Modular Termination of Context-Sensitive Rewriting

Bernhard Gramlich*
Institut für Computersprachen
Technische Universität Wien
Favoritenstr. 9, A-1040 Wien, Austria
gramlich@logic.at

Salvador Lucas†
DSIC
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
slucas@dsic.upv.es

ABSTRACT

Context-sensitive rewriting (CSR) has recently emerged as an interesting and flexible paradigm that provides a bridge between the abstract world of general rewriting and the (more) applied setting of declarative specification and programming languages such as OBJ*, CafeOBJ, ELAN, and Maude. A natural approach to study properties of programs written in these languages is to model them as context-sensitive rewriting systems. Here we are especially interested in proving termination of such systems, and thereby providing methods to establish termination of e.g. OBJ* programs. For proving termination of context-sensitive rewriting, there exist a few transformation methods, that reduce the problem to termination of a transformed ordinary term rewriting system (TRS). These transformations, however, have some serious drawbacks. In particular, most of them do not seem to support a modular analysis of the termination problem. In this paper we will show that a substantial part of the well-known theory of modular term rewriting can be extended to CSR, via a thorough analysis of the additional complications arising from context-sensitivity. More precisely, we will mainly concentrate on termination (properties). The obtained modularity results correspond nicely to the fact that in the above languages the modular design of programs and specifications is explicitly promoted, since it can now also be complemented by modular analysis techniques.

Categories and Subject Descriptors

D1.1 [**Software**]: Applicative (Functional) Programming; D3.1 [**Programming Languages**]: Formal Definitions and Theory (D.2.1, F.3.1, F.3.2, F.4.2, F.4.3); F3.1 [**Theory of Computation**]: Specifying and Verifying and Reasoning about Programs (D.2.1, D.2.4, D.3.1, E.1); F3.2 [**Theory of Computation**]: Semantics of Programming Languages (D.3.1); F4.2 [**Theory of Computation**]: Grammars and Other Rewriting Systems (D.3.1); I.1.3 [**Computing Methodologies**]: Languages and Systems—*Evaluation strategies*

General Terms

Languages, Theory, Verification

Keywords

Context-sensitive rewriting, modular proofs of termination, declarative programming, evaluation strategies, modular analysis and construction of programs, program verification

1. INTRODUCTION

Programmers usually organize the programs into components or *modules*. Components of a program are easier to develop, analyze, debug, and test. Eventually, the programmer wants that interesting computational properties like termination hold for the whole program if they could be proved for the individual components of the program. Roughly speaking, this is what being a modular property means.

Context-sensitive rewriting (CSR [26]) is a restriction of rewriting which forbids reductions on selected arguments of functions. In this way, the termination behavior of rewriting computations can be *improved*, by pruning (all) infinite rewrite sequences. Several methods have been developed to formally prove termination of CSR [8, 12, 13, 25, 43, 47]. Termination of (innermost) context-sensitive rewriting has been recently related to termination of declarative languages such as OBJ*, CafeOBJ, and Maude [27, 28]. These languages exhibit a strong orientation towards the modular design of programs and specifications. In this setting, achieving modular proofs of termination is desirable. For instance, borrowing Appendix C.5 of [16], in Figure 1 we show an specification of a program using *lazy lists*. Modules `TRUTH-VALUE` and `NAT` introduce (sorts and) the constructors for boolean and natural numbers. Module `ID-NAT` provides an specialization of the (built-in, syntactical) identity operator[1] '`===`' of OBJ (see module `IDENTICAL` on Appendix

*Work partially supported by ÖAD-Programm "Acciones Integradas 2002-2003" No. 3/2002, WTZ-Büro.

†Work partially supported by CICYT TIC2001-2705-C03-01, Acción Integrada hispano-austríaca HU2001-0019, and Generalitat Valenciana GV01-424.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PPDP'02, October 6-8, 2002, Pittsburgh, Pennsylvania, USA.
Copyright 2002 ACM 1-58113-528-9/02/0010 ...$5.00.

[1]The definition of binary predicate '`===`' is meaningful, provided that the rules are attempted from top to bottom. This is quite a reasonable assumption from the (OBJ) implementation point of view. Nevertheless, our discussion on termination of the program does not depend on this fact in any way.

```
obj TRUTH-VALUE is
  sort Bool .
  op false : -> Bool .
  op true : -> Bool .
endo

obj NAT is
  sort Nat .
  op 0 : -> Nat .
  op s : Nat -> Nat [strat (0)] .
endo

obj ID-NAT is
  protecting TRUTH-VALUE .
  op _===_ : Nat Nat -> Bool [strat (0)] .
  var M N  : Nat .
  eq 0 === 0 = true .
  eq s(M) === s(N) = M === N .
  eq M === N = false .
endo

obj LAZYLIST is
  sort List .
  op nil : -> List .
  op cons :
      List List -> List [assoc idr: nil strat (0)] .
endo

obj INF is
  protecting LAZYLIST[Nat] .
  op inf : Nat -> List [strat (1 0)] .
  var N  : Nat .
  eq inf N = cons(N,inf(s(N))) .
endo

obj TAKE is
  protecting LAZYLIST[Nat] .
  op take  : Nat List -> List [strat (1 2 0)] .
  var N X : Nat .
  var L : List .
  eq take(0,X) = nil .
  eq take(s(N),cons(X,L)) = cons(X,take(N,L)) .
endo

obj LENGTH is
  protecting LAZYLIST[Nat] .
  op length  : List -> Nat [strat (1 0)] .
  var X : Nat .
  var L : List .
  eq length(nil) = 0 .
  eq length(cons(X,L)) = s(length(L)) .
endo
```

Figure 1: Modular specification in OBJ3

D.3 of [16]). Module `INF` specifies a function `inf` which is able to build an infinite object: the list of all natural numbers `inf(`n`)` following number n. Module `TAKE` specifies a function `take` which is able to select the first n components of a (lazy) list given as the second argument of `take`. Finally, module `LENGTH` introduces a function for computing the length of a (finite) list. Here, the use of the *strategy annotation* `strat (0)` for the list constructor `cons` (in module `LAZYLIST`) is intended for both (1) allow for a real terminating behavior of the program due to disallowing the recursive call `inf(s(n))` in the definition of `inf` and (2) avoiding useless reductions on the first component of a list when computing its length. For instance, it is possible to obtain the value `s(s(0))` of `length(take(s(0),inf(s(0))))` without any risk of nontermination[2].

Although very simple, program of Figure 1 provides an interesting application of our modularity results. For instance, whereas it is not possible to prove termination of the program using automatic techniques for proving termination such as Knuth-Bendix, polynomial, or recursive path orderings (see [1, 5]), it is pretty simple to separately prove termination of modules `ID-NAT`, `INF`, `TAKE`, and `LENGTH`. Then, our modularity results permit a formal proof of termination which ultimately relies on the use of purely automatic techniques such as the recursive path ordering (see Example 9 below). Moreover, automatic tools for proving termination such as the C*i*ME 2.0 system[3] can also be used to prove termination of the corresponding modules. In this way, the user is allowed to ignore the details of termination proofs/ techniques by (only) relying on the use of software tools.

Before going into details, let us mention that there exists already abundant literature on rewriting with context-sensitive and other related strategies, cf. e.g. [3], [7], [9], [10], [11], [40].

2. PRELIMINARIES

2.1 Basics

Subsequently we will assume in general some familiarity with the basic theory of term rewriting (cf. e.g. [1], [5]). Given a set A, $\mathcal{P}(A)$ denotes the set of all subsets of A. Given a binary relation R on a set A, we denote the reflexive closure of R by $R^{=}$, its transitive closure by R^{+}, and its reflexive and transitive closure by R^{*}. An element $a \in A$ is an R-normal form, if there exists no b such that $a\ R\ b$; NF_R is the set of R-normal forms. We say that b is an R-normal form of a, if b is an R-normal form and $a\ R^{*} b$. We say that R is *terminating* iff there is no infinite sequence $a_1\ R\ a_2\ R\ a_3 \cdots$. We say that R is *locally confluent* if, for every $a, b, c \in A$, whenever $a\ R\ b$ and $a\ R\ c$, there exists $d \in A$ such that $b\ R^{*} d$ and $c\ R^{*} d$. We say that R is *confluent* if, for every $a, b, c \in A$, whenever $a\ R^{*} b$ and $a\ R^{*} c$, there exists $d \in A$ such that $b\ R^{*} d$ and $c\ R^{*} d$. If R is confluent and terminating, then we say that R is complete. Throughout the paper, $\mathcal{X}$ denotes a countable set of variables and $\mathcal{F}$ denotes a signature, i.e., a set of function symbols $\{f, g, \ldots\}$, each having a fixed arity given by a mapping $ar : \mathcal{F} \to \mathbb{N}$. The set of terms built from Σ and $\mathcal{X}$ is $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A term is said to be linear if it has no multiple occurrences of a single variable. Terms are viewed as labelled trees in the usual way. Positions $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms of t. Given positions p, q, we denote its concatenation as $p.q$. Positions are ordered by the standard prefix ordering $\leq$. Given a set of positions P, $maximal_{\leq}(P)$ is the set of maximal positions of P w.r.t. $\leq$. If p is a position, and Q is a set of positions, $p.Q = \{p.q \mid q \in Q\}$. We denote the empty chain by Λ. The set of positions of a term t is $\mathcal{P}os(t)$. Positions of non-variable symbols in t are denoted as $\mathcal{P}os_{\mathcal{F}}(t)$, and $\mathcal{P}os_{\mathcal{X}}(t)$ are the positions of variables. The subterm at position p of t is denoted as $t|_p$ and $t[s]_p$ is the term t with the subterm at position p replaced by s. The symbol labelling the root of t is denoted as $root(t)$.

A rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side (*lhs*) of the rule is l and r is the right-hand side (*rhs*). A TRS is a pair $\mathcal{R} = (\mathcal{F}, R)$ where R is a set of rewrite rules.

[2]This claim can be justified by using the results in [31, 30].

[3]Available at `http://cime.lri.fr`

$L(\mathcal{R})$ denotes the set of *lhs*'s of $\mathcal{R}$. An instance $\sigma(l)$ of a *lhs* l of a rule is a redex. The set of redex positions in t is $\mathcal{P}os_{\mathcal{R}}(t)$. A TRS $\mathcal{R}$ is left-linear if for all $l \in L(\mathcal{R})$, l is a linear term. Given TRS's $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{R}' = (\mathcal{F}', R')$, we let $\mathcal{R} \cup \mathcal{R}'$ be the TRS $(\mathcal{F} \cup \mathcal{F}', R \cup R')$.

