# The Journal of Functional and Logic Programming

*The MIT Press*

Volume 1998, Article 1

*16 January, 1998*

ISSN 1080–5230. MIT Press Journals, Five Cambridge Center, Cambridge, MA 02142-1493, USA; (617)253-2889; *journals-orders@mit.edu, journals-info @mit.edu*. Published one article at a time in LaTeX source form on the Internet. Pagination varies from copy to copy. For more information and other articles see:

- *http://www.cs.tu-berlin.de/journal/jflp/*
- *http://mitpress.mit.edu/JFLP/*
- *gopher.mit.edu*
- *ftp://mitpress.mit.edu/pub/JFLP*

# Context-Sensitive Computations in Functional and Functional Logic Programs

Salvador Lucas

16 January, 1998

**Abstract**

*Context-sensitive rewriting* is a refined form of rewriting that explores a smaller reduction space by imposing some fixed restrictions on the replacements. Any Term Rewriting System (TRS) can be given a context-sensitive rewrite relation. In this paper, we review the theory of context-sensitive rewriting and formulate conditions to guarantee the confluence of this relation. Also, for left-linear TRSs, we show that the (eventually obtained) computed value of a given expression can also be produced by context-sensitive rewriting, thus furnishing more efficient and still complete computations. We give the procedure for establishing the adequate replacement restrictions in order to achieve this. Finally, we raise the concept of context-sensitive restrictions from rewriting to narrowing, and provide the corresponding completeness results.[1]

## 1 Introduction

The operational semantics of functional languages is often given as term rewriting [Klo92, Lal93]. In this setting, a functional program is a set of equations that are interpreted as left-to-right rewrite rules [DJ90, Rea93]. The execution of a functional program for any input term consists of the evaluation of the term using the rewrite rules until it cannot be further reduced (i.e., until a *normal form* is reached). Variables occurring in the input term are considered to be universally quantified, and they cannot be

[1]This paper includes an extended and revised version of [Luc96a, Luc96c].

instantiated [Red85]. When we consider terms with logic (existentially quantified) variables, different instantiations of the variables can lead to different computed values. The result is a different computational paradigm that integrates functional and logic features, called *functional logic programming* (see [Han94] for a recent survey). Because rewriting is not able to instantiate these variables, a new operational principle is needed when we consider functional logic languages. Usually, this operational mechanism is (some form of) narrowing [Hul80, Red85].

In practice, at each reduction step, the concrete choice of the rule and the subterm to which the rule applies is determined by a *reduction strategy*. For instance, the evaluation of a function call $f(t_1, \ldots, t_k)$ using a *lazy* strategy only evaluates arguments $t_1, \ldots, t_k$ if it is necessary. An *eager* strategy first evaluates arguments, and then perfoms the reduction of $f$ applied to the evaluated arguments. The choice of a reduction strategy is crucial to ensure termination of the evaluation process. Lazy evaluation strategies allow us to deal with nonterminating programs and infinite data structures, because they delay the evaluation of arguments in a function call until they are really needed. As a drawback, the implementation of lazy strategies is expensive, owing to the need for closures for the delayed arguments, that introduce space and time overhead during program execution [PJ87]. We also need to (efficiently) decide which argument is *needed* in each reduction step, which is not a trivial matter (see [Dur94, HL91, KM91, O'D85, TSvEP93]). Eager evaluation strategies do not have this problem, because we decide to evaluate all arguments with no further considerations. However, they can easily progress into nontermination.

*Context-sensitive rewriting* (*csr*) represents a simple approach to both strategies. It is not a strategy, but a restriction of rewriting that is based on introducing syntactic restrictions for the allowed replacements. Context-sensitive rewriting always preserves termination, and can sometimes improve it [Luc96b, Zan97]. Even if we cannot obtain termination, we can define strategies that are able to use the restrictions that context-sensitive rewriting imposes as a basis for obtaining simple, efficient normalizing strategies [Luc97].

Context-sensitive rewriting has special strong connections with the lazy strategies of functional programming languages. Actually, the mechanism that allows us to prevent the evaluation of some expressions can also be used to declare that these reductions need not be done. Context-sensitive rewriting does allow us to easily express fixed, meaningful restrictions on

rewriting, similar to the way that lazy reduction techniques approximate "needed" reductions à la Huet and Lévy [HL91]. The link between these topics is analyzed in [Luc97].
In context-sensitive rewriting [Luc95], for each function symbol in the signature $\Sigma$, we fix the set of *replacing* positions by means of a mapping:

$$\mu : \Sigma \to \mathcal{P}(\mathbb{N})$$

This *replacement map* specifies the set of positions that can eventually be reduced. Given a term $t$, the subterm of $t$ at the occurrence $u$ is a redex of the context-sensitive rewrite relation, if it is a redex [DJ90] and the symbols in $t$ that label the occurrences above $u$ satisfy the particular *replacement condition*.

**Example 1** *Consider the TRS:*

$$\begin{array}{ll}
\mathsf{if}(\mathsf{true},\mathsf{x},\mathsf{y}) \to \mathsf{x} & \mathsf{and}(\mathsf{true},\mathsf{x}) \to \mathsf{x} \\
\mathsf{if}(\mathsf{false},\mathsf{x},\mathsf{y}) \to \mathsf{y} & \mathsf{and}(\mathsf{false},\mathsf{y}) \to \mathsf{false} \\
\mathsf{0}+\mathsf{x} \to \mathsf{x} & \\
\mathsf{s}(\mathsf{x})+\mathsf{y} \to \mathsf{s}(\mathsf{x}+\mathsf{y}) &
\end{array}$$

*For an input expression* $\mathsf{if}(\mathsf{and}(\mathsf{true},\mathsf{false}),\mathsf{0}+\mathsf{s}(\mathsf{0}),\mathsf{0})$*, we should not reduce the second and third arguments until the condition* $\mathsf{and}(\mathsf{true},\mathsf{false})$ *has been evaluated. By defining a replacement map such as* $\mu(\mathsf{if}) = \{1\}$ *(and even* $\mu(\mathsf{s}) = \mu(\mathsf{and}) = \mu(+) = \{1\}$*), undesirable reductions are avoided with no extra control. This also enforces the "intended meaning" of the* if-then-else *operation.*

In this paper, we summarize the basic theory of *csr* and analyze confluence of *csr*. We generalize two well-known results on confluence of unrestricted rewriting to *csr*. The first one is "terminating TRSs having joinable critical pairs are confluent" [Hue80]. The second one is "orthogonal TRSs are confluent" [HL91, O'D77]. The latter complements the previous one, since it also applies to nonterminating TRSs.

