ELSEVIER

Theoretical Computer Science 151 (1995) 487–512

Theoretical Computer Science

Modular proofs for completeness of hierarchical term rewriting systems ☆

M.R.K. Krishna Rao*

Computer Science Group, Tata Institute of Fundamental Research, Colaba, Bombay-400 005, India

Abstract

In this paper, we study modular aspects of hierarchical combinations of term rewriting systems. A combination $\mathscr{R}_0 \cup \mathscr{R}_1$ is hierarchical if the defined symbols of the two subsystems $\mathscr{R}_0$ and $\mathscr{R}_1$ are disjoint, some of the defined symbols of $\mathscr{R}_0$ are constructors in $\mathscr{R}_1$ and the defined symbols of $\mathscr{R}_1$ do not occur in $\mathscr{R}_0$. It is shown that in hierarchical combinations, a reduction can increase the rank of a term. Therefore, techniques employed in proving the modularity results for direct sums and constructor sharing systems are not applicable for hierarchical combinations.

We propose a set of sufficient conditions for the modularity of completeness of hierarchical combinations. The sufficient conditions are syntactic ones (about recursion) and can be easily tested for finite systems. First, the modularity of strong innermost normalization (SIN) for a class of hierarchical combinations is established. By imposing a restriction that $\mathscr{R}_0 \cup \mathscr{R}_1$ is an overlay system, the modularity of local confluence is established for this class. Then the modularity of completeness is obtained using a recent result relating strong innermost normalization and termination properties of locally confluent overlay systems.

1. Introduction

In the last few decades, term rewriting systems have played a fundamental role in the analysis and implementation of abstract data type specifications, decidability of word problems, theorem proving, computability theory, design of functional programming languages (e.g. Miranda), integration of functional and logic programming paradigms, etc. The study of properties which are preserved under combinations of term rewriting systems (called modular properties) is of both theoretical and practical importance.

A property **P** of term rewriting systems is *modular* if the following holds: *two rewriting systems* $\mathscr{R}_0$ *and* $\mathscr{R}_1$ *have property* **P** *if and only if their union* $\mathscr{R}_0 \cup \mathscr{R}_1$

☆ This is a revised and extended version of [8, 9].

* E-mail: krishna@tifrvax.bitnet.

SSDI 0304-3975(95)00075-5

has property **P**. A knowledge that property **P** is modular gives the following advantages:

1. *Analysis*: To check whether a (large) system satisfies **P**, one can decompose it into a set of smaller subsystems and check whether these subsystems satisfy **P**. This is very important because most of the interesting properties of rewrite systems are intractable (some are even undecidable). In other words, the modularity results facilitate the applicability of *divide-and-conquer* approach in the analysis of properties of rewrite systems.

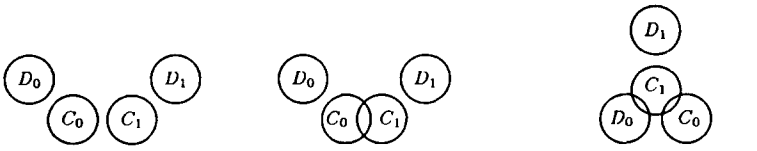
2. *Synthesis*: If a system S_1 satisfying a desirable property **P** is to be extended with a new set of rules, it is enough to check whether the new set of rules satisfy **P** for ensuring that the extended system still satisfies **P**. In other words, the modularity results facilitate *incremental development* of systems.

In a seminal paper [26], Toyama introduced the notions of modularity and *direct sum* of rewrite systems. The union $\mathscr{R}_0 \cup \mathscr{R}_1$ of two rewrite systems $\mathscr{R}_0$ and $\mathscr{R}_1$ is called a direct sum if alphabets of $\mathscr{R}_0$ and $\mathscr{R}_1$ are disjoint. Toyama proved the modularity of *confluence* property in [26] and refuted the modularity of *termination* through a counterexample. Klop and Barendregt (cf. [27]) and Drosten [4] have independently shown that termination is not preserved even if the two components are *complete* (confluent and terminating). Rusinowitch [24] and Middeldorp [17] formulated sufficient conditions for the modularity of termination based on the distribution of collapsing and duplicating rules in the constituent systems. Toyama et al. [28] established that left-linearity is sufficient for the modularity of completeness. Using a powerful technique, called alien-replacement, Kurihara and Ohuchi [14] proved an interesting result; *simple termination* is modular. All these results are for direct sums (i.e., sharing of function symbols is forbidden).

Kurihara and Ohuchi [15] and Middeldorp and Toyama [19] have obtained a few results on the modularity of termination when the two constituent systems share *constructors*. Function symbol f is a constructor in $\mathscr{R}$ if f does not occur as outermost symbol of the left-hand side term of any rewrite rule in $\mathscr{R}$, otherwise it is a defined symbol. Kurihara and Ohuchi [15] proved the modularity of simple termination for rewrite systems with shared constructors, whereas Middeldorp and Toyama [19] proved that completeness is modular for systems with constructor discipline.

Although the above results are elegant and interesting, they are not applicable in situations where defined symbols of $\mathscr{R}_0$ are used as constructor symbols in $\mathscr{R}_1$ (i.e., hierarchical combinations). This situation arises very naturally in an incremental development (or synthesis) of programs and algebraic specifications. This style of writing (and developing) programs is encouraged in logic and functional programming. Since termination of logic programs [12] and functional programs is closely related to that of term rewriting systems, the results which can be applied in this situation will be very useful and have a great significance from the practical point of view. However, the modular aspects of hierarchical combinations are not explored well in the literature.

The following diagram gives the pictorial view of direct sums, constructor sharing systems and hierarchical combinations. The sets of defined and constructor symbols of $\mathscr{R}_i$ are denoted by D_i and C_i respectively.



(a) Direct sum (b) Sharing Constructors (c) Hierarchical Combination

In this paper, we deal with modular aspects of hierarchical combinations, in particular *completeness*. A set of sufficient conditions for the modularity of completeness of hierarchical combinations is proposed. The conditions are syntactic ones (about recursion) and can be checked very easily. Our main result is a generalization of the main result of Middeldorp and Toyama [19].

It may be noted that techniques used in proving the modularity of termination for direct sums and constructor sharing systems are not applicable in hierarchical combinations because the following property is not valid for hierarchical combinations: if $t \Rightarrow^* t'$ then $rank(t) \geqslant rank(t')$. That is, in hierarchical combinations, a reduction can increase rank of the term. This complicates the proofs and necessitates a lot of machinery to deal with hierarchical combinations.

We employ the following approach for studying the modularity of completeness for a class of hierarchical combinations, called *proper-extensions*. To make the proofs simpler and avoid mixing of many issues, we start with a proper subclass of proper-extensions called *nice-extensions*. Using a result on abstract reduction systems, we show that the hierarchical combination $\mathscr{R}_0 \cup \mathscr{R}_1$ is strongly innermost normalizing (SIN), i.e., terminates under the innermost reduction strategy, if $\mathscr{R}_0$ and $\mathscr{R}_1$ are strongly innermost normalizing systems and $\mathscr{R}_1$ is a nice-extension of $\mathscr{R}_0$. That is, strong innermost normalization (SIN) is modular for this class of combinations. Then we point out that completeness is not modular for this class. To obtain the modularity of completeness, we impose a restriction that the combined system is an overlay system. Since overlay systems allow overlapping only at outermost level, it is very easy to prove the modularity of local confluence. Then, the modularity of completeness for this class follows from the modularity of innermost normalization and local confluence properties for this class and by a recent result of Gramlich. We then extend our results to the class of proper-extensions.

In fact, we consider a larger class of combinations than the hierarchical combinations. This class is called super-hierarchical combinations and allows (i) defined symbols to be shared and (ii) defined symbols of the higher system ($\mathscr{R}_1$) occurring on the left-hand sides of the base system ($\mathscr{R}_0$) as constructors, unlike in hierarchical combinations.

The rest of the paper is organized as follows. Section 2 gives the preliminary definitions and the results needed later. In Section 3, we give a brief overview of the existing results on modular aspects of term rewriting systems. In Section 4, various classes of hierarchical and super-hierarchical combinations, such as nice and proper-extensions, are defined. Section 5 establishes the modularity of innermost normalization for nice-extensions. Using this result, the modularity of completeness for a class of nice-extensions is established in Section 6. Section 7 relates proper-extensions with nice-extensions and establishes the modularity of completeness for a class of proper-extensions. Section 8 concludes with a discussion.

2. Preliminaries

We assume that the reader is familiar with the basic terminology of term rewriting systems, like contexts, substitutions and properties such as confluence (CR), local confluence (WCR), strong normalization (SN) and strong innermost normalization (SIN) etc. and give definitions only when they are required. The notations not defined in the paper can be found in [3, 7] or [18].

Definition 1 (*Critical pairs*). Let $l_1 \to r_1$ and $l_2 \to r_2$ be renamed versions of rewrite rules of a term rewriting system $\mathscr{R}$ such that they have no variables in common. Suppose $l_{1|p}$ is not a variable for some position p and $l_{1|p}$ unifies with l_2 through a most general unifier σ. *The pair of terms* $\langle l_1[r_2]_p\sigma,\ r_1\sigma\rangle$ *is called a critical pair of* $\mathscr{R}$. If $l_1 \to r_1$ and $l_2 \to r_2$ are renamed versions of the same rewrite rule, we do not consider the case $p = \varepsilon$. A critical pair $\langle l_1[r_2]_p\sigma,\ r_1\sigma\rangle$ with $p = \varepsilon$ is called an overlay and a critical pair $\langle s,\ t\rangle$ is trivial if $s \equiv t$.

The following definition defines the class of overlay systems.

Definition 2. A term rewriting system $\mathscr{R}$ is an *overlay system* (OS) if all its critical pairs are overlays.

Definition 3. A reduction step $C[l\sigma] \Rightarrow_{\mathscr{I}} C[r\sigma]$ is an *innermost* reduction step if no proper subterm of $l\sigma$ is reducible. A rewriting derivation is an *innermost* derivation if every reduction step in it is innermost. A term rewriting system $\mathscr{R}(\mathscr{F}, R)$ is *strongly innermost normalizing* (SIN) if *every* innermost derivation of $\mathscr{R}(\mathscr{F}, R)$ is of finite length.

The following theorem is proved in [5].

Theorem 4. *A locally confluent overlay system is complete if and only if it is strongly innermost normalizing* (*SIN*).

In the following, $\mathcal{T}(\mathcal{F},\mathcal{X})$ denotes the set of terms constructed from a set of function symbols $\mathcal{F}$ and a set of variables $\mathcal{X}$, and $F(t)$ denotes the set of function symbols occurring in term t. The root of a term t is defined as: $root(t) = f$ if $t \equiv f(s_1,\ldots,s_n)$, and $root(t) = t$ if $t \in \mathcal{X}$.

