal{I}, d)$ is labeled by d; i.e., $\underline{label}_{B(\mathcal{I},d)}\,(\underline{root}\,(B(\mathcal{I}, d))) = d$.
3. Let d' be the label of a node nd of $B(\mathcal{I}, d)$ and let the set of successor indices of d' be the set $I(d') = \{j_1, \ldots, j_n\}$ where $j_1 < j_2 < \ldots < j_n$. Then the node nd has n successors and for every $k \in [n]$: if $d' \vdash_{j_k} \hat{d}$, then the label $\underline{label}_{B(\mathcal{I},d)}\,(nd.k)$ of the k–th successor of nd is $\hat{d}$. □

The derivation tree for an indexed calculus $\mathcal{I}$ and a derivation form d does not depend on the set D_{success}. Until now we have not given any meaning to the set D_{success} besides a name for its elements as success forms. We can merely use it to mark particular derivation forms. The derivation trees for two indexed calculi that differ from each other only in the set D_{success} of success forms are equal. This fact is stated in the following lemma.

Lemma 3.9 Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ and $\mathcal{I}' = (D, D'_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ be indexed calculi. The following holds for every $d \in D$

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

□

Example 3.10 Let $\mathcal{I}$ be the indexed calculus of Example 3.6. The derivation tree $B(\mathcal{I}, (a * (a * (b + c)))$ is the following tree:

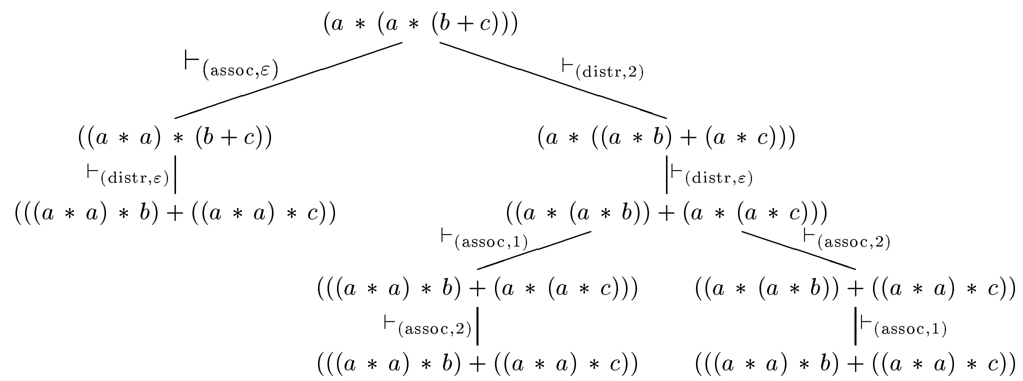

Figure 2: Derivation tree for $\mathcal{I}$ and $(a * (a * (b + c)))$

□

We note that every leaf of the tree in Figure 2 is labeled by a derivation form which is in normal form.

3.3 Deterministic Calculus

Let $\mathcal{I}$ be an indexed calculus. We now want to define a calculus for $\mathcal{I}$ that deterministically produces all derivation forms d' with $d \vdash^* d'$ for $\mathcal{I}$. In terms of the corresponding derivation tree this deterministic calculus will perform a depth-first left-to-right tree traversal through $B(\mathcal{I}, d)$. Roughly speaking, a derivation form of this calculus will be a sequence of pairs. Each pair consists of a derivation form of D and a set M that determines which branches of the tree are still to be traversed.

We now define two auxiliary functions *first* and *rest* that are needed for the definition of the deterministic calculus. The function *rest* returns the minimal element of a set $M \subseteq J$ with respect to a well-founded order $<$; the function *rest* returns all but this minimal element.

Definition 3.11 (*first, rest*)

Let the binary relation $<$ on J be a well–founded order. The functions *first*: $\mathcal{P}(J) \setminus \emptyset \to J$ and *rest*: $\mathcal{P}(J) \setminus \emptyset \to \mathcal{P}(J)$ are defined by: $\mathit{first}(M) = \mathit{min}_<(M)$ and $\mathit{rest}(M) = M \setminus \mathit{first}(M)$ for every $M \subseteq J$ with $M \neq \emptyset$. □

Definition 3.12 (deterministic calculus)

Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ be an indexed calculus. The *deterministic calculus for* $\mathcal{I}$, denoted by $\mathit{det}(\mathcal{I})$, is the calculus $(D', D'_{\text{success}}, \models)$ where:

1. $D' = D_0^+$, with $D_0 = \{(d, M) \mid d \in D, M \subseteq I(d)\}$
2. $D'_{\text{success}} = \{w(d, I(d)) \mid w \in D_0^*, d \in D_{\text{success}}\}$
3. $\models \subseteq D' \times D'$, such that:
   (a) for every $w \in D'$ and $d \in D$: $w(d, \emptyset) \models w$
   (b) for every $w \in D_0^*$ and $(d, M) \in D_0$ with $M \neq \emptyset$,

$$w(d, M) \models w(d, M')(\hat{d}, I(\hat{d})),$$

   if: $d \vdash_{\mathit{first}\,(\mathrm{M})} \hat{d}$ with $M' = \mathit{rest}(M)$. □

Thus, in particular, the indexed calculus $\mathit{det}(\mathcal{I})$ is a calculus itself.

Lemma 3.13 Let $\mathit{det}(\mathcal{I})$ be an indexed calculus for some calculus $\mathcal{I}$ and let d be a derivation form of $\mathit{det}(\mathcal{I})$. Then, for every $k \geq 0$, the set $\mathit{Suc}_k(d, \models)$ of k-successors of d contains at most one element. □

Example 3.14 Let $\mathcal{I}$ be the indexed calculus of Example 3.6. The deterministic calculus $\mathit{det}(\mathcal{I}) = (D', D'_{\text{success}}, \models)$ for $\mathcal{I}$ describes all depth-first left-to-to-right tree traversals through the derivation trees of $\mathcal{I}$ with arbitrary calculandi.

Let us consider the derivation tree for $\mathcal{I}$ (cf. Figure 2) and calculandum $(a * (a * (b + c)))$. The following is a derivation by $\models$ starting with $((a * (a * (b + c))), \{(\text{assoc}, \varepsilon), (\text{distr}, 2)\})$:

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

It should be realized that the choice of the depth–first left-to-right tree traversals does not exclude more general depth-first tree traversals. The arrangement of nodes of the derivation tree depends on the total order $<$ for the underlying indexed calculus. Hence, for instance a depth–first right-to-left tree traversal could be achieved by simply inverting the total order $<$. However the choice of a depth–first traversal is fixed by our definition of the deterministic calculus (therefore a breadth–first strategy could not be realized by our deterministic calculus).

4 E–Unification

In this section we will briefly recall the important definitions concerning E-unification. Recall that Ω denotes an arbitrary but fixed ranked alphabet.

Definition 4.1 (equation)

An *equation over* Ω *and* $\mathcal{V}$ is a pair (t, s) where $t, s \in T(\Omega, \mathcal{V})$. □

An equation (t, s) is usually written as $t = s$.

Definition 4.2 (E–equality)

Let E be a finite set of equations over Ω and $\mathcal{V}$. The *E–equality*, denoted by $=_E$, is the finest Ω–congruence containing every pair $(\psi(t), \psi(s))$ where $(t = s) \in E$ and ψ is an arbitrary $(\mathcal{V}, \Omega)$–substitution. If $t =_E s$, then t and s are called *E–equal.* □

E–unification now simply means unification with respect to an equational theory $=_E$ instead of the usual syntactic equality.

Definition 4.3 (set of E–unifiers)

Let E be a finite set of equations over Ω and $\mathcal{V}$.

- Two terms $t, s \in T(\Omega, \mathcal{V})$ are called *E–unifiable* if there is a $(\mathcal{V}, \Omega)$–substitution, such that $\varphi(t) =_E \varphi(s)$. Then φ is called *E–unifier* of t and s.
- The set $\{\varphi \mid \varphi(t) =_E \varphi(s)\}$ is called *set of E–unifiers of* t *and* s; it is denoted by $\mathcal{U}_E(t, s)$.

□

If E is the empty set, then the E–unification problem coincides with the usual unification problem which is decidable [Rob65]. However, we can easily code Post's correspondence problem (PCP) as an E–unification problem [Faß95a] to show that the E–unification problem is in general undecidable.

E–unifiers can be compared using the instantiation preorder.

Definition 4.4 (instantiation preorder)

Let E be a finite set of equations over Ω and $\mathcal{V}$. Let $V \subseteq \mathcal{V}$ be a finite set and let $\varphi, \varphi' \in$ *Sub* $(\mathcal{V}, \Omega)$ be substitutions. The *instantiation order* on $(\mathcal{V}, \Omega)$–substitutions, denoted by $\preceq_{E,V}$, is defined as:

$$\varphi \preceq_{E,V} \varphi' \text{ if there is a } (\mathcal{V}, \Omega)\text{–substitution } \psi,\text{, such that for all } v \in V : \\ \psi(\varphi(v)) =_E \varphi'(v).$$

□

For every two terms t and s, the set $\mathcal{U}_\emptyset\,(t, s)$ has a smallest element, called most general unifier (for short: mgu) of t and s. In general however, $\mathcal{U}_E\,(t, s)$ may contain more than one minimal element. A subset S of $\mathcal{U}_E\,(t, s)$ is called complete if it contains at least these minimal elements.

Definition 4.5 (complete set of E–unifiers)

Let E be a finite set of equations over Ω and $\mathcal{V}$, $t, s \in T(\Omega, \mathcal{V})$ terms, and W a finite set of variables which contains $V = \mathcal{V}(t) \cup \mathcal{V}(s)$, i.e., $V \subseteq W$. The set S is a *complete set of E–unifiers of t and s away from* W if the following three conditions hold:

1. For every $\varphi \in S : \mathcal{D}(\varphi) \subseteq V$ and $\mathcal{I}(\varphi) \cap W = \emptyset$.
2. $S \subseteq \mathcal{U}_E(t, s)$.
3. For every $\varphi \in \mathcal{U}_E(t, s)$ there is a $\psi \in S$, such that $\psi \preceq_{E,V} \varphi$.

The set S is *minimal* if

4. for all $\varphi, \varphi' \in S$: if $\varphi = \varphi'$ then $\varphi \not\preceq_{E,V} \varphi'$. □

As already mentioned, we will compute E-unifiers for two terms where the set E is induced by some term rewriting system.

Definition 4.6 (induced set of equations)

Let $\mathcal{R} = (\Omega, R)$ be a term rewriting system. The *set of equations induced by* $\mathcal{R}$, denoted by $E_\mathcal{R}$, is the set $\{(l, r) \mid (l \to r) \in R\}$. □

In some of the considered term rewriting systems, the set Ω is divided into two disjoint sets F and Δ of *function symbols* and *constructor symbols*, respectively. Later we will be mainly interested in $E_\mathcal{R}$-unifiers of which the ranges only contain constructor symbols and variables (but no function symbols).

Definition 4.7 ($(E_\mathcal{R}, \Delta)$–unifiers)

Let $\mathcal{R} = (\Omega, R)$ be a term rewriting system, such that Ω is divided into two disjoint sets F and Δ. Let $t, s \in T(\Omega, \mathcal{V})$ be terms and let $\varphi \in \mathcal{U}_{E_\mathcal{R}}(t, s)$ be an $E_\mathcal{R}$–unifier of t and s.

1. φ is an *$(E_\mathcal{R}, \Delta)$–unifier of t and s* if $\varphi \in \mathit{Sub}\,(\mathcal{V}, \Delta)$.
2. φ is a *ground $(E_\mathcal{R}, \Delta)$–unifier of t and s* if $\varphi \in \mathit{gSub}\,(\mathcal{V}, \Delta)$.

The sets of $(E_\mathcal{R}, \Delta)$–unifiers and of ground $(E_\mathcal{R}, \Delta)$–unifiers are denoted by $\mathcal{U}_{(E_\mathcal{R},\Delta)}\,(t, s)$ and $g\mathcal{U}_{(E_\mathcal{R},\Delta)}\,(t, s)$, respectively.

□

Definition 4.8 (ground complete set of $(E_\mathcal{R}, \Delta)$–unifiers)

Let $\mathcal{R} = (\Omega, R)$ be a term rewriting system, such that Ω is divided into two disjoint sets F and Δ. Let $t, s \in T(F \cup \Delta, \mathcal{V})$ be terms and W a finite set of variables containing $V = \mathcal{V}(t) \cup \mathcal{V}(s)$. A set S is called a *ground complete set of $(E_\mathcal{R}, \Delta)$–unifiers of t and s away from* W if the following three conditions hold:

1. For every $\varphi \in S : \mathcal{D}(\varphi) \subseteq V$ and $\mathcal{I}(\varphi) \cap W = \emptyset$
2. $S \subseteq \mathcal{U}_{(E_\mathcal{R},\Delta)}\,(t, s)$
3. For every $\varphi \in g\mathcal{U}_{(E_\mathcal{R},\Delta)}\,(t, s)$ there is a $\psi \in S$, such that $\psi \preceq_{E_\mathcal{R},V} \varphi$. □

5 Formal Calculi for E–Unification

In this section we want to construct four particular indexed calculi which compute $E_{\mathcal{R}}$–unifiers. We restrict our attention to equational theories which are induced by canonical term rewriting systems.

First we will construct the *reduction calculus for* $E_{\mathcal{R}}$*–unification* and the *narrowing calculus for* $E_{\mathcal{R}}$*–unification.* Then we will construct the *leftmost outermost narrowing calculus for* $(E_{\mathcal{R}}, \Delta)$*–unification* and the *unification driven leftmost outermost narrowing calculus for* $(E_{\mathcal{R}}, \Delta)$*–unification.* The latter two calculi assume an even more restricted class of equational theories, namely the class of equational theories induced by so called ctn term rewriting systems (cf. Definition 5.19).

5.1 Reduction Calculus for $E_{\mathcal{R}}$–Unification

First we will define the usual reduction relation for a term rewriting system $\mathcal{R}$. This is a relation that transforms a term $t \in T(\Omega, \mathcal{V})$ into a term $t' \in T(\Omega, \mathcal{V})$ by applying a rewrite rule. The application of a rewrite rule $l \to r$ is possible, if there is a subterm of t that matches with a left hand side of a rewrite rule.

Definition 5.1 (reduction relation)

Let $\mathcal{R} = (\Omega, R)$ be a term rewriting system. The *reduction relation for* $\mathcal{R}$, denoted by $\Rightarrow_{\mathcal{R}}$, is the binary relation over $T(\Omega, \mathcal{V})$, such that for every $t, s \in T(\Omega, \mathcal{V})$: $t \Rightarrow_{\mathcal{R}} s$ iff the following two conditions hold.

1. There is an occurrence $u \in O(t)$, such that $t/u \notin \mathcal{V}$.
2. There is a substitution $\varphi \in Sub(\mathcal{V}, \Omega)$ and a rule $l \to r \in R$, such that $\varphi(l) = t/u$ and $s = t[u \leftarrow \varphi(r)]$. □

If it is obvious from the context which term rewriting system $\mathcal{R}$ is meant, then we simply write $\Rightarrow$ instead of $\Rightarrow_{\mathcal{R}}$. Sometimes a particular derivation step $t \Rightarrow_{\mathcal{R}} t'$ is indexed by the occurrence u of t, the substitution φ, and the rule ξ, i.e., $t \Rightarrow_{\mathcal{R},u,\varphi,\xi} t'$.

Example 5.2 Let $\mathcal{R}_1 = (\Omega, R)$ be a term rewriting system where:

- $\Omega = \{append^{(2)}, CONS^{(2)}, NIL^{(0)}, A^{(0)}, B^{(0)}, C^{(0)}\}$ is a ranked alphabet.
- R is the set of the following rules:

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

Let t be the term:

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

The following is a derivation by $\Rightarrow_{\mathcal{R}_1}$ starting with t:

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

□

Definition 5.3 (canonical term rewriting system)

Let $\mathcal{R} = (\Omega, R)$ be a term rewriting system. $\mathcal{R}$ is a *canonical term rewriting system* (sometimes also called: *convergent*) if the following two conditions hold:

1. $\mathcal{R}$ is *confluent*; that is for every $s, t, t' \in T(\Omega, \mathcal{V})$: if $s \Rightarrow^*_{\mathcal{R}} t$ and $s \Rightarrow^*_{\mathcal{R}} t'$, then there is an $s' \in T(\Omega, \mathcal{V})$, such that $t \Rightarrow^*_{\mathcal{R}} s'$ and $t' \Rightarrow^*_{\mathcal{R}} s'$.
2. $\mathcal{R}$ is *noetherian* (also called: *terminating*); that is there is no infinite derivation by $\Rightarrow_{\mathcal{R}}$.

□

Lemma 5.4 (uniqueness of normal forms, cf. [Hue80])

Let $\mathcal{R} = (\Omega, R)$ be a canonical term rewriting system and let $t \in T(\Omega, \mathcal{V})$. Then there is a unique term t', such that the following two conditions hold: 1. $t \Rightarrow^*_{\mathcal{R}} t'$ and 2. there is no term t'' with $t' \Rightarrow_{\mathcal{R}} t''$. □

For t the unique term t' is called *($\mathcal{R}$-)normal form of* t and it is denoted by $nf_{\mathcal{R}}(t)$.

Let $\mathcal{R} = (\Omega, R)$ be an arbitrary canonical term rewriting system and let $\mathcal{C}_{\mathcal{R}} = (T(\Omega, \mathcal{V}), D_{\text{success}}, \Rightarrow_{\mathcal{R}})$ be a calculus. We now want to give an indexed calculus for $\mathcal{C}_{\mathcal{R}}$, such that an unambiguous representation of all possible derivations (starting with some calculandum t) with respect to $\Rightarrow_{\mathcal{R}}$ can be given. Hence we need an appropriate index set J and a total order $<$ on J. One should note that there are the following non-determinisms in a derivation step by $\Rightarrow_{\mathcal{R}}$:

- The choice of the occurrence u of t and
- given an occurrence u, the choice of a rewrite rule in R which is to be applied at u.

Thus, it is reasonable to choose the set $\mathbb{N}^* \times R$ as index set J and to fix an order on it, which is a combination of the lexicographical order on $\mathbb{N}^*$ and the total order which is induced by the enumeration π of the set R of rules (cf. Section 2.5).

How should the set D_{success} of success forms be defined? This amounts to the question of how the reduction relation $\Rightarrow_{\mathcal{R}}$ can be used to find E–unifiers. The next well-known lemma shows the connection.

Lemma 5.5 (cf. [HO80], [Hue80]) Let $\mathcal{R} = (\Omega, R)$ be a term rewriting system.

1. $=_{E_{\mathcal{R}}} = (\Rightarrow_{\mathcal{R}} \cup \Rightarrow^{-1}_{\mathcal{R}})^*$
2. If $\mathcal{R}$ is canonical and $t, s \in T(\Omega, \mathcal{V})$, then the following equivalence holds:
$t\ (\Rightarrow_{\mathcal{R}} \cup \Rightarrow^{-1}_{\mathcal{R}})^*\ s$ if and only if $nf_{\mathcal{R}}(t) = nf_{\mathcal{R}}(s)$. □

Hence, two terms t and s are $E_{\mathcal{R}}$–equal iff their normal forms are equal with respect to $\Rightarrow_{\mathcal{R}}$, and consequently, two terms t and s are $E_{\mathcal{R}}$–unifiable iff there is a substitution φ, such that the normal forms of $\varphi(t)$ and $\varphi(s)$ are equal.