When considering functional programs, we would like to use the replacement restrictions introduced by *csr* to improve evaluations without sacrificing completeness. Context-sensitive rewriting can compute different normal forms for a given expression, since the confluence and the length of the derivations might be modified by the replacement restrictions. Thus, even if we ensure that the context-sensitive computations are both finite and convergent, it is also essential to characterize conditions to guarantee that the value that is computed by unrestricted rewriting and the evaluation performed by *csr* do coincide. In this case, it makes sense to use *csr* instead of unrestricted rewriting. We formalize here the conditions ensuring these requirements. We show how to define a replacement map for a given TRS, in order to obtain completeness in computations leading to head-normal forms and values. Our results apply to the class of left-linear TRSs. We illustrate with some examples the definition of the replacement maps, and how it is possible to combine the improvements in termination with useful computations in functional programming.

Because narrowing can be considered to be a natural extension of rewriting to deal with logical variables, it seems natural to investigate whether the idea of restricting the positions of allowed reductions can be raised from rewriting to narrowing. The evaluation strategies are also an important subject of research in functional logic programming [Han95, Red85]. By introducing replacement restrictions, we expect to have similar improvements in efficiency, i.e., in avoiding useless computations. We define *context-sensitive narrowing* as a limited form of narrowing that incorporates replacement restrictions of a similar kind within the standard narrowing relation. We prove that the same kind of restrictions that provide complete evaluations using *csr* also ensure that context-sensitive narrowing does not lose completeness. This is the basis for using context-sensitive narrowing in evaluating goals with logic variables. A common application of logic variables is to use them to solve equations. We show that context-sensitive narrowing can also be used to obtain complete, more general sets of solutions in equational reasoning.

## 1.1 Related Work

Using replacement restrictions is not new in programming. A well-known example concerns the binary operator *cons* on lists (::) in its nonstrict version [FW76, HM76]. In [FW76, HM76], a modification of the evaluation modes for some basic operations of Lisp is proposed. In particular, the new *cons* operator does not evaluate its arguments. This is similar to impose $\mu(::) = \emptyset$. Since Lisp functions are evaluated by processing lists built from these operations, the restrictions over the basic operators, in some sense, are raised to arbitrary Lisp functions. In [KW95, Mar90], the specification of replacement restrictions is similar to ours: in a first stage, the authors also limit the replacements that are allowed on the arguments of function symbols. Next, the restriction is raised to occurrences of terms. Nevertheless, the reduction relations that are defined by using the replacement restrictions are different from *csr*, and the computational behavior and properties differ: [KW95] uses the replacement restrictions on the arguments of functions to implement lazy reductions in an eager mode by means of a program transformation. Therefore, the restricted rewrite relation (called *lazy rewriting*) is not used in practice. It is only an intermediate step for defining the transformation and its properties. In fact, few and weak results on computational properties for lazy rewriting (whose definition is more complex than ours) are given. Thus, a practical use of the restricted reduction relation itself is not feasible. A more-detailed comparison is given after our technical presentation.

In [Mar90] the implementation of lazy reductions by means of graph reduction techniques based on a restricted class of TRSs is proposed. In fact, the reduction relation defined by Maranget (called *conditional reduction*, although it is not related to conditional TRSs) is more similar to *csr* than is the lazy rewriting in [KW95]. Maranget proves some results on confluence and neededness of conditional reduction. Our results are more general than theirs, as we show below. This is mainly because the class of TRSs proposed by Maranget is very restrictive. Also, some details are missed in Maranget's discussion on confluence of conditional reduction, which we point out in this paper. In [Luc97], we compare the results on neededness.

The replacement map of *csr* describes purely syntactic information on restrictions that is easy to manage and implement, and, as opposed to [KW95], the restrictions on the occurrences are a simple extension of the restrictions on symbols of the signature. Computational properties of *csr* have been sufficiently analyzed to make it a suitable tool for computing using functional programs. Also, the previous approaches assume that either the programmer should specify the replacement restrictions [KW95, Mar90], or that the restrictions would be deduced from strictness information [HW87, KW95]. The first assumption is not suitable if we want to automatically introduce the optimization in compiling time (on unannotated programs). The second assumption can be as complex as the strictness analyses are: strictness analyses take into account not only the left-hand sides of rules but also the right-hand sides, and they use fixpoint techniques to establish the strictness information [CPJ85].

In this paper, we give an easy method for automatically calculating the most-restrictive replacement map that makes *csr* equivalent (but more efficient) to unrestricted rewriting in computations leading to head-normal forms and values. We only consider left-hand sides of rules to define them, and no fixpoint technique is used. We have formally proven the efficiency of context-sensitive computations using such replacement maps [Luc97].

The paper is organized as follows. In Section 2, we briefly review the technical concepts and results used in the remainder of the paper. In Section 3, we formulate some basic properties of the context-sensitive rewrite relation, and we illustrate the usefulness of *csr* in functional programming. Section 4 provides criteria for ensuring confluence of *csr* for a suitable class of programs. Section 5 characterizes the preservation of evaluations when using *csr*. Section 6 introduces context-sensitive narrowing, and proves its completeness. Section 7 concludes the paper.

## 2 Preliminaries

Let us first introduce the main notations used in the paper. For full definitions, we refer to [DJ90, Hue80, Klo92, Lal93]. Throughout the paper, $V$ denotes a countable set of variables, and $\Sigma$ denotes a signature: a set of function symbols $\{\mathsf{f}, \mathsf{g}, \ldots\}$, each with a fixed arity given by a function $ar_\Sigma : \Sigma \to \mathbb{N}$ (or just $ar$, when the signature is clear from the context). By $\mathcal{T}(\Sigma, V)$ we denote the set of terms built from symbols in the signature $\Sigma$ and variables in $V$. A $k$-tuple $t_1, \ldots, t_k$ of terms is denoted as $\tilde{t}$, where $k$ will be clarified from the context. Given a term $t$, $Var(t)$ is the set of variable symbols in $t$.

Terms are viewed as labeled trees in the usual way. Occurrences $u, v, \ldots$ are represented by chains of positive natural numbers used to address subterms of $t$. By $\epsilon$, we denote the empty chain. We denote as $|u|$ the length of a chain $u$. If $u$ is an occurrence, and $W$ is a set of occurrences, $u.W$ is the set $\{u.v \mid v \in W\}$. Occurrences are ordered by the standard prefix ordering: $u \leq v$ if and only if there is a chain $w$ such that $v = u.w$. By $u \parallel v$ we denote that $u, v$ are not comparable by means of $\leq$.