Definition 5. The set $D_{\mathcal{R}}$ of *defined* symbols of a term rewriting system $\mathcal{R}(\mathcal{F},R)$ is defined as $\{root(l) \mid l \to r \in R\}$ and the set $C_{\mathcal{R}}$ of *constructor* symbols of $\mathcal{R}(\mathcal{F},R)$ is defined as $\mathcal{F} - D_{\mathcal{R}}$.

To show the defined and constructor symbols explicitly, we often write the above rewrite system as $\mathcal{R}(D_{\mathcal{R}},C_{\mathcal{R}},R)$ and omit the subscript when such omission does not cause any confusion.

We need the following definitions in the sequel.

Definition 6 (*Dependency relation $\succeq_d$ over defined symbols*). The dependency relation of a rewrite system $\mathcal{R}(D,C,R)$ is the smallest quasi-order $\succeq_d$ over D satisfying the following conditions:

- $f \succeq_d f$ for each $f \in D$ (reflexivity),
- $f \succeq_d h$ if $f \succeq_d g$ and $g \succeq_d h$ (transitivity),
- $f \succeq_d g$ if there is a rewrite rule $l \to r \in R$ such that $f \equiv root(l)$ and $g \in F(r)$.

We say that a defined symbol $f \in D$ *depends on* a defined symbol $g \in D$ if $f \succeq_d g$. The set of symbols depending on a set of symbols S is defined as $\{f \mid f \succeq_d g$ and $g \in S\}$. Intuitively, $f \succeq_d g$ means that an evaluation of the defined function f for some arguments *may* involve an evaluation of the defined function g for some arguments (i.e., the definition of f depends in some sense on that of g). It also means that an appearance of f in a derivation might lead to a creation of g in the later part of the derivation.

Definition 7. Let $\mathcal{R}(\mathcal{F},R)$ be a rewrite system and t be a term in $\mathcal{T}(\mathcal{F},\mathcal{X})$. A set of function symbols $S \subseteq \mathcal{F}$ is *unreachable* (in $\mathcal{R}$) from t if $S \cap F(t') = \phi$ whenever $t \Rightarrow_{\mathcal{R}}^* t'$.

Lemma 8. *Let $\mathcal{R}(\mathcal{F},R)$ be a rewrite system, $S \subseteq \mathcal{F}$ be a set of function symbols and t be a term in $\mathcal{T}(\mathcal{F},\mathcal{X})$. Then, S is unreachable from t if no function symbol in t depends on S.*

3. Brief overview of existing results on modularity

In this section, we briefly discuss some of the major results (in our view) in the theory of modularity. This overview is meant for introducing the field of modularity to a general reader. We do not consider the conditional and higher order systems in this paper.

3.1. Direct sum

Toyama proved the modularity of *confluence* property in [26] and refuted the modularity of *termination* with the following counterexample [27].

Example 1. It is easy to see that the following $\mathscr{R}_0$ and $\mathscr{R}_1$ are terminating.

$$\mathscr{R}_0 : \mathtt{f}(0,1,\mathtt{x}) \to \mathtt{f}(\mathtt{x},\mathtt{x},\mathtt{x}) \qquad \mathscr{R}_1 : \mathtt{g}(\mathtt{x},\mathtt{y}) \to \mathtt{x}$$
$$\mathtt{g}(\mathtt{x},\mathtt{y}) \to \mathtt{y}$$

But their direct sum has a cyclic derivation

$$\mathtt{f}(0,1,\ \mathtt{g}(0,1)) \Rightarrow \mathtt{f}(\mathtt{g}(0,1),\ \mathtt{g}(0,1),\ \mathtt{g}(0,1)) \Rightarrow \mathtt{f}(0,\ \mathtt{g}(0,1),\ \mathtt{g}(0,1))$$
$$\Rightarrow \mathtt{f}(0,1,\ \mathtt{g}(0,1)) \Rightarrow \cdots$$

Rewrite system $\mathscr{R}_1$ in the above counterexample is not confluent. So one might expect that termination of confluent systems is preserved (i.e., the modularity of completeness). However this was refuted by Klop and Barendregt with a counterexample (see [27]). Drosten [4] provided the following simple counterexample.

Example 2. Following systems $\mathscr{R}_0$ and $\mathscr{R}_1$ are confluent and terminating.

$$\mathscr{R}_0 : \mathtt{f}(0,1,\mathtt{x}) \to \mathtt{f}(\mathtt{x},\mathtt{x},\mathtt{x}) \qquad \mathscr{R}_1 : \mathtt{g}(\mathtt{x},\mathtt{y},\mathtt{y}) \to \mathtt{x}$$
$$\mathtt{f}(\mathtt{x},\mathtt{y},\mathtt{z}) \to 2 \qquad\qquad \mathtt{g}(\mathtt{y},\mathtt{y},\mathtt{x}) \to \mathtt{x}$$
$$0 \to 2$$
$$1 \to 2$$

But the direct sum has the following cyclic derivation

$$\mathtt{f}(0,1,\mathtt{g}(0,1,1)) \Rightarrow \mathtt{f}(\mathtt{g}(0,1,1),\mathtt{g}(0,1,1),\mathtt{g}(0,1,1)) \Rightarrow \cdots \Rightarrow \mathtt{f}(0,1,\mathtt{g}(0,1,1)) \Rightarrow \cdots$$

In the above counterexample, both the left-hand side and the right-hand side terms of the first rule in $\mathscr{R}_0$ are reducible by the second rule. This may give an impression that termination is modular for irreducible confluent systems. However, this conjecture (of Hsiang) was also refuted by a counterexample in [27].

The first positive result on the modularity of termination was presented in [24], where it is proved that termination is modular for (i) collapse-free (i.e., no rule has just a variable on the right-hand side) and (ii) non-duplicating (i.e., no variable has more occurrences on the right-side than on the left-hand side of any rule) rewrite systems and conjectured that 'if direct sum of two terminating systems is non-terminating then one of them should contain collapsing rules and other contain duplicating rules'. Middeldorp [17] settled this conjecture positively and reformulated the result as 'direct sum of two terminating systems is terminating if one of them contains neither collapsing nor duplicating rules'. Toyama et al. [28] established the modularity of completeness for

left-linear term rewriting systems. Marchiori [16] and Schmidt-Schauss and Pintz [25] independently provided a simpler proof of this result.

Kurihara and Ohuchi [14] reported a nice result about the modularity of termination. The result is based on the nature of termination proofs rather than the syntactic properties of rewrite systems. The main theorem in [14] says that *simple-termination is modular* for finite systems. A term rewriting system $\mathscr{R}$ is *simply-terminating* if termination of $\mathscr{R}$ can be proved using some simplification-ordering.

Inspired by the works of Kurihara and Ohuchi [14, 15], Gramlich [6] revisited the results of [24, 17, 14, 15] in a uniform framework, with an assumption that the systems are finitely branching. He proved that 'if the direct sum $\mathscr{R}_0 \cup \mathscr{R}_1$ of two (finitely branching) terminating rewrite systems $\mathscr{R}_0$ and $\mathscr{R}_1$ is non-terminating then one of the two systems (say $\mathscr{R}_0$) is not termination-preserving under non-deterministic collapses (i.e., $\mathscr{R}_0 \cup \{G(x,y) \to x,\ G(x,y) \to y\}$ is non-terminating) and the other system $\mathscr{R}_1$ has a collapsing rule'. Ohlebusch [20, 21] proved this result without the assumption of finite branching (see [6, 20, 21, 23] for more details).

Kurihara and Kaji [13] took an alternative approach to the modularity by defining the notion of *modular reductions* and established very interesting results (e.g. they proved that there is no infinite sequence of modular reductions even if some of the modules are non-terminating).

3.2. Constructor sharing unions

We say that the union $\mathscr{R}_0 \cup \mathscr{R}_1$ of two systems $\mathscr{R}_0(D_0, C_0, R_0)$ and $\mathscr{R}_0(D_1, C_1, R_1)$ is a constructor sharing union if $C_1 \cap D_0 = C_0 \cap D_1 = D_0 \cap D_1 = \phi$.

Kurihara and Ohuchi [15] proved the modularity of simple-termination for (finite) rewrite systems with shared constructors. One of the surprising[1] results (negative) on rewrite systems with shared constructors is that *confluence is not modular* when constructors are shared as shown by Kurihara and Ohuchi [15] with the following counterexample. However, it can be easily shown that confluence is modular for constructor sharing unions of left-linear systems.

Example 3. Following two systems with a shared constructor, h, are confluent.

$\mathscr{R}_0$: f(x,x) → a $\qquad$ $\mathscr{R}_1$: g → h(g)

f(x, h(x)) → b

But $\mathscr{R}_0 \cup \mathscr{R}_1$ is not confluent; term f(g, g) has two different normal forms, a and b.

Middeldorp and Toyama [19] proved that completeness is modular for shared constructor systems. A term rewriting system is called a constructor system (and said to have constructor discipline) if the defined symbols do not occur in the proper subterms

[1] Surprising in view of the fact that confluence is the first property shown to be modular for direct-sums.

of the left-hand sides. The above two results ([15, 19]) are not comparable because [15] assumes that termination proofs (of constituent systems) are given by simplification orderings whereas [19] assumes constructor discipline and confluence of the constituent systems. There are systems whose termination can be established by one result but not by the other. See [15, 19] for such examples.

Gramlich in a recent paper [5] reported some nice results relating innermost normalization and strong normalization properties of rewrite systems. Using these results, he has given a simpler proof for the result of Middeldorp and Toyama [19]. To be precise, he proved the modularity of termination for *locally confluent overlay systems*. The main result of [5] relating strong innermost normalization and termination properties of rewrite systems is very useful in establishing our results below.

Kurihara and Ohuchi [15] proved that confluence is preserved if the constructor sharing systems are also simply terminating. That is, *confluence + simple termination* is a modular property for constructor sharing unions. Ohlebusch [22] established that *semi-completeness* (i.e., confluence + weak normalization) is modular for constructor sharing unions.

3.3. Composable unions

We say that the union $\mathscr{R}_0 \cup \mathscr{R}_1$ of two systems $\mathscr{R}_0(D_0, C_0, R_0)$ and $\mathscr{R}_0(D_1, C_1, R_1)$ is a composable union if (i) $C_1 \cap D_0 = C_0 \cap D_1 = \phi$ and (ii) $R_0 \cap R_1 = \{l \to r \in R_0 \cup R_1 \mid root(l) \in D_0 \cap D_1\}$. That is, sharing of defined symbols is allowed if the rules defining these symbols in the two systems are the same.

Ohlebusch [23] has generalized all the above results to the composable unions. It is interesting to note that none of the interesting properties differ on the modularity for constructor sharing unions and composable unions (i.e., it is not yet known if there is any natural property which is modular for constructor sharing unions but not modular for composable unions).