For technical convenience, we combine the terms $\varphi(t)$ and $\varphi(s)$ by placing them into the first and second argument position, respectively, of a new binary symbol *equ*. Thus, the calculandum has the form $equ(\varphi(t), \varphi(s))$. In order to derive such terms, we have to enrich the term rewriting system $\mathcal{R}$ slightly.

Definition 5.6 (equal term rewriting system)

Let $\mathcal{R} = (\Omega, R)$ be a term rewriting system. The *equal term rewriting system associated with* $\mathcal{R}$, denoted by $\mathcal{R}_{\text{equ}}$, is the term rewriting system $(\Omega \cup \{equ^{(2)}\}, R)$. □

We can now define the reduction calculus for $E_{\mathcal{R}}$–unification. The set of success forms D_{success} contains every term $equ(\Theta, \Theta)$ which is in normal form with respect to $\Rightarrow_{\mathcal{R}_{\text{equ}}}$.

Definition 5.7 (reduction calculus for $E_{\mathcal{R}}$–unification)

Let $\mathcal{R} = (\Omega, R)$ be a canonical term rewriting system.

1. The *reduction calculus for $E_{\mathcal{R}}$–unification for $\mathcal{R}$*, denoted by $\text{red}\mathcal{I}(\mathcal{R})$, is the indexed calculus

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

   where

   - $\text{Equ}(T(\Omega, \mathcal{V})) = \{equ(t, s) \mid t, s \in T(\Omega, \mathcal{V})\}$.
   - $D_{\text{success}} = \{equ(\Theta, \Theta) \mid \Theta \in T(\Omega, \mathcal{V}), equ(\Theta, \Theta)$ is in normal form w.r.t. $\Rightarrow_{\mathcal{R}_{\text{equ}}}\}$.
   - $<$ is the total order, such that for every $(u, l \to r), (u', l' \to r') \in \mathbb{N}^* \times R$: $(u, l \to r) < (u', l' \to l')$, if one of the following two conditions holds:

     (a) $u <_{lex} u'$

     (b) $u = u'$ and $\pi(l \to r) < \pi(l' \to r')$.

2. Let $t \in T(\Omega, \mathcal{V})$. The *reduction tree for $\mathcal{R}$ and $t \in T(\Omega, \mathcal{V})$* is the derivation tree $B(\text{red}\mathcal{I}(\mathcal{R}), t)$. □

Since the new symbol *equ* does not occur in any rule of R, the $E_{\mathcal{R}}$–equality is identical to the $E_{\mathcal{R}_{\text{equ}}}$–equality restricted to $T(\Omega, \mathcal{V})$. Moreover, for two terms t and s, the following holds: $equ(nf_{\mathcal{R}}(t), nf_{\mathcal{R}}(s)) = nf_{\mathcal{R}_{\text{equ}}}(equ(t, s))$. Hence, if a calculandum $equ(t, s)$ derives to a success form, then t and s are $E_{\mathcal{R}}$–equal. The important consequence of this fact is summarized in the following completeness theorem.

Theorem 5.8 (completeness of $\text{red}\mathcal{I}(\mathcal{R})$ for $E_{\mathcal{R}}$-unification)

Let $\mathcal{R} = (\Omega, R)$ be a canonical term rewriting system. Let $\text{red}\mathcal{I}(\mathcal{R})$ be the reduction calculus for $E_{\mathcal{R}}$–unification for $\mathcal{R}$ with success set D_{success}. Let $t, s \in T(\Omega, \mathcal{V})$ be terms and let $V = \mathcal{V}(t) \cup \mathcal{V}(s)$. Consider the set S of all $(\mathcal{V}, \Omega)$-substitutions φ, such that the following two conditions hold:

1. $nf_{\mathcal{R}_{\text{equ}}}(equ(\varphi(t), \varphi(s))) \in D_{\text{success}}$.
2. $D(\varphi) \subseteq V$ and $\mathcal{I}(\varphi) \cap V = \emptyset$.

Then S is a complete set of $E_{\mathcal{R}}$–unifiers of t and s away from V. □

Example 5.9 Let $\mathcal{R}_1$ be the term rewriting system of Example 5.2 and let $t = append(x, y)$, $s = CONS(A, CONS(B, NIL))$ and $\varphi = [x/CONS(A, CONS(B, NIL)), y/NIL]$. We show a derivation in the reduction calculus red$\mathcal{I}(\mathcal{R})$ (where we denote the equal term rewriting system belonging to $\mathcal{R}_1$ by $\mathcal{R}$):

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

and $\Theta = CONS(A, CONS(B, NIL)) \in T(\Omega, \mathcal{V})$. Therefore φ is an $E_{\mathcal{R}_1}$–unifier of t and s. □

5.2 Narrowing Calculus for $E_{\mathcal{R}}$–Unification

The narrowing calculus is a refinement of the reduction calculus. The idea of narrowing is to find the minimal substitution φ, such that a rewrite rule can be applied. Narrowing was first defined in [Sla74] and in [Lan75].

To keep track of the substitutions which are used in each narrowing step, we choose pairs of the form (t, ψ) as derivation forms of our narrowing calculus, where t is a term and ψ is the composition of all substitutions applied so far.

Definition 5.10 (narrowing relation)
Let $\mathcal{R} = (\Omega, R)$. The *narrowing relation associated with* $\mathcal{R}$, denoted by $\leadsto_{\mathcal{R}}$, is the binary relation over $T(\Omega, \mathcal{V}) \times Sub(\mathcal{V}, \Omega)$, such that, for every $t, s \in T(\Omega, \mathcal{V})$ and $\psi, \psi' \in Sub(\mathcal{V}, \Omega)$: $(t, \psi) \leadsto_{\mathcal{R}} (s, \psi')$ if the following three conditions hold:

1. There is an occurrence $u \in O(t)$ where $t/u \notin \mathcal{V}$, there is a substitution $\varphi \in Sub(\mathcal{V}, \Omega)$, there is a rule $(l \to r) \in R$, and there is a variable renaming ρ of the variables in l, such that:
   - $\mathcal{V}(\rho(l)) \cap \mathcal{V}(t) = \emptyset$ and
   - φ is the mgu of $\rho(l)$ and t/u.
2. $s = \varphi(t[u \leftarrow \rho(r)])$.
3. $\psi' = (\hat{\varphi}|_{\mathcal{V}(t)}) \circ \psi$. □

A particular derivation step $(t, \psi) \leadsto_{\mathcal{R}} (s, \psi')$ will sometimes be denoted by $(t, \psi) \leadsto_{\mathcal{R},u,\varphi|_{\mathcal{V}(t)},\xi}$ (s, ψ'), where u is the occurrence, φ the substitution, and ξ the rule involved in this step.

We want to use narrowing to find $E_{\mathcal{R}}$–unifiers for two terms t and s. As it was the case with reduction we will use the equal term rewriting system associated with $\mathcal{R}$ and start out with the pair $(equ\,(t,s),\varphi_\emptyset)$. In each step it has to be checked whether the two terms t' and s' of the derivation form $(equ\,(t',s'),\varphi)$ are unifiable. If so, then the substitution $\varphi\circ\sigma$ is an $E_{\mathcal{R}}$–unifier of t and s, where σ is the mgu of t' and s'. We will therefore choose the set of success forms D_{success} to be the set of pairs $(equ\,(\Theta_1,\Theta_2),\varphi)$, such that Θ_1 and Θ_2 are unifiable.

Definition 5.11 (narrowing calculus for $E_{\mathcal{R}}$–unification)

Let $\mathcal{R}=(\Omega,R)$ be a canonical term rewriting system.

1. The *narrowing calculus for $E_{\mathcal{R}}$–unification for $\mathcal{R}$*, denoted by $\text{nar}\mathcal{I}\,(\mathcal{R})$, is the following indexed calculus

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

   where

   - $\text{Equ}(T(\Omega,\mathcal{V}))=\{equ\,(t,s)\mid t,s\in T(\Omega,\mathcal{V})\}$
   - $D_{\text{success}}=\{(equ\,(\Theta_1,\Theta_2),\varphi)\mid \Theta_1,\Theta_2\in T(\Omega,\mathcal{V}),\ \Theta_1 \text{ and } \Theta_2 \text{ are unifiable}, \varphi\in \mathit{Sub}\,(\mathcal{V},\Omega)\}$
   - $<$ is the total order, such that for every $(u,l\to r),(u',l'\to r')\in\mathbb{N}^*\times R$: $(u,l\to r)<(u',l'\to l')$ if one of the following two conditions holds:
     (a) $u<_{lex}u'$
     (b) $u=u'$ and $\pi\,(l\to r)<\pi\,(l'\to r')$.

2. Let $(equ\,(t,s),\varphi_\emptyset)\in\text{Equ}(T(\Omega,\mathcal{V})\times\ \mathit{Sub}\,(\mathcal{V},\Omega)$. The *narrowing tree for $\mathcal{R}$ and $(equ\,(t,s)$, $\varphi_\emptyset)$* is the derivation tree $B(\text{nar}\mathcal{I}\,(\mathcal{R}),(equ\,(t,s),\varphi_\emptyset))$. □

In the following we might simply refer to $\text{nar}\mathcal{I}\,(\mathcal{R})$ as *narrowing calculus.*

Theorem 5.12 (completeness of $\text{nar}\mathcal{I}\,(\mathcal{R})$ for $E_{\mathcal{R}}$–unification)

Let $\mathcal{R}=(\Omega,R)$ be a canonical term rewriting system. Let $\text{nar}\mathcal{I}\,(\mathcal{R})$ be the narrowing calculus for $E_{\mathcal{R}}$–unification for $\mathcal{R}$ with success set D_{success}. Let $t,s\in T(\Omega,\mathcal{V})$ be terms and let $V=\mathcal{V}(t)\cup\mathcal{V}(s)$. Consider the set S of all $(\mathcal{V},\Omega)$-substitutions φ, such that the following two conditions hold:

1. There is a narrowing derivation

$$(equ\,(t,s),\varphi_\emptyset)\rightsquigarrow^*_{\mathcal{R}_{\text{equ}}}(equ\,(\Theta_1,\Theta_2),\mu)$$

   such that $(equ\,(\Theta_1,\Theta_2),\mu)\in D_{\text{success}}$ and Θ_1 and Θ_2 are unifiable by mgu ν.

2. $\varphi=(\mu\circ\nu)|_V$ and $\mathcal{I}(\varphi)\cap V=\emptyset$.

Then S is a complete set of $E_{\mathcal{R}}$–unifiers of t and s away from V.

Proof.

The proof follows immediately from the construction of the narrowing calculus $\text{nar}\mathcal{I}\,(\mathcal{R})$ and from Theorem 2 of [Hul80]. □

Example 5.13 Let $\mathcal{R}_1$ be the term rewriting system of Example 5.2. We now want to find $E_{\mathcal{R}_1}$–unifiers for the terms $t = append(x,y)$ and $s = CONS(A, CONS(B, NIL))$. In Example 5.9 it was shown that the substitution $\varphi = [x/CONS(A, CONS(B, NIL)),\ y/NIL]$ is an $E_{\mathcal{R}_1}$–unifier of t and s. How do we compute this substitution using narrowing? Consider the narrowing tree for $\mathcal{R}_{\text{equ}}$ and $equ(t,s)$ in Figure 3.

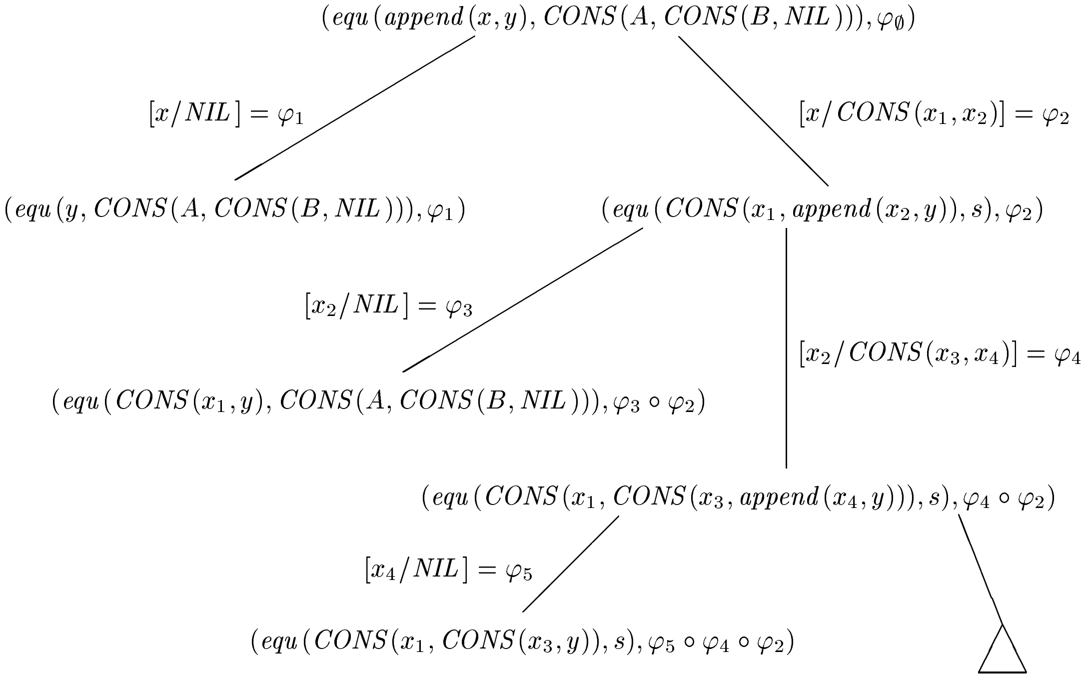

Figure 3: Narrowing tree for $equ(t,s)$.

All three leaves of the narrowing tree are labeled by success forms, namely:

1. $(equ(y,s), \varphi_1) \in D_{\text{success}}$, since there is an mgu $\sigma = [y/s]$, such that

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

   is an $E_{\mathcal{R}_1}$–unifier of t and s.

2. $(equ(CONS(x_1,y), CONS(A, CONS(B, NIL))), \varphi_3 \circ \varphi_2) \in D_{\text{success}}$, since there is an mgu $\sigma_2 = [x_1/A, y/CONS(B, NIL)]$, such that

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

   is an $E_{\mathcal{R}_1}$–unifier of t and s.

3. $(equ(CONS(x_1, CONS(x_3,y)), CONS(A, CONS(B, NIL))), \varphi_5 \circ \varphi_4 \circ \varphi_2) \in D_{\text{success}}$, since there is a mgu $\sigma_3 = [x_1/A, x_3/B, y/NIL]$, such that

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

   is an $E_{\mathcal{R}_1}$–unifier of t and s. □

It should be noted, that success forms of the narrowing calculus for $E_{\mathcal{R}}$–unification do not necessarily label leaves of the narrowing tree, i.e., they might label *arbitrary* nodes of the narrowing tree. Consider the term $\Theta = equ(append(CONS(A, NIL), NIL), x)$, then the narrowing tree $B(\text{nar}\mathcal{I}(\mathcal{R}_1), (\Theta, \varphi_\emptyset))$ with respect to the term rewriting system $\mathcal{R}_1$ of Example 5.2, would only consist of one node which is labeled by $(\Theta, \varphi_\emptyset)$, and thus $[x/append(CONS(A, NIL), NIL)]$ would be an $E_{\mathcal{R}_1}$–unifier, even though it still contains function symbols and is not in $\mathcal{R}_1$–normal form.

5.3 Leftmost Outermost Narrowing Calculus for $(E_{\mathcal{R}}, \Delta)$–Unification

We want to increase the efficiency of finding $E_{\mathcal{R}}$–unifiers by fixing a certain strategy for choosing a narrowing occurrence.

Definition 5.14 Let Ω be a ranked alphabet. A *strategy* is a total function ST: $T(\Omega, \mathcal{V}) \to \mathbb{N}^*$, such that, for every $t \in T(\Omega, \mathcal{V})$, the value $ST(t) \in O(t)$. □

The strategy that we want to deal with, will allow narrowing only at leftmost outermost occurrences. Consider a term $f(g(g(A, B), B), g(C, D), D)$ and a term rewriting system which includes a rule with the left-hand side $g(x, y)$. There are two leftmost occurrences of g in t, namely 11 and 1. The latter is the leftmost outermost occurrence; this is also the lexicographically minimal occurrence.

The leftmost outermost strategy can decrease the width and the height of a narrowing tree.

Definition 5.15 (set of narrowing occurrences)

Let $\mathcal{R} = (\Omega, R)$ be a term rewriting system and let $t \in T(\Omega, \mathcal{V})$ a term. The *set of narrowing occurrences for $\mathcal{R}$ and t*, denoted by $\text{nar}O(\mathcal{R}, t)$, is the set of occurrences u where $t/u \notin \mathcal{V}$ and: there is a substitution $\varphi \in Sub(\mathcal{V}, \Omega)$, a rule $(l \to r) \in R$ and a renaming ρ of the variables in l, such that the following two condition hold.

1. $\mathcal{V}(\rho(l)) \cap \mathcal{V}(t) = \emptyset$.
2. φ is the mgu of $\rho(l)$ and t/u. □

Definition 5.16 (leftmost outermost strategy)

Let $\mathcal{R} = (\Omega, R)$ be a term rewriting system and $t \in T(\Omega, \mathcal{V})$ a term to which the narrowing relation $\leadsto_{\mathcal{R}}$ can be applied, i.e., $\text{nar}O(\mathcal{R}, t) \neq \emptyset$. The *leftmost outermost narrowing strategy*, $ST_{lo} : T(\Omega, \mathcal{V}) \to \mathbb{N}^*$ is defined as:

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

□

The leftmost outermost narrowing relation is simply defined as narrowing respecting the leftmost outermost strategy.

Definition 5.17 (leftmost outermost narrowing relation)

Let $\mathcal{R} = (\Omega, R)$ be a term rewriting system. The *leftmost outermost narrowing relation associated with* $\mathcal{R}$, denoted by $\overset{lo}{\leadsto}_{\mathcal{R}}$, is the binary relation over $T(\Omega, \mathcal{V}) \times Sub(\mathcal{V}, \Omega)$, such that, for every $t, s \in T(\Omega, \mathcal{V})$ and $\psi, \psi' \in Sub(\mathcal{V}, \Omega)$: $(t, \psi) \overset{lo}{\leadsto}_{\mathcal{R}} (s, \psi')$ if the following three conditions hold.

1. There is an occurrence $u \in O(t)$ with $t/u \notin \mathcal{V}$ and $u = ST_{lo}(t)$, there is a substitution $\varphi \in Sub(\mathcal{V}, \Omega)$, there is a rule $(l \to r) \in R$, and there is a renaming ρ of the variables in l, such that
   (a) $\mathcal{V}(\rho(l)) \cap \mathcal{V}(t) = \emptyset$ and
   (b) φ is the mgu of $\rho(l)$ and t/u.
2. $s = \varphi(t[u \leftarrow \rho(r)])$.
3. $\psi' = (\hat{\varphi}|_{\mathcal{V}(t)}) \circ \psi$. □

Obviously, the leftmost outermost narrowing relation is a subset of the narrowing relation. Moreover, by imposing a strategy on the narrowing relation we have eliminated one non-determinism of the narrowing relation, namely the choice of an occurrence.