The set of occurrences of a term $t$ is denoted by $O(t)$. A linear term is a term having no multiple occurrences of the same variable. The subterm of $t$ at occurrence $u$ is denoted by $t|_u$. The set of occurrences of nonvariable symbols in $t$ is $O_\Sigma(t)$, and $O_V(t)$ is the set of variable occurrences. By $O_s(t)$, we denote the set of occurrences of $s$ in $t$, i.e., $u \in O_s(t)$ if and only if $t|_u = s$. Denote as $t[s]_u$ the term $t$ with the subterm at the occurrence $u$ replaced with $s$. Denote as $root(t)$ the symbol labeling the root of $t$. Denote as $\Sigma(t) = \{f \in \Sigma \mid \exists u \in O(t).\ root(t|_u) = f\}$ the set of symbols from $\Sigma$ appearing in $t$. The chain of symbols lying in occurrences above/on $u \in O(t)$ is $\mathit{prefix}_t(u)$, defined as follows: $\mathit{prefix}_t(\epsilon) = root(t)$, $\mathit{prefix}_t(i.u) = root(t).\mathit{prefix}_{t_i}(u)$. The strict prefix $\mathit{sprefix}_t(u)$ is defined by $\mathit{sprefix}_t(\epsilon) = \epsilon$, $\mathit{sprefix}_t(u.i) = \mathit{prefix}_t(u)$. We refer to any term $C$, which is the same as $t$ everywhere except below $u$, i.e., there exists a term $s$ such that $C[s]_u = t$, as the *context* within which the replacement occurs. Roughly speaking, a context is a term $C$ with a "hole" at a specific occurrence $u$. Sometimes we write $C[\ ]_u$ to represent the context itself.

Consider the following properties from [Hue80]. Assume $t_1, t_2, t_3 \in \mathcal{T}(\Sigma, V)$. If $u \in O(t_1)$, $v \in O(t_2)$, then $(t_1[t_2]_u)|_{u.v} = t_2|_v$ (*embedding*), and $t_1[t_2[t_3]_v]_u = t_1[t_2]_u[t_3]_{u.v}$ (*associativity*). If $u, v \in O(t_1)$ and $u \parallel v$ holds, we have $(t_1[t_2]_u)|_v = t_1|_v$ (*persistence*) and $t_1[t_2]_u[t_3]_v = t_1[t_3]_v[t_2]_u$ (*commutativity*). When $u \leq v$, assume $v = u.w$ to get $t_1|_v = (t_1|_u)|_w$ (*cancellation*), $t_1[t_2]_v[t_3]_u = t_1[t_3]_u$ (*dominance*), and $(t_1[t_2]_v)|_u = (t_1|_u)[t_2]_w$ (*distributivity*).

Recall the main concepts and notations about the lattice of first-order terms. A substitution is a mapping $\sigma : V \to \mathcal{T}(\Sigma, V)$. Denote as $\varepsilon$ the "identity" substitution: $\varepsilon(x) = x$ for all $x \in V$. The set $\mathcal{D}om(\sigma) = \{x \in V \mid \sigma(x) \neq x\}$ is called the *domain of* $\sigma$. Whenever $\mathcal{D}om(\sigma) \cap \mathcal{D}om(\sigma') = \emptyset$, for substitutions $\sigma, \sigma'$, we denote by $\sigma \cup \sigma'$ a substitution such that $(\sigma \cup \sigma')(x) = \sigma(x)$ if $x \in \mathcal{D}om(\sigma)$, and $(\sigma \cup \sigma')(x) = \sigma'(x)$ if $x \in \mathcal{D}om(\sigma')$. Given a subset $W \subseteq V$, denote as $\sigma\!\downarrow_W$ the restriction of $\sigma$ to variables in $W$: $\sigma\!\downarrow_W(x) = \sigma(x)$, if $x \in W$ and $\sigma\!\downarrow_W(x) = x$ if $x \notin W$. The quasi-ordering of subsumption $\leq$ in $\mathcal{T}(\Sigma, V)$ is $t \leq t' \Leftrightarrow \exists \sigma.\ t' = \sigma(t)$. We denote as $\sigma \leq \sigma' [\![W]\!]$ the fact that $\sigma(x) \leq \sigma'(x)$ for all $x \in W$. We write $\sigma \leq \sigma'$ if and only if $\sigma \leq \sigma' [\![\mathcal{D}om(\sigma) \cup \mathcal{D}om(\sigma')]\!]$.

The equivalence $\equiv$ induced by $\leq$ on $\mathcal{T}(\Sigma, V)$ is $t \equiv t' \Leftrightarrow t \leq t' \wedge t' \leq t$. The ordering $>$ can be defined as $t > t' \Leftrightarrow t' \leq t \wedge t \not\leq t'$. Let $\widehat{\mathcal{T}}$ be the quotient set $\mathcal{T}(\Sigma, V)/\equiv$ completed with a *maximum* element, $\top$. $\widehat{\mathcal{T}}$ is a complete lattice. We denote the least upper bound (lub) of two terms $t, t'$ as $t \sqcup t'$. This term is unique modulo $\equiv$, and can be found (if it exists) by a unification algorithm. If this lub exists, we write $t =^? t'$ and say that $t$ and $t'$ are unifiable. A term $s$ *overlaps* a term $t$ if it can be unified with some nonvariable subterm of $t$. Terms $s$ and $t$ are not overlapping if neither $s$ overlaps $t$ nor $t$ overlaps $s$.

A rewrite rule (labeled $\alpha$) is an ordered pair $(l, r)$, written $\alpha : l \to r$ (or just $l \to r$), with $l, r \in \mathcal{T}(\Sigma, V)$, $l \notin V$, and $Var(r) \subseteq Var(l)$. The left-hand side (lhs) of the rule is $l$, and $r$ is the right-hand side (rhs). A TRS is a pair $\mathcal{R} = (\Sigma, R)$ where $R$ is a set of rewrite rules. By $L(\mathcal{R})$, we denote the set of lhs's of $\mathcal{R}$. An instance $\sigma(l)$ of an lhs $l \in L(\mathcal{R})$ is a redex, and $O_{\mathcal{R}}(t) = \{u \in O(t) \mid \exists l \in L(\mathcal{R}).\ t|_u = \sigma(l)\}$ is the set of redex occurrences in a term $t$. If $O_{\mathcal{R}}(t) = \emptyset$, then $t$ is called a *normal form*.