4. Hierarchical combinations

In this section, we define a few classes of hierarchical combinations for which the modularity of completeness is studied in later sections. Before defining these classes, we show that completeness is not modular for hierarchical combinations (of even constructor systems) in general.

Example 4. It is easy to see that the following two systems $\mathscr{R}_0$ and $\mathscr{R}_1$ are *complete*.

$$\mathscr{R}_0 : \mathtt{f(x)} \to \mathtt{x} \qquad \mathscr{R}_1 : \mathtt{h(a)} \to \mathtt{h(f(a))}$$

To wit, the combined system has a cyclic derivation: $\mathtt{h(a)} \Rightarrow_{\mathscr{R}_1} \mathtt{h(f(a))} \Rightarrow_{\mathscr{R}_0} \mathtt{h(a)} \cdots$

The following example shows that confluence is not modular for hierarchical combinations of left-linear systems, even if they are (i) constructor systems and (ii) terminating.

Example 5. It is easy to see that the following two systems $\mathscr{R}_0$ and $\mathscr{R}_1$ are *complete*.

$\mathscr{R}_0 : \mathtt{a} \to \mathtt{b}$ $\quad$ $\mathscr{R}_1 : \mathtt{f(a)} \to \mathtt{c}$

But, the combined system is not confluent; term `f(a)` has two different normal forms, `c` and `f(b)`.

For the discussions in the sequel, it is convenient to classify defined symbols in D_1 into two sets (i) $D_1^0 = \{f \mid f \in D_1$ and $f \succeq_d g$ for some $g \in D_0\}$ consisting of function symbols depending on D_0 and (ii) $D_1^1 = D_1 - D_1^0$ consisting of function symbols not depending on D_0. All throughout the paper, $\succeq_d$ denotes the dependency relation of the combined system.

The following definition characterizes the main class of hierarchical combinations we are interested in.

Definition 9. A term rewriting system $\mathscr{R}_1(D_1, C_1, R_1)$ is a *proper-extension* of another term rewriting system $\mathscr{R}_0(D_0, C_0, R_0)$ if the following conditions are satisfied:

1. $D_0 \cap D_1 = C_0 \cap D_1 = \phi$ (i.e., $\mathscr{R}_0 \cup \mathscr{R}_1$ is a hierarchical combination).
2. Each rewrite rule $l \to r \in R_1$ satisfies the following condition:

(H1): For every subterm s of r, if $root(s) \in D_1^0$ and $root(s) \succeq_d root(l)$, then s contains no function symbol (in $D_0 \cup D_1^0$) depending on D_0 except at the outermost level (of s).

The second (and the main) condition essentially says that no symbol depending on D_0 occurs below the defined symbols (in D_1^0) which are in mutual recursion with $root(l)$. The intuition behind this condition will be clear in the sequel.

Example 6. The following system $\mathscr{R}_1$ is a proper-extension of $\mathscr{R}_0$.

$\mathscr{R}_0 :$ `add(0, x)` $\to$ `x` $\qquad$ $\mathscr{R}_1 :$ `mult(0, x)` $\to$ `0`

`add(S(x), y)` $\to$ `S(add(x, y))` $\qquad$ `mult(S(x), y)` $\to$ `add(y, mult(x, y))`

The diagram representing hierarchical combinations in the introduction suggests that the two components do not share any defined symbols and rewrite rules. In many practical situations a need might arise to allow two systems to share some rewrite rules (and hence defined symbols). This is in particular needed while studying properties like weak normalization, innermost normalization, confluence and semi-completeness, which do not have the following hereditary property; *if R has property* **P** *and R' is a subsystem of R then R' has property* **P**. The lack of this property forces us to allow two components to share some rules (so that the subsystems have the property

(e.g. confluence) of the whole system) while studying (and proving) these properties in a modular way. So, we now consider the following situation: two systems $\mathscr{R}_0(D_0 \uplus D, C_0, R_0)$ and $\mathscr{R}_1(D_1 \uplus D, C_1, R_1)$ sharing the defined symbols in D and the rules $R_0 \cap R_1 = \{l \to r \mid root(l) \in D\}$. We also consider such combinations with a restriction $D_0 \cap D_1 = C_0 \cap D_1 = \phi$ as hierarchical combinations since some of the defined symbols of $\mathscr{R}_0$ are used as built-in functions in $\mathscr{R}_1$. It is interesting to note that this notion of hierarchical combinations is a generalization of composable unions.

The notations, D_1^1 and D_1^0 need slight changes now.

Notation: $D_1^0 = \{f \mid f \in (D_1 \cup D)$ and $f \succeq_d g$ for some $\overset{*}{g} \in D_0\}$ and $D_1^1 = (D_1 \cup D) - D_1^0$. We denote the set of constructors $(C_0 \cup C_1) - (D_0 \cup D_1 \cup D)$ of the combined system by *Constr*. $\mathscr{T}_i$ denotes $\mathscr{T}(D_i \cup D \cup Constr, \mathscr{X})$ and $\mathscr{C}_i$ denotes the set of contexts of $D_i \cup D \cup Constr$, i.e., terms in $\mathscr{T}(D_i \cup D \cup Constr \cup \{\Box\}, \mathscr{X})$. By $\mathscr{C}_0^1$, we denote the set of contexts of $(Constr \cup D \cup D_0 \cup D_1^1)$. We assume that $D \subseteq D_1^1$.

Definition 10. A term rewriting system $\mathscr{R}_1(D_1 \uplus D, C_1, R_1)$ is a *generalized proper-extension* of another term rewriting system $\mathscr{R}_0(D_0 \uplus D, C_0, R_0)$ if the following conditions are satisfied:

1. $D_0 \cap D_1 = C_0 \cap D_1 = \phi$ and $R_0 \cap R_1 = \{l \to r \mid root(l) \in D\}$.
2. Each rewrite rule $l \to r \in R_1$ satisfies the following condition:

(H1): For every subterm s of r, if $root(s) \in (D_1^0 - D)$ and $root(s) \succeq_d root(l)$, then s contains no function symbol (in $D_0 \cup D_1^0$) depending on D_0 except at the outermost level (of s).

4.1. Super-hierarchical combinations

In hierarchical combinations, defined symbols of $\mathscr{R}_1$ are not allowed to occur in $\mathscr{R}_0$. In a few (very rare) situations, it may not be possible to divide a system into two subsystems $\mathscr{R}_0$ and $\mathscr{R}_1$ such that the combination is hierarchical, but it might be possible to divide that system into two subsystems $\mathscr{R}_0$ and $\mathscr{R}_1$ such that the defined symbols of $\mathscr{R}_0$ do not depend on the defined symbols of $\mathscr{R}_1$. Basically, the defined symbols of $\mathscr{R}_1$ are allowed to occur on the left-hand side terms of $\mathscr{R}_0$ and defined symbols of $\mathscr{R}_0$ can occur on both the left and the right-hand side terms of $\mathscr{R}_1$. Such combinations are called super-hierarchical combinations. It may be noted that such a situation can occur with the rewrite systems generated by completion procedures. Now, we generalize the notion of proper-extension to the super-hierarchical combinations.

Definition 11. A term rewriting system $\mathscr{R}_1(D_1 \uplus D, C_1, R_1)$ is a *generalized proper-extension** of another term rewriting system $\mathscr{R}_0(D_0 \uplus D, C_0, R_0)$ if the following conditions are satisfied:

1. $R_0 \cap R_1 = \{l \to r \mid root(l) \in D\}$.
2. $\forall f \in (D_0 \cup D)$, $\forall g \in D_1$, $f \not\succeq_d g$ (i.e., $\mathscr{R}_0 \cup \mathscr{R}_1$ is a super-hierarchical combination).

3. Each rewrite rule $l \to r \in R_1$ satisfies the following condition:
(H1): For every subterm s of r, if $root(s) \in (D_1^0 - D)$ and $root(s) \succeq_d root(l)$, then s contains no function symbol (in $D_0 \cup D_1^0$) depending on D_0 except at the outermost level (of s).

Remark. Note that condition 2 implies $D_0 \cap D_1 = \phi$.

Our main aim is to study modularity of completeness for the class of *generalized proper-extension*s*. To make the proofs simpler and avoid mixing up many issues, we first study modularity of completeness for a proper subclass of generalized proper-extension*s, called generalized nice-extension*s, from which we derive the results for generalized proper-extension*s.

Definition 12. A term rewriting system $\mathscr{R}_1(D_1 \uplus D, C_1, R_1)$ is a *generalized nice-extension** (resp. *nice-extension*, *generalized nice-extension*) of another term rewriting system $\mathscr{R}_0(D_1 \uplus D, C_1, R_1)$ if $\mathscr{R}_1$ is a *generalized proper-extension** (resp. *proper-extension*, *generalized proper-extension*) of $\mathscr{R}_0$ and the following condition is satisfied:

Each rewrite rule $l \to r \in R_1$ satisfies the following condition:
(H2): For every subterm s of r, if $root(s) \in (D_1^0 - D)$, then s contains no function symbol (in $D_0 \cup D_1^0$) depending on D_0 except at the outermost level (of s).

This condition essentially says that the nesting of defined symbols from $(D_1^0 - D)$ is not allowed on the right-hand side terms of rules and no symbol from D_0 occurs below $(D_1^0 - D)$-symbols. The following example shows that the class of (generalized) nice-extension* is a proper subclass of (generalized) proper-extension*.

Example 7. The following system $\mathscr{R}_1$ is a (generalized) proper-extension* of $\mathscr{R}_0$.

$\mathscr{R}_0$:
```
add(0, x) →x
add(S(x), y) → S(add(x, y))
```

$\mathscr{R}_1$:
```
mult(0,x) → 0
mult(S(x),y) →add(y, mult (x, y))
fact(0) → 1
fact(S(x)) →mult(S(x), fact(x))
```

However, $\mathscr{R}_1$ is not a (generalized) nice-extension* of $\mathscr{R}_0$; notice the occurrence of the function `fact` (which depends on D_0) below function `mult` in the last rule violating condition H1.

Remark. Introduction of so many classes of combinations is justified as follows. The notion of hierarchical combinations is very natural from a programming point of view as one defines new functions in terms of the already defined functions. The need to allow sharing of rewrite rules (and hence defined symbols) arises in the analysis of systems, particularly while analyzing for the properties such as innermost normalization and confluence as explained above. We call the combinations sharing defined symbols such that the (non-shared) defined symbols of one system occur as constructors (or built-ins) in the other system but not vice versa, also as hierarchical combinations as the basic idea – some of the functions defined in one system are used as built-in

functions (constructors) in the other system – is the same. The notion of super-hierarchical combinations may look artificial at the first glance. However, one can be easily convinced about the practicality of super-hierarchical combinations by just looking at the following rewrite rules derived by the Knuth–Bendix completion procedure from the group axioms. Another motivation for introducing the notion of super-hierarchical combinations is to characterize the largest class of combinations for which our techniques apply. This class is given a new name, super-hierarchical combinations, as it properly includes the class of hierarchical combinations and the defined symbols of $\mathscr{R}_1$ are allowed to occur in $\mathscr{R}_0$ as constructors (which is beyond the scope of hierarchical combinations).