Leftmost outermost narrowing is not complete for arbitrary canonical term rewriting systems (cf. Example 1 of [Ech88]). Rather the class of term rewriting systems has to be restricted to so called ctn term rewriting system. Also it does not yield a complete set of $E_{\mathcal{R}}$–unifiers but rather produces a ground complete set of $(E_{\mathcal{R}}, \Delta)$–unifiers (cf. Definition 4.8). As preparation for the definition of ctn term rewriting system we recall the notion of strictly sub-unifiable from [Ech88].

Definition 5.18 (strictly sub–unifiable)

Two terms $t, t' \in T(\Omega, \mathcal{V})$ are *strictly sub–unifiable* if there is an occurrence $u \in O(t) \cap O(t')$, such that the following two conditions hold.

- Let ρ be a variable renaming, such that: $\mathcal{V}(t/u) \cap \mathcal{V}(\rho(t'/u)) = \emptyset$. The terms t/u and $\rho(t'/u)$ are unifiable with mgu σ_u, and σ_u is neither a variable renaming nor the empty substitution.
- For every occurrence w with $w < u$: $l[w] = l'[w]$. □

Definition 5.19 (ctn term rewriting system)

Let $\mathcal{R} = (\Omega, R)$ be a term rewriting system. $\mathcal{R}$ is a *canonical, totally defined, not strictly sub–unifiable term rewriting system*, for short ctn term rewriting system, if the following conditions hold:

1. $\mathcal{R}$ is canonical.
2. There are two ranked alphabets F and Δ, such that $\Omega = F \cup \Delta$ and $F \cap \Delta = \emptyset$. F is called *set of function symbols* and Δ is called *set of constructor symbols*.
3. Every left hand side of R is linear in $\mathcal{V}$.

4. Every left hand side of R is of the form $f(t_1, \ldots, t_n)$ where $f \in F$ and for every $i \in [n]$: $t_i \in T(\Delta, \mathcal{V})$.

5. Every function symbol in F is totally defined; that means for every $t \in T(\Omega, \mathcal{V})$: $nf_{\mathcal{R}}(t) \in T(\Delta, \mathcal{V})$.

6. The left hand sides of rules of R are pairwise *not strictly sub–unifiable.*

We will usually write $\mathcal{R} = (F, \Delta, R)$ for a ctn term rewriting systems. □

In fact, the term rewriting system in Example 5.2 is a ctn term rewriting system where $F = \{append^{(2)}\}$ is the set of function symbols and $\Delta = \{CONS^{(2)}, NIL^{(0)}, A^{(0)}, B^{(0)}, C^{(0)}\}$ is the set of constructor symbols.

The class of (term–to–term) functions which can be computed by ctn term rewriting systems is reasonably large for practical purposes, because it contains the class of primitive recursive tree functions [Hup78]. This follows from the fact that the class of primitive recursive tree function is characterized by modular tree transducers [EV91], and from the easy observation that every modular tree transducer is a particular ctn term rewriting system.

Analogous to Definition 5.6, we denote an equal ctn term rewriting system associated with a ctn term rewriting system $\mathcal{R} = (F, \Delta, R)$ by $\mathcal{R}_{\mathrm{equ}} = (F \cup \{equ^{(2)}\}, \Delta, R)$. Let us now define the lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification.

Definition 5.20 (lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification)

Let $\mathcal{R} = (F, \Delta, R)$ be a ctn term rewriting system.

1. The *lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification for $\mathcal{R}$*, denoted by lo–nar$\mathcal{I}(\mathcal{R})$, is the indexed calculus

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

   where

   - $\mathrm{Equ}(T(\Omega, \mathcal{V})) = \{equ\,(t, s) \mid t, s \in T(\Omega, \mathcal{V})\}$
   - $D_{\mathrm{success}} = \{(equ\,(\Theta_1, \Theta_2), \varphi) \mid \Theta_1, \Theta_2 \in T(\Delta, \mathcal{V}),\ \Theta_1$ and Θ_2 are unifiable, $\varphi \in \mathit{Sub}\,(\mathcal{V}, \Delta)\}$
   - $<$ is a total order, such that for all $(l \to r), (l' \to r') \in R$: $(l \to r) < (l' \to r')$ if $\pi(l \to r) < \pi(l' \to r')$.

2. Let $(equ\,(t, s), \varphi_\emptyset) \in \mathrm{Equ}(T(\Omega, \mathcal{V})) \times \mathit{Sub}\,(\mathcal{V}, \Omega)$. The *lo–narrowing tree for $\mathcal{R}$ and* $(equ\,(t, s), \varphi_\emptyset)$ is the derivation tree $B(\text{lo–nar}\mathcal{I}(\mathcal{R}), (equ\,(t, s), \varphi_\emptyset))$. □

In the following we might simply refer to lo–nar$\mathcal{I}(\mathcal{R})$ as *lo–narrowing calculus.*

Theorem 5.21 (completeness of lo–nar$\mathcal{I}(\mathcal{R})$ for $(E_{\mathcal{R}}, \Delta)$—unification)

Let $\mathcal{R} = (F, \Delta, R)$ be a ctn term rewriting system. Let lo–nar$\mathcal{I}(\mathcal{R})$ be the leftmost–outermost narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification with success set D_{success}. Let $t, s \in T(F \cup \Delta, \mathcal{V})$ be terms and let $V = \mathcal{V}(t) \cup \mathcal{V}(s)$. Consider the set S of all $(\mathcal{V}, \Delta)$-substitutions φ, such that the following two conditions hold:

1. There is a lo–narrowing derivation

$$(equ\,(t,s),\varphi_\emptyset)\overset{lo}{\rightsquigarrow}{}^{*}_{\mathcal{R}_{\mathrm{equ}}}(equ\,(\Theta_1,\Theta_2),\mu)$$

   such that $(equ\,(\Theta_1,\Theta_2),\mu)\in D_{\mathrm{success}}$ and Θ_1 and Θ_2 are unifiable by ν.

2. $\varphi=(\mu\circ\nu)|_V$ and $\mathcal{I}(\varphi)\cap V=\emptyset$.

Then S is a ground complete set of $(E_{\mathcal{R}},\Delta)$–unifiers of t and s away from V.

Proof.

The proof follows immediately from the construction of the lo–narrowing calculus lo–nar$\mathcal{I}\,(\mathcal{R})$ and from Theorem 3 of [Ech88]. □

Example 5.22 We reconsider the term rewriting system $\mathcal{R}_1$ of Example 5.2, but this time we write it as a ctn term rewriting system (F,Δ,R).

- $F=\{append^{(2)}\}$
- $\Delta=\{CONS^{(2)},NIL^{(0)},A^{(0)},B^{(0)},C^{(0)}\}$
- R is composed of the following rules:

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

We want to $(E_{\mathcal{R}_1},\Delta)$–unify the terms $t=CONS(B,CONS(A,NIL))$ and $s=append(z_1,$ $append(z_2,NIL))$. Consider the leftmost outermost narrowing tree in Figure 4. For the sake of brevity, we have omitted the substitutions in the derivation forms and we have abbreviated *append* by *ap*. Now we list the three $(E_{\mathcal{R}_1},\Delta)$–unifiers (underlined in Figure 4):

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

It should be noted that the subtrees next to the nodes which are labeled by success forms, contain infinite branches. For the deterministic calculus for lo–nar$\mathcal{I}\,(\mathcal{R})$, this implies that only the first $(E_{\mathcal{R}_1},\Delta)$–unifier, i.e., φ_1 will be found, because the deterministic calculus will not return from the infinite subtree next to the node labeled by the success form $(equ\,(CONS(x,CONS(z_3,$ $NIL)),\,t),\varphi_1)$. □

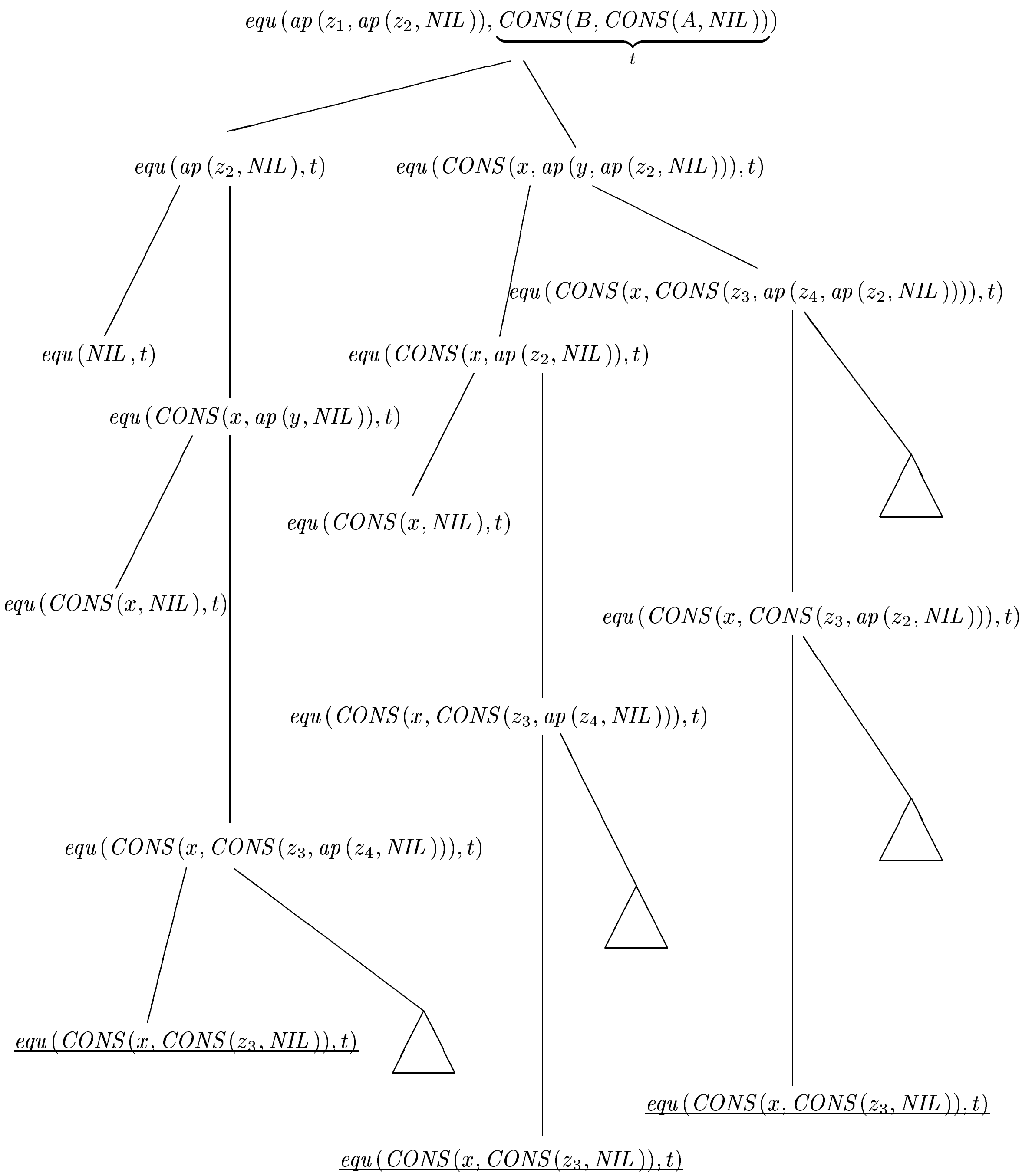

Figure 4: lo–narrowing tree for the terms s and t

5.4 Unification Driven LO–Narrowing Calculus for $(E_\mathcal{R}, \Delta)$–Unification

In this section we will construct the fourth calculus that will again increase the efficiency of finding $(E_\mathcal{R}, \Delta)$–unifiers as compared to lo–narrowing. A sequence of term decomposition steps will be incorporated into the lo–narrowing derivation; these steps simulate the usual term unification which was done at the end of the lo–narrowing derivation. The gain of efficiency is due to the fact that the decomposition steps take place as early as possible. Thereby, some of the lo–narrowing steps will not be performed; and some of the infinite trees paths can be omitted. Clearly, the decomposition steps also depend on an occur check. Here we restrict the occur check to the skeleton of the considered term which is, roughly speaking, the upper part of the term up to function symbols. The integration of the decomposition steps and the occur check into the lo–narrowing relation leads to the unification-driven leftmost-outermost narrowing relation which was introduce in [FV94].

Definition 5.23 (decompositions rules, extension)
Let $\mathcal{R} = (F, \Delta, R)$ be a ctn term rewriting system.

- Let $\sigma \in \Delta^{(k)}$, with $k \geq 0$. The *decomposition rule for* σ is of the form:
$$equ\,(\sigma(x_1, \ldots, x_k), \sigma(x_{k+1}, \ldots, x_{2k})) \rightarrow \sigma(equ\,(x_1, x_{k+1}), \ldots, equ\,(x_k, x_{2k})).$$
- The *decomposition part of* $\mathcal{R}$, denoted by $\mathcal{R}(\Delta)$, is the tuple $(\hat{F}, \Delta, R(\Delta))$, where $\hat{F} = F \cup \{equ^{(2)}\}$ and $R(\Delta)$ is the set of all decomposition rules for the elements of Δ.
- The *extension of* $\mathcal{R}$, denoted by $\hat{\mathcal{R}}$, is the ctn term rewriting system $(\hat{F}, \Delta, \hat{R})$, where $\hat{R} = R \cup R(\Delta)$. □

In Section 2.5 we have assumed that a bijection π is associated with the set R of rules of the term rewriting system. We extend this bijection to $\hat{\pi} : \hat{R} \rightarrow [card(\hat{R})]$, such that $\hat{\pi}|_R = \pi$ and the enumeration of the decomposition rules is arbitrary.

Definition 5.24 (important occurrence, skeleton, occur check)
Let $\mathcal{R} = (F, \Delta, R)$ be a ctn term rewriting system and let $t \in T(\hat{F} \cup \Delta, \mathcal{V})$.

- The *important occurrence in* t, denoted by $impO\,(t)$ is the occurrence $min_{lex}(\{u \in O(t) \mid t[u] = equ\})$.
- t is *in binding mode* if $t[impO\,(t)1], t[impO\,(t)2] \in \mathcal{V}$.
- The $(\Delta \cup \mathcal{V})$*–skeleton of* t is the set $\{u \in O(t) \mid$ there is no $v \in O(t)$, such that $v \leq u$ and $t[v] \in F\}$
- The *occur check for* t *succeeds* if the following conditions hold:
  1. The term t is not in binding mode.
  2. Let $t/impO\,(t) = equ\,(s, s')$. Either
     - $s \in \mathcal{V}, s' \notin \mathcal{V}$ and there is an occurrence u in s', such that $s'/u = s$, or
     - $s \notin \mathcal{V}, s' \in \mathcal{V}$ and there is an occurrence u in s, such that $s/u = s'$. □

Definition 5.25 (ulo–narrowing relation)
Let $\mathcal{R} = (F, \Delta, R)$ be a ctn term rewriting system. The *unification driven leftmost outermost narrowing relation associated with* $\hat{\mathcal{R}}$ (for short: ulo-narrowing relation), denoted by $\overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}}$, is the binary relation over $T(F \cup \Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Delta)$ defined as follows: for every $t, s \in T(\hat{F} \cup \Delta, \mathcal{V})$ and $\psi, \psi' \in Sub\,(\mathcal{V}, \Delta) : (t, \psi) \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}} (s, \psi')$ if $t/impO\,(t) = equ\,(t_1, t_2)$, where $t_1, t_2 \in T(F \cup \Delta, \mathcal{V})$ and one of the following five conditions holds:

1. $(t_1[\varepsilon], t_2[\varepsilon] \in \Delta$ and $t_1[\varepsilon] = t_2[\varepsilon])$ or $(((t_1[\varepsilon] \in \Delta$ and $t_2[\varepsilon] \in \mathcal{V})$ or $(t_1[\varepsilon] \in \mathcal{V}$ and $t_2[\varepsilon] \in \Delta))$ and the occur check for t does not succeed) and the following conditions hold:
   (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. $t_1, t_2 \in \mathcal{V}, t_1 \neq t_2$, and the following three conditions hold:
   Let $k = min\{i \mid z_i \in \mathcal{V} \setminus \mathcal{V}(t) \cup \mathcal{D}(\psi) \cup \mathcal{I}(\psi))\}$:
   (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. $t_1, t_2 \in \mathcal{V}, t_1 = t_2$, and the following two conditions hold:
   (a) $s = t[impO\,(t) \leftarrow t_1]$.
   (b) $\psi' = \psi$.
4. $t_1[\varepsilon] \in F$ and the following three conditions hold:
   (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. $t_1[\varepsilon] \notin F$ and $t_2[\varepsilon] \in F$ and the following three conditions hold:
   (a) $(t_2, \varphi_\emptyset) \overset{lo}{\rightsquigarrow}_{\mathcal{R}} (t', \varphi')$.
   (b) $s = \varphi'(t[impO\,(t)2 \leftarrow t'])$.
   (c) $\psi' = \varphi' \circ \psi$. □

If a rule $(l \to r) \in \hat{R}$ is applied, i.e., in cases 1, 4, and 5, then we write $\overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}, l \to r}$. In cases 2 and 3 we write $\overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}, bm}$ to indicate that the current term is in binding mode. Let us now define the ulo–narrowing calculus. As index set we choose the set $\hat{R} \cup \{bm\}$ with bm as minimal element. We will use $<$ for natural numbers as total order on the index set and compare the enumeration $\hat{\pi}$ wherein the symbol bm is arbitrarily defined as minimal element.

Definition 5.26 (ulo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification)
Let $\mathcal{R} = (F, \Delta, R)$ be a ctn term rewriting system with enumeration π.

1. The *unification driven lo–narrowing calculus for* $(E_{\mathcal{R}}, \Delta)$*–unification*, denoted by ulo-nar$\mathcal{I}\,(\mathcal{R})$, is the indexed calculus

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

   where

- $D_{\text{success}} = T(\Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Delta)$
- $<$ is the total order on $\hat{R} \cup \{bm\}$, such that for every $\xi, \xi' \in \hat{R} \cup \{bm\}$, $\xi < \xi'$ if one of the following conditions holds:
  (a) $\xi = bm$ and $\xi' \neq bm$.
  (b) $\xi \neq bm, \xi' \neq bm$ and $\hat{\pi}(\xi) < \hat{\pi}(\xi')$.

2. Let $(equ(t, s), \varphi_\emptyset) \in T(F \cup \Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Omega)$. The *ulo–narrowing tree for $\mathcal{R}$ and $(equ(t, s), \varphi_\emptyset)$* is the derivation tree $B(\text{ulo–nar}\mathcal{I}(\mathcal{R}), (equ(t, s), \varphi_\emptyset))$. □

In the following we might simply refer to ulo–nar$\mathcal{I}(\mathcal{R})$ as *ulo–narrowing calculus*.