A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ rewrites to s (at position p), written $t \xrightarrow{p}_{\mathcal{R}} s$ (or just $t \to s$), if $t|_p = \sigma(l)$ and $s = t[\sigma(r)]_p$, for some rule $\rho : l \to r \in R$, $p \in \mathcal{P}os(t)$ and substitution σ. A TRS is terminating if $\to$ is terminating. We say that t *innermost* rewrites to s, written $t \to_i s$, if $t \xrightarrow{p} s$ and $p \in maximal_{\leq}(\mathcal{P}os_{\mathcal{R}}(t))$. A TRS is *innermost* terminating if $\to_i$ is terminating.

2.2 Context-Sensitive Rewriting

Given a signature $\mathcal{F}$, a mapping $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ is a *replacement map* (or $\mathcal{F}$-map) if for all $f \in \mathcal{F}$, $\mu(f) \subseteq \{1, \ldots, ar(f)\}$ [26]. Let $M_{\mathcal{F}}$ (or $M_{\mathcal{R}}$ if $\mathcal{R} = (\mathcal{F}, R)$ determines the considered symbols), the set of all $\mathcal{F}$-maps. For the sake of simplicity, we will apply a replacement map $\mu \in M_{\mathcal{F}}$ on symbols $f \in \mathcal{F}'$ of any signature $\mathcal{F}'$ by assuming that $\mu(f) = \varnothing$ whenever $f \notin \mathcal{F}$. A replacement map μ specifies the *argument* positions which can be reduced for each *symbol* in $\mathcal{F}$. Accordingly, the set of *μ-replacing positions* $\mathcal{P}os^{\mu}(t)$ of $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is: $\mathcal{P}os^{\mu}(t) = \{\Lambda\}$, if $t \in \mathcal{X}$ and

$$\mathcal{P}os^{\mu}(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(root(t))} i.\mathcal{P}os^{\mu}(t|_i),$$

if $t \notin \mathcal{X}$. The set of positions of replacing redexes in t is $\mathcal{P}os^{\mu}_{\mathcal{R}}(t) = \mathcal{P}os_{\mathcal{R}}(t) \cap \mathcal{P}os^{\mu}(t)$. A context-sensitive rewrite system (CSRS) is a pair $(\mathcal{R}, \mu)$, where $\mathcal{R}$ is a TRS and μ is a replacement map. In *context-sensitive rewriting* (CSR [26]), we (only) contract *replacing* redexes: t μ-rewrites to s, written $t \hookrightarrow_{\mu} s$ (or just $t \hookrightarrow s$), if $t \xrightarrow{p}_{\mathcal{R}} s$ and $p \in \mathcal{P}os^{\mu}(t)$.

Example 1. Consider the TRS $\mathcal{R}$:

$$\begin{aligned} 0 === 0 &\to true \\ s(x) === s(y) &\to x === y \\ x === y &\to false \\ inf(x) &\to x : inf(s(x)) \\ take(0, x) &\to nil \\ take(s(x), y : z) &\to y : take(x, z) \\ length(nil) &\to 0 \\ length(x : y) &\to s(length(l)) \end{aligned}$$

with $\mu(s) = \mu(:) = \mu(===) = \varnothing$, $\mu(inf) = \mu(length) = \{1\}$ and $\mu(take) = \{1, 2\}$. The CSRS $(\mathcal{R}, \mu)$ corresponds to the OBJ3 program of Figure 1 (with *cons* replaced by : here; see [27] for further details about this correspondence). Then, we have:

$$take(s(0), \underline{inf(0)}) \hookrightarrow take(s(0), 0 : inf(s(0)))$$

Since $2.2 \notin \mathcal{P}os^{\mu}(take(s(0),0:inf(s(0))))$, the redex *inf(s(0))* cannot be μ-rewritten.

The $\hookrightarrow$-normal forms are called $(\mathcal{R}, \mu)$-normal forms. A CSRS $(\mathcal{R}, \mu)$ is terminating (resp. locally confluent, confluent, complete) if $\hookrightarrow_{\mu}$ is terminating (resp. locally confluent, confluent, complete). Slightly abusing terminology, we shall also sometimes say that the TRS $\mathcal{R}$ is $\mu-$terminating if the CSRS $(\mathcal{R}, \mu)$ is terminating. With *innermost* CSR, $\hookrightarrow_i$, we only contract *maximal* positions of replacing redexes: $t \hookrightarrow_i s$ if $t \xrightarrow{p}_{\mathcal{R}} s$ and $p \in maximal_{\leq}(\mathcal{P}os^{\mu}_{\mathcal{R}}(t))$. We say that $(\mathcal{R}, \mu)$ is *innermost* terminating if $\hookrightarrow_i$ is terminating.

3. MODULAR PROOFS OF TERMINATION OF CSR BY TRANSFORMATION

Subsequently we assume some basic familiarity with the usual notions, notations and terminology in modularity of term rewriting (cf. e.g. [45], [33], [17], [38, 39][4]). We say that some property of CSRS's is modular (under disjoint, constructor sharing, composable unions) if, whenever two (resp. disjoint, constructor sharing, composable) CSRS's $(\mathcal{R}_1, \mu_1)$ and $(\mathcal{R}_2, \mu_2)$ have the property, then the union $(\mathcal{R}_1 \cup \mathcal{R}_2, \mu_1 \cup \mu_2)$ also enjoys this property.[5] Two CSRS's $(\mathcal{R}_1, \mu_1)$ and $(\mathcal{R}_2, \mu_2)$ are *disjoint* if the signatures of $\mathcal{R}_1$ and $\mathcal{R}_2$ are disjoint. CSRS's $(\mathcal{R}_1, \mu_1)$ and $(\mathcal{R}_2, \mu_2)$ are *constructor sharing* if they only share constructor symbols (see Definition 7 for details). Finally, CSRS's $(\mathcal{R}_1, \mu_1)$ and $(\mathcal{R}_2, \mu_2)$ are *composable* if they share defined symbols provided that they share all of their defining rules, too (cf. [39]). Note that disjoint TRS's are sharing constructor, and sharing constructor TRS's are composable.

Termination of CSRS's $(\mathcal{R}, \mu)$ is usually proved by demonstrating *termination* of a transformed TRS $\mathcal{R}^{\mu}_{\Theta}$ obtained from $\mathcal{R}$ and μ by using a transformation[6] Θ [8, 12, 13, 25, 43, 47]. The simplest (and trivial) correct transformation for proving termination of CSRS's is the identity: if $\mathcal{R}^{\mu}_{ID} = \mathcal{R}$ is terminating, then $(\mathcal{R}, \mu)$ is terminating for every replacement map μ.

When considering the interaction between modularity and transformations for proving termination of a CSRS $(\mathcal{R}, \mu)$, we can imagine two main scenarios:

- *First modularize, next transform* (M&T): first, given $(\mathcal{R}, \mu)$, we look for (or have) some decomposition $\mathcal{R} = \mathcal{S} \cup \mathcal{T}$ that satisfies some standard criteria $\mathsf{M}(\mathcal{S}, \mathcal{T})$ for modularity (e.g., disjointness, constructor-sharing, composability, etc.). Then, we prove termination of both $\mathcal{S}^{\mu}_{\Theta}$ and $\mathcal{T}^{\mu}_{\Theta}$ (for the *same* transformation Θ).

- *First transform, next modularize* (T&M): first, we transform $(\mathcal{R}, \mu)$ into $\mathcal{R}^{\mu}_{\Theta}$; then, we look for a suitable decomposition $\mathcal{R}^{\mu}_{\Theta} = \mathcal{S} \cup \mathcal{T}$ such that termination of $\mathcal{S}$ and $\mathcal{T}$ ensures termination of $\mathcal{R}^{\mu}_{\Theta}$ (hence, that of $(\mathcal{R}, \mu)$).

The second approach (T&M) is actually a standard problem of (being able to achieve) a modular proof of termination of $\mathcal{R}^{\mu}_{\Theta}$.

The first approach (M&T) succeeds if we have both:

1. $(\mathcal{S} \cup \mathcal{T})^{\mu}_{\Theta} \subseteq \mathcal{S}^{\mu}_{\Theta} \cup \mathcal{T}^{\mu}_{\Theta}$ (in this way, termination of $\mathcal{S}^{\mu}_{\Theta} \cup \mathcal{T}^{\mu}_{\Theta}$ implies termination of $(\mathcal{S} \cup \mathcal{T})^{\mu}_{\Theta}$ which entails termination of $(\mathcal{S} \cup \mathcal{T}, \mu)$), and

2. The transformation is 'compatible' with M, i.e., $\mathsf{M}(\mathcal{S}, \mathcal{T})$ implies $\mathsf{M}'(\mathcal{S}^{\mu}_{\Theta}, \mathcal{T}^{\mu}_{\Theta})$ for some (possibly the same) modularity criterion M' (in this way, termination of $\mathcal{S}^{\mu}_{\Theta}$ and $\mathcal{T}^{\mu}_{\Theta}$ would imply termination of $\mathcal{S}^{\mu}_{\Theta} \cup \mathcal{T}^{\mu}_{\Theta}$).

[4] Further relevant works on modularity not mentioned elsewhere include (this list is highly incomplete): [22, 23, 24], [20], [21], [34, 35, 36], [42], [46]

[5] Typically, the inverse implication is trivial.

[6] See `http://www.dsic.upv.es/users/elp/slucas/muterm` for a tool, MU-TERM 1.0, that implements these transformations.

Indeed, the first requirement $(\mathcal{S} \cup \mathcal{T})^\mu_\Theta \subseteq \mathcal{S}^\mu_\Theta \cup \mathcal{T}^\mu_\Theta$ is satisfied by the transformations reported in the literature, as they have, in fact, a 'modular' definition (based on each individual rule or symbol in the signature disregarding any 'interaction'), i.e., we actually have $(\mathcal{S} \cup \mathcal{T})^\mu_\Theta = \mathcal{S}^\mu_\Theta \cup \mathcal{T}^\mu_\Theta$ for all these transformations. On the other hand, the second requirement is not fulfilled by many of these transformations (in general).

According to this, we review the main (nontrivial) correct transformations for proving termination of CSR regarding their suitability for modular proofs of termination.