Example 8. The following system is a subsystem of the canonical term rewrite system derived by the Knuth–Bendix completion procedure from the group axioms.

$$\begin{aligned}
&(x + 0) \to x\\
&(x + i(x)) \to 0\\
&i(x+y) \to i(x)+i(y)\\
&i(i(x)) \to x\\
&i(0) \to 0
\end{aligned}$$

If we want to study properties of this system in a modular fashion, it has to be divided into the following two systems.

$$\mathscr{R}_0 : \begin{aligned}&(x + 0) \to x\\&(x + i(x)) \to 0\end{aligned} \qquad \mathscr{R}_1 : \begin{aligned}&i(i(x)) \to x\\&i(0) \to 0\\&i(x+y) \to i(x)+i(y)\end{aligned}$$

This combination is not hierarchical as the defined symbol i of $\mathscr{R}_1$ occurs on the left-hand sides of $\mathscr{R}_0$. But it is a super-hierarchical combination.

The following lemma characterizes the rewrite rules on generalized nice-extension*s.

Lemma 13. *If $\mathscr{R}_1$ is a generalized nice-extension* of $\mathscr{R}_0$ then for each rule $l \to r \in R_1$, r is of the form $C[t_1,\ldots,t_n]$, where $C \in \mathscr{C}_0^1$, $root(t_i) \in (D_1^0 - D)$ and $t_i \in \mathscr{T}_1$, $1 \leqslant i \leqslant n$ $(n \geqslant 0)$. Further, no proper subterm of t_i contains any function symbol depending on D_0.*

Proof. Follows from the condition (H2) of Definition 12. □

Example 9. It is easy to see that the following system is $\mathscr{R}_1$ a nice-extension* of $\mathscr{R}_0$.

$$\mathscr{R}_0 : b \to c \qquad \mathscr{R}_1 : \begin{aligned}&a \to b\\&f(x) \to h(f(c))\\&f(x) \to a\\&h(x) \to d\end{aligned}$$

Here, $D_0 = \{\mathtt{b}\}$, $C_0 = \{\mathtt{c}\}$, $D_1 = \{\mathtt{a}, \mathtt{f}, \mathtt{h}\}$, $C_1 = \{\mathtt{b}, \mathtt{c}, \mathtt{d}\}$, $D_1^1 = \{\mathtt{h}\}$, $D_1^0 = \{\mathtt{a}, \mathtt{f}\}$ and $Constr = \{\mathtt{c}, \mathtt{d}\}$. The right-hand sides of $\mathscr{R}_1$ are of the above form; (i) $\mathtt{b} \in D_0$ and $\mathtt{d} \in Constr$ are contexts (without any $\Box$) in $\mathscr{C}_0^1$, (ii) $\mathtt{h}(\mathtt{f}(\mathtt{c})) \equiv \mathtt{C}[\mathtt{f}(\mathtt{c})]$, where $\mathtt{C} \equiv \mathtt{h}(\Box)$ is a context in $\mathscr{C}_0^1$ and (iii) $\mathtt{a} \equiv \mathtt{C}[\mathtt{a}]$, where C is a trivial context (i.e., $\Box$) in $\mathscr{C}_0^1$.

Before proving the results about the modularity of completeness of hierarchical combinations (nice/proper-extension*s), we point out that the following property: '$t \Rightarrow^* t'$ *implies* $rank(t) \geqslant rank(t')$' is not valid in hierarchical combinations, as the example given below illustrates. This contributes to the difficulty in proving the modularity results for hierarchical combinations. Now, we define the notion of *rank* of a term.

Definition 14. Let $path(t)$ denotes the set of paths from root to leaves in the tree representation of term t. Each path is a list of symbols ending in a variable or a constant. The *rank* of a path is the number of alternations of D_0 symbols and D_1 symbols (forgetting the other symbols) plus 1. The *rank* of a term t is the maximum rank of its paths.

Rank of a term is basically a measure of layer structure of D_1 symbols and D_0 symbols in the given term.

Example 10. Consider example 6 again. Term $\mathtt{mult}(\mathtt{S}(\mathtt{x}), \mathtt{y})$ has rank 1 and can be rewritten by the second rule in $\mathscr{R}_1$ to the term $\mathtt{add}(\mathtt{y}, \mathtt{mult}(\mathtt{x}, \mathtt{y}))$, which has rank 2.

5. Innermost normalization

In this section, we establish that strong innermost normalization (SIN) is modular for generalized nice-extension*s. We first prove that strong innermost normalization (SIN) is modular for a proper subclass of generalized nice-extension*s, called crosswise independent unions, where defined symbols of one system do not depend on the defined symbols of the other system. Then we establish the result for the whole class of generalized nice-extension*s as follows: (i) we identify a set $\mathscr{S}$ of terms of a special form, (ii) show that $\mathscr{R}_0 \cup \mathscr{R}_1$ is strongly innermost normalizing (over all the terms) if and only if it is strongly innermost normalizing over the set $\mathscr{S}$ and then (iii) show that $\mathscr{R}_0 \cup \mathscr{R}_1$ is strongly innermost normalizing over the set $\mathscr{S}$. Throughout this section, (a) $\mathscr{R}_i$ denotes $\mathscr{R}_i(D_i \uplus D, C_i, R_i)$, (b) $\mathscr{R}_0$, the subsystem $\mathscr{R}_1' = \{l \to r \mid root(l) \in (D_1^1 \cup D)\}$ and $\mathscr{R}_1$ are strongly innermost-normalizing and (c) $\mathscr{R}_1$ is a generalized nice-extension*s of $\mathscr{R}_0$.

5.1. *Innermost normalization of crosswise independent unions*

In this subsection, we study modularity of strong innermost normalization (SIN) for crosswise independent unions. The notion of crosswise independent unions is a generalization of (i) constructor sharing unions, (ii) composable unions and (iii) Plump's crosswise disjoint unions.

Definition 15. We say that two term rewriting systems $\mathscr{R}_0(D_0 \uplus D, C_0, R_0)$ and $\mathscr{R}_1(D_1 \uplus D, C_1, R_1)$ are *crosswise independent* if (i) $R_0 \cap R_1 = \{l \to r \mid root(l) \in D\}$ *and* (ii) $f \not\trianglerighteq_d g$ *for each* $f \in D_i \cup D$ *and* $g \in D_{1-i}$, *where* $i \in \{0,1\}$.

We say that $\mathscr{R}_0 \cup \mathscr{R}_1$ *is a crosswise independent union if* $\mathscr{R}_0$ *and* $\mathscr{R}_1$ *are crosswise independent.*

In crosswise independent unions, the non-shared defined symbols of one system do not appear on the right-hand sides of the other system, but they may appear in the left-hand sides. It is useful to note that the above definition implies (a) $D_0 \cap D_1 = \phi$ and (b) $f \not\trianglerighteq_d g$ for each $f \in D$ and $g \in (D_0 \cup D_1)$.

The following lemma is a characteristic of crosswise independent unions.

Lemma 16. *If* $\mathscr{R}_0$ *and* $\mathscr{R}_1$ *are crosswise independent term rewriting systems and* t *is a term such that no subterm (say,* s*) of* t *with* $root(s) \in D_i$ *is reducible by* $\mathscr{R}_0 \cup \mathscr{R}_1$, *then* $t \Rightarrow^*_{\mathscr{R}_{1-i}} t'$ *implies that no subterm (say,* s'*) of* t' *with* $root(s') \in D_i$ *is reducible by* $\mathscr{R}_0 \cup \mathscr{R}_1$ *for* $i \in \{0,1\}$.

Proof. Follows from the fact that the symbols from D_i do not occur on the right-hand side terms of $\mathscr{R}_{1-i}$. □

In fact, we can replace $\Rightarrow^*_{\mathscr{R}_{1-i}}$ by $\Rightarrow^*_{\mathscr{R}_0 \cup \mathscr{R}_1}$ in the above lemma as the condition that no subterm (say, s) of t with $root(s) \in D_i$ is reducible by $\mathscr{R}_0 \cup \mathscr{R}_1$ implies that no rule from $\{l \to r \mid root(l) \in D_i\}$ is applicable on any term derived from t.

Lemma 17. *If* $\mathscr{R}_0$ *and* $\mathscr{R}_1$ *are crosswise independent term rewriting systems and* t *is a term such that no subterm (say,* s*) of* t *with* $root(s) \in D_i$ *is reducible by* $\mathscr{R}_0 \cup \mathscr{R}_1$, *then* $t \Rightarrow^*_{\mathscr{R}_0 \cup \mathscr{R}_1} t'$ *implies that no subterm (say,* s'*) of* t' *with* $root(s') \in D_i$ *is reducible by* $\mathscr{R}_0 \cup \mathscr{R}_1$ *for* $i \in [0,1]$.

The following theorem establishes the modularity of *strong innermost normalization* (SIN) for crosswise independent unions.

Theorem 18. *If* $\mathscr{R}_0$ *and* $\mathscr{R}_1$ *are crosswise independent strongly innermost normalizing (SIN) term rewriting systems, then* $\mathscr{R}_0 \cup \mathscr{R}_1$ *is strongly innermost normalizing too.*

Proof. Induction on term structure.

Basis: It is obvious that there is no infinite innermost derivation starting from any constructor symbol of arity 0 (constant). Strong Innermost normalization of a defined symbol in $D_i \cup D$ of arity 0 follows from the strong innermost normalization of $\mathscr{R}_i$ (remember that no function symbol in D_{1-i} is reachable from a term in $\mathscr{T}(D_i \cup D \cup C_i, \mathscr{X})$).

Induction step: Consider a term $t \equiv f(t_1, \ldots, t_n)$. By induction hypothesis, each t_i is strongly innermost normalizing. Then by König's lemma, there can be only a finite

number of *innermost reduction* steps before the reduction at root (i.e., at $f(\ldots)$) takes place. Therefore, if there is an infinite innermost derivation from t, there must be an infinite innermost derivation from $t' \equiv f(t'_1,\ldots,t'_n)$, where each t'_k is a normal form derived from t_k, $k\in[1,n]$ using an innermost derivation. Now, we have three cases.

1. If f is a constructor, it is obvious that t' is a normal form and the theorem holds.

2. If $f\in D$, no rule from $\{l\to r \mid root(l)\in(D_0\cup D_1)\}$ is applicable on any term derived from t', by Lemma 17. From this it follows that t' is strongly innermost normalizing.