Theorem 5.27 (completeness of ulo–nar$\mathcal{I}(\mathcal{R})$ for $(E_\mathcal{R}, \Delta)$–unification)
Let $\mathcal{R} = (F, \Delta, R)$ be a ctn term rewriting system. Let ulo–nar$\mathcal{I}(\mathcal{R})$ be the unification driven lo–narrowing calculus for $(E_\mathcal{R}, \Delta)$–unification with success set D_{success}. Let $t, s \in T(F \cup \Delta, \mathcal{V})$ be terms and let $V = \mathcal{V}(t) \cup \mathcal{V}(s)$. Consider the set S of all $(\mathcal{V}, \Delta)$-substitutions φ, such that the following two conditions hold:

1. There is a ulo–narrowing derivation
$$(equ(t, s), \varphi_\emptyset) \overset{u}{\rightsquigarrow}{}^{*}_{\hat{\mathcal{R}}} (\Theta, \mu)$$
such that $(\Theta, \mu) \in D_{\text{success}}$.
2. $\varphi = \mu|_V$ and $\mathcal{I}(\varphi) \cap V = \emptyset$.

Then S is a ground complete set of $(E_\mathcal{R}, \Delta)$–unifiers of t and s away from V.

Proof.

The proof follows immediately from the construction of the unification driven lo–narrowing calculus ulo–nar$\mathcal{I}(\mathcal{R})$ and from Theorem 4.7 of [FV94]. □

Example 5.28 Let $\mathcal{R}_1$ be the ctn term rewriting system of Example 5.2. The decomposition rules for the constructor symbols are:

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

Figures 5 and 6 show the ulo–narrowing tree for $\mathcal{R}_1$ and the term $(equ(t, s), \varphi_\emptyset)$ where $s = append(z_1, append(z_2, NIL))$ and $t = CONS(B, CONS(A, NIL))$ and where we have again abbreviated *append* by *ap*. The terms p and q which label two of the leaves are:

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

At the nodes labeled by p and q the derivation stops due to the constructor consistency check. □

The previous example illustrates that the ulo–narrowing calculus is capable of reducing the infinite lo–narrowing tree of Example 5.2 to a finite ulo–narrowing tree. Of course this is not generally the case, because then the $(E_{\mathcal{R}}, \Delta)$–unification problem would be decidable for ctn term rewriting systems which, in its turn, is not true.

Let us consider a final example that shows how a small change in the enumeration function π yields an infinite ulo–narrowing tree.

Example 5.29 Let $\mathcal{R}_1$ be the ctn term rewriting system of Example 5.22 and let the enumeration function $\hat{\pi}$ be defined as follows (we drop the decomposition rules for A, B, and C, because they are not used here):

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

The terms $t = append\,(x, NIL)$ and $s = append\,(y, x)$ are to be $(E_{\mathcal{R}_1}, \Delta)$–unified. The following ulo–narrowing derivation yields a success:

$$\begin{array}{ll} (equ\,(ap\,(x, NIL), ap\,(y,x)), \varphi_\emptyset) & \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}_1,1,(2)}\\ (equ\,(NIL, ap\,(y, NIL)), \varphi_1 = [x/NIL]) & \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}_1,2,(2)}\\ (equ\,(NIL, NIL), \varphi_2 = [y/NIL] \circ \varphi_1) & \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}_1,\varepsilon,(4)}\\ (NIL, \varphi_2) & \end{array}$$

However this success form labels a node which is on the rightmost path of the corresponding ulo–narrowing tree. Let us take a look at the leftmost path which is the path that the deterministic ulo–narrowing calculus would traverse first (again *append* is abbreviated by *ap*):

$$\begin{array}{ll} (equ\,(ap\,(x, NIL), ap\,(y,x)), \psi_\emptyset) & \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}_1,1,(1)}\\ (equ\,(CONS(z_1, ap\,(z_2, NIL)), ap\,(y, CONS(z_1,z_2))), \psi_1 = [x/CONS(z_1,z_2)]) & \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}_1,2,(1)}\\ (equ\,(CONS(z_1, ap\,(z_2, NIL)), & \\ \quad CONS(z_3, ap\,(z_4, CONS(z_1,z_2)))), \psi_2 = [y/CONS(z_3,z_4)]) & \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}_1,\varepsilon,(3)}\\ (CONS(equ\,(z_1,z_3), equ\,(ap\,(z_2, NIL), ap\,(z_4, CONS(z_1,z_2)))), \psi_2) & \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}_1,1,bm}\\ (CONS(z_5, equ\,(ap\,(z_2, NIL), ap\,(z_4, CONS(z_5,z_2)))), \psi_3 = [z_1/z_5, z_3/z_5] \circ \psi_2) & \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}_1,21,(1)}\\ (CONS(z_5, equ\,(CONS(z_6, ap\,(z_7, NIL)), & \\ \quad ap\,(z_4, CONS(z_5, CONS(z_6,z_7))))), \psi_4 = [z_2/CONS(z_6,z_7)] \circ \psi_3) & \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}_1,22,(1)}\\ (CONS(z_5, equ\,(CONS z_6, ap\,(z_7, NIL)), & \\ \quad CONS(z_8, ap\,(z_9, CONS(z_5, CONS(z_6,z_7)))))), \psi_5 = [z_4/CONS(z_8,z_9)] \circ \psi_4) & \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}_1,2,(3)}\\ (CONS(equ\,(z_6,z_8), equ\,(ap\,(z_7, NIL), ap\,(z_9, CONS(z_5, CONS(z_6,z_7))))), \psi_5) & \overset{u}{\rightsquigarrow}_{\hat{\mathcal{R}}_1,21,bm}\\ (CONS(z_5, CONS(z_{10}, equ\,(ap\,(z_7, NIL), & \\ \quad ap\,(z_9, CONS(z_5, CONS(z_6,z_7))))), \psi_6 = [z_6/z_{10}, z_8/z_{10}] \circ \psi_5) & \\ \quad\vdots & \end{array}$$

Obviously this is an infinite branch of the ulo–narrowing tree, although at all nodes where rule (1) can be applied we can apply rule (2). After repeatedly applying rule (2), $(E_{\mathcal{R}_1}, \Delta)$–unifiers of the form $CONS(z_i, CONS(z_j, \ldots))$ would be derived.

The substitution ψ_5 in the above derivation is the following:

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

One can observe that, in general, after each fourth derivation step we get a substitution $\psi = [x/CONS(z_5, CONS(z_{10}, CONS(\ldots, CONS(z_{5n}, z_{5n-2})))), [y/CONS(z_5, CONS(z_{10}, CONS(\ldots, CONS(z_{5n}, z_{5n-1})))).$

Therefore this branch is infinite; the deterministic ulo–narrowing calculus does not find a success form and does not terminate. □

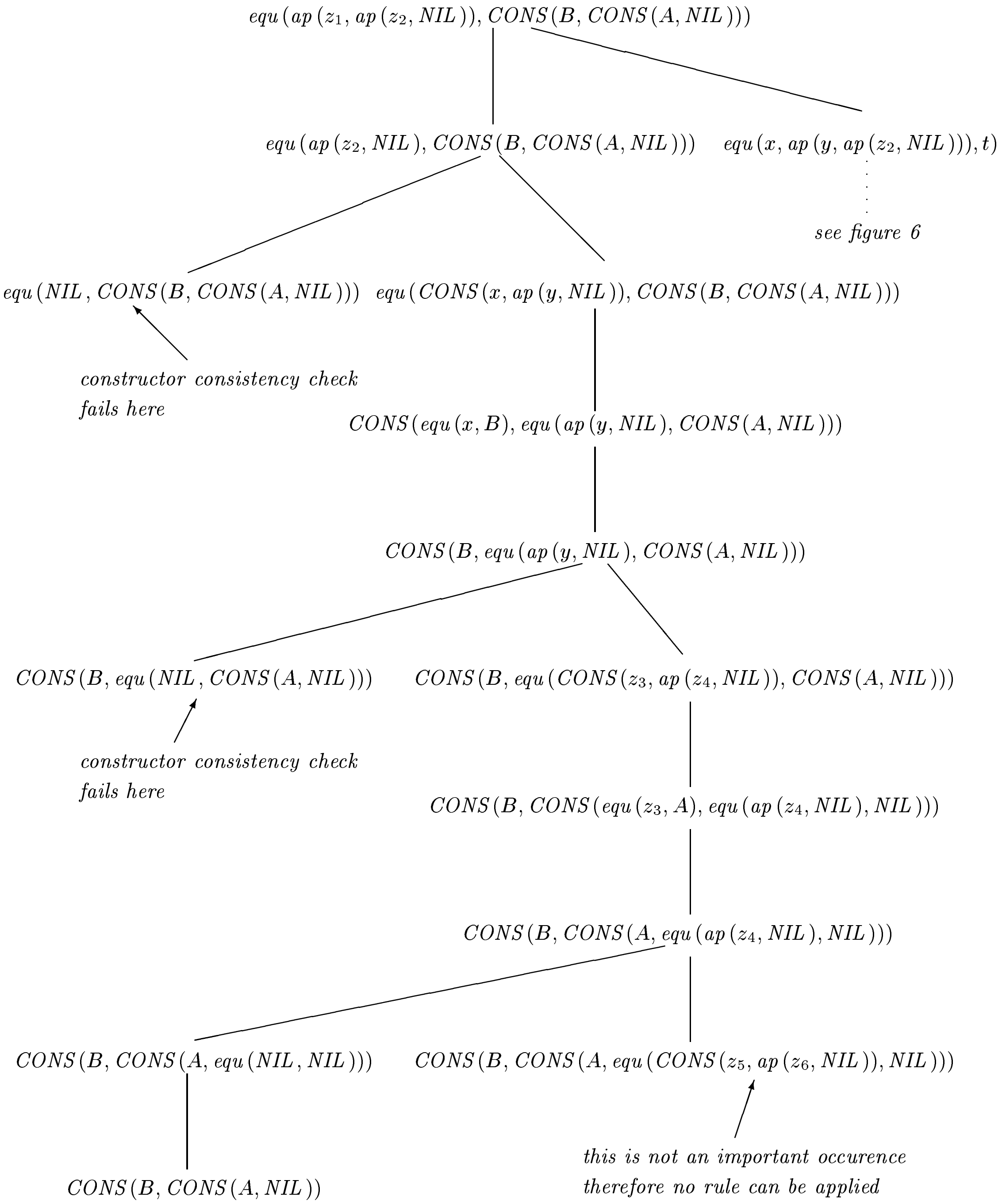

Figure 5: ulo–narrowing tree for $equ(ap(z_1, ap(z_2, NIL)), t)$

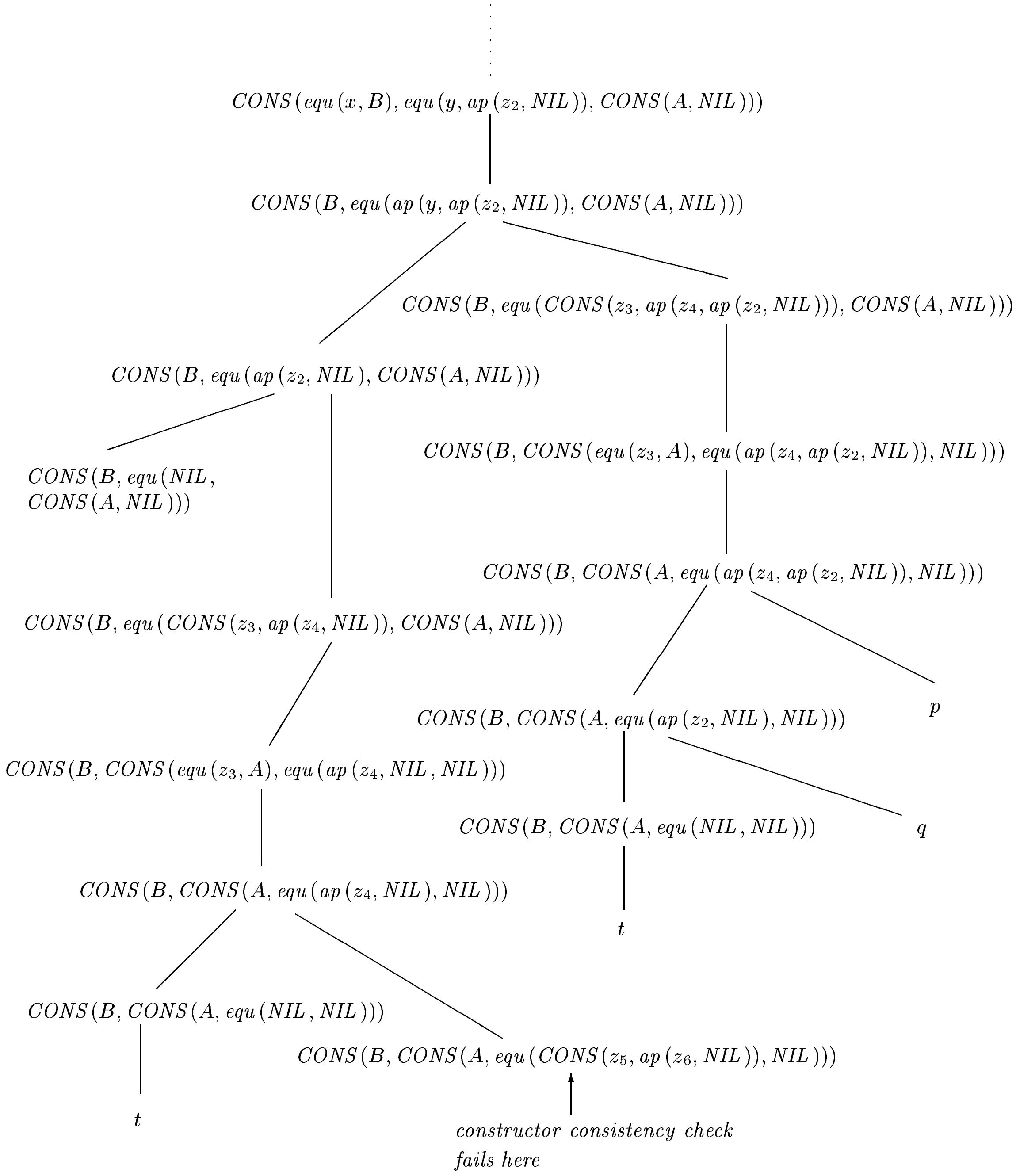

Figure 6: right branch of the ulo–narrowing tree for $equ(ap(z_1, ap(z_2, NIL)), t)$

6 Efficiency Measures

In this section we want to define efficiency measures that will allow us to compare the efficiencies of finding success forms of two indexed calculi $\mathcal{I}$ and $\mathcal{I}'$. Efficiencies will always be represented by rational numbers (in fact, mostly even by natural numbers) or by the symbol ∞ if no success form can be derived. The comparison of efficiencies is based on the usual "less than" order on rational numbers. Instead of this rather simple measuring space $(\mathbb{Q}, <)$ one could also try to consider more general measuring spaces $(N, \leq)$, where N is an arbitrary set and $\leq$ is a *partial* order on N. Here, however, we stick to $(\mathbb{Q}, <)$ as measuring space.

In general, one can understand a measuring space as an *abstract interpretation*; namely that an indexed calculus $\mathcal{I}$ and a calculandum d is interpreted over the abstract domain of the natural numbers. The notion of abstract interpretations goes back to [CC79] who interpreted programs over abstract domains.

In particular, we focus our attention on the comparison of efficiencies of the calculi for $E_{\mathcal{R}}$-unification and for $(E_{\mathcal{R}}, \Delta)$-unification that were introduced in Section 5. Each of these calculi is an indexed calculus. There, the derivation relation $\vdash$ is partitioned into relations $\vdash_j$ where $j \in J$. This fixes one unambiguous arrangement of the nodes in the corresponding derivation tree. The deterministic calculus for an indexed calculus defines a depth–first left-to-right tree traversal over this tree. Hence, one possible efficiency measure is the number of steps needed by the deterministic calculus for an indexed calculus $\mathcal{I}$, to derive the first success form; we call this measure *single success efficiency.* We could also consider the number of steps needed to find the first n different success forms (*multi success efficiency*).

Of course, for derivation trees which contain many infinite branches, these two basic measures might not be sufficient, because efficiencies will tend to be considered equal then. For this reason, we define two other efficiency measures. One measure is the number of success forms depending on the depth of their occurrences in the derivation tree. We call this efficiency measure *success density efficiency.* Another measure is the number of infinite paths in a derivation tree. This measure is called *infinite path efficiency.*

In the following subsections, we will introduce these efficiency measures formally. More precisely, we introduce

- the single success efficiency (in Section 6.1)
- the multi success efficiency (in Section 6.2)
- the success density efficiency (in Section 6.3.1)
- the infinite path efficiency (in Section 6.3.2),

all of which are defined for indexed calculi. Then, in Section 7, we will compare

1. the narrowing calculus $\mathrm{nar}\mathcal{I}(\mathcal{R})$ with the lo-narrowing calculus $\mathrm{lo{-}nar}\mathcal{I}(\mathcal{R})$ and
2. the lo-narrowing calculus $\mathrm{lo{-}nar}\mathcal{I}(\mathcal{R})$ with the ulo-narrowing calculus $\mathrm{ulo{-}nar}\mathcal{I}(\mathcal{R})$

with respect to both, the single success efficiency and the multi success efficiency. The other two measures will not be considered.

6.1 Single Success Efficiency

The single success efficiency is a function that assigns a natural number to every calculandum d of some indexed calculus $\mathcal{I}$. It measures how many derivation steps the deterministic calculus for $\mathcal{I}$ needs to find the first success form. If no success form can be derived, then the symbol ∞ will be assigned.

Definition 6.1 (single success efficiency)
Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ be an indexed calculus and let $det(\mathcal{I}) = (D', D'_{\text{success}}, \models)$ be the deterministic calculus for $\mathcal{I}$. For every $d \in D$, the set $E(d)$ of those numbers of derivation steps of the deterministic calculus for $\mathcal{I}$, which yield success forms, is defined by

$$E(d) = \{k \mid k \geq 0 \text{ and } Suc_k\,((d, I(d)), \models) \cap D'_{\text{success}} \neq \emptyset\}.$$

The *single success efficiency of* $\mathcal{I}$, denoted by $\eta(\mathcal{I})$, is the function $\eta(\mathcal{I}) : D \to (\mathbb{N} \cup \{\infty\})$, such that for all $d \in D$:

$$(\eta(\mathcal{I}))\,(d) = \begin{cases} \min\,(E(d)), & \text{if } E(d) \neq \emptyset \\ \infty, & \text{otherwise} \end{cases}$$

□

The definition of the single success efficiency makes use of the set $Suc_k\,((d, I(d)), \models)$, which contains the derivation forms of $det\,(\mathcal{I})$ that can be derived from $(d, I(d))$ by $\models$ in k derivation steps. By Lemma 3.13 this set contains at most one element, and therefore there is one unique k in $E(d)$, for every success form that can be reached by the deterministic calculus for $\mathcal{I}$.

Example 6.2 Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ be the indexed calculus of Example 3.6. Construct the indexed calculus $\mathcal{I}'$ from $\mathcal{I}$ by replacing the success set D_{success} by the set

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

i.e., $\widehat{D}_{\text{success}}$ contains all derivation forms that are in normal form. Consider the calculandum $d = (a * (a * (b + c)))$.

According to Lemma 3.9, the derivation tree $B(\mathcal{I}', d)$ for $\mathcal{I}'$ and d is identical to the derivation tree shown in Figure 2, because derivation trees do not depend on the set of success forms. Then

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

because (let $\models$ be the derivation relation of the deterministic calculus for $\mathcal{I}'$)

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

□

$$\begin{tikzcd}
\mathcal{I}\colon & d & \vdash_j & \bar{d} \\
\mathcal{I}'\colon & d' \arrow[u, "h"] & \vdash'^{+} & \bar{d}' \arrow[u, "h"']
\end{tikzcd}$$

Figure 7: $\mathcal{I}$ is h–comparable with $\mathcal{I}'$.