3.1 The Contractive Transformation

Let $\mathcal{F}$ be a signature and $\mu \in M_\mathcal{F}$ be a replacement map. With the contractive transformation [25], the non-μ-replacing arguments of all symbols in $\mathcal{F}$ are *removed* and a new, *μ-contracted* signature $\mathcal{F}^\mu_L$ is obtained (possibly reducing the *arity* of symbols). The function $\tau_\mu : \mathcal{T}(\mathcal{F}, \mathcal{X}) \to \mathcal{T}(\mathcal{F}^\mu_L, \mathcal{X})$ *drops* the non-replacing immediate subterms of a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and constructs a 'μ-contracted' term by joining the (also transformed) replacing arguments below the corresponding operator of $\mathcal{F}^\mu_L$. A CSRS $(\mathcal{R}, \mu)$, where $\mathcal{R} = (\mathcal{F}, R)$ is μ-contracted into $\mathcal{R}^\mu_L = (\mathcal{F}^\mu_L, \{\tau_\mu(l) \to \tau_\mu(r) \mid l \to r \in R\})$.

Example 2. Consider the CSRS $(\mathcal{R}, \mu)$ of Example 1. Then, $\mathcal{R}^\mu_L$ is:

$$\begin{aligned} ===(x,y) &\to true(x,y)\\ ===(x,y) &\to ===(x,y)\\ ===(x,y) &\to false(x,y)\\ inf(x) &\to :(x, inf(s(x)))\\ take(0,x) &\to nil\\ take(s(x),:(x,y)) &\to :(x, take(x,y))\\ length(nil) &\to 0\\ length(:(x,y)) &\to s(length(y)) \end{aligned}$$

According to this definition, it is not difficult to see that, given TRS's $\mathcal{S}$ and $\mathcal{T}$, $(\mathcal{S} \cup \mathcal{T})^\mu_L = \mathcal{S}^\mu_L \cup \mathcal{T}^\mu_L$. It is also clear that $\mathsf{M}(\mathcal{S}, \mathcal{T})$ implies $\mathsf{M}(\mathcal{S}^\mu_L, \mathcal{T}^\mu_L)$ for $\mathsf{M} \in \{\mathsf{Disj}, \mathsf{ShCons}, \mathsf{Compos}\}$, i.e., the usual criteria for modularity are preserved (as are) by the transformation.

3.2 Zantema's Transformation

Zantema's transformation *marks* the *non-replacing arguments* of function symbols (disregarding their positions within the term) [47]. Given $\mathcal{R} = (\mathcal{F}, R)$ and $\mu \in M_\mathcal{F}$, $\mathcal{R}^\mu_Z = (\mathcal{F} \cup \mathcal{F}' \cup \{activate\}, R^\mu_Z)$ where R^μ_Z consists of two parts. The first part results from R by replacing every function symbol f occurring in a left or right-hand side with f' (a fresh function symbol of the same arity as f which, then, is included in $\mathcal{F}'$) if it occurs in a non-replacing *argument* of the function symbol directly above it. These new function symbols are used to block further reductions at this position. In addition, if a variable x occurs in a non-replacing position in the *lhs* l of a rewrite rule $l \to r$, then all occurrences of x in r are replaced by *activate(x)*. Here, *activate* is a new unary function symbol which is used to activate blocked function symbols again.

The second part of R^μ_Z consists of rewrite rules that are needed for blocking and unblocking function symbols:

$$\begin{aligned} f(x_1, \ldots, x_k) &\to f'(x_1, \ldots, x_k)\\ activate(f'(x_1, \ldots, x_k)) &\to f(x_1, \ldots, x_k) \end{aligned}$$

for every $f' \in \mathcal{F}'$, together with the rule $activate(x) \to x$ The problem is that *activate* is a new defined symbol having a defining rule for each new 'blocked' symbol appearing in the signature. This means that, starting from composable modules $\mathcal{S}$ and $\mathcal{T}$, $\mathcal{S}^\mu_Z$ and $\mathcal{T}^\mu_Z$ are composable only if blocked symbols in both $\mathcal{S}$ and $\mathcal{T}$ are *the same*.

Example 3. Consider the TRS's:

$$\mathcal{S}: \quad inf(x) \to x : inf(s(x))$$

and

$$\mathcal{T}: \quad \begin{aligned} 0 === 0 &\to true\\ s(x) === s(y) &\to x === y\\ x === y &\to false \end{aligned}$$

that correspond to modules `INF` and `ID-NAT` in Figure 1. Viewed as modules, they are sharing constructor. Let μ be as in Example 1. Now we have that

$$\mathcal{S}^\mu_Z: \quad \begin{aligned} inf(x) &\to x : inf'(s(x))\\ inf(x) &\to inf'(x)\\ activate(inf'(x)) &\to inf(x)\\ activate(x) &\to x \end{aligned}$$

and

$$\mathcal{T}^\mu_Z: \quad \begin{aligned} 0' === 0' &\to true\\ s'(x) === s'(y) &\to activate(x) === activate(y)\\ x === y &\to false\\ 0 &\to 0'\\ s(x) &\to s'(x)\\ activate(0') &\to 0\\ activate(s'(x)) &\to s(x)\\ activate(x) &\to x \end{aligned}$$

are not composable. The problem is that $\mathcal{S}^\mu_Z$ has a blocked symbol *inf'* which is not present in $\mathcal{T}^\mu_Z$.

Note that, since the rule $activate(x) \to x$ is present in every transformed system, composability is the best that we can achieve after applying the transformation. For instance, disjoint TRS's $\mathcal{S}$ and $\mathcal{T}$ lose disjointness after applying the transformation.

In [8], Ferreira and Ribeiro propose a variant of Zantema's transformation which has been proved strictly more powerful than Zantema's one (see [13]). This transformation has the same problems regarding modularity.

3.3 Giesl and Middeldorp's Transformations

Giesl and Middeldorp introduced a transformation that explicitly *marks* the replacing positions of a term (by using a new symbol *active*). Given a TRS $\mathcal{R} = (\mathcal{F}, R)$ and $\mu \in M_\mathcal{F}$, the TRS $\mathcal{R}^\mu_{GM} = (\mathcal{F} \cup \{active, mark\}, R^\mu_{GM})$ consists of the following rules (for all $l \to r \in R$ and $f \in \mathcal{F}$):

$$\begin{aligned} active(l) &\to mark(r)\\ mark(f(x_1, \ldots, x_k)) &\to active(f([x_1]_f, \ldots, [x_k]_f))\\ active(x) &\to x \end{aligned}$$

where $[x_i]_f = mark(x_i)$ if $i \in \mu(f)$ and $[x_i]_f = x_i$ otherwise [12].

Unfortunately, unless $\mathcal{R} = \mathcal{S}$, this transformation will never yield a pair of composable TRS's. Note that two different composable systems $\mathcal{R}$ and $\mathcal{S}$ cannot share *all* symbols: if they have the same defined symbols (i.e., $\mathcal{D}_\mathcal{R} = \mathcal{D}_\mathcal{S}$), then all rules must coincide too (up to renaming of variables).

Hence $\mathcal{R} \neq \mathcal{S}$ implies that they differ (at least) in a constructor symbol. However, if, e.g., $f \in \mathcal{F}_\mathcal{R} - \mathcal{F}_\mathcal{S}$, then a new rule $mark(f(x_1, \ldots, x_k)) \to active(f([x_1]_f, \ldots, [x_k]_f))$ is in $\mathcal{R}^\mu_{GM}$ but not in $\mathcal{S}^\mu_{GM}$. Since *mark* is a defined symbol, $\mathcal{R}^\mu_{GM}$ and $\mathcal{S}^\mu_{GM}$ are not composable. Thus, we have proven the following:

THEOREM 1. *Let $(\mathcal{R}, \mu)$ and $(\mathcal{S}, \mu)$ be different composable CSRS's. Then, $\mathcal{R}^\mu_{GM}$ and $\mathcal{S}^\mu_{GM}$ are not composable.*

Note that, since disjoint TRS's are sharing constructor; and sharing constructor TRS's are composable, it follows that Giesl and Middeldorp's transformation does not provide any possibility for a M&T-analysis of termination of CSRS's (at least regarding the modularity criteria considered here).

In [12], Giesl and Middeldorp suggest a slightly different presentation $\mathcal{R}^\mu_{mGM}$ of the previous transformation. In this presentation, symbol *active* is not used anymore. However, since, regarding modularity, the conflicts are due to the use of symbol *mark*, this new transformation has exactly the same problem.

Giesl and Middeldorp also introduced a transformation which is *complete*, i.e., every terminating CSRS $(\mathcal{R}, \mu)$ is transformed into a terminating TRS $\mathcal{R}^\mu_C$ [12].

Given a TRS $\mathcal{R} = (\mathcal{F}, R)$ and a replacement map μ, the TRS $\mathcal{R}^\mu_C = (\mathcal{F} \cup \{f' \mid f \in \mathcal{F} \wedge ar(f) > 0\} \cup \{active, mark, ok, proper, top\}, R^\mu_C)$ consists of the following rules (see [12] for a more detailed explanation): for all $l \to r \in R$, $f \in \mathcal{F}$ such that $k = ar(f) > 0$, $i \in \mu(f)$, and constants $c \in \mathcal{F}$,

$$\begin{array}{rcl}
active(l) & \to & mark(r) \\
active(f(x_1, \ldots, x_i, \ldots, x_k)) & \to & f'(x_1, \ldots, active(x_i), \ldots, x_k) \\
f'(x_1, \ldots, mark(x_i), \ldots, x_k) & \to & mark(f(x_1, \ldots, x_i, \ldots, x_k)) \\
proper(c) & \to & ok(c) \\
proper(f(x_1, \ldots, x_k)) & \to & f(proper(x_1), \ldots, proper(x_k)) \\
f(ok(x_1), \ldots, ok(x_k)) & \to & ok(f(x_1, \ldots, x_k)) \\
top(mark(x)) & \to & top(proper(x)) \\
top(ok(x)) & \to & top(active(x))
\end{array}$$

Unfortunately, it is not difficult to see that, regarding a M&T-modular analysis of termination (and due to the rules defining *proper*), we have the following.