3. If $f\in D_i$, $i\in[0,1]$, no rule from $\{l\to r \mid root(l)\in D_{1-i}\}$ is applicable on any term derived from t', by Lemma 17. The strong innermost normalization of t' follows from the strong innermost normalization of $\mathscr{R}_i$.

Therefore, $\mathscr{R}_0\cup\mathscr{R}_1$ is strongly innermost normalizing. □

5.2. *Innermost normalization of nice-extensions*

Now, we consider strong innermost normalization of generalized nice-extension*s. We identify a set $\mathscr{S}$ of terms of a special form and show that $\mathscr{R}_0\cup\mathscr{R}_1$ is strongly innermost normalizing (over all the terms) if and only if it is strongly innermost normalizing over the set $\mathscr{S}$.

Definition 19. Let $\mathscr{S}$ be the set of all terms of the form $C[s_1,\ldots,s_n]$, $C\in\mathscr{C}_0^1$, $n\geqslant 0$ such that for all $i\in[1,n]$

(i) $root(s_i)\in(D_1^0-D)$ and

(ii) if u is a proper subterm of s_i and $root(u)\in(D_0\cup D_1^0)$ then u *is not reducible by* $\mathscr{R}_0\cup\mathscr{R}_1$.

Condition (ii) implies that no rule from $\{l\to r \mid root(l)\in D_0\}$ is applicable on s_i and s_i *contains no reducible proper subterms with root-symbol depending on* D_0. This ensures the following property: there are no two different positions p and q in a term $t\in\mathscr{S}$ such that (i) $root(t_{|p})\in(D_1^0-D)$ and $t_{|p}$ is reducible, (ii) $root(t_{|q})\in(D_1^0-D)$ and $t_{|q}$ is reducible and (iii) p is above q or q is above p. That is, (D_1^0-D) symbols are not nested above any redex. In view of this property, we call $\mathscr{S}$, the set of *single layered terms*. Further, $\mathscr{S}$ is closed under $\Rightarrow_{\mathscr{R}_0\cup\mathscr{R}_1}$.

Now, we prove that $\mathscr{R}_0\cup\mathscr{R}_1$ is strongly innermost normalizing (over all the terms) if and only if it is strongly innermost normalizing over $\mathscr{S}$.

Theorem 20. *If* $\mathscr{R}_1$ *is a generalized nice-extension* of* $\mathscr{R}_0$, *then the combined system* $\mathscr{R}_0\cup\mathscr{R}_1$ *is strongly innermost normalizing if and only if it is strongly innermost normalizing over* $\mathscr{S}$.

Proof. The only-if part follows from the definition of the strong innermost normalization (SIN) property of rewrite systems. The if-part is proved by establishing that every term is strongly innermost normalizing using induction on term structure.

Basis: It is easy to see that every constant (in the signature, $D \cup D_0 \cup D_1 \cup C_0 \cup C_1$) is an element of $\mathscr{S}$. Hence every constant is strongly innermost normalizing.

Induction step: Consider a term $t \equiv f(t_1, \ldots, t_n)$. By induction hypothesis, each t_i is strongly innermost normalizing. By König's lemma, there can be only a finite number of *innermost reduction* steps before the reduction at root (i.e., at $f(\ldots)$) takes place. Therefore, if there is an infinite innermost derivation from t, there must be an infinite innermost derivation from $t' \equiv f(t'_1, \ldots, t'_n)$, where each t'_i is a normal form derived from t_i, $1 \leqslant i \leqslant n$ using an innermost derivation. Now, we show that t' is an element in $\mathscr{S}$, thereby establishing its strong innermost normalization and hence *strong innermost normalization of t*.

Since each t'_i is a normal form of $\mathscr{R}_0 \cup \mathscr{R}_1$, it is obvious that $t' \in \mathscr{S}$ if $f \in (D_1^0 - D)$. That is, t' is of the form $C[s]$, where C is the trivial context ($\Box$) and $s \equiv t'$. If $f \notin (D_1^0 - D)$, let $s_1, \ldots, s_m$ be the maximal subterms (from left to right) in t' such that $root(s_j) \in (D_1^0 - D)$, $1 \leqslant j \leqslant m$. That is, t' can be written as $C[s_1, \ldots, s_m]$ such that $C \in \mathscr{C}_0^1$ and $root(s_j) \in (D_1^0 - D)$. It is obvious that each s_j is a subterm of some t'_i and hence a normal form of $\mathscr{R}_0 \cup \mathscr{R}_1$. Therefore, $t' \in \mathscr{S}$. $\square$

5.3. *Innermost normalization of* $\mathscr{R}_0 \cup \mathscr{R}_1$ *over* $\mathscr{S}$

In this subsection, we prove that $\mathscr{R}_0 \cup \mathscr{R}_1$ is strongly innermost normalizing over $\mathscr{S}$, using a result on abstract reduction systems (ARS). First, we prove that $\mathscr{S}$ is closed under $\Rightarrow_{\mathscr{R}_0 \cup \mathscr{R}_1}$.

The following lemma establishes that $\mathscr{S}$ is closed under $\Rightarrow_{\mathscr{R}_1}$. The proof can be intuitively explained as follows. The reductions in C does not create any new $(D_1^0 - D)$ symbols and hence cannot increase the nesting of $(D_1^0 - D)$ symbols. The reduction at the root of any s_i does not increase the nesting of $(D_1^0 - D)$ symbols by condition H2 of Definition 12. The reduction in a proper subterm of s_i does not create any new $(D_1^0 - D)$ symbols and there is no nesting of $(D_1^0 - D)$ symbols above this redex position (by Definition 19). Therefore, it can only result in a term in $\mathscr{S}$.

Lemma 21. *If* $t \in \mathscr{S}$ *and* $t \Rightarrow_{\mathscr{R}_1} t'$ *then* $t' \in \mathscr{S}$ *too.*

Proof. By Definition 19, the term t is of the form $C[s_1, \ldots, s_n]$, $n \geqslant 0$ with $C \in \mathscr{C}_0^1$ and for all $i \in [1, n]$, s_i satisfying the above properties. Let $l \to r \in R_1$ and σ be the rule and the substitution respectively applied in the reduction step $t \Rightarrow_{\mathscr{R}_1} t'$. There are two cases: (a) $root(l) \notin (D_1^0 - D)$ and (b) $root(l) \in (D_1^0 - D)$.

Case (a): $root(l) \notin (D_1^0 - D)$. That is, $root(l) \in D_1^1 \cup (D \cap D_1^0)$. There are two subcases. (1) The reduction took place in C. By definition, no function symbol from $(D_1^0 - D)$ occurs in r and hence, t' is of the form $C'[t_1, \ldots, t_m]$, $C' \in \mathscr{C}_0^1$, $root(t_i) \in (D_1^0 - D)$ such that each t_i is a (not necessarily proper) subterm of some s_j. The lemma holds. (2) The reduction took place in a proper subterm of some s_i and $t' \equiv C[s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_n]$. In this case, by Definition 19, $root(l)$ cannot be from

D_1^0 and hence $root(l) \in D_1^1$. Let s_i be of the form $C_1[u_1,\ldots,u_m]$ such that $root(u_j) \in (D_0 \cup D_1^0)$ and no symbol from $(D_0 \cup D_1^0)$ occurs in C_1 except at the root. By Definition 19, each u_k is irreducible. Since no function symbol in $(D_0 \cup D_1^0)$ occurs in r, s_i' is of the form $C_1'[v_1,\ldots,v_{m'}]$ such that $root(v_j) \in (D_0 \cup D_1^0)$ and no symbol from $(D_0 \cup D_1^0)$ occurs in C_1' except at the root. Further, each v_j is a subterm of some u_k and hence v_j is irreducible. The lemma holds.

Case (b): $root(l) \in (D_1^0 - D)$. In this case, the reduction should take place at the root of some s_i. It follows from Lemma 13 and irreducibility of proper subterms (with root in $(D_0 \cup D_1^0)$) of s_i that $r\sigma$ is of the form $C'[u_1,\ldots,u_m]$, $C' \in \mathscr{C}_0^1$, $root(u_i) \in (D_1^0 - D)$ such that no proper subterm (say, s) of u_i, $1 \leqslant i \leqslant m$ with $root(s) \in (D_0 \cup D_1^0)$ is reducible by $\mathscr{R}_0 \cup \mathscr{R}_1$. It is easy to see that $t' \equiv C[s_1,\ldots,s_{i-1},r\sigma,s_{i+1},\ldots,s_n]$ can be written as $C''[v_1,\ldots,v_{m+n-1}]$, $C'' \in \mathscr{C}_0^1$, $root(v_i) \in D_1^0$ such that no proper subterm (say, s) of v_i, $1 \leqslant i \leqslant m+n-1$ with $root(s) \in (D_0 \cup D_1^0)$ is reducible by $\mathscr{R}_0 \cup \mathscr{R}_1$. The lemma holds. □

The case (b) of the above proof is the technical origin of our condition H2 in Definition 12.

The following lemma establishes that $\mathscr{S}$ is closed under $\Rightarrow_{\mathscr{R}_0}$.

Lemma 22. *If $t \in \mathscr{S}$ and $t \Rightarrow_{\mathscr{R}_0} t'$ then $t' \in \mathscr{S}$ too.*

Proof. Let $l \to r \in R_0$ and σ be the rule and the substitution respectively applied in the reduction step $t \Rightarrow_{\mathscr{R}_0} t'$. If $root(l) \in D$, this rule is also in $\mathscr{R}_1$ and lemma holds in this case by the above lemma. Let us now consider the case $root(l) \in D_0$. By Definition 19, the term t is of the form $C[s_1,\ldots,s_n]$ with $C \in \mathscr{C}_0^1$, $root(s_i) \in (D_1^0 - D)$ for all $i \in [1,n]$ and the reduction must take place in C. Since no function symbol from $(D_1^0 - D)$ occurs in r, it follows that t' is of the form $C'[t_1,\ldots,t_m]$, $C' \in \mathscr{C}_0^1$, $root(t_i) \in (D_1^0 - D)$ such that each t_i is a subterm of some s_j. The lemma holds. □

Now, we establish that $\mathscr{R}_0 \cup \mathscr{R}_1$ is *strongly innermost normalizing* over terms in $\mathscr{S}$. This is done using a result on strong normalization and quasi-commutation of abstract reduction systems (ARSs). We need the following definition.

Definition 23. Let $\to_0$ and $\to_1$ be two relations on a set S. We say, relation $\to_1$ *quasi-commutes over* relation $\to_0$ if *for all terms $s,u,t \in S$ with $s \to_0 u \to_1 t$, there exists a term $v \in S$ such that $s \to_1 v \to_{01}^* t$.* ($\to_{01}^*$ is transitive-reflexive closure of $\to_0 \cup \to_1$).