The question arises which conditions should be put on two indexed calculi $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ and $\mathcal{I}' = (D', D'_{\text{success}}, (J', <'), (\vdash'_j \mid j \in J'))$ to be able to compare their single success efficiencies. If we required that the sets D and D' are equal, then we would, for instance, not be able to compare the lo–narrowing calculus lo–nar$\mathcal{I}(\mathcal{R})$ with the ulo–narrowing calculus ulo–nar$\mathcal{I}(\mathcal{R})$, because then $D \neq D'$.

Hence, we just require that there is a partial function $h : D' \to D$ such that, roughly speaking, for an element $d' \in D'$, the element $h(d')$ is the element represented by d' in the calculus $\mathcal{I}$. Moreover, we require that every derivation step induced by $\vdash_j$, for some $j \in J$, can be simulated by a number of derivation steps induced by $\vdash' = \bigcup_{i \in J} \vdash'_i$ (cf. Figure 7). Finally, we require that, for every success form of $\mathcal{I}'$, there is a corresponding success form of $\mathcal{I}$. This notion of representation is closely related to the data representation of [Hoa72] (also cf. [EV86]).

Definition 6.3 (h–comparable indexed calculi)
Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ and $\mathcal{I}' = (D', D'_{\text{success}}, (J', <'), (\vdash'_j \mid j \in J'))$ be two indexed calculi. Let $h : D' \to D$ be a partial function. Then $\mathcal{I}$ is *h–comparable with $\mathcal{I}'$*, if the following two conditions hold.

1. $\hat{h}(D'_{\text{success}}) \subseteq D_{\text{success}}$, where $\hat{h}$ is the usual extension of h to sets.
2. For every $d, \bar{d} \in D$ and $j \in J$ with $d \vdash_j \bar{d}$ there are $d', \bar{d}' \in D'$, such that $h(d') = d$, $h(\bar{d}') = \bar{d}$, and $d' \vdash'^{+} \bar{d}'$; where $\vdash' = \bigcup_{i \in J'} \vdash'_i$. □

We want to note that our definition of comparability is an attempt to put weak conditions on the indexed calculi, while still guaranteeing that the semantics of the calculi is similar. Of course it does not make sense to compare efficiencies for $(\eta(\mathcal{I}'))(d') = 0$ because then no derivation step occurs, and the calculandum d' remains unchanged (obviously there is no more efficient way to leave a calculandum unchanged than applying no derivation steps at all).

For the following definitions of *more single success efficient*, *more k–success efficient*, *more success density efficient*, and *more infinite path efficient* it will always be required that there is a partial function h, such that the two indexed calculi being compared are h-comparable.

Definition 6.4 (more single success efficient)
Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ and $\mathcal{I}' = (D', D'_{\text{success}}, (J', <'), (\vdash'_j \mid \in J'))$ be two indexed calculi and let $h : D' \to D$ be a partial function. Then $\mathcal{I}$ is *more single success efficient w.r.t. h than $\mathcal{I}'$* if the following two conditions hold:

1. $\mathcal{I}$ is h–comparable with $\mathcal{I}'$.

2. For every $d' \in D'$ with $(\eta(\mathcal{I}'))\,(d') \geq 1$: $(\eta(\mathcal{I}))\,(h(d')) \leq (\eta(\mathcal{I}'))\,(d')$, and there is a $\bar{d}' \in D'$, such that $(\eta(\mathcal{I}))\,(h(\bar{d}')) < (\eta(\mathcal{I}'))\,(\bar{d}')$. □

6.2 Multi Success Efficiency

In general, the set D_{success} of success forms will contain more than one element. Sometimes the single success efficiency does not provide a characteristic statement about the efficiencies of the indexed calculi involved. For example, the deterministic calculus for an indexed calculus $\mathcal{I}$ could derive the first success form after very few steps, but then derive no more success forms, because it moves into an infinite branch. However, the indexed calculus $\mathcal{I}$ would be more single success efficient than the calculus $\mathcal{I}'$, if the deterministic calculus for $\mathcal{I}'$ needs more steps to compute the first success form, but on the other hand derives more success forms in total. Thus, if the derivation of more than one success form is important, then we need a different measure. The *k–success efficiency* or *multi success efficiency* is such a measure. It depends on the derivation steps needed by the deterministic calculus for an indexed calculus $\mathcal{I}$ to derive the first k different success forms starting with a calculandum d.

Under which conditions are two success forms considered to be different? If we simply consider syntactical difference, then two success forms $d = (\mathit{CONS}(z_1, \mathit{NIL}), \underbrace{[x/\mathit{CONS}(z_1, \mathit{NIL})]}_{\varphi})$ and $d' = (\mathit{CONS}(z_7, \mathit{NIL}), \underbrace{[x/\mathit{CONS}(z_7, \mathit{NIL})]}_{\varphi'})$ would be considered different, even though the substitution φ' is merely a variable renaming of φ. This means that syntactical difference does *not* mean semantical difference in this context. Therefore it seems reasonable to include an equivalence relation $\sim$ in the definition of the k–success efficiency, which is defined on the set of derivation forms D of the corresponding indexed calculus $\mathcal{I}$. Then, for two derivation forms d and d': if $d \not\sim d'$, then d and d' are considered to be different.

Let $\mathcal{I}$ be an indexed calculus and d a calculandum. Before we define the notion of k–success efficiency, we will define a set that contains all success forms of $\mathcal{I}$ that can be derived from d in at most n derivation steps and which are pairwise different with respect to $\sim$.

Definition 6.5 (set of n–derivable success forms)

Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ be an indexed calculus, $\mathit{det}(\mathcal{I}) = (D', D'_{\text{success}}, \models)$ the deterministic calculus for $\mathcal{I}$, $d \in D$ a calculandum, $n \in \mathbb{N}$, and $\sim\, \subseteq D \times D$ an equivalence relation. The *set of n–derivable, $\sim$–different success forms from d*, denoted by $E(n, d, \mathcal{I}, \sim)$ is the set

$$S(n, d, \mathcal{I})/_{\sim}$$

where $S(n, d, \mathcal{I}) = \bigcup_{i \in [n]} \{d' \in D_{\text{success}} \mid \beta(d', I(d')) \in \mathit{Suc}_i\,((d, I(d)), \models) \text{ for some } \beta \in D'\}$. □

Definition 6.6 (k–success efficiency)

Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ be an indexed calculus, $\sim\subseteq D \times D$ an equivalence relation, and k a natural number with $k \geq 2$. The *k–success efficiency of $\mathcal{I}$ and $\sim$*, denoted by $\eta_k(\mathcal{I}, \sim)$, is a function $\eta_k(\mathcal{I}, \sim) : D \to (\mathbb{N} \cup \{\infty\})$, such that for every $d \in D$:

$$(\eta_k(\mathcal{I}, \sim))\,(d) = \begin{cases} \min\{j \mid |E(j, d, \mathcal{I}, \sim)| = k\}, & \text{if there is a } j\text{, such that } |E(j, d, \mathcal{I}, \sim)| = k \\ \infty, & \text{otherwise.} \end{cases}$$

□

Example 6.7 (for k–success efficiency)

Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ be the indexed calculus which we only define informally by a figure of the derivation tree of $\mathcal{I}$ and the calculandum $d = (\mathit{append}\,(x, \mathit{append}\,(y, \mathit{NIL})), \varphi_\emptyset)$ (see Figure 8). Note that this derivation tree just serves as an example and does *not* correspond to the term rewriting system of Example 5.2.

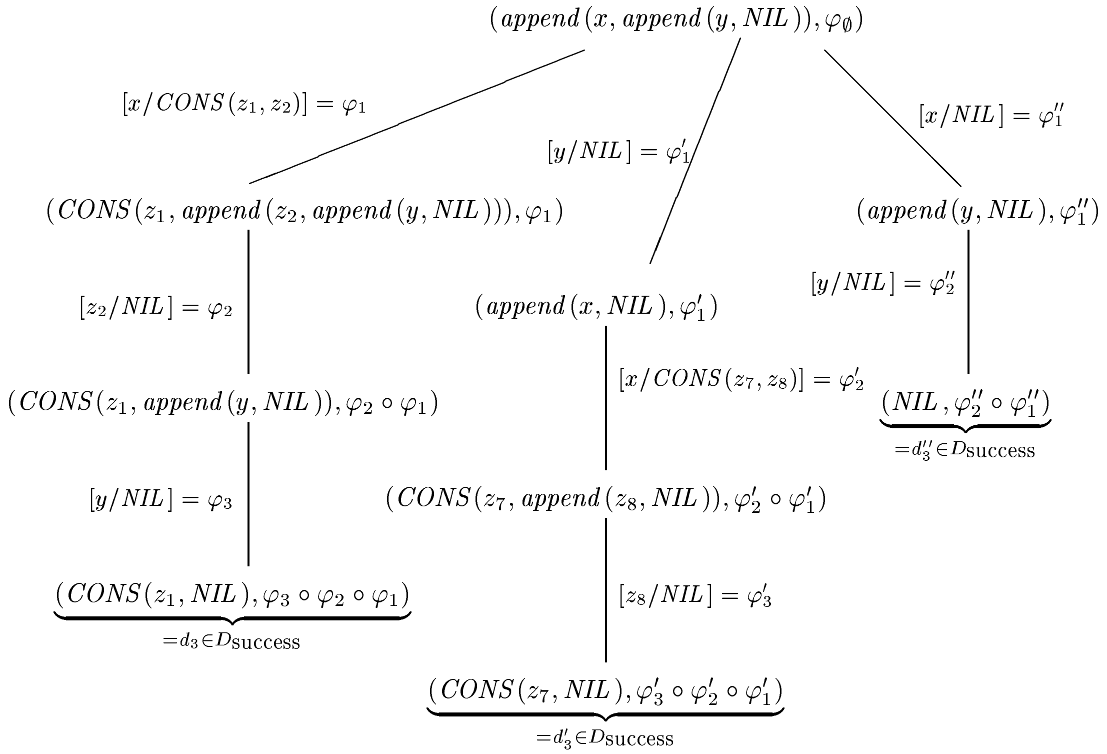

Figure 8: Derivation tree for $\mathcal{I}$ and calculandum d

We denote the derivation forms in the leftmost path of the derivation tree by d_i, where i is the level in the tree; e.g., $d_0 = d$ and $d_3 = (\mathit{CONS}(z_1, \mathit{NIL}), \varphi_3 \circ \varphi_2 \circ \varphi_1)$. For the path in the middle we choose d'_i and for the rightmost path d''_i.

We define an equivalence relation $\sim$ as follows:
For all $d, d' \in D$ with $d = (t, [v_1/t_1, \ldots v_n/t_n])$: $d \sim d'$, iff there is a variable renaming ρ, such that $d' = (\hat{\rho}(t), [\rho(v_1)/\hat{\rho}(t_1), \ldots \rho(v_n)/\hat{\rho}(t_n)])$; where $\hat{\rho}$ is the homomorphical extension of $\rho : \mathcal{V} \to \mathcal{V}$ to terms $T(\Omega, \mathcal{V})$.

We now want to compute the 2–success efficiency of $\mathcal{I}$ and $\sim$ for the calculandum d. The two derivation forms $d_3 = (t, [x/t_1, y/t_2])$ and d'_3 are related by $\sim$ since there is a variable renaming $\rho = [z_1/z_7]$, such that $d'_3 = (\hat{\rho}(t), [x/\hat{\rho}(t_1), y/\hat{\rho}(t_2)])$.

It is:

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

and therefore:

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

□

Analogous to the previous subsection we now define what is meant by more k–success efficient. The conditions that are put on the two calculi are the same as in Definition 6.4.

Definition 6.8 (more k–success efficient)
Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ and $\mathcal{I}' = (D', D'_{\text{success}}, (J', <'), (\vdash'_j \mid j \in J'))$ be indexed calculi, $h : D' \to D$ a partial function, $\sim\, \subseteq D \times D$, and $\sim' \subseteq D' \times D'$ equivalence relations, and $k \geq 2$ a natural number. Then, $\mathcal{I}$ is *more k–success efficient w.r.t.* $(h, \sim, \sim')$ *than* $\mathcal{I}'$ if the following two conditions hold:

1. $\mathcal{I}$ is h–comparable with $\mathcal{I}'$.
2. For every $d' \in D'$ with $(\eta_k(\mathcal{I}', \sim'))\,(d') \geq 1$: $(\eta_k(\mathcal{I}, \sim))\,(h(d')) \leq (\eta_k(\mathcal{I}', \sim'))\,(d')$, and there is a $\bar{d}' \in D'$, such that $(\eta_k(\mathcal{I}, \sim))\,(h(\bar{d}')) < (\eta_k(\mathcal{I}', \sim'))\,(\bar{d}')$. □

6.3 Other efficiency measures

We now want to define two efficiency measures that are capable of dealing with infinite subtrees in the derivation trees. The efficiency measures defined so far assign the symbol ∞, if the deterministic calculus for an indexed calculus $\mathcal{I}$ traverses an infinite subtree before finding a success form. We need measures that will characterize the efficiency of finding success forms independently from the deterministic calculus. The *success density efficiency* and the *infinite path efficiency* are such measures.

6.3.1 Success Density Efficiency

The success density efficiency $\mathrm{d}\eta(\mathcal{I}, \sim, f)$ of $\mathcal{I}, \sim$, and f associates a function $\mathrm{d}\eta(\mathcal{I}, \sim, f)(d) : \mathbb{N} \to \mathbb{Q}$ with every calculandum d, such that, for every $n \in \mathbb{N}$, $\mathrm{d}\eta(\mathcal{I}, \sim, f)(d)(n)$ is the number of $\sim$-different success forms which can be derived from d in at most n steps. More precisely, we will give a weight to every number of derivation steps in order to formalize the following intuitively clear observation: the fewer steps an indexed calculus needs to derive a success form the better it is. Hence, typically, the weight function f will assign a greater number to levels close to the calculandum than to levels which are far from the calculandum.

Definition 6.9 (success density efficiency)
Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ be an indexed calculus, $\sim\subseteq D \times D$ an equivalence relation, $d \in D$ a calculandum, and $f : \mathbb{N} \to \mathbb{Q}$ a weight function. The *success density efficiency*

of $\mathcal{I}$, $\sim$, and f, denoted by $\mathrm{d}\eta(\mathcal{I},\sim,f)$, is a function $\mathrm{d}\eta(\mathcal{I},\sim,f) : D \to (\mathbb{N} \to \mathbb{Q})$, such that for every $d \in D$ and $n \in \mathbb{N}$:

$$(\mathrm{d}\eta(\mathcal{I},\sim,f))\,(d)(n) = \sum_{\nu=0}^{n} f(\nu) \cdot |(Suc_\nu\,(d,\vdash) \cap D_{\text{success}})/_\sim|$$

where $\vdash = \bigcup_{i\in J} \vdash_i$. □

Example 6.10 Let us again consider Figure 8 and let $\mathcal{I} = (D, D_{\text{success}}, (J,<), (\vdash_j \mid j \in J))$ be the indexed calculus which is defined informally by the figure of the derivation tree of $\mathcal{I}$ and the calculandum $d = (append\,(x, append\,(y, NIL)),\ \varphi_\emptyset)$ in Figure 8. The equivalence relation $\sim$ is defined as in Example 6.7. Let d be the derivation form $(append\,(x, append\,(y, NIL)), \varphi_\emptyset)$. As weight function we choose $f(\nu) = 1/(1+\nu)$.

We now want to compute the success density efficiency of $\mathcal{I},\sim$ and f up to level 3:

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

where $\vdash = \bigcup_{i\in J} \vdash_i$. The success density efficiency up to level 3 is computed by summing up the values in the rightmost column: $(\mathrm{d}\eta(\mathcal{I},\sim,f))(d)(3) = 7/12$. □

Analogous to the previous subsections we now define the notion of more success density efficient.

Definition 6.11 (more success density efficient)

Let $\mathcal{I} = (D, D_{\text{success}}, (J,<), (\vdash_j \mid j \in J))$ and $\mathcal{I}' = (D', D'_{\text{success}}, (J',<'), (\vdash'_j \mid j \in J'))$ be indexed calculi, $h : D' \to D$ a partial function, and $\sim \subseteq D \times D$ and $\sim' \subseteq D' \times D'$ equivalence relations. Then, $\mathcal{I}$ is *more success density efficient w.r.t. h and f than $\mathcal{I}'$* if the following two conditions hold:

1. $\mathcal{I}$ is h–comparable with $\mathcal{I}'$.
2. For every $d' \in D' \setminus D'_{\text{success}}$ and $n \in \mathbb{N}$: $(\mathrm{d}\eta(\mathcal{I},\sim,f)\,(h(d'))(n) \geq (\mathrm{d}\eta(\mathcal{I}',\sim',f)\,(d'))(n)$ and there is a $\bar{d}' \in D'$ and an $n' \in \mathbb{N}$, such that $(\mathrm{d}\eta(\mathcal{I},\sim,f)\,(h(\bar{d}'))(n') > (\mathrm{d}\eta(\mathcal{I}', \sim', f)\,(\bar{d}'))(n')$. □

6.3.2 Infinite Path Efficiency

The infinite path efficiency simply counts the number of infinite paths in a derivation tree. Let us first define the notion of an infinite path.

Definition 6.12 (infinite path)

Let $\mathcal{I} = (D, D_{\text{success}}, (J,<), (\vdash_j \mid j \in J))$ be an indexed calculus, $d \in D$ a calculandum, $i_0, i_1, i_2 \ldots \in J$ indices, and $d_0, d_1, \ldots \in D$ success forms. An *infinite path of d* is an infinite sequence $w = (i_0, i_1, i_2 \ldots)$, such that:

$$d \vdash_{i_0} d_0 \vdash_{i_1} d_1 \vdash_{i_2} \ldots$$

The set of all infinite paths of d is denoted by $IP(\mathcal{I}, d)$. □

Example 6.13 Consider an indexed calculus that models commutativity for addition for arithmetic terms over a ranked alphabet $\{+^{(2)}, a^{(0)}, b^{(0)}\}$, i.e.,

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

where $(\vdash_j \mid j \in \mathbb{N}^*)$ is a family with the following property: Let $t \in D, u \in O(t)$ and $t' = t/u$, then: if $t' = (t_1 + t_2)$ with $t_1, t_2 \in T(\{+^{(2)}, a^{(0)}, b^{(0)}\}, \emptyset)$, then $(t, t[u \leftarrow (t_2 + t_1)]) \in \vdash_u$.

The sequence $w = (\varepsilon, \varepsilon, \varepsilon, \ldots)$ is an infinite path of $d = (a + b)$, because there is no d' in the derivation $(a + b) \vdash_\varepsilon (b + a) \vdash_\varepsilon (a + b) \ldots \vdash_\varepsilon d'$, such that $\bigcup_{j \in \mathbb{N}^*} Suc(d', \vdash_j) = \emptyset$. □

Let $\mathcal{I}$ be an indexed calculus and d a calculandum. The infinite path efficiency depends on the number of infinite paths of d in the derivation tree of $\mathcal{I}$ and d.