A TRS $(\Sigma, R)$ is *left-linear*, if all lhs's of rules in $R$ are linear terms. Two rules $l \to r$ and $l' \to r'$ (having disjoint variables) *overlap* if there is a nonvariable occurrence $u \in O_\Sigma(l)$ such that $l|_u$ and $l'$ unify with the most general unifier (mgu) $\sigma$. Recall that a substitution $\sigma$ is an mgu of terms $t, s$ if, for every other unifier $\sigma'$ of $t$ and $s$, it holds that $\sigma \leq \sigma'$. The pair $\langle \sigma(l)[\sigma(r')]_u, \sigma(r) \rangle$ is called a *critical pair*. A critical pair $\langle \sigma(l)[\sigma(r')]_u, \sigma(r) \rangle$ with $u = \epsilon$ is an overlay. A critical pair $\langle t, s \rangle$ is trivial if $t = s$. A TRS is almost nonambiguous if all its critical pairs are trivial overlays. A TRS is nonambiguous if there are no overlapping lhs's (trivial overlap in the same lhs is not considered). A left-linear, almost nonambiguous TRS is called *almost orthogonal*. A left-linear, nonambiguous TRS is called orthogonal.

For a given TRS $\mathcal{R} = (\Sigma, R)$, a term $t$ rewrites to a term $s$ (at the occurrence $u$), written $\overset{[u,\alpha]}{\to}_{\mathcal{R}}$ (or just $t \overset{u}{\to}_{\mathcal{R}} s$, $t \to_{\mathcal{R}} s$, or $t \to s$) if $t|_u = \sigma(l)$ and $s = t[\sigma(r)]_u$, for some rule $\alpha : l \to r \in R$, occurrence $u \in O(t)$, and substitution $\sigma$. The one-step rewrite relation for $\mathcal{R}$ is $\to_{\mathcal{R}}$. The inner reduction relation is $\xrightarrow{>\epsilon} = \to \setminus \overset{\epsilon}{\to}$. A term $t$ is a head-normal form (hnf), if there is no derivation $t = t_1 \to t_2 \to \ldots$ starting from $t$ that reduces the root of a term $t_i$, $i \geq 1$.

A term $t$ narrows to $s$, written $t \leadsto_{[u,\alpha,\sigma]} s$ (or just $t \leadsto_\sigma s$), if there is $u \in O_\Sigma(t)$ and a variant (i.e., a renamed version) of a rule $\alpha : l \to r$ such that $t|_u$ and $l$ unify with (idempotent) mgu $\sigma$, and $s = \sigma(t[r]_u)$. A narrowing derivation (or $\leadsto$derivation) $t \leadsto_\theta^* s$ is such that either $t = s$ and $\theta = \epsilon$ or $t \leadsto_{\sigma_0} t_1 \leadsto_{\sigma_1} \ldots t_{n-1} \leadsto_{\sigma_{n-1}} s$ and $\theta = \sigma_{n-1} \ldots \sigma_1 \sigma_0$.

$\mathbb{N}_k^+$ is an initial segment $\{1, 2, \ldots k\}$ of the set of positive natural numbers $\mathbb{N}^+$, where $\mathbb{N}_0^+ = \emptyset$. $\mathcal{P}(\mathbb{N})$ is the power set of natural numbers. Given a finite set $A$, $|A|$ is the number of elements in $A$.

## 3 Context-Sensitive Rewriting

In *csr* [Luc95], we impose a syntactically based restriction that prevents our having to perform some reductions. This is achieved by the replacement map.

**Definition 1 (Replacement Map)** *Let $\Sigma$ be a signature. A mapping $\mu : \Sigma \to \mathcal{P}(\mathbb{N})$ is a replacement map (or $\Sigma$-map) for the signature $\Sigma$ if and only if for all $f \in \Sigma$. $\mu(f) \subseteq \mathbb{N}^+_{ar(f)}$.*

A $\Sigma$-map $\mu$ determines the argument positions $\mu(f)$ that can be reduced for each symbol $f$ in the signature $\Sigma$. If we assume an arbitrary ordering in the signature[2] $\Sigma = \{f_1, \ldots, f_n\}$, then we can express a $\Sigma$-map as follows: $\mu = \langle I_1, \ldots, I_n \rangle$, where $I_j = \mu(f_j)$ for $1 \leq j \leq n$.

The set of replacement maps that can be defined for a given signature $\Sigma$ is denoted as $M_\Sigma$. In particular, we consider $M_\emptyset = \{\mu_\perp^\perp\}$, where $\mu_\perp^\perp$ is the unique replacement map that corresponds to the empty signature (the unique mapping that can be defined from an empty set to any other set).

The ordering $\subseteq$ on $\mathcal{P}(\mathbb{N})$ extends pointwise to an ordering $\sqsubseteq$ on replacement maps.

**Definition 2 (Partial Order on Replacement Maps)** *Let $\Sigma$ be a signature. Let $M_\Sigma$ be the set of all $\Sigma$-maps, and let $\mu, \mu' \in M_\Sigma$. We define a partial order $\sqsubseteq$ on $M_\Sigma$ as follows: $\mu \sqsubseteq \mu' \Leftrightarrow \forall f \in \Sigma.\ \mu(f) \subseteq \mu'(f)$.*

Therefore, $\mu \sqsubseteq \mu'$ means that $\mu$ considers less positions than $\mu'$ for reduction. The distributive complete lattice $(\mathcal{P}(\mathbb{N}), \subseteq, \emptyset, \mathbb{N}, \cup, \cap)$ induces a distributive complete lattice $(M_\Sigma, \sqsubseteq, \mu_\perp, \mu_\top, \sqcup, \sqcap)$. The (bottom) $\Sigma$-map $\mu_\perp$ is $\mu_\perp(f) = \emptyset$ for all $f \in \Sigma$: the (top) $\Sigma$-map $\mu_\top$ is $\mu_\top(f) = \mathbb{N}^+_{ar(f)}$ for all $f \in \Sigma$. The least upper bound (lub) binary operation, $\sqcup$, gives the least replacement map $\mu \sqcup \mu'$, which is greater than or equal to both $\mu$ and $mu'$. It is given by $(\mu \sqcup \mu')(f) = \mu(f) \cup \mu'(f)$ for all $f \in \Sigma$. The greatest lower bound (*glb*) binary operation, $\sqcap$, gives the greatest replacement map $\mu \sqcap \mu'$, which is less than or equal to both $\mu$ and $\mu'$. It is given by $(\mu \sqcap \mu')(f) = \mu(f) \cap \mu'(f)$ for all $f \in \Sigma$. We denote as $\mu \parallel \mu'$ the fact $\mu \not\sqsubseteq \mu'$ and $\mu' \not\sqsubseteq \mu$. The following lemma is used later.