The importance of quasi-commutation can be seen from the following theorem of Bachmair and Dershowitz [1].

Theorem 24. *If the relations $\to_0$ and $\to_1$ in an ARS $(A, \to_0, \to_1)$ are strongly normalizing and $\to_1$ quasi-commutes over $\to_0$, the relation $\to_0 \cup \to_1$ is strongly normalizing too.*

Henceforth, abstract reduction system $\mathscr{A}$ stands for the following.

Definition 25. We define abstract reduction system $\mathscr{A}$ as $\mathscr{A} \equiv (\mathscr{S}, \rightarrow_0, \rightarrow_1)$, where $\rightarrow_i$ are the binary relations over $\mathscr{S}$ defined as follows. Let $s \equiv C[s_1,\ldots,s_n]$ be a term in $\mathscr{S}$ such that $C \in \mathscr{C}_0^1$ and $root(s_i) \in (D_1^0 - D)$ for each $i \in [1,n]$. Then (a) $s \rightarrow_0 t$ *if* $s \Rightarrow_{\mathscr{R}_0 \cup \mathscr{R}_1} t$ *is an innermost reduction step and the reduction took place in* C and (b) $s \rightarrow_1 t$ *if* $s \Rightarrow_{\mathscr{R}_0 \cup \mathscr{R}_1} t$ *is an innermost reduction step and the reduction took place in one of the subterms* $s_1,\ldots,s_n$.

The relation $\rightarrow_0 \cup \rightarrow_1$ is *precisely* the innermost reduction relation over $\mathscr{S}$ of $\mathscr{R}_0 \cup \mathscr{R}_1$. The following lemma establish that $\rightarrow_0$ and $\rightarrow_1$ are strongly normalizing relations. Therefore, to establish the strong innermost normalization of $\mathscr{R}_0 \cup \mathscr{R}_1$ over $\mathscr{S}$, it is enough to prove that the relation $\rightarrow_1$ *quasi-commutes over* the relation $\rightarrow_0$ in ARS $\mathscr{A}$.

Lemma 26. *In ARS* $\mathscr{A}$, *the relations* $\rightarrow_0$ *and* $\rightarrow_1$ *are strongly normalizing.*

Proof. Let $s \equiv C[s_1,\ldots,s_n]$ be a term in $\mathscr{S}$ such that $C \in \mathscr{C}_0^1$ and $root(s_i) \in (D_1^0 - D)$ for each $i \in [1,n]$. By the definition of $\mathscr{S}$, no rule from $\{l \rightarrow r \mid root(l) \in D_0\}$ is applicable on s_i. Therefore, relation $\rightarrow_1$ is a subrelation of the innermost reduction relation of term rewriting system $\mathscr{R}_1$ and hence strong normalization of $\rightarrow_1$ follows from strong innermost normalization of $\mathscr{R}_1$.

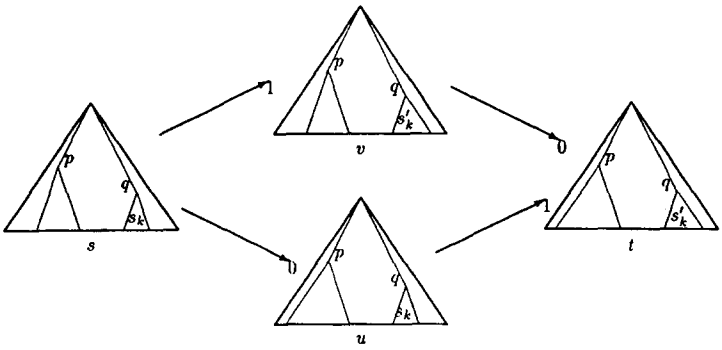
Now consider the subsystem $\mathscr{R}_1' = \{l \rightarrow r \in \mathscr{R}_1 \mid root(l) \in (D_1^1 \cup D)\}$. It is easy to see that $\mathscr{R}_0$ and $\mathscr{R}_1'$ are crosswise independent and hence $\mathscr{R}_0 \cup \mathscr{R}_1'$ is strongly innermost normalizing by Lemma 18. From the definition of $\mathscr{S}$, it is easy to see that the relation $\rightarrow_0$ is a subrelation of the innermost reduction relation of term rewriting system $\mathscr{R}_0 \cup \mathscr{R}_1'$ and hence strong normalization of $\rightarrow_0$ follows from strong innermost normalization of $\mathscr{R}_0 \cup \mathscr{R}_1'$. □

From this lemma, it is clear that, to establish the innermost normalization of $\mathscr{R}_0 \cup \mathscr{R}_1$ over $\mathscr{S}$, it is enough to prove that the relation $\rightarrow_1$ *quasi-commutes over* the relation $\rightarrow_0$ in ARS $\mathscr{A}$.

Theorem 27. *In ARS* $\mathscr{A}$, *the relation* $\rightarrow_1$ quasi-commutes over *the relation* $\rightarrow_0$.

Proof. We have to prove that *for all terms* $s,u,t \in \mathscr{S}$ *with* $s \rightarrow_0 u \rightarrow_1 t$, *there exists a term* $v \in \mathscr{S}$ *such that* $s \rightarrow_1 v \rightarrow_{01}^* t$. Consider a term $s \equiv C[s_1,\ldots,s_n] \in \mathscr{S}$. Since $s \rightarrow_0 u$, term u is of the form $C'[t_1,\ldots,t_m]$, $C' \in \mathscr{C}_0^1$, $root(t_i) \in (D_1^0 - D)$ such that each t_i is a subterm of some s_j. Since $u \rightarrow_1 t$, term $t \equiv C'[t_1,\ldots,t_k',\ldots,t_m]$ (that is, $t_k \Rightarrow_{\mathscr{R}_1} t_k'$). Now, assume that t_k is a subterm of $s_{k'}$. Then $s \equiv C[s_1,\ldots,s_n] \in \mathscr{S}$ can be reduced to $v \equiv C[s_1,\ldots,s_{k'-1},s_{k'}',s_{k'+1},\ldots,s_n] \in \mathscr{S}$ (reduce the subterm t_k in $s_{k'}$). That is, $s \rightarrow_1 v$.

Let p and q be the positions at which two reductions $s \Rightarrow u$ and $s \Rightarrow v$ took place. Since s_k is reducible and the reduction $s \Rightarrow u$ is *innermost* reduction (by the definition of $\rightarrow_0$), p is not above q. Since p is in C and q is in s_k, it is obvious that p is not below q. That is, p and q are disjoint positions. From the following diagram, it becomes clear that $v \rightarrow_0 t$ establishing that $\rightarrow_1$ *quasi-commutes over* $\rightarrow_0$. □



It may be noted that $\rightarrow_0$ does not quasi-commute over $\rightarrow_1$ as illustrated by the following example. The quasi-commutation of $\rightarrow_1$ over $\rightarrow_0$ (and the absence of quasi-commutation of $\rightarrow_0$ over $\rightarrow_1$) can be intuitively explained as follows: a $\rightarrow_0$-step does not create a new $\rightarrow_1$-step whereas a $\rightarrow_1$-step can create a new $\rightarrow_0$-step.

Example 11. Consider example 6 again. It is easy to see that $s \equiv \mathtt{add(mult(y,0),x)} \rightarrow_1 u \equiv \mathtt{add(0,x)} \rightarrow_0 t \equiv \mathtt{x}$, but there is no v such that $s \rightarrow_0 v \rightarrow_{01}^* t$. In fact, s is not reducible by $\mathscr{R}_0$. Therefore, $\rightarrow_0$ does not quasi-commute over $\rightarrow_1$.

Strong innermost normalization of $\mathscr{R}_0 \cup \mathscr{R}_1$ over $\mathscr{S}$ follows from the above two theorems.

Theorem 28. *The combined system $\mathscr{R}_0 \cup \mathscr{R}_1$ is strongly innermost normalizing over $\mathscr{S}$.*

Now, we are in a position to state one of the main results of the paper.

Theorem 29. *If $\mathscr{R}_0$ and $\mathscr{R}_1$ are two strongly innermost normalizing systems such that $\mathscr{R}_1$ is a generalized nice-extension* of $\mathscr{R}_0$, then the combined system $\mathscr{R}_0 \cup \mathscr{R}_1$ is strongly innermost normalizing too.*

Proof. Follows from the above theorem and Theorem 20. □

6. Completeness of nice-extensions

In this section, we study modularity of completeness for the class of generalized nice-extension*s. Unlike strong innermost normalization, completeness is not modular for the whole class of generalized nice-extension*s as demonstrated by the counter examples of Klop and Barendregt [27] and Drosten [4] (see Example 2) – note that

the classes of direct-sums, constructor sharing unions and composable unions are subclasses of generalized nice-extension*s. The best result known about the modularity of completeness for composable unions is that completeness is modular for overlay systems. Now, we consider overlay systems and show that this result extends to the class of generalized nice-extension*s.

By definition, every complete system is terminating and hence strongly innermost normalizing. In the previous section, we established the modularity of strong innermost normalization for the class of generalized nice-extension*s. Therefore, completeness of $\mathscr{R}_0 \cup \mathscr{R}_1$ follows from Theorem 4, if we can establish local confluence of $\mathscr{R}_0 \cup \mathscr{R}_1$.

Lemma 30. *If* $\mathscr{R}_0$ *and* $\mathscr{R}_1$ *are two* locally confluent *systems such that* (i) $\mathscr{R}_1$ *is a* generalized nice-extension* *of* $\mathscr{R}_0$ *and* (ii) $\mathscr{R}_0 \cup \mathscr{R}_1$ *is a overlay system then* the combined system $\mathscr{R}_0 \cup \mathscr{R}_1$ *is locally confluent.*

Proof. According to *critical pair lemma*, it is enough to prove that every critical pair is convergent. Since each $\mathscr{R}_i$ is locally confluent, all the critical pairs obtained from overlapping of the left-hand sides of rules in $\mathscr{R}_i$ are convergent. So, we only have to prove convergence of critical pairs obtained from overlapping of the left-hand side of a rule in $\mathscr{R}_i$ with the left-hand side of a rule in $\mathscr{R}_{1-i}$, $i \in \{0,1\}$. Since $\mathscr{R}_0 \cup \mathscr{R}_1$ is a overlay system, overlapping is only possible at the topmost level and since $D_0 \cap D_1 = \phi$, no overlapping is possible between the left-hand sides of $\{l \to r \in \mathscr{R}_i \mid root(l) \in D_i\}$ and the left-hand sides of $\{l \to r \in \mathscr{R}_{1-i} \mid root(l) \in D_{1-i}\}$. Overlapping is possible only between the rules defining the symbols in D. Since the rules defining these symbols are the same in both the systems, these critical pairs are included in the critical pairs of individual components and hence are convergent. Therefore, $\mathscr{R}_0 \cup \mathscr{R}_1$ is locally confluent. □

Now we state the main result of this section.