Definition 6.14 (infinite path efficiency)

Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ be an indexed calculus. The *infinite path efficiency of* $\mathcal{I}$, denoted by $\mathrm{i}\eta(\mathcal{I})$, is a function $\mathrm{i}\eta(\mathcal{I}) : D \to \mathbb{N}$, such that for all $d \in D$:

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

In Example 6.13 there is only one derivation starting with the term $(a + b)$. Obviously the infinite path efficiency for the calculandum $(a + b)$ is

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

Definition 6.15 (more infinite path efficient)

Let $\mathcal{I} = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ and $\mathcal{I}' = (D', D'_{\text{success}}, (J', <'), (\vdash'_j \mid j \in J'))$ be indexed calculi and let $h : D' \to D$ be a partial function. Then, $\mathcal{I}$ is *more infinite path efficient w.r.t. h than $\mathcal{I}'$* if the following two conditions hold:

1. $\mathcal{I}$ is h–comparable with $\mathcal{I}'$.
2. For every $d' \in D'$: $(\mathrm{i}\eta(\mathcal{I}))\,(h(d')) \leq (\mathrm{i}\eta(\mathcal{I}'))\,(d'))$ and there is a $\bar{d'} \in D'$, such that $(\mathrm{i}\eta(\mathcal{I}))\,(h(\bar{d'})) < (\mathrm{i}\eta(\mathcal{I}'))\,(\bar{d'}))$ □

7 Comparison of Efficiencies

Now we compare

1. the narrowing calculus for $E_{\mathcal{R}}$–unification with the lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification and
2. the lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification with the ulo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification.

The class of equational theories induced by ctn term rewriting systems is the biggest class, for which all narrowing techniques which we want to compare, are complete. Thus, in our comparison, we always consider equational theories which are induced by ctn term rewriting systems. Furthermore, we will restrict the set of success forms to ground complete sets of $(E_{\mathcal{R}}, \Delta)$–unifiers. Indeed, this is only a limitation for the narrowing calculus for $E_{\mathcal{R}}$–unification which computes a superset of the set of ground complete $(E_{\mathcal{R}}, \Delta)$–unifiers (cf. Theorem 5.12).

7.1 Narrowing vs. LO–Narrowing

Let us shortly repeat the two indexed calculi that we want to compare now. We change the narrowing calculus for $E_{\mathcal{R}}$–unification to the narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification by simply restricting the set of success forms to terms in $T(\Delta, \mathcal{V})$. Then, success forms label leaves only, and not arbitrary nodes as it was possible in the narrowing calculus for $E_{\mathcal{R}}$–unification. Since the derivation trees are left unchanged, the algorithm inherent in Theorem 5.12 now computes a ground complete set of $(E_{\mathcal{R}}, \Delta)$–unifiers.

Let $\mathcal{R} = (F, \Delta, R)$ be a ctn term rewriting system, $\Omega = F \cup \Delta$, and $\mathcal{R}' = (\Omega, R)$ a term rewriting system.

- Lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification:

  lo–nar$\mathcal{I}(\mathcal{R}) = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ where

  - $D = \text{Equ}\,(T(\Omega, \mathcal{V})) \times \mathit{Sub}\,(\mathcal{V}, \Delta)$
  - $D_{\text{success}} = \{(\mathit{equ}\,(\Theta_1, \Theta_2), \varphi) \mid \Theta_1, \Theta_2 \in T(\Delta, \mathcal{V}),\ \Theta_1$ and Θ_2 are unifiable, $\varphi \in \mathit{Sub}\,(\mathcal{V}, \Delta)\}$
  - $J = R$
  - $<$ is defined as in Definition 5.20
  - for $j = (l \to r) \in J$, the relation $\vdash_j = \overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\text{equ}}, l \to r}$ (see Definition 5.17).

- Narrowing calculus for $(E_{\mathcal{R}'}, \Delta)$–unification:

  nar$\mathcal{I}(\mathcal{R}') = (D', D'_{\text{success}}, (J', <'), (\vdash'_j \mid j \in J'))$ where

  - $D' = \text{Equ}\,(T(\Omega, \mathcal{V})) \times \mathit{Sub}\,(\mathcal{V}, \Omega)$
  - $D'_{\text{success}} = \{(\mathit{equ}\,(\Theta_1, \Theta_2), \varphi) \mid \Theta_1, \Theta_2 \in T(\Delta, \mathcal{V}),\ \Theta_1$ and Θ_2 are unifiable, $\varphi \in \mathit{Sub}\,(\mathcal{V}, \Delta)\}$
  - $J' = \mathbb{N}^* \times R$
  - $<'$ is defined as in Definition 5.11

– for $j = (u, l \to r) \in J'$, the relation $\vdash'_j = \rightsquigarrow_{\mathcal{R}'_{\text{equ}}, u, l \to r}$ (see Definition 5.10)

In the rest of this section we will adopt the names of sets as stated above.

Since lo–narrowing is only a limitation of narrowing, namely the limitation to one specific narrowing occurrence, the calculi are very similar.

Lemma 7.1 lo–nar$\mathcal{I}(\mathcal{R})$ is h–comparable with nar$\mathcal{I}(\mathcal{R}')$, where h is the identity.

Proof: The first condition of h–comparability is satisfied, because $D_{\text{success}} = D'_{\text{success}}$. The second condition is satisfied, because every lo–narrowing step is also a narrowing step, i.e., for every $d, \bar{d}$ with $d \overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\text{equ}}} \bar{d}$: $d \rightsquigarrow_{\mathcal{R}'_{\text{equ}}} \bar{d}$. □

7.1.1 Single Success Efficiency

Consider the derivation trees for nar$\mathcal{I}(\mathcal{R}')$ and lo–nar$\mathcal{I}(\mathcal{R})$ and a calculandum d. The narrowing tree is either

- identical to the lo–narrowing tree or
- it contains *more* branches than the lo–narrowing tree.

Intuitively it is clear that the single success efficiency of lo–nar$\mathcal{I}(\mathcal{R})$, i.e., the number of derivation steps of the deterministic calculus for lo–nar$\mathcal{I}(\mathcal{R})$, needed to find the first success form, is *less than or equal to* the single success efficiency of nar$\mathcal{I}(\mathcal{R}')$. Let us give an example for the fact that there is a d with $(\eta(\text{lo–nar}\mathcal{I}(\mathcal{R})))(d) < (\eta(\text{nar}\mathcal{I}(\mathcal{R}')))(d)$.

Example 7.2 Let $\mathcal{R}$ be a ctn term rewriting system with:

- $F = \{f^{(1)}, g^{(1)}\}$
- $\Delta = \{A^{(0)}, B^{(0)}, G^{(2)}\}$
- R contains the following rules:

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

We will not use rule (3), but since f has to be totally defined, it has to be part of R. The narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification is defined for term rewriting systems; therefore we define $\mathcal{R}' = (F \cup \Delta, R)$. Figure 9 depicts the derivation trees for nar$\mathcal{I}(\mathcal{R}')$ and lo–nar$\mathcal{I}(\mathcal{R})$ for the calculandum $d = (equ(f(x), B))$.

The dotted area is not part of the lo–narrowing tree but only part of the narrowing tree for d. Obviously it is:

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

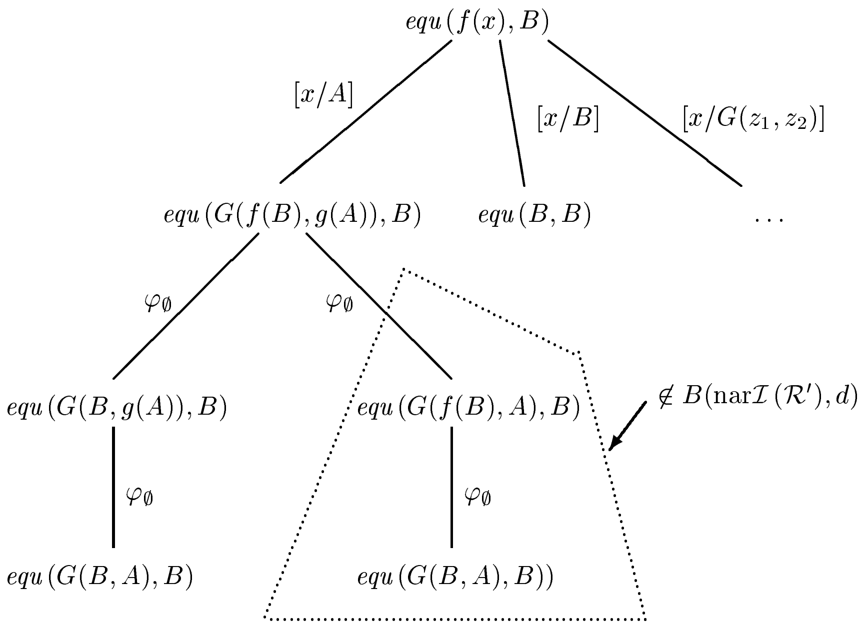

Figure 9: Narrowing– and lo–narrowing tree for the calculandum $equ\,(f(x),B)$

Therefore the lo–narrowing calculus for $(E_{\mathcal{R}},\Delta)$–unification has a lower single success efficiency than the narrowing calculus for $(E_{\mathcal{R}},\Delta)$–unification for this calculandum. □

Although it seems obvious that, also in general, lo–narrowing is more single success efficient than narrowing, one might suspect that the arrangement shown in Figure 10 is possible. There, success form 1 can not be derived by the lo–narrowing calculus for $(E_{\mathcal{R}},\Delta)$–unification but only by the narrowing calculus for $(E_{\mathcal{R}},\Delta)$–unification. The success forms 2 and 1' are derivable by both calculi, and success form 1' is equal to success form 1 modulo variable renaming. Then obviously, the single success efficiency of lo–narrowing would be larger than the one for narrowing, i.e., $(\eta(\text{lo–nar}\mathcal{I}\,(\mathcal{R}))\,(d) > (\eta(\text{nar}\mathcal{I}\,(\mathcal{R}')))\,(d)$. The following lemma states that such an arrangement can not exist.

Let us first define the set *left* (x) which contains all nodes of a derivation tree that are left of the node x (not including the path to x).

Definition 7.3 (*left*)

Let $\mathcal{I} = (D, D_{\text{success}}, (J,<), (\vdash_j \mid j \in J))$ be an indexed calculus, $d \in D$ a calculandum, and $x \in \underline{node}\,(B(\mathcal{I},d))$ a node. The node x is represented by a sequence of indices in J that describes the path from $\underline{root}\,(B(\mathcal{I},d))$ to x; i.e., $x = (j_1 \dots j_n)$ where $j_i \in J$ for $1 \leq i \leq n$. The sets *prefix* and *left* are defined as follows:

- $\textit{prefix}\,(x,\mathcal{I},d) = \{w \mid ww' = x \text{ and } w,w' \in J^*\}$
- $\textit{left}\,(x,\mathcal{I},d) = \{x' \mid x' \in \underline{node}\,(B(\mathcal{I},d)) \text{ with } x' <_{\text{lex}} x\} \setminus \textit{prefix}\,(x,\mathcal{I},d)$. □

The following lemma describes that if there is a node x in the narrowing tree which is labeled by a success form d_1 and which does not occur in the lo–narrowing tree, then there is a node

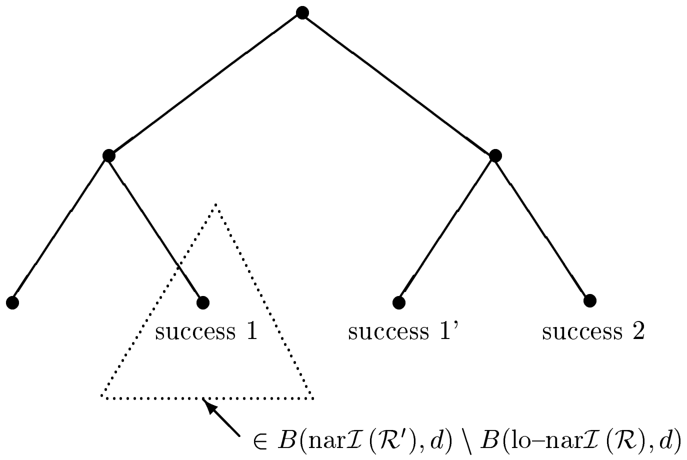

Figure 10: Narrowing- and lo–narrowing tree with $(\eta(\text{lo–nar}\mathcal{I}(\mathcal{R})))(d) > (\eta(\text{nar}\mathcal{I}(\mathcal{R}')))(d)$.

x' left of x which is part of the lo–narrowing tree and which is labeled by a success form d_2 which is equivalent to d_1; in this context, equivalence means variable renaming.

Lemma 7.4 Let $\mathcal{R} = (F, \Delta, R)$ be a ctn term rewriting system, $\mathcal{R}' = (F \cup \Delta, R)$ a term rewriting system, and d a calculandum. If $x \in \underline{\mathit{node}}(B(\text{nar}\mathcal{I}(\mathcal{R}'), d)) \setminus \underline{\mathit{node}}(B(\text{lo–nar}\mathcal{I}(\mathcal{R}), d))$ and $\underline{\mathit{label}}_{B(\text{nar}\mathcal{I}(\mathcal{R}'), d)}(x) \in D_{\text{success}}$, then there is an $x' \in \mathit{left}(x, \text{nar}\mathcal{I}(\mathcal{R}'), d) \cap \underline{\mathit{node}}(B(\text{lo–nar}\mathcal{I}(\mathcal{R}), d))$ where $\underline{\mathit{label}}_{B(\text{lo–nar}\mathcal{I}(\mathcal{R}), d)}(x')$ is obtained from $\underline{\mathit{label}}_{B(\text{nar}\mathcal{I}(\mathcal{R}'), d)}(x)$ by variable renaming.

Proof:

Let d be a calculandum, $x \in \underline{\mathit{node}}(B(\text{nar}\mathcal{I}(\mathcal{R}'), d)) \setminus \underline{\mathit{node}}(B(\text{lo–nar}\mathcal{I}(\mathcal{R}), d))$ a node with $\underline{\mathit{label}}_{B(\text{nar}\mathcal{I}(\mathcal{R}'), d)}(x) \in D_{\text{success}}$. Since we have restricted the set of success forms D_{success} to $(E_{\mathcal{R}}, \Delta)$–unifiers, the node x is a leaf of the narrowing tree. Since x is not derivable by leftmost–outermost narrowing, there must be a node in $\mathit{prefix}(x, \text{nar}\mathcal{I}(\mathcal{R}'), d)$ with at least two narrowing occurrences. We choose the lowest such node y which is still part of the lo–narrowing tree, that is

$$y = \max_{\text{lex}} \{\hat{x} \mid \hat{x} \in \mathit{prefix}(x, \text{nar}\mathcal{I}(\mathcal{R}'), d) \cap \underline{\mathit{node}}(B(\text{lo–nar}\mathcal{I}(\mathcal{R}), d)) \text{ and } \underline{\mathit{label}}_{B(\text{lo–nar}\mathcal{I}(\mathcal{R}), d)}(\hat{x}) \text{ contains two narrowing occurrences}\}.$$

Let the set F of function symbols of the ctn term rewriting system $\mathcal{R}$ contain the symbols f and g and assume that node y is labeled by (t, φ). There are two possibilities how the two narrowing occurrences in t can be related to each other:

1. $t = \ldots f(\ldots) \ldots g(\ldots) \ldots$; i.e., g is *not* an argument of f.
2. $t = \ldots f(\ldots g(\ldots) \ldots) \ldots$; i.e., g is an argument of f.

We can assume that the symbols in "..." are elements of $\Delta \cup \mathcal{V}$. This assumption is justified, because function symbols (which are always narrowing occurrences) can only be nested in exactly those possibilities mentioned above, and therefore the same explanation will apply.

Case 1: $t = \dots f(\dots) \dots g(\dots) \dots$

Let $u_f, u_g \in \mathbb{N}^*$ with $t/u_f = f(\dots)$ and $t/u_g = g(\dots)$. Then u_f is a lo–narrowing occurrence and u_g is a narrowing occurrence. It has to be shown, that, for every success form $\hat{d}$ which labels a leaf in $B(\text{nar}\mathcal{I}(\mathcal{R}'), d) \setminus B(\text{lo–nar}\mathcal{I}(\mathcal{R}), d)$ there is a lo–narrowing derivation yielding a success form labeling a node left of the path from y to x. Moreover, this success form is equal to $\hat{d}$ modulo variable renaming. Let the node y be the lowest node on the path from the root to x with at least two narrowing occurrences that is still part of the lo–narrowing tree. This means, that by narrowing at occurrence u_g and further narrowing, we get from y to x. We distinguish between the following two kinds of rules:

(a) rules with right–hand sides containing more that one function symbol, i.e., which increase the number of function symbols in t.

(b) rules with right–hand sides each containing at most one function symbol.

We only have to consider rules of the form (b), because the node y could not be the *lowest* node labeled by a derivation form with at least two narrowing occurrences, if rules of the form (a) are applied below y.

The narrowing derivation starting with t looks like this (substitutions are left out for clarity reasons):

$$t = \dots f(\dots) \dots g(\dots) \dots \rightsquigarrow_{\mathcal{R}'_{\text{equ}}, \delta_1} \cdots \rightsquigarrow_{\mathcal{R}'_{\text{equ}}, \delta_n} \dots t_f \dots t_g \dots = \bar{t}$$

where $t_f, t_g \in T(\Delta, \mathcal{V})$ and for $1 \le i \le n : \delta_i = (u_i, \varphi_i, \xi_i)$ with $u_i \in \mathbb{N}^*$, $\varphi_i \in Sub(\mathcal{V}, \Omega)$, $\xi_i \in R$ and the rules ξ_i are of form (b).

The sequence $\delta_1 \dots \delta_n$ of narrowing triples is partitioned as follows

$$\underbrace{\delta_1, \dots, \delta_{n-1}}_{\text{occ.: } u_g w}, \underbrace{\delta_n}_{\text{occ.: } u_f}$$

More precisely, for every step i of the first $n-1$ narrowing steps there is a $w_i \in \mathbb{N}^*$, such that narrowing is applied at an occurrence of the form $u_g w_i$ (such occurrences will simply be denoted by $u_g w$ in the following).

There are two reasons, why the sequence of narrowing triples has precisely the above form:

(1) The narrowing triple which corresponds to narrowing at occurrence u_f, i.e., δ_n can only occur as *last* step of the derivation. Assume this triple would be in between narrowing steps at occurrence $u_g w$. Then, the sequence of narrowing triples would be:

$$\underbrace{\delta_1, \dots, \delta_{k-1}}_{u_g w}, \underbrace{\delta_k}_{u_f}, \underbrace{\delta_{k+1}, \dots, \delta_n}_{u_g w}.$$

But then, the node labeled by the derivation form which is derived by the narrowing triple δ_{k-1} would be the lowest node labeled by a derivation form which contains at least two narrowing occurrences, and therefore this node would have to be the node y.

(2) Every rule which can be applied at occurrence u_f does not have function symbols in its right–hand side. If there were two or more steps applicable to u_f and sub–occurrences, then the node labeled by the success form which is derived by the narrowing triple δ_{n-1} would be the lowest node y containing at least two narrowing occurrences, and therefore there can only be one narrowing step at occurrence u_f.