**Lemma 1** *$\mu \not\sqsubseteq \mu'$ if and only if $\exists f \in \Sigma. \exists i \in \mathbb{N}^+_{ar(f)}.\ i \in \mu(f) \wedge i \notin \mu'(f)$.*

[2]We usually take the order in which the symbols are written.

**Proof of Lemma 1**

$$\neg(\exists f \in \Sigma . \exists i \in \mathbb{N}^+_{ar(f)}.\ i \in \mu(f) \wedge i \notin \mu'(f))$$

if and only if

$$\forall f \in \Sigma . \forall i \in \mathbb{N}^+_{ar(f)}.\ i \notin \mu(f) \vee i \in \mu'(f)$$

if and only if

$$\forall f \in \Sigma . \forall i \in \mathbb{N}^+_{ar(f)}.\ i \in \mu(f) \Rightarrow i \in \mu'(f)$$

if and only if $\mu \sqsubseteq \mu'$.

**Proof of Lemma 1** $\Box$

We are also interested in considering replacement maps from different signatures $\Sigma, \Sigma'$. We define a generic operation $\overline{\phantom{x}} : M_\Sigma \rightarrow M_{\Sigma'}$ which, given a replacement map $\mu \in M_\Sigma$, provides the *insertion* $\overline{\mu}$ of $\mu$ into $M_{\Sigma'}$: for all $f \in \Sigma'$,

$$\overline{\mu}(f) = \begin{cases} \mu(f) & \text{if } f \in \Sigma \text{ and } ar_\Sigma(f) = ar_{\Sigma'}(f) \\ \emptyset & \text{otherwise} \end{cases}$$

Note that when we consider a signature $\Sigma$, we always have $\overline{\mu}^{\perp}_{\perp} = \mu_\perp \in M_\Sigma$. This motivates the symbol $\mu^{\perp}_{\perp}$.

This operation makes it feasible to extend algebraic operations on replacement maps in $M_\Sigma$ to operations between replacement maps from different signatures. For instance, given $\mu \in M_\Sigma$ and $\mu' \in M_{\Sigma'}$, we write $\mu \sqcup \mu' \in M_{\Sigma \cup \Sigma'}$ instead of $\overline{\mu} \sqcup \overline{\mu'}$. In this way, we actually use $\overline{\phantom{x}} : M_\Sigma \rightarrow M_{\Sigma \cup \Sigma'}$, $\overline{\phantom{x}} : M_{\Sigma'} \rightarrow M_{\Sigma \cup \Sigma'}$, and the lub operation of $M_{\Sigma \cup \Sigma'}$. We will do so (mainly in Section 5) unless we specify differently.

Also, given a subset of function symbols $\Delta \subseteq \Sigma$, we denote as $\mu\downarrow_\Delta$ the $\Delta$-map, which is the restriction of the $\Sigma$-map $\mu$ to symbols in $\Delta$, i.e., $\mu\downarrow_\Delta(f) = \mu(f)$ for all $f \in \Delta$.

The replacement map determines the positions of the arguments that can be reduced for a given symbol of the signature. The replacement condition indicates the occurrences that can be rewritten.

**Definition 3 (Replacement Condition)** *Let $\Sigma$ be a signature, and $\mu$ be a $\Sigma$-map. Let $t \in \mathcal{T}(\Sigma, V)$ be a term. The replacement condition is a predicate $\gamma_{\mu,t}$ defined on the set of occurrences $O(t)$ as follows:*

$$\gamma_{\mu,t}(\epsilon)$$
$$\gamma_{\mu,f(t_1,\ldots,t_k)}(i.u) \Leftrightarrow (i \in \mu(f)) \wedge \gamma_{\mu,t_i}(u)$$

*We say that $u$ is a $\mu$-replacing occurrence of $t$ or that $t|_u$ is a $\mu$-replacing subterm if and only if $\gamma_{\mu,t}(u)$ holds.*

We write $\gamma_t(u)$ when the replacement map is clear from the context. We introduce the following notation derived from the concept of the replacement condition. Denote as $O^\mu(t)$ the set of *replacing* occurrences of a term $t$, i.e., $u \in O^\mu(t) \Leftrightarrow u \in O(t) \wedge \gamma_t(u)$. The set of *nonreplacing* occurrences is $\widetilde{O^\mu}(t) = O(t) \backslash O^\mu(t)$. The set of replacing variable occurrences is $O^\mu_V(t) = O_V(t) \cap O^\mu(t)$. The set of replacing nonvariable occurrences is $O^\mu_\Sigma(t) = O_\Sigma(t) \cap O^\mu(t)$. The set of replacing occurrences of a subterm $s$ of $t$ is $O^\mu_s(t) = O_s(t) \cap O^\mu(t)$. Denote as $Var^\mu(t)$ the set of replacing variables of a term $t$, i.e., $Var^\mu(t) = \{x \in Var(t) \mid O^\mu_x(t) \neq \emptyset\}$. The set of *exclusively* nonreplacing variables is $\widetilde{Var^\mu}(t) = Var(t) \backslash Var^\mu(t)$.

## 3.1 Basic Properties of the Replacement Condition

**Proposition 1 ( Compositionality of the Evaluation of $\gamma_t$)** *Let $\Sigma$ be a signature, $t \in \mathcal{T}(\Sigma, V)$, and $u \in O(t)$. Then the following statement holds: $u = u_1.u_2 \Rightarrow (\gamma_t(u) \Leftrightarrow (\gamma_t(u_1) \wedge \gamma_{t|_{u_1}}(u_2)))$.*

**Proof of Proposition 1** By induction on the length of $u_1$. If $u_1 = \epsilon$, then $u = \epsilon.u_2$. Therefore, $\gamma_t(u) \Leftrightarrow \mathit{true} \wedge \gamma_{t|_\epsilon}(u) \Leftrightarrow \gamma_t(\epsilon) \wedge \gamma_{t|_\epsilon}(u) \Leftrightarrow \gamma_t(u_1) \wedge \gamma_{t|_{u_1}}(u_2)$. If $u_1 = i.u'_1$, then $t = f(t_1, \ldots, t_k)$, $1 \leq i \leq k$, and $u = i.u'_1.u_2$. By the induction hypothesis, $\gamma_t(u) \Leftrightarrow \gamma_t(i.u'_1.u_2) \Leftrightarrow i \in \mu(f) \wedge \gamma_{t_i}(u'_1.u_2) \Leftrightarrow i \in \mu(f) \wedge \gamma_{t_i}(u'_1) \wedge \gamma_{t_i|_{u'_1}}(u_2) \Leftrightarrow \gamma_t(i.u'_1) \wedge \gamma_{t|_{i.u'_1}}(u_2) \Leftrightarrow \gamma_t(u_1) \wedge \gamma_{t|_{u_1}}(u_2)$.