Theorem 31. *If* $\mathscr{R}_0$ *and* $\mathscr{R}_1$ *are two* complete *systems such that* (i) $\mathscr{R}_1$ *is a* generalized nice-extension* *of* $\mathscr{R}_0$ *and* (ii) $\mathscr{R}_0 \cup \mathscr{R}_1$ *is a overlay system, then* the combined system $\mathscr{R}_0 \cup \mathscr{R}_1$ *is* complete too.

Proof. Follows from Corollary 1, Theorem 29 and Lemma 30. □

This result is a generalization of the main result in [19]. The main result in [19] says that the composable union $\mathscr{R}_0 \cup \mathscr{R}_1$ of two constructor systems $\mathscr{R}_0(D_0 \uplus D, C_0, R_0)$ and $\mathscr{R}_1(D_1 \uplus D, C_1, R_1)$ is complete if $\mathscr{R}_0$ and $\mathscr{R}_1$ are complete. By the composability restriction, it easily follows that none of the defined symbols in D_1 depend on D_0 and all the rules in $\mathscr{R}_1$ obviously satisfy condition H2 of our Definition 12. Therefore, the above result is a generalization of the main result in [19]. For example, results of [19] are not applicable for the following set of rewrite systems as well as the rewrite systems given in Example 6.

Example 12. It is easy to see that the following two rewrite systems are complete.

$\mathscr{R}_0$: apnd(nil,y) → y
apnd(c(x,xs),y) → c(x,apnd(xs,y))

$\mathscr{R}_1$: rev(nil) → nil
rev(c(x,xs)) → apnd(rev(xs),c(x,nil))

Using our result, we can establish the completeness of $\mathscr{R}_0 \cup \mathscr{R}_1$ whereas the result of [19] is not applicable because the symbol *apnd* defined in $\mathscr{R}_0$ is used as constructor in $\mathscr{R}_1$.

The following example demonstrates that we cannot weaken our condition H2 (in Definition 12) to '*For every subterm s of r, if root*$(s) \in D_1^0$ *then s contains no function symbol of* D_0 *except at the outermost level (of s)*'.

Example 13. It is easy to see that the following two systems $\mathscr{R}_0$ and $\mathscr{R}_1$ are *complete.*

$\mathscr{R}_0$: f(x) → x

$\mathscr{R}_1$: g(y) → f(y)
h(a) → h(g(a))

To wit, the combined system has a cyclic derivation:

$$\mathtt{h(a)} \Rightarrow_{\mathscr{R}_1} \mathtt{h(g(a))} \Rightarrow_{\mathscr{R}_1} \mathtt{h(f(a))} \Rightarrow_{\mathscr{R}_0} \mathtt{h(a)} \cdots$$

The above theorem is a generalization[2] of the main results of Krishna Rao [9] and Dershowitz [2].[3] The main result of Dershowitz can be stated as follows.

Theorem 32. *Let* $\mathscr{R}_0$ *and* $\mathscr{R}_1$ *be two complete overlay systems (OS) such that*

1. *defined symbols of the two systems are disjoint, i.e.,* $D_0 \cap D_1 = \phi$;
2. $\mathscr{R}_1$ *is* flat, *i.e., no nesting of defined symbols are allowed on the left and the right-hand-sides of rewrite rules in* $\mathscr{R}_1$;
3. *defined symbols of* $\mathscr{R}_0$ *do not occur on the left-hand-sides of rewrite rules in* $\mathscr{R}_1$;
4. *if t is a subterm of the right-hand side of a rewrite rule in* $\mathscr{R}_1$ *such that root*$(t) \in D_1$ *then* no defined symbols of $\mathscr{R}_0$ occur in *t*.

Then the combined system $\mathscr{R}_0 \cup \mathscr{R}_1$ *is complete too.*

Remark. The main advantages of our result over that of Dershowitz is that we do not need *flatness* condition (we also do not need the above condition 3). This gives the following two additional advantages: (a) The flatness requirement means that Theorem 32 is not a generalization of the known results (in particular, of [19]),

[2] In [8], no distinction is made between D_1^0 (defined symbols of $\mathscr{R}_1$ depending on D_0) and $D_1^!$ (symbols not depending on D_0) and (weaker) precondition *root*$(s) \in D_1$ is used in condition (H2) instead of the present precondition *root*$(s) \in D_1^0$.

[3] The results of [8, 2] are obtained independently and contemporaneously.

whereas our result is a generalization of the known results. (b) Theorem 32 cannot be extended to many hierarchies as our result will be extended in the next section.

In a revised version of [2], Dershowitz proposed some results orthogonal to our results on hierarchical combinations.

7. Completeness of proper-extensions

In this section, we extend the results of the previous sections for the class of generalized proper-extension*s. Our approach is to relate the notions of generalized nice-extension*s and generalized proper-extension*s and use induction. To relate the notion of generalized proper-extension* with generalized nice-extension*, we need the following definition.

Definition 33 (Equivalence relation $\approx$ over $(D_1^0 - D)$ and partial order $\sqsupset$ over the equivalence classes). From the dependency relation, $\succeq_d$ (see Definition 6) of $\mathscr{R}_1$, we define

1. Equivalence relation $\approx$ ($f \approx g$ if $f \approx_d g$ and $g \succeq_d f$) on the set of defined symbols $(D_1^0 - D)$. We denote the equivalence class containing f by $[f]$.
2. Partial ordering $\sqsupset$ ($[f] \sqsupset [g]$ if $f \succeq_d g$ and $g \not\succeq_d f$) on the set of equivalence classes.

Assumption. In the following we assume that the relation $\sqsupset$ on $(D_1^0 - D)$ is noetherian.

Since signature of any term rewriting system is a countable set, the equivalence relation $\approx$ partitions $(D_1^0 - D)$ into a countable set E of equivalence classes and this partition is called *stratification*. Since relation $\sqsupset$ is noetherian, one can easily extend it to a well-ordering of order type λ, where λ is a countable ordinal.

Notation. For any ordinal α, we denote the αth element in the above well-ordering by E_α (for all ordinals $\alpha > \lambda$, we let $E_\alpha = \phi$) and the rewrite system $\{l \to r \in R_1 \mid root(l) \in (D \cup D_1^\dagger \cup E_\alpha)\}$ by R_α and the combined system $(\bigcup_{\beta<\alpha} R_\beta) \cup \mathscr{R}_0$ by S_α. In particular, S_0 is $\mathscr{R}_0$ and S_κ is $\mathscr{R}_0 \cup \mathscr{R}_1$ for any ordinal κ above λ.

The following theorem relates generalized proper-extension*s with generalized nice-extension*s.

Theorem 34. *Let $\mathscr{R}_0$ and $\mathscr{R}_1$ be two term rewriting systems such that $\mathscr{R}_1$ is a generalized proper-extension* of $\mathscr{R}_0$ and $\sqsupset$ is noetherian. Then R_α is a generalized nice-extension* of S_α for every ordinal α, where R_α and S_α denote the objects explained in the above notation.*

Proof. It is easy to see that the first two conditions in the definition of generalized nice-extension* (see Definition 12 and 11) are satisfied by R_α and S_α. To prove condition 3, we have to prove that *every rule $l \to r$ in R_α satisfies the following*: if s is

a subterm of r such that $root(s) \in E_\alpha$, then no proper subterm of s contains any defined symbol depending on $(Def(S_\alpha) - Def(R_\alpha))$, where $Def(R_\alpha)$ and $Def(S_\alpha)$ are the sets of defined symbols of R_α and S_α respectively. Since $root(s) \in E_\alpha$, it follows from the definition of the above equivalence relation, that $root(s) \succeq_d root(l)$. Since (i) $\mathscr{R}_1$ is a generalized proper-extension of $\mathscr{R}_0$, (ii) $l \to r \in R_1$, (iii) $root(s) \in D_1^0$ and (iv) $root(s) \succeq_d root(l)$, it follows from the definition of generalized proper-extension*s that *no proper subterm of s contains any defined symbol depending on* D_0. Since $(Def(S_\alpha) - Def(R_\alpha)) \subset (D_0 \cup D_1^0)$, any function symbol depending on $(Def(S_\alpha) - Def(R_\alpha))$ also depends on D_0. Hence no proper subterm of s contains any defined symbol depending on $(Def(S_\alpha) - Def(R_\alpha))$. Therefore, R_α is a generalized nice-extension* of S_α. □

Now, we are in a position to establish the main result of the paper.

Theorem 35. *If* $\mathscr{R}_0$ *and* $\mathscr{R}_1$ *are two* complete *systems such that* (i) $\mathscr{R}_1$ *is a* generalized proper-extension* *of* $\mathscr{R}_0$, (ii) $\mathscr{R}_0 \cup \mathscr{R}_1$ *is a overlay system and* (iii) $\sqsupset$ *is noetherian, then the combined system* $\mathscr{R}_0 \cup \mathscr{R}_1$ *is complete too.*

Proof. Proving local confluence of the combined system $\mathscr{R}_0 \cup \mathscr{R}_1$ is again easy as in Lemma 30. Therefore, *completeness of* $\mathscr{R}_0 \cup \mathscr{R}_1$ *follows if we can establish its strong innermost normalization.* We establish this by showing that S_α is strongly innermost normalizing for each α using transfinite induction.

Basis: $\alpha = 0$. By assumption $S_0 = \mathscr{R}_0$ is (complete and hence) strongly innermost normalizing.

Induction: There are two cases: (a) $\alpha \neq 0$ is a successor ordinal and (b) $\alpha \neq 0$ is a limit ordinal.

Case (a): Let $\alpha = \beta + 1$. By definition, $S_\alpha = S_\beta \cup R_\beta$. By the above theorem, R_β is a generalized nice-extension* of S_β. Since R_β is a subsystem of $\mathscr{R}_1$, it is (strongly normalizing and hence) strongly innermost normalizing and S_β is strongly innermost normalizing by induction hypothesis. Therefore, $S_\alpha = S_\beta \cup R_\beta$ is strongly innermost normalizing by Theorem 29.

Case (b): α is a limit ordinal. By definition, $S_\alpha = \bigcup_{\beta<\alpha} S_\beta$. By induction hypothesis, each S_β is strongly innermost normalizing. We show that every term t is strongly innermost normalizing under S_α. Since the set of function symbols occurring in t is finite, there exists a $\beta < \alpha$ such that every function symbol in $Def(S_\alpha) \cap F(t)$ is in $\bigcup_{\gamma \leqslant \beta} E_\gamma$. Therefore, strong innermost normalization of t follows from strong innermost normalization of S_β. □

Since relation $\sqsupset$ is noetherian for any finite system, we have the following.