Obviously, there is a branch in $B(\text{lo–nar}\mathcal{I}(\mathcal{R}'), d)$ which starts from y and the first step of which corresponds to the triple δ_n. Moreover, this branch occurs left of the branch from y to x. By (2), f disappears and no other function appears after application of δ_n. Thus the occurrences involved in the triples $\delta_1, \ldots, \delta_{n-1}$ are now leftmost outermost and hence, $B(\text{lo–nar}\mathcal{I}(\mathcal{R}'), d)$ must contain a branch which performs the steps $\delta_n, \delta_1, \ldots, \delta_{n-1}$. Thus the lo–narrowing derivation will look as follows.

$$t = \ldots f(\ldots) \ldots g(\ldots) \ldots \overset{lo}{\leadsto}_{\mathcal{R}_{\text{eqy}}, \varphi_n, \xi_n} \ldots t'_f \ldots g(\ldots) \ldots \overset{lo}{\leadsto}_{\mathcal{R}_{\text{equ}}, \varphi_1, \xi_1} \cdots \overset{lo}{\leadsto}_{\mathcal{R}_{\text{equ}}, \varphi_{n-1}, \xi_{n-1}} \ldots t'_f \ldots t'_g \ldots = \bar{t}'$$

Moreover the terms t'_f and t'_g are equal to t_f and t_g, respectively, modulo variable renaming. Thus, the term $\bar{t}'$ is equal to $\bar{t}$ modulo variable renaming. q.e.d.

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

Let $u_f, u_g \in \mathbb{N}^*$ with $t/u_f = f(\ldots)$ and $t/u_g = g(\ldots)$. Then u_f is a lo–narrowing occurrence and u_g is a narrowing occurrence. Let $g(\ldots)$ be contained in the m-th argument position of f.

The narrowing derivation starting at the label of y and ending at the label of x is of the following form:

$$t = \ldots f(\ldots g(\ldots) \ldots) \ldots \leadsto_{\mathcal{R}'_{\text{equ}}, \delta_1} \cdots \leadsto_{\mathcal{R}'_{\text{equ}}, \delta_k} \ldots f(\ldots t_g \ldots) \ldots \cdots \leadsto_{\mathcal{R}'_{\text{equ}}, \delta_n} \ldots t_f \ldots = \bar{t}$$

where $t_g, t_f \in T(\Delta, \mathcal{V})$, $1 \leq k \leq n$ and for $1 \leq i \leq n : \delta_i = (u_i, \varphi_i, \xi_i)$ with $u_i \in \mathbb{N}^*$, $\varphi_i \in Sub(\mathcal{V}, \Omega)$, $\xi_i \in R$ and the rules ξ_i are of the form (b).

Since $\mathcal{R}$ is a ctn term rewriting system, it holds that exactly one function symbol occurs in the left–hand side of every rule. Moreover, this function symbol occurs at the root of the left–hand side. Thus, in the application of rule ξ_{k+1} at occurrence u_f, the subterm $g(\ldots)$ has to be matched with some variable of the left–hand side of ξ_{k+1}. Let us denote this variable by z.

Then, we have to consider two cases:

(i) The variable z occurs in the right hand side of the rule ξ_{k+1}.

For the same reasons as in Case 1 the narrowing triples can be partitioned as follows:

$$\underbrace{\delta_1, \ldots, \delta_{n-1}}_{\text{occ.: } u_g w}, \underbrace{\delta_n}_{\text{occ.: } u_f}$$

Then, as in Case 1, it is obvious that $B(\text{lo–nar}\mathcal{I}(\mathcal{R}), d)$ contains a branch which starts from y, whose first step corresponds to δ_n, and which is left of the branch from y to x. Again f disappears and no other function symbol is introduced. Hence, the occurrences $u_1, \ldots, u_{n-1}$ are now leftmost outermost and thus the branch continues by applying the rules $\xi_1, \ldots, \xi_{n-1}$. Then, there is a lo–narrowing derivation of the form:

$$t = \ldots f(\ldots g(\ldots) \ldots) \ldots \overset{lo}{\leadsto}_{\mathcal{R}_{\text{equ}}, \varphi_n, \xi_n} t' \overset{lo}{\leadsto}_{\mathcal{R}_{\text{equ}}, \varphi_1, \xi_1} \cdots \overset{lo}{\leadsto}_{\mathcal{R}_{\text{equ}}, \varphi_{n-1}, \xi_{n-1}} \ldots t_f \ldots = \bar{t}'$$

where $t' \in T(\Omega, \mathcal{V})$.

(ii) The variable z does *not* occur in the right hand side of the rule ξ_{k+1}.

Now there might be more than one narrowing step being applied at occurrences of the form $u_f w$ with $w \in \mathbb{N}^*$. The reason for this is that the narrowing occurrence u_g will not exist after applying the rule ξ_{k+1}, because z is not contained in the right hand side. Still, the application of rules at $u_g w$ and $u_f w$ can not be interleaved, because then the node y would be at a lower position. Thus, the sequence of narrowing triples looks like this:

$$\underbrace{\delta_1, \ldots, \delta_k}_{\text{occ.: } u_g w}, \underbrace{\delta_{k+1}, \cdots, \delta_n}_{\text{occ.: } u_f w}$$

Again it is clear that $B(\text{lo–nar}\mathcal{I}(\mathcal{R}), d)$ contains a branch which starts at y by the application of the rules $\xi_{k+1}, \ldots, \xi_n$ and which is left of the branch from y to x. Then the branch has reached a leaf already. Note that the lo–narrowing steps which correspond to the narrowing triples δ_i with $1 \leq i \leq k-1$ do not have to be applied at all, because the result is still at argument position m and will be deleted by the lo–narrowing step corresponding to the narrowing triple δ_{k+1}. Therefore the lo–narrowing derivation consists only of the steps applied at occurrences $u_f w$, namely:

$$t = \ldots f(\ldots g(\ldots)\ldots)\ldots \overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\text{equ}}, \varphi_{k+1}, \xi_{k+1}} \cdots \overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\text{equ}}, \varphi_n, \xi_n} \ldots t_f \ldots = \bar{t}''$$

Obviously $\bar{t}$ and $\bar{t}'$ and $\bar{t}''$ are equal to each other modulo variable renaming. q.e.d. □

Based on this lemma we can now formulate a theorem which compares the single success efficiency of the lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification and the narrowing calculus for $(E_{\mathcal{R}'}, \Delta)$–unification.

Theorem 7.5 Leftmost outermost narrowing is more single success efficient than narrowing, i.e.,

1. There is a ctn term rewriting system $\mathcal{R} = (F, \Delta, R)$ and a term rewriting system $\mathcal{R}' = (F \cup \Delta, R)$, such that

   lo–nar$\mathcal{I}(\mathcal{R})$ is more single success efficient w.r.t. the identity than nar$\mathcal{I}(\mathcal{R}')$

   and

2. there is no ctn term rewriting system $\mathcal{R} = (F, \Delta, R)$ and term rewriting system $\mathcal{R}' = (F \cup \Delta, R)$ and calculandum $d \in D' \setminus D'_{\text{success}}$, such that

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

Proof of Theorem 7.5

To prove Statement 1 of the theorem, we must show that there is a ctn term rewriting system $\mathcal{R} = (F, \Delta, R)$ and a term rewriting system $\mathcal{R}' = (F \cup \Delta, R)$, such that lo–nar$\mathcal{I}(\mathcal{R})$ is more single success efficient w.r.t. the identity than nar$\mathcal{I}(\mathcal{R}')$. In particular this means that we have to prove the following statements (i) and (ii):

(i) There is an h, such that lo–nar$\mathcal{I}(\mathcal{R})$ is h–comparable with nar$\mathcal{I}(\mathcal{R}')$.

(ii) Let D and D' be the sets of derivation forms of lo–nar$\mathcal{I}\,(\mathcal{R})$ and nar$\mathcal{I}\,(\mathcal{R}')$, respectively.

(a) For every $d' \in D'$ with $(\eta(\text{nar}\mathcal{I}\,(\mathcal{R}')))\,(d') \geq 1$: $(\eta(\text{lo–nar}\mathcal{I}\,(\mathcal{R}))\,(h(d')) \leq (\eta(\text{nar}\mathcal{I}\,(\mathcal{R}')))\,(d')$ and

(b) there is a $\bar{d}' \in D'$, such that $(\eta(\text{lo–nar}\mathcal{I}\,(\mathcal{R}))\,(h(\bar{d}')) < (\eta(\text{nar}\mathcal{I}\,(\mathcal{R}'))\,(\bar{d}')$.

Let $\mathcal{R} = (F, \Delta, R)$ be the ctn term rewriting system from Example 7.2 and let $\mathcal{R}' = (F \cup \Delta, R)$ be a term rewriting system.

Proof of (i): By Lemma 7.1 it holds that lo–nar$\mathcal{I}\,(\mathcal{R})$ is h–comparable with nar$\mathcal{I}\,(\mathcal{R}')$ where h is the identity.

Proof of (ii): For the calculandum $d = (equ\,(f(x), B))$, the following holds: $(\eta(\text{lo–nar}\mathcal{I}\,(\mathcal{R}))\,(d)= 7 < (\eta(\text{nar}\mathcal{I}\,(\mathcal{R}'))\,(d) = 11$. Thus, we have shown part (b) of (ii). Part (a) will be shown later.

We now prove Statement 2 of the lemma by contradiction.

Let us start with the assumption that there is a ctn term rewriting system $\mathcal{R} = (F, \Delta, R)$, a term rewriting system $\mathcal{R}' = (F \cup \Delta, R)$, and a calculandum $d \in \text{Equ}\,(T(\Omega, \mathcal{V})) \times \mathit{Sub}\,(\mathcal{V}, \Omega) \setminus D'_{\text{success}}$, such that

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

Now we derive a contradiction to this assumption. The above means that the deterministic narrowing calculus finds the first success form in less steps than the deterministic lo–narrowing calculus. Therefore there is a node $x \in \underline{node}\,(B(\text{nar}\mathcal{I}\,(\mathcal{R}'), d)) \setminus \underline{node}\,(B(\text{lo–nar}\mathcal{I}\,(\mathcal{R})), d)$, such that $\alpha = \underline{label}_{\,B(\text{nar}\mathcal{I}\,(\mathcal{R}'),d)}\,(x)$ is a success form for the narrowing calculus. Moreover there is *no* success form α' that is equal to α modulo to variable renaming and that labels a node of the lo–narrowing calculus left of x. However Lemma 7.4 states that such a case does *not* exist since for every node y that is only part of the narrowing calculus and which is labeled by a success form there is a node left of y in the lo–narrowing tree, which is labeled by an equivalent success form. This is a contradiction to our assumption and hence Statement 2 of the theorem is proved. Part (a) of (ii) of Statement 1 is an immediate consequence of Statement 2. □

7.1.2 Multi Success Efficiency

In this subsection the results for the single success efficiency are expanded to the multi success case. This means that not only finding the first success form is relevant, but rather finding the first k success forms (that are $\sim$–different). The equivalence relation $\sim$ will represent variable renaming.

Theorem 7.6 Leftmost outermost narrowing is more multi success efficient than narrowing, i.e.,

1. There is a ctn term rewriting system $\mathcal{R} = (F, \Delta, R)$ and a term rewriting system $\mathcal{R}' = (F \cup \Delta, R)$, such that there is a $k \geq 2$ with

$$\text{lo–nar}\mathcal{I}\,(\mathcal{R}) \text{ is more } k\text{–success efficient w.r.t. to } (id, \sim, \sim) \text{ than } \text{nar}\mathcal{I}\,(\mathcal{R}')$$

   where id is the identity, and

2. there is no ctn term rewriting system $\mathcal{R} = (F, \Delta, R)$, and term rewriting system $\mathcal{R}' = (F \cup \Delta, R)$ and calculandum $d \in D'$, such that there is a $k \geq 2$ with

$$(\eta_k(\text{nar}\mathcal{I}\,(\mathcal{R}'), \sim))\,(d) < (\eta_k(\text{lo–nar}\mathcal{I}\,(\mathcal{R}), \sim))\,(d)$$

where the equivalence relation $\sim$ is defined as follows:
for all $d, d' \in D$ with $d = (t, [v_1/t_1, \ldots v_n/t_n])$: $d \sim d'$, iff there is a variable renaming ρ, such that $d' = (\hat{\rho}(t), [\rho(v_1)/\hat{\rho}(t_1), \ldots \rho(v_n)/\hat{\rho}(t_n)])$, where $\hat{\rho}$ is the homomorphical extension of $\rho : \mathcal{V} \to \mathcal{V}$ to terms Equ$(T(\Omega, \mathcal{V}))$.

<u>Proof of Theorem 7.6</u>

To prove Statement 1 of the theorem, we must show that there is a ctn term rewriting system $\mathcal{R} = (F, \Delta, R)$ and a term rewriting system $\mathcal{R} = (F \cup \Delta, R)$, such that lo–nar$\mathcal{I}\,(\mathcal{R})$ is more k–success efficient w.r.t. $(id, \sim, \sim)$ than nar$\mathcal{I}\,(\mathcal{R}')$, with $k \geq 2$. In particular this means that

(i) There is an h, such that lo–nar$\mathcal{I}\,(\mathcal{R})$ is h–comparable with nar$\mathcal{I}\,(\mathcal{R}')$

(ii) Let D' be the set of derivation forms of lo–nar$\mathcal{I}\,(\mathcal{R})$ and D the set of derivation forms of nar$\mathcal{I}\,(\mathcal{R}')$.

   (a) For every $d' \in D'$ with $(\eta_k(\text{nar}\mathcal{I}\,(\mathcal{R}'), \sim'))\,(d') \geq 1$: $(\eta_k(\text{lo–nar}\mathcal{I}\,(\mathcal{R}), \sim))\,(h(d')) \leq (\eta_k(\text{nar}\mathcal{I}\,(\mathcal{R}'), \sim'))\,(d')$ and

   (b) there is a $\bar{d'} \in D'$, such that $(\eta_k(\text{lo–nar}\mathcal{I}\,(\mathcal{R}), \sim))\,(h(\bar{d'})) < (\eta_k(\text{nar}\mathcal{I}\,(\mathcal{R}'), \sim'))\,(\bar{d'})$.

Let us consider the ctn term rewriting system $\mathcal{R}$ with the set $F = \{f^{(1)}, g^{(1)}\}$ of function symbols, the set $\Delta = \{A^{(0)}, B^{(0)}, C^{(0)}, G^{(2)}\}$ of constructor symbols, and the following set of rules:

$$
\begin{array}{rcll}
f(A) & \rightarrow & G(f(B), g(A)) & (1) \\
f(B) & \rightarrow & B & (2) \\
f(C) & \rightarrow & B & (3) \\
g(x) & \rightarrow & A & (4) \\
f(G(x,y)) & \rightarrow & G(f(x), f(y)) & (5)
\end{array}
$$

and the term rewriting system $\mathcal{R}' = (F \cup \Delta, R)$.

Proof of (i): By Lemma 7.1 it holds that lo–nar$\mathcal{I}\,(\mathcal{R})$ is h–comparable with nar$\mathcal{I}\,(\mathcal{R}')$ where h is the identity.

Proof of (ii): Consider the 2–success efficiency for the calculandum $d = equ\,(f(x), B)$, then the following holds:

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

Thus, we have shown part (b) of (ii). Part (a) of (ii) will be proved later.

Now we prove Statement 2; this is again done by contradiction. Assume that there is a ctn term rewriting system $\mathcal{R} = (F, \Delta, R)$ and a term rewriting system $\mathcal{R}' = (F \cup \Delta, R)$ and a calculandum $d \in \text{Equ}\,(T(\Omega, \mathcal{V})) \times\ Sub\,(\mathcal{V}, \Omega)$, such that for a $k \geq 2$:

$$
(\eta_k(\text{nar}\mathcal{I}\,(\mathcal{R}), \sim))\,(d) < (\eta_k(\text{lo–nar}\mathcal{I}\,(\mathcal{R}), \sim))\,(d)
$$

This means that the k–th $\sim$–different success form is found earlier by the deterministic narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification than by the deterministic lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification. Therefore, there is a node $x \in \underline{node}\,(B(\text{nar}\mathcal{I}(\mathcal{R}'), d)) \setminus \underline{node}\,(B(\text{lo–nar}\mathcal{I}(\mathcal{R}), d))$ which is labeled by a success form Θ. But Lemma 7.4 states that then there is a node y in the lo–narrowing tree, which is equal to Θ modulo variable renaming and which is left of y. Since y is left of x, this success form is found earlier by the lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification than the success form labeling x. This is a contradiction to our assumption and thus Statement 2 is proved. Again, Part (a) of (ii) follows immediately from Statement 2. □

7.2 LO–Narrowing vs. ULO–Narrowing

We now want to compare the lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification with the ulo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification with respect to single success efficiency and multi success efficiency. The ulo–narrowing calculus differs from the lo–narrowing calculus in two points. First, it interleaves narrowing steps with unification steps, such that the final substitution is an $(E_{\mathcal{R}}, \Delta)$–unifier already; and second, the constructor consistency check stops derivations with terms having different constructor symbols at their roots. In contrast to the first difference, using the lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification, one has to unify the terms t and s of a success form $(equ\,(t, s), \varphi)$. Then, the composition of φ and the mgu ψ of t and s is an $(E_{\mathcal{R}}, \Delta)$–unifier. Thus, when comparing the single success efficiency of these two calculi, it would not be fair to count the ulo–narrowing steps that are responsible for unification. Therefore we will define a calculus very similar to the ulo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification called ulo$_{tu}$–calculus for $(E_{\mathcal{R}}, \Delta)$–unification. It combines a lo–narrowing step with the ulo–narrowing steps that are responsible for term–unification and the constructor consistency check, into one ulo$_{tu}$-narrowing step (the subscript tu stands for term–unification).

Definition 7.7 (ulo$_{tu}$–narrowing relation)
Let $\mathcal{R} = (F, \Delta, R)$ be a ctn term rewriting system. The *ulo$_{tu}$–narrowing relation associated with $\hat{\mathcal{R}}$*, denoted by $\overset{u_{tu}}{\leadsto}$, is defined as: for all $t, t' \in T(\hat{F} \cup \Delta, \mathcal{V})$ and $\psi, \varphi, \psi' \in Sub(\mathcal{V}, \Delta)$: $(t, \psi) \overset{u_{tu}}{\leadsto}_{\hat{\mathcal{R}}} (s, \psi')$, if $t/impO(t) = equ(t_1, t_2)$, where $t_1, t_2 \in T(F \cup \Delta, \mathcal{V})$ and either $t_1[\varepsilon] \in F$ or ($t_1[\varepsilon] \notin F$ and $t_2[\varepsilon] \in F$) and the following holds:

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

and there is no $d' \in T(\hat{F} \cup \Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Delta)$ with $(s, \psi') \overset{u}{\leadsto}_{\mathcal{R}(\Delta)} d'$, i.e., (s, ψ') is in normal form with respect to $\overset{u}{\leadsto}_{\mathcal{R}(\Delta)}$. □

We write $\overset{u_{tu}}{\leadsto}_{\hat{\mathcal{R}}, l \to r}$ if the rule $(l \to r)$ of the $\overset{lo}{\leadsto}_{\mathcal{R}_{\text{equ}}}$–relation is applied.
Now let us show the two calculi involved in our present comparison.