**Proof of Proposition 1** □

This proposition allows us to realize the evaluation of replacement conditions in a compositional mode. As immediate consequences of this result, we have the following corollaries.

**Corollary 1** *Let $\Sigma$ be a signature, $t \in \mathcal{T}(\Sigma, V)$, and $u, v \in O(t)$. The following statements hold:*

1. $(u \leq v \wedge \gamma_t(v)) \Rightarrow \gamma_t(u)$
2. $(u \leq v \wedge \neg\gamma_t(u)) \Rightarrow \neg\gamma_t(v)$

**Proof of Corollary 1**

1. If $v = u.v'$ then by Proposition 1, $\gamma_t(v) \Leftrightarrow \gamma_t(u) \wedge \gamma_{t|_u}(v')$. By contradiction, assume that $\neg\gamma_t(u)$, and thus $\neg\gamma_t(v)$. Since we have $\gamma_t(v)$, the claim holds.

2. If $v = u.v'$ and $\neg\gamma_t(u)$, then by Proposition 1 it is obvious that $\neg\gamma_t(v)$ holds.

**Proof of Corollary 1** □

**Corollary 2** *Let $\Sigma$ be a signature, $t \in \mathcal{T}(\Sigma, V)$, and $u \in O(t)$. The following statement holds: $\gamma_t(u) \Rightarrow (u = u_1.u_2 \Rightarrow \gamma_{t|_{u_1}}(u_2))$.*

**Proof of Corollary 2** Similar to Corollary 1(1).

**Proof of Corollary 2** □

We can assert some properties about the persistence of evaluating the replacement condition $\gamma_t$ with respect to two transformations of the term $t$: substitution application and subterm replacement.

**Proposition 2 (Persistence, e.g., Substitutions)** *Let $\Sigma$ be a signature, $t \in \mathcal{T}(\Sigma, V)$, and $u \in O(t)$. Let $\sigma$ be a substitution. Then, $\gamma_t(u) \Leftrightarrow \gamma_{\sigma(t)}(u)$.*

**Proof of Proposition 2** By structural induction on $t$. Base case: if $t$ is a variable or a constant symbol, we have $O(t) = \epsilon$ and $\gamma_t(\epsilon) \Leftrightarrow \gamma_{\sigma(t)}(\epsilon)$. For the induction step, let $t = f(t_1, \ldots, t_k)$ and $\sigma(t) = f(\sigma(t_1), \ldots, \sigma(t_k))$. Then, by the induction hypothesis, $\gamma_{\sigma(t)}(u) \Leftrightarrow \gamma_{\sigma(t)}(i.u') \Leftrightarrow i \in \mu(f) \wedge \gamma_{t_i}(u') \Leftrightarrow \gamma_t(u)$ since $u \in O(t)$.

**Proof of Proposition 2** □

**Proposition 3 (Persistence, e.g., Subterm Replacements)** *Let $\Sigma$ be a signature, $t, t' \in \mathcal{T}(\Sigma, V)$, and $u, v \in O(t)$. Then,*

*1.* $u \leq v \Rightarrow (\gamma_t(u) \Leftrightarrow \gamma_{t[t']_v}(u))$

*2.* $u \parallel v \Rightarrow (\gamma_t(u) \Leftrightarrow \gamma_{t[t']_v}(u))$

**Proof of Proposition 3**

1. By induction on the length of $u$. Base: $u = \epsilon$. Then, $\gamma_t(\epsilon) \Leftrightarrow \gamma_{t[t']_v}(\epsilon)$. Induction step: Let $u = i.u'$ and $v = i.v'$. By distributivity and by the induction hypothesis, we have $\gamma_{t[t']_v}(i.u') \Leftrightarrow i \in \mu(f) \wedge \gamma_{(t[t']_v)|_i}(u') \Leftrightarrow i \in \mu(f) \wedge \gamma_{t|_i[t']_{v'}}(u') \Leftrightarrow i \in \mu(f) \wedge \gamma_{t|_i}(u') \Leftrightarrow \gamma_t(u)$.

2. If $u \parallel v$, then there exists $w \in O(t)$, which is the glb of $\{u, v\}$. Therefore, $u = w.i.u'$ and $v = w.j.v'$ with $i \neq j$. Let $s = t|_w = f(t_1, \ldots, t_k)$. By distributivity, we get $(t[t']_v)|_w = (t[t']_{w.j.v'})|_w = (t|_w)[t']_{j.v'} = s[t']_{j.v'}$. Since $i, j.v' \in O(s)$, then by the persistence property, $(s[t']_{j.v'})|_i = s|_i = t_i$ with $i \parallel j.v'$. By Proposition 1, $\gamma_{t[t']_v}(u) \Leftrightarrow \gamma_{t[t']_v}(w.i.u') \Leftrightarrow \gamma_{t[t']_v}(w) \wedge \gamma_{(t[t']_v)|_w}(i.u')$. Now consider the following facts. First, by Proposition 3(1), $\gamma_{t[t']_v}(w) \Leftrightarrow \gamma_t(w)$, since $w < v$. Second, by the previous statements we have: $\gamma_{(t[t']_v)|_w}(i.u') \Leftrightarrow \gamma_{s[t']_{j.v'}}(i.u') \Leftrightarrow i \in \mu(f) \wedge \gamma_{(s[t']_{j.v'})|_i}(u') \Leftrightarrow i \in \mu(f) \wedge \gamma_{s|_i}(u') \Leftrightarrow \gamma_s(i.u') \Leftrightarrow \gamma_{t|_w}(i.u')$. Therefore, we conclude $\gamma_{t[t']_v}(u) \Leftrightarrow \gamma_t(w) \wedge \gamma_{t|_w}(i.u') \Leftrightarrow \gamma_t(w.i.u') \Leftrightarrow \gamma_t(u)$.

**Proof of Proposition 3** □

The following proposition establishes that replacement restrictions only depend on symbols lying on occurrences above a given occurrence.