Corollary 36. *If* $\mathscr{R}_0$ *and* $\mathscr{R}_1$ *are two finite* complete *systems such that* (i) $\mathscr{R}_1$ *is a* generalized proper-extension* *of* $\mathscr{R}_0$, *and* (ii) $\mathscr{R}_0 \cup \mathscr{R}_1$ *is a overlay system, then the combined system* $\mathscr{R}_0 \cup \mathscr{R}_1$ *is complete too.*

Remark. It may be noted that the relation between the generalized proper-extension*s and the generalized nice-extension*s established in Theorem 34 does not hold between the proper-extensions and the nice-extensions as illustrated by the following example. That is, a proper-extension cannot be seen as a sequence of nice-extensions. However, a generalized proper-extension can be seen as a sequence of generalized nice-extensions.

Example 14. Consider the following two systems.

$$\mathscr{R}_0 : \mathtt{f(x)} \to \mathtt{c(x)} \qquad \mathscr{R}_1 : \begin{array}{l} \mathtt{g_2(c(x))} \to \mathtt{g_1(g_2(h(x))} \\ \mathtt{g_1(x)} \to \mathtt{h(f(x))} \\ \mathtt{h(x)} \to \mathtt{x} \end{array}$$

The system $\mathscr{R}_1$ is a proper-extension (but not a nice-extension) of the system $\mathscr{R}_0$. The dependency relation suggests the following stratification: $E_0 = \{\mathtt{g_1}\}$ and $E_1 = \{\mathtt{g_2}\}$. Now, S_1 is $\{\mathtt{f(x)} \to \mathtt{c(x)}, \mathtt{g_1(x)} \to \mathtt{h(f(x))}, \mathtt{h(x)} \to \mathtt{x}\}$ and R_1 is $\{\mathtt{g_2(c(x))} \to \mathtt{g_1(g_2(h(x))}, \mathtt{h(x)} \to \mathtt{x}\}$. It is easy to see that R_1 is not a nice-extension of S_1 as they share a defined symbol, h.

Further, it is not possible to remove $\mathtt{h(x)} \to \mathtt{x}$ from either S_1 of R_1. That is,

1. $R_1 - \{\mathtt{h(x)} \to \mathtt{x}\}$ is not a nice-extension of S_1 as h is occurring below $\mathtt{g_2}$ on the right-hand-side of the rule in $R_1 - \{\mathtt{h(x)} \to \mathtt{x}\}$ and
2. R_1 is not a nice-extension of $S_1 - \{\mathtt{h(x)} \to \mathtt{x}\}$ because the combination is first of all not hierarchical as $S_1 - \{\mathtt{h(x)} \to \mathtt{x}\}$ uses h (defined in R_1) as constructor and R_1 uses $\mathtt{g_1}$ (defined in $S_1 - \{\mathtt{h(x)} \to \mathtt{x}\}$) as constructor.

Therefore, it may not be possible to see a proper-extension as a sequence of nice-extensions. □

Remark. In the above example, the main reason for the inability to view the proper-extension as a sequence of nice-extensions is *the presence of functions in* $D_1^!$. If $D_1^! = \phi$, a proper-extension can indeed be seen as a sequence of nice-extensions – note that S_α and R_α do not share defined symbols and rewrite rules for each α, in this case.

8. Conclusion

The study of modular aspects is very important in an incremental synthesis of programs and systems. If two systems S_1 and S_2 satisfy a property P and P is known to be modular, *one can infer that P is satisfied by the union of* S_1 *and* S_2 without giving a separate proof. This is very important because most of the properties of rewrite systems are intractable. The modularity results of *direct-sums* can be used when two subsystems are defined over different domains, e.g. one on natural numbers and other on lists. The modularity results of *constructor sharing unions* can be used when two subsystems define two *independent* functions (none of the two systems use the procedures defined in the other system) over some domain. The modularity results of *hierarchical combinations* can be used if new procedures (i.e., second system) use the

procedures defined in the other (first) system. This is the most important situation and these results are of great practical significance.

In this paper, modular aspects of hierarchical combinations are investigated. We identified a class of hierarchical combinations for which *completeness* property is modular. Our result generalizes the main result of Middeldorp and Toyama [19]. The nontriviality of the result can be seen from the fact that the techniques employed in getting the modularity results over direct-sums and constructor sharing systems are not applicable in the case of hierarchical combinations as a reduction in hierarchical combinations can increase *rank* of a term.

It would be interesting to extend our results for conditional term rewrite systems. Our investigations in this direction are presently at a very preliminary stage. Modularity of simple termination, weak normalization and semi-completeness of hierarchical combinations has been studied in [10, 11]. Though the classes of combinations considered in [10, 11] are comparable to the classes considered in this paper, the techniques applied/needed there are very different from the techniques used in this paper.

Acknowledgements

The author gratefully acknowledges the discussions he had with Aart Middeldorp, Jan Willem Klop, Nachum Dershowitz, Fred Otto, Enno Ohlebusch, Stefan Kahrs and Narayan Kumar. The author is grateful to the referees for many suggestions in improving the presentation very much.

References

[1] L. Bachmair and N. Dershowitz, Commutation, transformation and termination, in: *Proc. of CADE'8*, Lecture Notes in Computer Science, Vol. 230 (Springer, Berlin, 1986) 5–20.

[2] N. Dershowitz, Hierarchical termination, draft, Hebrew University, Dec. 1992. Revised version to appear in *Proc. CTRS'94*, Lecture Notes in Computer Science (Springer, Berlin) to appear.

[3] N. Dershowitz and J.-P. Jouannaud, Rewrite systems, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science, Vol. B* (North-Holland, Amstredam, 1990) 243–320.

[4] K. Drosten, *Termersetzungssysteme*, Informatik-Fachberichte, Vol. 210 (Springer, Berlin, 1989).

[5] B. Gramlich, Relating innermost, weak, uniform and modular termination of term rewrite systems, in: *Proc. of Logic Prog. and Automated Reasoning, LPAR'92*, Lecture Notes in Computer Science Vol. 624 (Springer, Berlin, 1992) 285–296. Revised version to appear as Abstract relations between restricted termination and confluence properties of rewrite systems in a special issue of *Fundamenta Informatice* on TRSs.

[6] B. Gramlich, Generalized sufficient conditions for modular termination of rewriting, in: *AAECC* (Applicable Algebra in Engineering, Communication and Computing) **5** (1994) 131–158. Preliminary version in *Proc. of ALP'92*, Lecture Notes in Computer Science Vol. 632 (Springer, Berlin, 1992) 53–68.

[7] J.W. Klop, Term rewriting systems, Tech Rep. CS-R9073, CWI, Amsterdam. Also appears as a chapter in S. Abramsky, D. Gabbay and T. Maibaum, ed., *Handbook of Logic in Computer Science, Vol. 2* (Oxford Press, Oxford, 1992).

[8] M.R.K. Krishna Rao, Modular proofs for completeness of hierarchical systems, Technical report, TIFR, Bombay, December 1992.

[9] M.R.K. Krishna Rao, completeness of hierarchical combinations of term rewriting systems, in: *Proc. of 13th Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'93*, Lecture Notes in Computer Science, Vol. 761 (Springer, Berlin, 1993) 125–138.

[10] M.R.K. Krishna Rao, Simple termination of hierarchical combinations of term rewriting systems, in: *Proc. of Theoretical Aspects of Computer Science, TACS'94*, Lecture Notes in Computer Science Vol. 789 (Springer, Berlin, 1994) 203–223.

[11] M.R.K. Krishna Rao, Semi-completeness of hierarchical and super-hierarchical combinations of term rewriting systems, in: *Proc. of Theory and Practice of Software Development, TAPSOFT'95*, Lecture Notes in Computer Science, Vol. 915 (Springer, Berlin, 1995) 379–393.

[12] M.R.K. Krishna Rao, D. Kapur and R.K. Shyamasundar, A Transformational methodology for proving termination of logic programs, in: *Proc. of Computer Science Logic, CSL'91*, Lecture Notes in Computer Science, Vol. 626 (Springer, Berlin, 1991) 213–226.

[13] M. Kurihara and I. Kaji, Modular term rewriting systems and the termination, *Inform. Process. Lett.* **34** (1990) 1–4.

[14] M. Kurihara and A. Ohuchi, Modularity of simple termination of term rewriting systems, *J. IPS, Japan* **34** (1990) 632–642.

[15] M. Kurihara and A. Ohuchi, Modularity of simple termination of term rewriting systems with shared constructors, *Theoret. Comput. Sci.* **103** (1992) 273–282.

[16] M. Marchiori, Modularity of completeness revisited, Technical Report, University of Padova (1994).

[17] A. Middeldorp, A sufficient condition for the termination of the direct sum of term rewriting systems, *Proc. of LICS'89* (1989) 396–401.

[18] A. Middeldorp, Modular properties of term rewriting systems, Ph.D. Thesis, Free University, Amsterdam, 1990.

[19] A. Middeldorp and Y. Toyama, Completeness of combinations of constructor systems, in: *Proc. of RTA'91*, Lecture Notes in Computer Science Vol. 488 (Springer, Berlin, 1991) 188–199 Also appears in *J. Symbolic Comput.* **15** (1991) 331–348.

[20] E. Ohlebusch, A simple proof of sufficient conditions for the termination of disjoint union of TRSs, *Bulletin of EATCS* **49** (1993) 178–183.

[21] E. Ohlebusch, On the modularity of termination of term rewriting systems, Report no. 11, Universität Bielefeld, 1993.

[22] E. Ohlebusch, On the modularity of confluence of constructor-sharing term rewriting systems, in: *Proc. of CAAP'94*, Lecture Notes in Computer Science, Vol. 787 (Springer, Berlin, 1994) 261–275.

[23] E. Ohlebusch, Modular properties of composable term rewriting systems, Ph.D. Thesis, Universität Bielefeld, 1994.

[24] M. Rusinowitch, On termination of the direct sum of term rewriting systems, *Inform. Process. Lett.* **26** (1987) 65–70.

[25] M. Schmidt-Schauss and S. Pintz, Modular termination of consistent and left-linear TRSs, Draft, May, 1994.

[26] Y. Toyama, On the Church-Rosser property for the direct sum of term rewriting systems, *J. ACM* **34** (1987) 128–143.

[27] Y. Toyama, Counterexamples to termination for the direct sum of term rewriting systems, *Inform. Process. Lett.* **25** (1987) 141–143.

[28] Y. Toyama, J.W. Klop and H.P. Barendregt, Termination for the direct sum of left-linear term rewriting systems, in: *Proc. of RTA'89*, Lecture Notes in Computer Science, Vol. 355 (Springer, Berlin, 1989) 477–491.