- ulo$_{tu}$–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification
  ulo$_{tu}$–nar$\mathcal{I}(\mathcal{R}) = (D, D_{\text{success}}, (J, <), (\vdash_j \mid j \in J))$ where
  - $D = T(\hat{F} \cup \Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Delta)$
  - $D_{\text{success}} = T(\Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Delta)$
  - $J = R$
  - $<$ ist defined as in Definition 5.26
  - for $j \in J$, $\vdash_j = \overset{u_{tu}}{\leadsto}_{\hat{\mathcal{R}}, j}$
- lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification
  lo–nar$\mathcal{I}(\mathcal{R}) = (D', D'_{\text{success}}, (J', <'), (\vdash'_j \mid j \in J'))$ where
  - $D' = \text{Equ}(T(\Delta, \mathcal{V})) \times Sub(\mathcal{V}, \Omega)$
  - $D'_{\text{success}} = \{(equ(\Theta_1, \Theta_2), \varphi) \mid \Theta_1, \Theta_2 \in T(\Delta, \mathcal{V}),\ \Theta_1 \text{ and } \Theta_2 \text{ are unifiable}, \varphi \in Sub(\mathcal{V}, \Delta)\}$
  - $J' = R$
  - $<'$ is defined as in Definition 5.20
  - for $j = (l \to r) \in J'$, $\vdash'_j = \overset{lo}{\leadsto}_{\mathcal{R}_{\text{equ}}, l \to r}$ (see Definition 5.17).

7.2.1 Single Success Efficiency

In the following we will show that the ulo$_{tu}$–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification is more single success efficient than the lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification.
We need a partial function h of the form $h : \text{Equ}(T(\Delta, \mathcal{V})) \times Sub(\mathcal{V}, \Omega) \to T(\hat{F} \cup \Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Delta)$, such that

- $\hat{h}(D'_{\text{success}}) \subseteq D_{\text{success}}$ where D_{success} and D'_{success} are defined as stated above.
- for every $d, \bar{d} \in D$ and $j \in J$ with $d \vdash_j \bar{d}$ there are $d', \bar{d}' \in D'$, such that $h(d') = d$, $h(\bar{d}') = \bar{d}$, and $d' \vdash'^{+} \bar{d}'$; where $\vdash' = \bigcup_{i \in J'} \vdash'_i$.

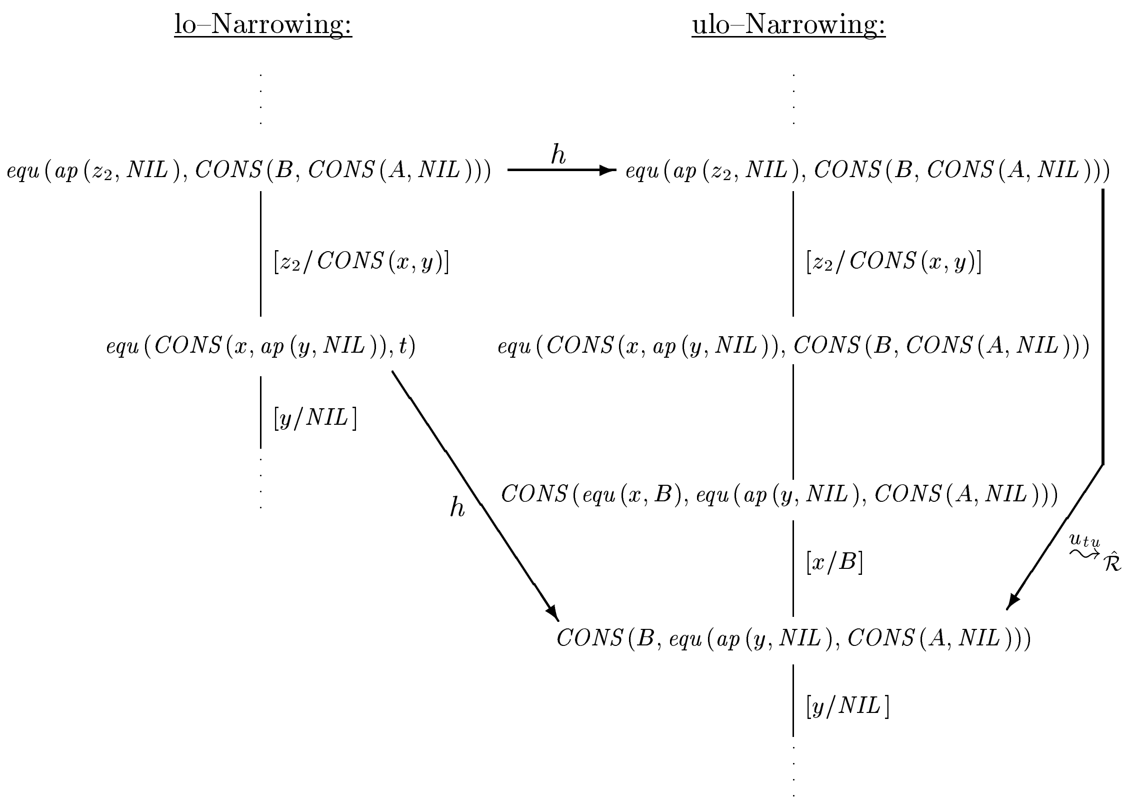

Figure 11: Branches from lo– and ulo–narrowing trees

How does such a function h look like? Consider two corresponding branches of the derivation trees for the two calculi involved as, e.g., in Figure 11. There, the relation between lo-, ulo- and ulo$_{tu}$–narrowing and the function h for an example of the ap–function as defined in Example 5.22 are indicated. Let us now define this function h.

Definition 7.8 ($h_{\text{lo–ulo}_{tu}}$)
The function $h_{\text{lo–ulo}_{tu}} : D' \to D$ is defined as follows: for every $d' \in D'$: $h_{\text{lo–ulo}_{tu}}(d) = \Theta$ where $d \overset{u}{\rightsquigarrow}{}^{*}_{\mathcal{R}(\Delta)} \Theta$ and Θ is in normal form with respect to $\overset{u}{\rightsquigarrow}_{\mathcal{R}(\Delta)}$. $\square$

Lemma 7.9 ulo$_{tu}$–nar$\mathcal{I}\,(\mathcal{R})$ is $h_{\text{lo–ulo}_{tu}}$–comparable with lo–nar$\mathcal{I}\,(\mathcal{R})$.

Proof:
To prove that the first condition of Definition 6.3 holds, we must show that $\hat{h}_{\text{lo–ulo}_{tu}}(D'_{\text{success}}) \subseteq D_{\text{success}}$.
Success forms for the lo–narrowing calculus for $(E_{\mathcal{R}}, \Delta)$–unification are of the form $d = (equ\,(\Theta, \Theta'), \varphi)$ where Θ and Θ' are unifiable by mgu ψ. By Lemma 4.5 of [FV94] two terms Θ, Θ' are unifiable, iff there exists a derivation by $\overset{u}{\rightsquigarrow}_{\mathcal{R}(\Delta)}$ of the form

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

and $(t, \psi) \in T(\Delta, \mathcal{V}) \times Sub(\mathcal{V}, \Delta)$. Thus, $h_{\text{lo–ulo}_{tu}}(d) \in D_{\text{success}}$ for every $d \in D'_{\text{success}}$ and therefore $\hat{h}_{\text{lo–ulo}_{tu}}(D'_{\text{success}}) \subseteq D_{\text{success}}$, and thereby the first condition of Definition 6.3 is proved.

To prove that the second condition of Definition 6.3 holds, we must show now that, for every $d, \bar{d} \in D$ and $j \in J$ with $d \overset{u_{tu}}{\rightsquigarrow}_{\mathcal{R},j} d'$, there are $d', \bar{d}' \in D'$, such that $h_{\text{lo–ulo}_{tu}}(d') = d$, $h_{\text{lo–ulo}_{tu}}(\bar{d}') = \bar{d}$ and $d' \vdash'^{+} \bar{d}'$ where $\vdash' = \bigcup_{i \in J'} \vdash'_i$.

Let $d, \bar{d} \in D$, $j \in J$ and $d \overset{u_{tu}}{\rightsquigarrow}_{\mathcal{R},j} \bar{d}$. We define a $d' \in D'$ by the following construction:

> Let d have the following form:
> there is an $r \geq 0, s \in T(\Delta, \{z_1, \ldots, z_r\})$ in which every z_i occurs exactly once, and for every $i \in [r]$ there is a term $equ(\zeta_{i,1}, \zeta_{i,2}) \in \text{Equ}(T(\Omega, \mathcal{V}))$, and there is a $\varphi \in Sub(\mathcal{V}, \Omega)$, such that
>
> $$d = (s[z_1/equ(\zeta_{1,1}, \zeta_{1,2}), z_2/equ(\zeta_{2,1}, \zeta_{2,2}), \ldots, z_r/equ(\zeta_{r,1}, \zeta_{r,2})], \varphi)$$
>
> Construct d' as follows:
>
> $$d' = (equ(s[z_1/\zeta_{1,1}, z_2/\zeta_{2,1}, \ldots z_r/\zeta_{r,1}], s[z_1/\zeta_{1,2}, z_2/\zeta_{2,2}, \ldots z_r/\zeta_{r,2}]), \varphi)$$

And $\bar{d}'$ is unambiguously defined by the following lo–narrowing step: $d' \overset{lo}{\rightsquigarrow}_{\mathcal{R}_{\text{equ}},j} \bar{d}'$.

Quite obviously, the above construction is the inverse of $h_{\text{lo–ulo}_{tu}}$, and therefore it holds that $h_{\text{lo–ulo}_{tu}}(d') = d$. Since the ulo$_{tu}$–narrowing step will apply exactly one narrowing step, the applied rule of which is the rule j, it must also hold that $h_{\text{lo–ulo}_{tu}}(\bar{d}') = \bar{d}$. This concludes the proof of the second condition of Definition 6.3 and the proof of the lemma. □

Somehow it is clear that ulo$_{tu}$–narrowing is more single success efficient than lo–narrowing, because the constructor consistency check eliminates subtrees of the lo–narrowing tree in the ulo$_{tu}$–narrowing tree. However, we now provide a formal proof.

Theorem 7.10 Ulo$_{tu}$–narrowing is more single success efficient than lo–narrowing, i.e.,

1. there is a ctn term rewriting system $\mathcal{R}$, such that

   $$\text{ulo}_{tu}\text{–nar}\mathcal{I}(\mathcal{R}) \text{ is more single success efficient w.r.t. } h_{\text{lo–ulo}_{tu}} \text{ than lo–nar}\mathcal{I}(\mathcal{R})$$

   and

2. there is no ctn term rewriting system $\mathcal{R}$ and calculandum $d \in D' \setminus D'_{\text{success}}$, such that

   $$(\eta(\text{lo–nar}\mathcal{I}(\mathcal{R})))(d) < (\eta(\text{ulo}_{tu}\text{–nar}\mathcal{I}(\mathcal{R})))(d)$$

Proof:

To prove Statement 1 of the theorem, we must show that there is a ctn term rewriting system $\mathcal{R}$, such that ulo$_{tu}$–nar$\mathcal{I}(\mathcal{R})$ is more single success efficient with respect to $h_{\text{lo–ulo}_{tu}}$ than lo–nar$\mathcal{I}(\mathcal{R})$. In particular this means that we have to prove (i) and (ii):

(i) There is an h, such that ulo$_{tu}$–nar$\mathcal{I}(\mathcal{R})$ is h–comparable with lo–nar$\mathcal{I}(\mathcal{R})$.

(ii) Let D be the set of derivation forms of ulo$_{tu}$–nar$\mathcal{I}(\mathcal{R})$ and let D' be the set of derivation forms of lo–nar$\mathcal{I}(\mathcal{R})$.

(a) For every $d' \in D'$ with $(\eta(\text{lo–nar}\mathcal{I}(\mathcal{R})))(d') \geq 1$: $(\eta(\text{ulo}_{tu}\text{–nar}\mathcal{I}(\mathcal{R})))(h_{\text{lo–ulo}_{tu}}(d')) \leq (\eta(\text{lo–nar}\mathcal{I}(\mathcal{R})))(d')$ and

(b) there is a $\bar{d}' \in D'$, such that $(\eta(\text{ulo}_{tu}\text{–nar}\mathcal{I}(\mathcal{R})))(h_{\text{lo–ulo}_{tu}}(d')) < (\eta(\text{lo–nar}\mathcal{I}(\mathcal{R})))(d')$.

Let us consider the ctn term rewriting system $\mathcal{R} = (\{ap^{(2)}, \{A^{(0)}, NIL^{(0)}, CONS^{(2)}\}, R)$ where R consists of the following rules:

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

Proof of (i): By Lemma 7.9 it holds that $\text{ulo}_{tu}\text{–nar}\mathcal{I}(\mathcal{R})$ is $h_{\text{lo–ulo}_{tu}}$–comparable with $\text{lo–nar}\mathcal{I}(\mathcal{R})$.

Proof of (ii): For the calculandum $d = (equ(ap\,(x, NIL), CONS(A, NIL)), \varphi_\emptyset)$ the following holds. A derivation by the deterministic ulo_{tu}–narrowing calculus for $(E_\mathcal{R}, \Delta)$–unification takes four steps to derive the first success form. On the other hand, the deterministic lo–narrowing calculus for $(E_\mathcal{R}, \Delta)$–narrowing can not derive a success form, because the leftmost branch of the lo–narrowing tree is infinite. Therefore

$$\begin{array}{rcl} (\eta(\text{ulo}_{tu}\text{–nar}\mathcal{I}(\mathcal{R})))(d) & = & 4 \\ (\eta(\text{lo–nar}\mathcal{I}(\mathcal{R})))(d) & = & \infty, \end{array}$$

Thus, we have shown part (b) of (ii). Part (a) will be shown later.

And now we proof Statement 2 of the theorem. Let us assume that there is a calculandum d, such that

$$(\eta(\text{lo–nar}\mathcal{I}(\mathcal{R})))(d) < (\eta(\text{ulo}_{tu}\text{–nar}\mathcal{I}(\mathcal{R})))(h_{\text{lo–ulo}_{tu}}(d)).$$

Consider the derivation trees of $\text{lo–nar}\mathcal{I}(\mathcal{R})$ and $\text{ulo}_{tu}\text{–nar}\mathcal{I}(\mathcal{R})$ for a ctn term rewriting system $\mathcal{R}$ and calculandum d and $h_{\text{lo–ulo}_{tu}}(d)$, respectively. Since the two calculi only differ in the constructor consistency check, which causes the ulo_{tu}–narrowing calculus to stop certain derivations, the success forms of $\text{lo–nar}\mathcal{I}(\mathcal{R})$ and $\text{ulo}_{tu}\text{–nar}\mathcal{I}(\mathcal{R})$ label the "same" nodes, i.e., nodes the paths to which are equal. Thus, the paths to the nodes which are labeled by d and $h_{\text{lo–ulo}_{tu}}(d)$, respectively, are equal. This means that differences in the single success efficiency can only emerge from the part of the ulo_{tu}–narrowing tree, which is left of the path from the root to the node labeled by $h_{\text{lo–ulo}_{tu}}(d)$.

This part must contain more nodes than the corresponding part of the lo–narrowing tree for our assumption to be true. Obviously this can *not* be the case, because

- either lo–narrowing and ulo_{tu}–narrowing steps lead to derivation forms which are equal with respect to $h_{\text{lo–ulo}_{tu}}$ or
- the ulo_{tu}–derivation stops due to the constructor consistency check, whereas the lo–narrowing derivation continues.

This is a contradiction to our assumption, and thus Statement 2 is proved. Again, Part (a) of (ii) follows immediately from Statement 2. □

As in the previous comparison we will now expand this result to the multi success case.

7.2.2 Multi Success Efficiency

It is quite obvious that Theorem 7.10 holds in the multi success case, too.

Theorem 7.11 Ulo$_{tu}$–narrowing is more multi success efficient than lo–narrowing, i.e.,

1. there is a ctn term rewriting system $\mathcal{R}$, such that for a $k \geq 2$:

   $$\text{ulo}_{tu}\text{–nar}\mathcal{I}\,(\mathcal{R}) \text{ is more } k\text{–success efficient w.r.t. } (h_{\text{lo–ulo}_{tu}}, \sim, \sim') \text{ than lo–nar}\mathcal{I}\,(\mathcal{R})$$

   and

2. there is no ctn term rewriting system $\mathcal{R}$, such that there is a $k \geq 2$ with:

   $$(\eta(\text{lo–nar}\mathcal{I}\,(\mathcal{R}), \sim'))\,(d) < (\eta(\text{ulo}_{tu}\text{–nar}\mathcal{I}\,(\mathcal{R}), \sim))\,(d)$$

where the equivalence relations $\sim$ and $\sim'$ are defined as follows:

- For all $d, d' \in D$ with $d = (t, [v_1/t_1, \ldots v_n/t_n])$: $d \sim d'$, iff there is a variable renaming ρ, such that $d' = (\hat{\rho}(t), [\rho(v_1)/\hat{\rho}(t_1), \ldots \rho(v_n)/\hat{\rho}(t_n)])$; where $\hat{\rho}$ is the homomorphical extension of $\rho : \mathcal{V} \to \mathcal{V}$ to terms $T(\hat{F} \cup \Delta, \mathcal{V})$.
- For all $d, d' \in D'$ with $d = (t, [v_1/t_1, \ldots v_n/t_n])$: $d \sim' d'$, iff there is a variable renaming ρ, such that $d' = (\hat{\rho}(t), [\rho(v_1)/\hat{\rho}(t_1), \ldots \rho(v_n)/\hat{\rho}(t_n)])$; where $\hat{\rho}$ is the homomorphical extension of $\rho : \mathcal{V} \to \mathcal{V}$ to terms $\text{Equ}\,(T(\Omega, \mathcal{V}))$.

The proof is analogous to the proof of Theorem 7.10. Merely, the nodes which are labeled by the k–th success forms which the deterministic lo–narrowing and the deterministic ulo$_{tu}$–narrowing calculus find have to be considered in place of the nodes which are labeled by the first success forms found by those calculi.

8 Conclusion

We have given a formal definition of a calculus. A more specialized calculus is the indexed calculus which fixes an unambiguous derivation tree for a given calculandum. The deterministic calculus describes a depth–first left-to-right tree traversal over the derivation tree. Due to its determinism it can serve as basis for an abstract machine which itself can serve as basis for an implementation of a functional–logic programming language.

In Section 5 we have instantiated our formal calculi with methods for finding $E_{\mathcal{R}}$–unifiers.

It would be interesting to put other narrowing methods like e.g. LSE–narrowing [WBK94] or needed narrowing [AEH94] into the formal framework of calculi. Then, it would be possible to compare their efficiencies as it was done in Section 7. Also the additional efficiency measures success density efficiency and infinite path efficiencies defined in Section 6 could be applied. As mentioned in the beginning of Section 7 optimality is a representative efficiency measure. It should therefore be defined in the context of our calculi.

References

[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.

[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.

[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.

[EV86] J. Engelfriet and H. Vogler. Pushdown machines for the macro tree transducer. *Theoret. Comput. Sci.*, 42:251–368, 1986.

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

[Faß95a] H. Faßbender. *E–Unifikation für primitiv–rekursive Baumfuktionen*. PhD thesis, University of Ulm, March 1995. Abteilung Theoretische Informatik.

[Faß95b] H. Faßbender. A verified implementation of narrowing. In *to appear in proceedings of PLILP' 95*, 1995.

[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.

[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.

[Hoa72] C. A. R. Hoare. Proof of correctness of data representation. *Acta Informatica*, 1:271–, 1972.

[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.

[Klo92] J. W. Klop. Term rewrite systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Vol. 2*, pages 1–116. Oxford Science Publications, 1992.

[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.

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

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

[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.