THEOREM 2. *Let $(\mathcal{R}, \mu)$ and $(\mathcal{S}, \mu)$ be different composable CSRS's. Then, $\mathcal{R}^\mu_C$ and $\mathcal{S}^\mu_C$ are not composable.*

3.4 A Non-Transformational Approach to Modularity?

The previous discussion shows that only the contractive transformation seems to be a suitable choice for performing a modular analysis of termination of CSRS's. However, consider again the OBJ program of Figure 1 (represented by the CSRS $(\mathcal{R}, \mu)$ of Example 1). Note that a direct proof of termination of $(\mathcal{R}, \mu)$ is not possible with the contractive transformation (as shown in Example 2, $\mathcal{R}^\mu_L$ is not terminating). Of course, in this setting, modularity is not useful either. On the other hand, we note that $\mathcal{S}^\mu_Z$ (resp. $\mathcal{T}^\mu_Z$) in Example 3 is *not* kbo-terminating (resp. rpo-terminating). Therefore, $\mathcal{R}^\mu_Z$ (which contains both $\mathcal{S}^\mu_Z$ and $\mathcal{T}^\mu_Z$) is not either kbo- or rpo-terminating. Moreover, our attempts to prove termination of $\mathcal{R}^\mu_Z$ by using C*i*ME failed for every considered combination (including techniques that potentially deal with non-simply terminating TRS's like the use of dependency pairs together with polynomial orderings) of proof criteria. Similarly, termination of $\mathcal{R}^\mu_{GM}$ or $\mathcal{R}^\mu_C$ cannot be proved either using *kbo* or *rpo* (see [2] for a formal justification of this claim). In fact, we could even prove that they are *not* simply terminating (see [31]). On the other hand, our results in this section shows that M&T or T&M approaches are not able to provide a (simpler) proof of termination of $(\mathcal{R}, \mu)$. Hence, termination of $(\mathcal{R}, \mu)$ remains difficult to (automatically) prove. The following section shows that this situation dramatically changes using a *direct* modular analysis of termination of CSR.

4. MODULAR TERMINATION OF CSR

In this main section we investigate whether, and if so, how known modularity results from ordinary term (cf. e.g. the early pioneering work of Toyama [45, 44] and later surveys on the state-of-the-art about modularity in rewriting like [33], [37], [19]) extend to context-sensitive rewriting. Since any TRS $\mathcal{R}$ can be viewed as the CSRS $(\mathcal{R}, \mu_\top)$, all modularity results about TRS's cover a very specific case of CSRS's, namely, with no replacement restrictions at all. Yet, the interesting case, of course, arises when there are proper replacement restrictions. In this paper we only concentrate on termination properties. First we study how to obtain criteria for the modularity of termination of CSRS's. Later on we'll also consider weak termination properties, which surprisingly may help to guarantee full termination (and confluence) under some conditions. Then we generalize the setting and consider also to some extent certain non-disjoint combinations. As for ordinary term rewriting, a deep understanding of the disjoint union case appears to be indispensable for properly treating non-disjoint unions. For that reason we mainly focus here on the case of disjoint unions. For practical purposes it is obvious, that non-disjoint combinations are much more interesting. Yet, the lessons learned from (the nowadays fairly well understood) modularity analysis in term rewriting suggest to be extremely careful with seemingly plausible conjectures and "obvious facts".

4.1 Modularity of Termination in Disjoint Unions

In this section, we investigate modularity of termination of disjoint unions of CSRS's. For simplicity, as a kind of global assumption we assume that all considered CSRS's are finite. Most of the results (but not all) do also hold for systems with arbitrarily many rules.

The very first positive results on modularity of termination, after the negative ones in [45, 44], were given by Rusinowitch [41], who showed that the absence of *collapsing* rules or the absence of *duplicating* rules suffice for the termination of the disjoint union of two terminating TRS's. Later, Middeldorp [32] refined and generalized this criterion by showing that it suffices that one system doesn't have any collapsing or duplicating rules. A careful inspection of the proofs actually shows that these results do also for CSRS's. Even more interestingly, there is an additional source for generalization. Consider e.g. the following variant of Toyama's basic counterexample.

Example 4. The systems

$$\mathcal{R}_1: \quad f(a,b,x) \to f(x,x,x) \quad \text{and} \quad \mathcal{R}_2: \quad \begin{array}{l} G(x,y) \to x \\ G(x,y) \to y \end{array}$$

with $\mu(f) = \{3\}$ are both terminating CSRS's, as well as their disjoint union (the latter is a consequence of Theorem 3 below). In Toyama's original version, where the union is

non-terminating, there are no restrictions, $\mathcal{R}_2$ is collapsing and $\mathcal{R}_1$ duplicating.

A careful inspection of these two CSRS's and of their interaction shows that the duplicating $\mathcal{R}$-rule is not a problem any more regarding non-termination, because the first two occurrences of x in the rhs of the $\mathcal{R}_1$-rule become blocked after applying the rule. In particular, any further extraction of subterms at these two positions, that is crucial for Toyama's counterexample to work, is prohibited by $\mu(f) = \{3\}$. This observation naturally leads to the conjecture that blocked/inactive variables in rhs's shouldn't count for *duplication*.

DEFINITION 1. *A rule $l \to r$ in a CSRS $(\mathcal{R}, \mu)$ is* non-duplicating *if for every $x \in \mathcal{V}ar(l)$ the multiset of replacing occurrences of x in r is contained in the multiset of replacing occurrences of x in l. $(\mathcal{R}, \mu)$ is* non-duplicating *if all its rules are.*

Of course, in order to sensibly combine two CSRS's, one should require some basic *compatibility* condition regarding the respective replacement restrictions.

DEFINITION 2. *Two CSRS's $(\mathcal{R}_1, \mu_1)$, $(\mathcal{R}_2, \mu_2)$ are said to be* compatible *if they have the same replacement restrictions for shared function symbols, i.e., if $\mathcal{R}_1 = (\mathcal{F}_1, R_1)$ and $\mathcal{R}_2 = (\mathcal{F}_2, R_2)$, we have $\mu_1(f) = \mu_2(f)$ for every $f \in \mathcal{F}_1 \cap \mathcal{F}_2$.*

Disjoint CSRS's are trivially compatible.

THEOREM 3. *Let $(\mathcal{R}_1, \mu_1)$, $(\mathcal{R}_2, \mu_2)$ be two disjoint, terminating CSRS's, and let $(\mathcal{R}, \mu)$ be their union. Then the following hold:*

(i) $(\mathcal{R}, \mu)$ terminates, if both $\mathcal{R}_1$ and $\mathcal{R}_2$ are non-collapsing.

(ii) $(\mathcal{R}, \mu)$ terminates, if both $\mathcal{R}_1$ and $\mathcal{R}_2$ are non-duplicating.

(iii) $(\mathcal{R}, \mu)$ terminates, if one of the systems is both non-collapsing and non-duplicating.

PROOF. We sketch the proof idea and point out the differences to the TRS case. All three properties follow immediately from the following observations. For any infinite $(\mathcal{R}, \mu)$-derivation $D : s_1 \hookrightarrow s_2 \hookrightarrow \ldots$ of minimal rank (i.e. in any minimal counterexample) we have:

(a) There are infinitely many outer reduction steps in D.

(b) There are infinitely many inner reduction steps in D which are destructive at level 2.

(c) There are infinitely many duplicating outer reduction steps in D.

(a) and (b) are proved as for TRS's, cf. e.g. the minimal counterexample approach of [17], and (c) is proved as in [38], but with a small adaptation: Instead of the well-founded measure there, that is shown to be decreasing, namely $\#s = [rank(t) \mid t \in S_2(s)]$ (the multiset of ranks of all special subterms of level 2), we only take $\#s = [rank(t) \mid t \in S_2(s), t \text{ active in } s]$, i.e., we only count special subterms at active positions. With this modification the proof goes through as before. □

With this result, we're able to explain termination of Example 4 without having to use any sophisticated proof method for the combined system.

In fact, in the case of TRS's, the above syntactical conditions (non-collapsingness and non-duplication) had turned out to be very special cases (more precisely, consequences) of an abstract structure theorem that characterizes minimal counterexamples (cf. [17][7]). For CSRS's this powerful result also holds. To show this, we first need another definition (where $\mathcal{R} \oplus \mathcal{S}$ mean $\mathcal{R} \cup \mathcal{S}$ provided that $\mathcal{R}$ and $\mathcal{S}$ are disjoint).

DEFINITION 3. *A TRS $\mathcal{R}$ is said to be* terminating under free projections*, FP-terminating for short, if $\mathcal{R} \oplus \{G(x, y) \to x, G(x, y) \to y\}$ is terminating.*[8] *A CSRS $(\mathcal{R}, \mu)$ is said to be FP-terminating, if $(\mathcal{R} \oplus \{G(x, y) \to x, G(x, y) \to y\}, \mu)$, where $\mu(G) = \{1, 2\}$, is terminating.*

THEOREM 4 (EXTENDS [17, THEOREM 7]). *Let $(\mathcal{R}_1, \mu_1)$, $(\mathcal{R}_2, \mu_2)$ be two disjoint, terminating CSRS's, such that their union $(\mathcal{R}, \mu)$ is non-terminating. Then one of the systems is not FP-terminating and the other system is collapsing.*

PROOF. The quite non-trivial proof is practically the same as for TRS's as a careful inspection of [17] reveals. All the abstracting constructions and the resulting transformation of a minimal counterexample in the disjoint union into a counterexample in one of the systems extended by the free projection rules for some fresh binary operator work as before. □

As already shown for TRS's, this abstract and powerful structure result has a lot of – more or less straightforward – direct and indirect consequences and corollaries. To mention only two:

COROLLARY 1. *Termination is modular for* non-deterministically collapsing[9] *disjoint CSRS's.*

COROLLARY 2. *FP-termination is modular for disjoint CSRS's.*

Next we will have a look at (the modularity of) weak termination properties.

[7] For the construction in [17] the involved TRS's must be *finitely branching*, which in practice is always satisfied. The case of *infinitely branching* systems is handled in [38] by a similar but more involved abstraction function, based on the same underlying idea of extracting all relevant information from deeper alien subterms.

[8] This important property was later called *$\mathcal{C}_\mathcal{E}$-termination* in [38] which however isn't really telling or natural. A slightly different property in [17] had been called *termination preserving under non-deterministic collapses* which is precise but rather lengthy. Here we prefer to use FP-termination, since it naturally expresses that a rewrite system under the addition of the projection rules for a *free* (i.e., fresh) function symbol (of arity ≥ 2) is still terminating.

[9] A CSRS is *non-deterministically collapsing* if there is a term that reduces to two distinct variables (in a finite number of context-sensitive rewrite steps).

4.2 Modularity of Weak Termination Properties

Weak termination properties are clearly interesting on their own, since full termination may be unrealistic or need not really correspond to the computational process being modelled. Certain processes or programs are inherently non-terminating. But still, one may wish to compute normal forms for certain inputs (and guarantee their existence).