**Proposition 4** *If $u \in O(t) \cap O(s)$ and $sprefix_t(u) = sprefix_s(u)$, then $u \in O^\mu(t) \Leftrightarrow u \in O^\mu(s)$.*

**Proof of Proposition 4** By induction on the length of $u$. If $u = \epsilon$, then the result is immediate, since $\epsilon \in O^\mu(t)$ for all $t$. Otherwise, let $u = v.w$. By Proposition 1, $u \in O^\mu(t) \Leftrightarrow v \in O^\mu(t) \wedge w \in O^\mu(t|_v)$. By the induction hypothesis, $v \in O^\mu(t) \Leftrightarrow v \in O^\mu(s)$ and $w \in O^\mu(t|_v) \Leftrightarrow w \in O^\mu(s|_v)$. Hence, by Proposition 1, $u \in O^\mu(t) \Leftrightarrow v \in O^\mu(s) \wedge w \in O^\mu(s|_v) \Leftrightarrow u \in O^\mu(s)$.

**Proof of Proposition 4** □

The previous results allow us to devise more efficient methods for evaluating the replacement condition in context-sensitive systems. In practice, this is important in implementing a method for computing the replacement condition. Our results allow us to do this compositionally and incrementally, with respect to the context-sensitive rewriting process that we describe in the following section. In particular, Proposition 3 expresses the independence of the calculus of the replacement condition from the context *below* and *around* a given occurrence of a term. Proposition 4 expresses this fact in a complementary way. These features are useful in parallel implementations of *csr* and, in general, in ensuring the locality of the calculus of the replacement condition. As a consequence, this entails the locality of context-sensitive computations.

The evaluation of the replacement condition is preserved by the ordering $\sqsubseteq$ on replacement maps.

**Proposition 5 (Monotonicity of $\gamma_t$, e.g., the Order $\sqsubseteq$)** *Let $\Sigma$ be a signature, $t \in \mathcal{T}(\Sigma, V)$, and $u \in O(t)$. Let $\mu, \mu'$ be two $\Sigma$-maps such that $\mu \sqsubseteq \mu'$. Then we get $\gamma_{\mu,t}(u) \Rightarrow \gamma_{\mu',t}(u)$.*

**Proof of Proposition 5** By induction on the length of $u$. If $u = \epsilon$, the result is immediate. For the induction step, let $u = i.u'$ and $t = f(t_1, \ldots, t_k)$. Since $\gamma_{\mu,t}(u) \Leftrightarrow i \in \mu(f) \wedge \gamma_{t_i}(u')$, if we have $\gamma_{\mu,t}(u)$, we also have $i \in \mu(f)$ and $\gamma_{\mu,t_i}(u')$. By the induction hypothesis, we get $\gamma_{\mu',t_i}(u')$, and since $i \in \mu(f) \Rightarrow i \in \mu'(f)$, then $i \in \mu'(f)$, and thus $\gamma_{\mu',t}(u)$.

**Proof of Proposition 5** □

In Figure 1, we summarize the previous properties, but we express them without using the replacement condition. Thus, they are given in terms of the set of replacing occurrences only. In some cases, a simpler formulation arises. We need the following propositions from Huet [Hue80], which are used below.

**Proposition 6 ([Hue80])** *Let $\Sigma$ be a signature, $t \in \mathcal{T}(\Sigma, V)$, and $\sigma$ be a substitution. Then we have $O(\sigma(t)) = O(t) \cup \bigcup_{t|_u \in V} \{u.v | v \in O(\sigma(t|_u))\}$, and therefore, for all $u \in O(t)$:*

1. *if $t|_u = t' \notin V$, then $\sigma(t)|_u = \sigma(t')$;*
2. *if $t|_u = x \in V$, then $\sigma(t)|_{u.v} = \sigma(x)|_v$ for all $v \in O(\sigma(x))$.*

**Proposition 7 ([Hue80])** *Let $\Sigma$ be a signature, $t, t' \in \mathcal{T}(\Sigma, V)$, and $\sigma$ be a substitution. Then, for all $u \in O(t)$, we have $\sigma(t)[\sigma(t')]_u = \sigma(t[t']_u)$.*

Assume the meaning of symbols to be as in the corresponding formal statements.

$$O^\mu(x) = \{\epsilon\}$$
$$O^\mu(f(\tilde{t})) = \{\epsilon\} \cup \bigcup_{i\in\mu(f)} i.O^\mu(t_i)$$

| | |
|---|---|
| **Proposition 1** | $u_1.u_2 \in O^\mu(t)$ if and only if $u_1 \in O^\mu(t) \wedge u_2 \in O^\mu(t\|_{u_1})$ |
| **Corollary 1(1)** | $u \le v \wedge v \in O^\mu(t) \Rightarrow u \in O^\mu(t)$ |
| **Corollary 1(2)** | $u \le v \wedge u \notin O^\mu(t) \Rightarrow v \notin O^\mu(t)$ |
| **Corollary 2** | $u_1.u_2 \in O^\mu(t) \Rightarrow u_2 \in O^\mu(t\|_{u_1})$ |
| **Proposition 2** | $O^\mu(t) \subseteq O^\mu(\sigma(t))$ |
| **Proposition 3(1)** | $u \le v \Rightarrow (u \in O^\mu(t) \Leftrightarrow u \in O^\mu(t[t']_v))$ |
| **Proposition 3(2)** | $u \parallel v \Rightarrow (u \in O^\mu(t) \Leftrightarrow u \in O^\mu(t[t']_v))$ |
| **Proposition 5** | $\mu \sqsubseteq \mu' \Rightarrow O^\mu(t) \subseteq O^{\mu'}(t)$ |

Figure 1: Properties of the replacing occurrences.

## 3.2 The Context-Sensitive Rewrite Relation

Next we introduce the context-sensitive rewrite relation. Essentially, we only allow replacements on occurrences that satisfy the replacement condition.

**Definition 4 (Context-Sensitive Rewrite Relation)** *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, and $\mu$ be a $\Sigma$-map. A term $t$ $\mu$-rewrites to a term $s$, written $t \overset{u}{\hookrightarrow}_{\mathcal{R}(\mu)} s$, if $t \xrightarrow{u}_{\mathcal{R}} s$ and $u \in O^\mu(t)$. The one-step context-sensitive rewrite relation of $\mathcal{R}$, e.g., $\mu$, is $\hookrightarrow_{\mathcal{R}(\mu)}$. The context-sensitive rewrite relation of $\mathcal{R}$, e.g., $\mu$, is $\hookrightarrow^*_{\mathcal{R}(\mu)}$.*

In the sequel, when it is clear from the context, we drop references to the TRS $\mathcal{R} = (\Sigma, R)$ and $\Sigma$-map $\mu$, writing $\hookrightarrow$ instead of $\hookrightarrow_{\mathcal{R}(\mu)}$. Let us now give a first example of *csr*.