On the other hand, interestingly, for TRS's it has turned out that weak termination properties can be very helpful in order to obtain in a *modular* way the full termination property under certain assumptions.

DEFINITION 4. *Let $(\mathcal{R},\mu)$ be a CSRS. $(\mathcal{R},\mu)$ (and $\hookrightarrow$) is said to be* weakly terminating (WN)*, if $\hookrightarrow$ is weakly terminating. $(\mathcal{R},\mu)$ (and $\hookrightarrow$) is* weakly innermost terminating (WIN) *if the innermost context-sensitive rewrite relation $\hookrightarrow_i$ is weakly terminating, and* (strongly) innermost terminating (SIN) *if the innermost context-sensitive rewrite relation $\hookrightarrow_i$ is (strongly) terminating.*

For ordinary TRS's it is well-known (and not difficult to prove) that weak termination, weak innermost termination and (strong) innermost termination are all modular properties (cf. e.g. [6], [17, 18]), w.r.t. disjoint unions. Surprisingly, this does not hold in general for CSRS's as shown by the following counterexample.

Example 5. Consider the disjoint CSRS's

$$\mathcal{R}_1:\ f(a,b,x)\to f(x,x,x) \quad\text{and}\quad \mathcal{R}_2:\ \begin{array}{l} G(x,y)\to x\\ G(x,y)\to y\end{array}$$

with $\mu(f)=\mu(G)=\{1,2\}$ are both innermost terminating, in fact even terminating, but their union $(\mathcal{R},\mu)$ is neither terminating nor innermost terminating. We have $f(a,b,G(a,b)) \hookrightarrow_i f(G(a,b),G(a,b),G(a,b)) \hookrightarrow_i f(a, G(a,b),G(a,b)) \hookrightarrow_i f(a,b,G(a,b)) \hookrightarrow_i \ldots$. Note, however, that $(\mathcal{R},\mu)$ is weakly innermost terminating, hence also weakly terminating: $f(a,b,G(a,b)) \hookrightarrow_i f(G(a,b),G(a,b), G(a,b)) \hookrightarrow_i f(a,G(a,b),G(a,b)) \hookrightarrow_i f(a,a,G(a,b))$, where the latter term is in $(\mathcal{R},\mu)$-normal form.

But even WIN and WN are not modular in general for CSRS's as we next illustrate by a modified version of Example 5.

Example 6. Consider the disjoint CSRS's

$$\mathcal{R}_1:\ \begin{array}{l} f(a,b,x)\to f(x,x,x)\\ f(b,a,x)\to f(x,x,x)\\ f(a,a,x)\to f(x,x,x)\\ f(b,b,x)\to f(x,x,x)\\ f(x,x,x)\to x\end{array} \quad\text{and}\quad \mathcal{R}_2:\ \begin{array}{l} G(x,y)\to x\\ G(x,y)\to y\end{array}$$

with $\mu(f)=\mu(G)=\{1,2\}$. $\mathcal{R}_1$ is WIN (and WN), but not SIN. $\mathcal{R}_2$ is WIN, WN, and even SN. But their union $(\mathcal{R},\mu)$ is neither WIN nor WN.

$$f(a,b,G(a,b)) \hookrightarrow_i f(G(a,b),G(a,b),G(a,b))$$

is the (only) first innermost $(\mathcal{R},\mu)$-step issuing from $f(a,b,G(a,b))$. Then we can innermost reduce the first argument of f and then the second one, or vice versa, but the subsequent innermost step must be using one of the first four $\mathcal{R}_1$-rules, hence necessarily yielding a cycle

$$f(a,b,G(a,b)) \hookrightarrow_i^+ f(a,b,G(a,b))\ .$$

Therefore, $f(a,b,G(a,b))$ doesn't have an innermost $(\mathcal{R},\mu)$-normal form, and also no $(\mathcal{R},\mu)$-normal form at all.

A careful inspection of what goes wrong here, as well as an inspection of the corresponding proofs for the context-free case shows that the problem comes from (innermost) redexes which, at some point, are blocked (inactive) because they are below some forbidden position, but become unblocked (active) again later on. The condition that is needed to prevent this phenomenon is the following:

DEFINITION 5 (CONSERVATIVELY BLOCKING). *A CSRS $(\mathcal{R},\mu)$ is said to be* conservatively blocking*, (CB for short), if the following holds: For every rule $l\to r\in\mathcal{R}$, for every variable x occurring in l at an inactive position, all occurrences of x in r are inactive, too.*[10]

Under this condition now, the properties WIN, WN, and SIN turn out to be indeed modular for CSRS's.

THEOREM 5 (MODUL. CRIT. FOR WIN, WN, SIN).

(a) WIN is modular for disjoint CSRS's satisfying CB.

(b) WN is modular for disjoint CSRS's satisfying CB.

(c) SIN is modular for disjoint CSRS's satisfying CB.

PROOF. (a), (b) and (c) are all proved by structural induction as in the case of TRS's, cf. e.g. [18, 19]. Condition CB ensures that the innermost term rewrite derivation that is constructed in these proofs in the induction step, is also still innermost, i.e., that the proof goes through for CSRS's as well. □

4.3 Relating Innermost Termination and Termination

For TRS's a powerful criterion is known under which SIN implies, hence is equivalent to, termination (SN). Namely, this equivalence holds for locally confluent *overlay TRS's* [18]. Via the modularity of SIN for TRS's this gave rise to immediate new modularity results for termination (and completeness) in the case of context-free rewriting (cf. [18]). Fortunately, the above equivalence criterion between SIN and SN also extends to CSRS's, but not directly. The non-trivial proof requires a careful analysis and a subtle additional assumption (which is vacuously satisfied for TRS's).[11]

DEFINITION 6. *A CSRS $(\mathcal{R},\mu)$ is said to be a* (context-sensitive) overlay system *or* overlay CSRS *or* overlaying*, if there are no critical pairs*[12] *$\langle\sigma(l_1)[\sigma(r_2)]_\pi,\sigma(r_1)\rangle$ such that π is an active non-root position in l_1. $(\mathcal{R},\mu)$ has* left-homogeneous replacing variables *(LHRV for short) if, for every replacing variable x in l, all occurrences of x in both l and r are replacing.* [13]

[10] Formally: For every rule $l\to r\in\mathcal{R}$, for every variable $x\in\mathcal{V}ar(l)$: If $\mathcal{P}os_x(l)\setminus\mathcal{P}os^\mu_x(l)\neq\varnothing$, then $\mathcal{P}os^\mu_x(r)=\varnothing$.

[11] A similar claim was recently made in [14] without proof, but it remains unclear whether this claim without our condition LHRV is true.

[12] Here $l_1\to r_1$ and $l_2\to r_2$ are rewrite rules having no variable in common, and π is a non-variable position in l_1 such that $l_1|\pi$ and l_2 are unifiable with most general unifier σ. Observe that by definition any overlay TRS is an overlay CSRS, but not vice versa (when considering a CSRS $(\mathcal{R},\mu)$ as TRS $\mathcal{R}$)!

[13] Formally: For every $l\to r\in(\mathcal{R},\mu)$, for every $x\in\mathcal{V}ar^\mu(l)$ we have $\mathcal{P}os^\mu_x(l)=\mathcal{P}os_x(l)$ and $\mathcal{P}os^\mu_x(r)=\mathcal{P}os_x(r)$.

THEOREM 6 (LOCAL COMPLETENESS CRITERION). *Let $(\mathcal{R},\mu)$ be a locally confluent overlay CSRS satisfying LHRV and let $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$. If t is innermost terminating, then t is terminating.*

PROOF. Since the proof uses essentially the same approach and construction as the one in [18] for TRS's, we only focus on the differences and problems arising due to context-sensitivity. Basically, the proof is by minimal counterexample. Suppose, there is an infinite $(\mathcal{R},\mu)$-derivation issuing from s. Then it is not difficult to see that one can construct an infinite *minimal* derivation of the following form (with $s_1 := s$):

$$D : s_1 \hookrightarrow^*_{>p_1} s_1' \hookrightarrow^*_{p_1} s_2 \hookrightarrow^*_{>p_1p_2} s_2' \hookrightarrow_{p_1p_2} s_3 \hookrightarrow^* \dots .$$

Here, $s_i|_{p_1p_2\dots p_i}$ is non-terminating, but all proper subterms are terminating, hence complete. Since reduction steps strictly below $p_1p_2\dots p_i$ in s_i are only finitely often possible, eventually there must again be a root reduction step after $s_i|_{p_1p_2\dots p_i}$: $s_i \hookrightarrow_{>p_1\dots p_i} s_i' \hookrightarrow_{p_1\dots p_i} s_{i+1}$. Now the idea is to transform the infinite derivation D into an infinite innermost derivation D' in such a way that the reduction steps $s_i' \hookrightarrow_{p_1\dots p_i} s_{i+1}$ (for $i \geq 1$) are still proper (and innermost) reduction steps using the same rules, whereas the other reductions $s_i \hookrightarrow^*_{>p_1\dots p_i} s_i'$ "disappear". Technically this was achieved for TRS's by a transformation Φ which (uniquely) normalizes all complete subterms of a given term, but doesn't touch the top parts of a given non-terminating term. Formally:[14]

$$\Phi(t) = C[t_1{\downarrow_{\mathcal{R}}}, \dots, t_n{\downarrow_{\mathcal{R}}}]$$

where $t = C[t_1,\dots,t_n]$ such that $t_1,\dots,t_n$ are all maximal terminating (hence complete) subterms of t. Now the crucial (and subtle) issue is to make sure that $\Phi(s_i) = \dots = \Phi(s_i')$ and to guarantee that the rule $l_i \to r_i$ applied in $s_i' \to_{p_1\dots p_i} s_{i+1}$ is still applicable to $\Phi(s_i')$ at the same position $p_1\dots p_i$, i.e., that the pattern of l_i is not destroyed by Φ. In the case of TRS's this was guaranteed by the overlay property combined with completeness. In the (more general) case of overlay CSRS's there may be (context-free) rewrite steps in the pattern of l (strictly below the root) which would invalidate this argument. To account for this problem, we slightly modify the definition of Φ as follows:

$$\Phi_\mu(t) = C[t_1{\downarrow_{(\mathcal{R},\mu)}}, \dots, t_n{\downarrow_{(\mathcal{R},\mu)}}]$$

where $t = C[t_1,\dots,t_n]$ such that $t_1,\dots,t_n$ are all maximal terminating (hence complete) subterms at active positions of t (i.e., maximal complete subterms at inactive positions of t are left untouched!). Now we are almost done. However, there is still a problem, namely with the "variable parts" of the lhs l_i of the rule $l_i \to r_i$. In the TRS case we did get $\Phi(\sigma_i(l_i)) = (\Phi(\sigma))(l_i)$, hence again a redex. Here, for CSRS's, we may have the problem, that (for non-left-linear rules) "synchronization of normalization within variable parts" becomes impossible, because e.g. one occurrence of x is active, while another one is inactive. Consequently, the resulting transformed term $\Phi_\mu(\sigma_i(l_i))$ would not be an instance of l_i any more, and $l_i \to r_i$ not applicable. To avoid this, we need the additional requirement LHRV, which also accounts for enabling "synchronization of non-linear variables". With these adaptations and modifications the proof finally goes through as in the TRS case. [15] □

Clearly, this stronger local version directly implies a global completeness criterion as corollary.

THEOREM 7 (GLOBAL COMPLETENESS CRITERION). *Let $(\mathcal{R},\mu)$ be a locally confluent overlay CSRS satisfying LHRV. If $(\mathcal{R},\mu)$ is innermost terminating, then it is also terminating (hence complete).*

4.4 Modularity of Completeness

Combining previous results, we get another new criterion for the modularity of termination of CSRS's, in fact also for the modularity of completeness.

THEOREM 8 (MODULARITY CRIT. FOR COMPLETENESS). *Let $(\mathcal{R}_1,\mu_1)$, $(\mathcal{R}_2,\mu_2)$ be two disjoint, terminating CSRS's satisfying LHRV and CB. Suppose both $(\mathcal{R}_1,\mu_1)$ and $(\mathcal{R}_2,\mu_2)$ are locally confluent and overlaying. Then their disjoint union $(\mathcal{R},\mu)$ is also overlaying, terminating and confluent, hence complete.*

PROOF. Termination of $(\mathcal{R}_i,\mu_i)$ clearly implies innermost termination of $(\mathcal{R}_i,\mu_i)$, for $i = 1,2$. Theorem 5 thus yields innermost termination of $(\mathcal{R},\mu)$. $(\mathcal{R},\mu)$ is an overlay CSRS, too, by definition of this notion. Similarly, LHRV and CB also hold for $(\mathcal{R},\mu)$, since these syntactical properties are purely rule-based. Now, the only assumption missing, that we need to apply Theorem 7, is local confluence. But this is indeed guaranteed by the *critical pair lemma* of [26, Theorem 4, pp. 25] for CSRS's (which in turn crucially relies on the condition LHRV). Hence, applying Theorem 7 yields termination of $(\mathcal{R},\mu)$, which together with local confluence shows (via Newman's Lemma) confluence and completeness. □

4.5 Extensions to the Constructor-Sharing Case

As in the case of TRS's there is justified hope that many modularity results that hold for disjoint unions can also be extended to more general combinations. The natural next step are *constructor sharing* unions. Here we will concentrate on the case of (at most) constructor sharing CSRS's. The slightly more general setting of unions of *composable* CSRS's is beyond the scope of the present paper and will only be touched. But we expect our approach and analysis also to be applicable to this setting (which, already for TRS's, is technically rather complicated).

DEFINITION 7. *For a CSRS $(\mathcal{R},\mu)$, where $\mathcal{R} = (\mathcal{F},R)$, the set of* defined (function) symbols *is $\mathcal{D} = \{root(l) \mid l \to r \in R\}$, its set of* constructors *is $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$ (thus $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$). Let $(\mathcal{R}_1,\mu_1)$, $(\mathcal{R}_2,\mu_2)$ be CSRS's with $\mathcal{F}_1,\mathcal{F}_2$; $\mathcal{C}_1$,*

[14] Note that normalization is performed with $\hookrightarrow$ and not with $\to$.

[15] Currently, we have no counterexample to the statement of Theorem 6 without the LHRV assumption. But the current proof doesn't seem to work without it. It remains to be investigated whether imposing linearity restrictions would help. Observe also, that the LHRV property plays a crucial role in known (local and global) confluence criteria for CSRS's. Very little is known about how to prove (local) confluence of CSRS's without LHRV, cf. [26].

$\mathcal{C}_2$, and $\mathcal{D}_1$, $\mathcal{D}_2$ denoting their respective signatures, sets of constructors, and defined function symbols. Then $(\mathcal{R}_1, \mu_1)$ and $(\mathcal{R}_2, \mu_1)$ are said to be (at most) constructor sharing *if $\mathcal{D}_1 \cap \mathcal{F}_2 = \mathcal{D}_2 \cap \mathcal{F}_1 = \varnothing$. The set of* shared constructors *between them is $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2$. A rule $l \to r \in \mathcal{R}_i$ is said to be* (shared) constructor lifting *if $root(r) \in \mathcal{C}$, for $i = 1, 2$. $\mathcal{R}_i$ is said to be* (shared) constructor lifting *if it has a constructor lifting rule ($i = 1, 2$). A rule $l \to r \in \mathcal{R}_i$ is said to be* shared symbol lifting *if $root(r)$ is a variable or a shared constructor. $\mathcal{R}_i$ is said to be* shared symbol lifting *if it is collapsing or has a constructor lifting rule. $\mathcal{R}_i$ is* layer preserving *if it is not shared symbol lifting.*

For TRS's the main problems in disjoint unions arise from the additional interference between the two systems in "mixed" terms. This interference stems from (a) non-left-linearity, and (b) from rewrite steps that destroy the "layered structure" of mixed terms thereby potentially enabling new rewrite steps that have not been possible before. Now, (a) is usually not a severe problem and can be dealt with by synchronizing steps. However, (b) is a serious issue and the main source of (almost) all problems. In disjoint unions such "destructive" steps are only possible via collapsing rules (cf. e.g. Theorems 3, 4). In constructor sharing unions, interference and fusion of previously separated layers is also possible via (shared) constructor lifting rules. The basic example in term rewriting is the following variant of Toyama's counterexample.

Example 7. The two constructor sharing TRS's

$$\mathcal{R}_1: \quad f(a,b,x) \to f(x,x,x) \quad \text{and} \qquad \mathcal{R}_2: \quad \begin{array}{l} c \to a \\ c \to b \end{array}$$

are terminating, but their union admits a cycle

$$f(a,b,c) \to f(c,c,c) \to f(a,c,c) \to f(a,b,c)\ .$$

Observe how the application of the two constructor lifting rules enables the application of the $\mathcal{R}_1$-rule previously not possible.

Taking this additional source of interference into account, namely, besides collapsing rules also constructor lifting rules, some results for disjoint unions also extend to the constructor sharing case.

First let us look at an illuminating example.

Example 8. Consider the two constructor sharing CSRS's

$$\mathcal{R}_1: \quad f(c(x,y)) \to f(y) \qquad \text{and} \quad \mathcal{R}_2: \quad g(x) \to c(x, g(x))$$

with shared constructor c and $\mu(c) = \mu(f) = \mu(g) = \{1\}$. Both systems are obviously terminating, but their union admits a cycle

$$f(g(x)) \hookrightarrow f(c(x, g(x))) \hookrightarrow f(g(x))\ .$$

Observe that the CSRS $(\mathcal{R}_1, \mu)$ is not shared symbol lifting and non-duplicating as TRS but duplicating (as CSRS), whereas $(\mathcal{R}_2, \mu)$ is constructor lifting and non-duplicating (as CSRS).

For the next general result we need an additional definition.

Definition 8. *Let $((\mathcal{F}, R, \mathcal{F}), \mu)$ be a CSRS and $f \in \mathcal{F}$. We say that f is* fully replacing *if $\mu(f) = \{1, \ldots, n\}$ where n is the arity of f.*

Now we are ready to generalize Theorem 4 to the constructor sharing case (cf. [17, Theorem 34]).

Theorem 9 (Theorem 4 extended).
Let $(\mathcal{R}_1, \mu_1)$, $(\mathcal{R}_2, \mu_2)$ be two constructor sharing, compatible, terminating CSRS's with all shared constructors fully replacing, such that their union $(\mathcal{R}, \mu)$ is non-terminating. Then one of the systems is not FP-terminating and the other system is shared symbol lifting (i.e., collapsing or constructor lifting).[16]

Proof. We just sketch the proof idea. The proof is very similar to the one in the TRS case, i.e., as for Theorem 34 in [17]. Given a minimal counterexample in the union, i.e., an infinite (ground) derivation D of minimal *rank*, let's say, with the *top layer* from $\mathcal{F}_1$, an *abstracting transformation* Ψ is defined, which abstracts from the concrete syntactical from of *inner* parts of the terms but retains all relevant syntactical $\mathcal{F}_1$-information that may eventually pop up and fuse with the topmost $\mathcal{F}_1$-layer. The only difference is that in the recursive definition of this abstraction function Ψ we use $\hookrightarrow_{\mathcal{R}}$ instead of $\to$ as in [17]. The abstracted $\mathcal{F}_1$-information is collected and brought into a unique syntactical form via a fresh binary function symbol G with $\mu(G) = \{1, 2\}$ (together with a fresh constant A). With these preparations it is not very difficult to show:

(a) D contains infinitely many outer reduction steps.

(b) Any outer step in D translates into a corresponding outer ($(\mathcal{R}_1, \mu_1)$-)step in $\Psi(D)$ (using the same rule at the same position).

(c) D contains infinitely many inner reduction steps that are destructive at level 2 (hence must be $(\mathcal{R}_2, \mu_2)$-steps).

(d) Any inner step in D which is destructive at level 2 translates into a (non-empty) sequence of rewrite steps in $\Psi(D)$ using (only) the projection rules for G, i.e., $\{G(x,y) \to x, G(x,y) \to y\}$.

(e) Any inner step which is not destructive at level 2 (hence it can be an $(\mathcal{R}_1, \mu_1)$- or an $(\mathcal{R}_2, \mu_2)$-step) translates into a (possibly empty) sequence of rewrite steps in $\Psi(D)$ using $(\mathcal{R}_1 \cup \{G(x,y) \to x, G(x,y) \to y\}, \mu)$.

Observe that without the assumption that all shared constructors are fully replacing, the above properties (b), (d) and (e) need not hold any more in general.[17] Now, (c) implies that $(\mathcal{R}_2, \mu_2)$ is shared lifting. And from (a), (b), (d) and (e) we obtain the infinite $(\mathcal{R}_1 \cup \{G(x,y) \to x, G(x,y) \to y\}, \mu)$-derivation $\Psi(D)$ which means that $(\mathcal{R}_1, \mu_1)$ is not FP-terminating. □

[16] Note that, as in the case of TRS's, this result holds not only for finite CSRS's, but also for finitely branching ones. But, in contrast to the disjoint union case, it doesn't hold any more for infinitely branching systems, cf. [38] for a corresponding counterexample of infinitely branching constructor sharing TRS's.

[17] Roughly speaking, this failure is due to the fact that, for non-fully replacing constructors, context-sensitivity makes the abstracting transformation interfere with reduction steps in a "non-monotonic" way.

Without the above assumption the statement of the Theorem does not hold in general as is witnesses by Example 8 above. Clearly, both CSRS's $\mathcal{R}_1$ and $\mathcal{R}_2$ in this example are FP-terminating, but their union is not even terminating. Note that the (only) shared constructor c here is **not** fully replacing.

Next let us consider the extension of the syntactical modularity criteria of Theorem 3 to constructor sharing unions.

THEOREM 10. *Let $(\mathcal{R}_1, \mu_1)$, $(\mathcal{R}_2, \mu_2)$ be two constructor sharing, compatible, terminating CSRS's, and let $(\mathcal{R}, \mu)$ be their union. Then the following hold:*

(i) $(\mathcal{R}, \mu)$ terminates, if both $\mathcal{R}_1$ and $\mathcal{R}_2$ are layer-preserving.

(ii) $(\mathcal{R}, \mu)$ terminates, if both $\mathcal{R}_1$ and $\mathcal{R}_2$ are non-duplicating.

(iii) $(\mathcal{R}, \mu)$ terminates, if one of the systems is both layer-preserving and non-duplicating.

PROOF. The proof is essentially analogous to the one of Theorem 3 one for disjoint CSRS's. □

Example 9. Now we are ready to give a modular proof of termination of the OBJ program of Figure 1: consider the CSRS $(\mathcal{R}, \mu)$ of Example 1 as the (constructor sharing, compatible) union of:

$$(\mathcal{S}, \mu): \quad \begin{array}{r} 0 === 0 \to true \\ s(x) === s(y) \to x === y \\ x === y \to false \end{array}$$

and

$$(\mathcal{T}, \mu): \quad \begin{array}{r} inf(x) \to x : inf(s(x)) \\ take(0, x) \to nil \\ take(s(x), y : z) \to y : take(x, z) \\ length(nil) \to 0 \\ length(x : y) \to s(length(l)) \end{array}$$

Note that $\mathcal{S}$ is *rpo*-terminating (use the precedence $=== > true, false$). Hence, $(\mathcal{S}, \mu)$ is terminating. On the other hand:

$$\mathcal{T}_L^\mu: \quad \begin{array}{r} inf(x) \to : \\ take(0, x) \to nil \\ take(s, :) \to : \\ length(nil) \to 0 \\ length(:) \to s \end{array}$$

is *rpo*-terminating too: use precedence $take > nil, :$; $inf > :$, and $length > 0, s$. Hence, $(\mathcal{T}, \mu)$ is terminating. Also, polynomial termination[18] of $\mathcal{S}$ and $\mathcal{T}_L^\mu$ can easily be proved by using the C*i*ME 2.0 system. Since $(\mathcal{S}, \mu)$ is layer-preserving and non-duplicating, by Theorem 10 we conclude termination of $(\mathcal{R}, \mu)$. According to [27], this implies termination of the OBJ program.

Example 10. Consider the two constructor sharing TRS's

$$\mathcal{R}_1: \quad f(a, b, x) \to f(x, x, x) \quad \text{and} \qquad \mathcal{R}_2: \quad \begin{array}{l} c \to a \\ c \to b \end{array}$$

[18] i.e., termination by using some well-founded polynomial ordering

together with $\mu(f) = \{3\}$. In [12] Giesl and Middeldorp show that termination of $(\mathcal{R}_1 \cup \mathcal{R}_2, \mu)$ cannot be proved by any existing transformation (but the complete one, see Section 3.3). However, no proof of termination of $(\mathcal{R}_1 \cup \mathcal{R}_2, \mu)$ has been reported in the literature yet. Now, we are able to give a very simple (modular) proof: note that $\mathcal{R}_1$ and $\mathcal{R}_2$ are terminating, hence $(\mathcal{R}_1, \mu)$ and $(\mathcal{R}_2, \mu)$ are terminating. Since $(\mathcal{R}_1, \mu)$ is layer-preserving and non-duplicating, termination of $(\mathcal{R}_1 \cup \mathcal{R}_2, \mu)$ follows by Theorem 10.

As for TRS's, Theorem 9 has a whole number of direct or indirect corollaries stating more concrete modularity criteria for termination in the case of constructor sharing unions. We will not detail this here, but rather focus on some other results along the lines of Sections 4.2 and 4.4.

Of course, the negative counterexamples of Section 4.2, Examples 5 and 6, immediately extend to the constructor sharing case, too. Yet, the positive results regarding modularity of the "weak termination properties" WIN, WN, and SIN extend from disjoint CSRS's to constructor sharing ones.

THEOREM 11 (THEOREM 5 EXTENDED).

(a) WIN is preserved under unions of constructor sharing CSRS's satisfying CB.

(b) WN is preserved under unions of constructor sharing CSRS's satisfying CB.

(c) SIN is preserved under unions of constructor sharing CSRS's satisfying CB.

PROOF. The proofs of (a), (b) and (c) are essentially the same as for Theorem 5 above, namely by structural induction and case analysis. The only difference now is that in the case of a shared constructor at the root, we may have both *top-white* and *top-black principal subterms* below (in the common modularity terminology, cf. e.g. [17]) such a constructor symbol at the root. But this doesn't disturb the reasoning in the proofs. Again condition CB ensures that the innermost term rewrite derivation that is constructed in these proofs in the induction step, is also still innermost, i.e., that the proofs go through for CSRS's as well. □

Similarly we obtain

THEOREM 12 (THEOREM 8 EXTENDED).
Let $(\mathcal{R}_1, \mu_1)$, $(\mathcal{R}_2, \mu_2)$ be two constructor sharing, compatible, terminating CSRS's satisfying LHRV and CB. Suppose both $(\mathcal{R}_1, \mu_1)$ and $(\mathcal{R}_2, \mu_2)$ are locally confluent and overlaying. Then their (constructor sharing) union $(\mathcal{R}, \mu)$ is also overlaying, terminating and confluent, hence complete.

PROOF. Analogous to the proof of Theorem 8 using Theorem 11 instead of Theorem 5. □

5. RELATED WORK

As far as we know our results are the first to deal with the analysis of modular properties of CSRS's. Some properties of CSRS's have by now been fairly well investigated, especially regarding termination proof techniques, but also concerning other properties and verification criteria (cf. e.g. [25, 26, 28, 27, 29, 31, 30], [47], [8], [12, 13, 15, 14], [2]).

Recent interesting developments include in particular the approach of Giesl & Middeldorp for proving innermost termination of CSRS's via transformations to ordinary TRS's along the lines of [12], as well as the rpo-style approach of [2] for directly proving termination of CSRS's without any intermediate transformations and without recurring to ordinary TRS's. A comparison of our results and approach with the latter ones mentioned remains to be done.

5.1 Perspectives and Open Problems

In this paper we have started to systematically investigate modular aspects of context-sensitive rewriting. We have almost exclusively focussed on termination (properties). Of course, this is only the beginning of more research to be done. We have shown that, taking the additional complications arising from context-sensitivity carefully into account, it is indeed possible to extend a couple of fundamental modularity results for TRS's to the more general case of CSRS's. In this sense, the obtained results are quite encouraging. They also seem to indicate that a considerable amount of the structural knowledge about modularity in term rewriting can be taken over to context-sensitive rewriting. However, it has also turned that there are a couple of new phenomena and ugly properties that crucially interfere with the traditional approach for TRS's. In particular, it turns out that the syntactical restrictions CB and LHRV of the replacement μ play a crucial role. These conditions are certainly a considerable restriction in practice, and hence should also be more thoroughly investigated. Apart from the disjoint union case, we have also shown that the obtained results for disjoint unions extend nicely to the case of shared constructors. On the other hand, of course, modularity results do not always help. A simple example is the following.

Example 11. The two CSRS's

$$\mathcal{R}_1: \quad inf(x) \to x : inf(s(x))$$

$$\mathcal{R}_2: \quad \begin{array}{c} nth(0, x : y) \to x \\ nth(s(x), y : z) \to nth(x, z) \end{array}$$

with $\mu(:) = \{1\}$ are constructor sharing and both terminating, and their union $\mathcal{R}$ is terminating, too! However, none of our modularity results is applicable here as the reader is invited to verify. Intuitively, the reason for this non-applicability of (generic) modularity results lies in the fact, that any termination proof of $\mathcal{R}$ must somehow exploit internal (termination) arguments about $\mathcal{R}_2$. A bit more precisely, the decrease in the first argument of the second $\mathcal{R}_2$-rule "lexicographically dominates" what happens in the second argument. To make this more explicit, consider also the following, semantically meaningless, variant of $\mathcal{R}_2$:

$$\mathcal{R}_3: \quad \begin{array}{c} nth(0, x : y) \to x \\ nth(s(x), y : z) \to nth(s(x), z) \end{array}$$

with μ as before. Clearly, $\mathcal{R}_3$ is also terminating. However, now the union of the constructor sharing CSRS's $\mathcal{R}_1$ and $\mathcal{R}_3$ becomes non-terminating: $nth(s(x), inf(y)) \hookrightarrow nth(s(x), y : inf(s(y))) \hookrightarrow nth(s(x), inf(s(y))) \hookrightarrow \ldots$. Here the failure of our modularity criteria becomes comprehensible. Namely, $\mathcal{R}_1$ and $\mathcal{R}_2$ do have the same syntactical modularity structure as $\mathcal{R}_1$ combined with $\mathcal{R}_3$. And in the former case we got termination, in the latter one non-termination. Thus it is unrealistic to expect the applicability of a general modularity result in this particular example. Yet, if we now consider still another system

$$\mathcal{R}_4: \quad \begin{array}{c} take(0, x) \to nil \\ take(s(x), y : z) \to take(x, z) \end{array}$$

with $\mu(:)$ as above and consider the union of the terminating *composable* CSRS's $\mathcal{R}_1 \cup \mathcal{R}_2$ and $\mathcal{R}_1 \cup \mathcal{R}_4$, then we might wish to conclude termination of the combination by some modularity criterion. This does not seem to be hopeless.

In other words, we expect that many results that hold for constructor sharing CSRS's also extend to unions of *composable* CSRS's which, additionally, may share defined function symbols provided they then share all their defining rules, too (cf. [39]), and to some extent also for hierarchical combinations of CSRS's (cf. e.g. [21], [4]). However, since modularity is known to be a very error-prone domain, any concrete such claim has to be carefully verified. This will be the subject of future work.

6. CONCLUSION

We have presented some first steps of a thorough modularity analysis in context-sensitive rewriting. In the paper we have mainly focussed on termination properties. The results obtained by now are very encouraging. But there remains a lot of work to be done.

7. REFERENCES

[1] F. Baader and T. Nipkow. *Term rewriting and All That.* Cambridge University Press, 1998.

[2] C. Borralleras, S. Lucas, and A. Rubio. Recursive path orderings can be context-sensitive. In A. Voronkov, ed., *Proc. 18th Int. Conf. on Automated Deduction (CADE'02), Copenhagen, Denmark, July 2002*, *LNAI*, Springer-Verlag, to appear.

[3] M. Clavel, S. Eker, P. Lincoln, and J. Meseguer. Principles of Maude. In J. Meseguer, ed., *Proc. 1st Int. Workshop on Rewriting Logic and its Applications (WRLA'96)*, volume 4 of *Electronic Notes in Theoretical Computer Science*, Pacific Grove, California, Sept. 1996. Elsevier. 25 pages.

[4] N. Dershowitz. Hierarchical termination. In N. Dershowitz and N. Lindenstrauss, eds.*Proc. 4th Int. Workshop on Conditional and Typed Rewriting Systems, Jerusalem, Israel (1994)*, *LNCS 968*, pp. 89–105. Springer-Verlag, 1995.

[5] N. Dershowitz and D. Plaisted. Rewriting. In J. Robinson and A. Voronkov, eds., *Handbook of Automated Reasoning*, volume 1, chapter 9, pp. 535–610. Elsevier and MIT Press, 2001.

[6] K. Drosten. *Termersetzungssysteme.* Informatik-Fachberichte 210. Springer-Verlag, 1989. In German.

[7] S. Eker. Term rewriting with operator evaluation strategies. In C. Kirchner and H. Kirchner, eds., *Proc. 2nd Int. Workshop on Rewriting Logic and its Applications (WRLA'98)*, volume 15 of *Electronic Notes in Theoretical Computer Science*, pages 1–20, Pont-à-Mousson, Nancy, France, Sept. 1998. Elsevier.

[8] M. Ferreira and A. Ribeiro. Context-sensitive AC-rewriting. In P. Narendran and M. Rusinowitch, eds., *Proc. 10th Int. Conference on Rewriting Techniques and Applications (RTA'99)*, *LNCS 1631*, pp. 286–300, Trento, Italy, July 1999. Springer-Verlag.

[9] O. Fissore, I. Gnaedig, and H. Kirchner. Termination of rewriting with local strategies. In M.P. Bonacina and B. Gramlich, eds., *4th International Workshop on Strategies in Automated Deduction (STRATEGIES 2001) – Selected Papers*, volume 58 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers,

2001. also available as Techical Report A01-R-177, LORIA, Nancy, France.

[10] K. Futatsugi, J. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Conference Record of the 12th Annual ACM Symposium on Principles of Programming Languages (POPL'85)*, pp. 52–66. ACM Press, 1985.

[11] K. Futatsugi and A. Nakagawa. An overview of CAFE specification environment – an algebraic approach for creating, verifying, and maintaining formal specifications over networks. In *Proc. 1st IEEE Int. Conference on Formal Engineering Methods (ICFEM'97)*, pp. 170–182, Hiroshima, Japan, 1997. IEEE Computer Society. http://computer.org/proceedings/icfem/8002/8002toc.htm.

[12] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. In P. Narendran and M. Rusinowitch, eds., *Proc. 10th Int. Conference on Rewriting Techniques and Applications (RTA'99)*, *LNCS 1631*, pp. 271–287, Trento, Italy, July 1999. Springer-Verlag.

[13] J. Giesl and A. Middeldorp. Transforming context-sensitive rewrite systems. In Y. Toyama, ed., *Proc. Int. Workshop on Rewriting in Proof and Computation (RPC'01)*, RIEC, pp. 14–33, Tohoku University, Japan, 2001.

[14] J. Giesl and A. Middeldorp. Innermost termination of context-sensitive rewriting. Aachener Informatik-Berichte (AIBs) 2002-04, RWTH Aachen, 2002.

[15] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. Aachener Informatik-Berichte (AIBs) 2002-02, RWTH Aachen, 2002.

[16] J. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducing OBJ. In J. Goguen and G. Malcolm, eds., *Software Engineering with OBJ: algebraic specification in action*. Kluwer, 2000.

[17] B. Gramlich. Generalized sufficient conditions for modular termination of rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 5:131–158, 1994.

[18] B. Gramlich. Abstract relations between restricted termination and confluence properties of rewrite systems. *Fundamenta Informaticae*, 24:3–23, 1995.

[19] B. Gramlich. *Termination and Confluence Properties of Structured Rewrite Systems*. PhD thesis, Fachbereich Informatik, Universität Kaiserslautern, Jan. 1996.

[20] J. W. Klop, A. Middeldorp, Y. Toyama, and R. Vrijer. Modularity of confluence: A simplified proof. *Inf. Process. Lett.*, 49:101–109, 1994.

[21] M. Krishna Rao. Modular proofs for completeness of hierarchical term rewriting systems. *Theoretical Comput. Sci.*, 151(2):487–512, Nov. 1995.

[22] M. Kurihara and A. Ohuchi. Modularity of simple termination of term rewriting systems with shared constructors. *Theoretical Comput. Sci.*, 103:273–282, 1992.

[23] M. Kurihara and A. Ohuchi. Termination of combination of composable term rewriting systems. In *Proc. 7th Australian Joint Conference on Artificial Intelligence*, Nov. 1994.

[24] M. Kurihara and A. Ohuchi. Modularity in noncopying term rewriting. *Theoretical Comput. Sci.*, 152(1):139–169, Dec. 1995.

[25] S. Lucas. Termination of context-sensitive rewriting by rewriting. In F. Meyer auf der Heide and B. Monien, eds., *Proc. 23rd Int. Colloquium on Automata, Languages and Programming (ICALP'96)*, *LNCS 1099*, pp. 122–133, Paderborn, Germany, July 1996. Springer-Verlag.

[26] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1), Jan. 1998. The MIT Press.

[27] S. Lucas. Termination of on-demand rewriting and termination of OBJ programs. In H. Sondergaard, ed., *Proc. 3rd Int. ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP'01)*, pp. 82–93, Firenze, Italy, Sept. 2001. ACM Press, New York.

[28] S. Lucas. Termination of rewriting with strategy annotations. In R. Nieuwenhuis and A. Voronkov, eds., *Proc. 8th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'01)*, *LNCS 2250*, pages 669–684, Havana, Cuba, Dec. 2001. Springer-Verlag.

[29] S. Lucas. Transfinite rewriting semantics for term rewriting systems. In A. Middeldorp, ed., *Proc. 12th Int. Conference on Rewriting Techniques and Applications, (RTA'01)*, *LNCS 2051*, pp. 216–230, Utrecht, The Netherlands, May 2001. Springer-Verlag.

[30] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 2002. to appear.

[31] S. Lucas. Termination of (canonical) context-sensitive rewriting. In S. Tison, ed., *Proc. 13th Int. Conference on Rewriting Techniques and Applications (RTA'02)*, *LNCS*, Copenhagen, Denmark, July 2002. Springer-Verlag. To appear.

[32] A. Middeldorp. A sufficient condition for the termination of the direct sum of term rewriting systems. In *Proceedings of the 4th IEEE Symposium on Logic in Computer Science*, pp. 396–401, Pacific Grove, 1989.

[33] A. Middeldorp. *Modular Properties of Term Rewriting Systems*. PhD thesis, Free University, Amsterdam, 1990.

[34] A. Middeldorp. Modular properties of conditional term rewriting systems. *Information and Computation*, 104(1):110–158, May 1993.

[35] A. Middeldorp. Completeness of combinations of conditional constructor systems. *Journal of Symbolic Computation*, 17:3–21, 1994.

[36] A. Middeldorp and Y. Toyama. Completeness of combinations of constructor systems. *Journal of Symbolic Computation*, 15:331–348, Sept. 1993.

[37] E. Ohlebusch. *Modular Properties of Composable Term Rewriting Systems*. PhD thesis, Universität Bielefeld, 1994. Report 94-01.

[38] E. Ohlebusch. On the modularity of termination of term rewriting systems. *Theoretical Comput. Sci.*, 136:333–360, 1994.

[39] E. Ohlebusch. Modular properties of composable term rewriting systems. *Journal of Symbolic Computation*, 20(1):1–42, July 1995.

[40] J. Pol. Just-in-time: On strategy annotations. In B. Gramlich and S. Lucas, eds., *1st Int. Workshop on Reduction Strategies in Rewriting and Programming (WRS 2001)*, volume 57 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2001.

[41] M. Rusinowitch. On termination of the direct sum of term rewriting systems. *Inf. Process. Lett.*, 26:65–70, 1987.

[42] M. Schmidt-Schauß, M. Marchiori, and S. Panitz. Modular termination of r-consistent and left-linear term rewriting systems. *Theoretical Comput. Sci.*, 149(2):361–374, Oct. 1995.

[43] J. Steinbach and H. Xi. Freezing – termination proofs for classical, context-sensitive and innermost rewriting. Technical report, Institut für Informatik, TU München, Jan. 1998.

[44] Y. Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Inf. Process. Lett.*, 25:141–143, 1987.

[45] Y. Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *J. ACM*, 34(1):128–143, 1987.

[46] Y. Toyama, J. Klop, and H. Barendregt. Termination for direct sums of left-linear complete term rewriting systems. *J. ACM*, 42(6):1275–1304, Nov. 1995.

[47] H. Zantema. Termination of context-sensitive rewriting. In H. Comon, ed., *Proc. 8th Int. Conference on Rewriting Techniques and Applications (RTA'97)*, *LNCS 1232*, pp. 172–186, Sitges, Spain, June 1997. Springer-Verlag.