**Example 2** *Consider the TRS and replacement map $\mu$ of Example 1, and the input expression $t = \mathsf{if}(\mathsf{and}(\mathsf{true},\mathsf{false}), \mathsf{0}+\mathsf{s}(\mathsf{0}), \mathsf{0})$. Then, since $1 \in O^\mu(t)$, we have (redexes underlined):*

$$\mathsf{if}(\underline{\mathsf{and}(\mathsf{true},\mathsf{false})}, \mathsf{0}+\mathsf{s}(\mathsf{0}), \mathsf{0}) \hookrightarrow \mathsf{if}(\mathsf{false}, \mathsf{0}+\mathsf{s}(\mathsf{0}), \mathsf{0})$$

*However,*

$$\mathsf{if}(\mathsf{and}(\mathsf{true},\mathsf{false}), \underline{\mathsf{0}+\mathsf{s}(\mathsf{0})}, \mathsf{0}) \not\hookrightarrow \mathsf{if}(\mathsf{and}(\mathsf{true},\mathsf{false}), \mathsf{s}(\mathsf{0}), \mathsf{0})$$

*since* $2$ *is not a* $\mu$*-replacing occurrence of* $t$*, i.e.,* $2 \notin O^\mu(t)$.

**Remark 1** *Note that with the (top)* $\Sigma$*-map* $\mu_\top$*,* $\gamma_{\mu_\top,t}(u)$ *holds for all terms* $t$ *and occurrences* $u \in O(t)$*. Hence,* $O^{\mu_\top}(t) = O(t)$*. That is to say that given a TRS* $\mathcal{R}$*, the context-sensitive rewrite relation for* $\mu_\top$ *coincides with the standard rewrite relation, i.e.,* $\hookrightarrow_{\mathcal{R}(\mu_\top)} = \rightarrow_{\mathcal{R}}$.

We denote as $O^\mu_{\mathcal{R}}(t) = O_{\mathcal{R}}(t) \cap O^\mu(t)$ the set of replacing redexes. If $O^\mu_{\mathcal{R}}(t) = \emptyset$, then $t$ is called a *$\mu$-normal form*.

The following proposition basically states that *csr* is closed under replacing context application, as is unrestricted term rewriting.

**Proposition 8 (Restricted Context Replacements)** *Let* $\mathcal{R} = (\Sigma, R)$ *be a TRS,* $\mu$ *be a* $\Sigma$*-map, and* $u \in O^\mu(C[t]_u)$*. If* $t \hookrightarrow^* s$*, then* $C[t]_u \hookrightarrow^* C[s]_u$.

**Proof of Proposition 8** By induction on the length $n$ of the $\mu$-derivation. If $n = 0$, the result is immediate. If $n > 0$, then we consider $t \hookrightarrow t' \hookrightarrow^* s$. If $t \hookrightarrow t'$, there is an occurrence $v \in O^\mu(t)$ and a rule $l \to r \in R$ such that $t|_v = \sigma(l)$ for some substitution $\sigma$. Since $u \in O^\mu(C[t]_u)$, by Proposition 1, $u.v \in O^\mu(C[t]_u)$. Therefore the same rule $l \to r$ applies to the occurrence $u.v$ of $C[t]_u$, and then $C[t]_u \hookrightarrow C[t']_u$. By Proposition 3(1), $u \in O^\mu(C[t']_u)$. Hence, by the induction hypothesis, the conclusion follows.

**Proof of Proposition 8** □

The following proposition shows how the context-sensitive relations induced by different replacement maps compare.

**Proposition 9 ( Monotonicity of $\hookrightarrow$, e.g. $\sqsubseteq$)** *Let* $\mathcal{R} = (\Sigma, R)$ *be a TRS and* $\mu, \mu'$ *be* $\Sigma$*-maps. Then,* $\mu \sqsubseteq \mu' \Rightarrow \hookrightarrow_{\mathcal{R}(\mu)} \subseteq \hookrightarrow_{\mathcal{R}(\mu')}$.

**Proof of Proposition 9** If $t \hookrightarrow_{\mathcal{R}(\mu)} s$, then there exist $u \in O^\mu(t)$, $l \to r \in R$, and substitution $\sigma$ such that $t|_u = \sigma(l)$. Since we have $\gamma_{\mu,t}(u)$, by Proposition 5 we also have $\gamma_{\mu',t}(u)$, and then $t \hookrightarrow_{\mathcal{R}(\mu')} s$.

**Proof of Proposition 9** □

From a computational point of view, this proposition mainly states that *csr* gives rise to a reduction space which is smaller than the one for standard term rewriting (consider Remark 1). Context-sensitive rewriting is also proven stable under substitution.

**Proposition 10 ( Stability of $\hookrightarrow$)** *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, $\mu$ be a $\Sigma$-map, and $\sigma$ be a substitution. Then, $t \hookrightarrow s \Rightarrow \sigma(t) \hookrightarrow \sigma(s)$.*

**Proof of Proposition 10** If $t \hookrightarrow s$, then there exist $u \in O^{\mu}(t)$, $l \to r \in R$, and substitution $\sigma'$ such that $t|_u = \sigma'(l)$ and $s = t[\sigma'(r)]_u$. Since $l \notin V$, then by Proposition 6 we have that $t|_u = t' \notin V$ and $\sigma(t)|_u = \sigma(t') = \sigma(\sigma'(l)) = \sigma''(l)$. By Proposition 2, $\gamma_{\sigma(t)}(u)$. Then $\sigma(t) \hookrightarrow \sigma(t)[\sigma(\sigma'(r))]_u$, and by Proposition 7, $\sigma(t) \hookrightarrow \sigma(t[\sigma'(r)]_u)$, i.e., $\sigma(t) \hookrightarrow \sigma(s)$.

**Proof of Proposition 10** $\square$

## 3.3 Applications

### 3.3.1 Improving the Evaluation of Expressions

As we illustrated in Examples 1 and 2, in some cases it makes sense to evaluate some fixed argument up to its normal form whenever this outcome is necessary to determine which rule has to be applied to an outer occurrence. The following example further develops the advantages of this strategy.

**Example 3** *Consider the rules that define the short-cut Boolean operators and/or of Lisp:*

$$\begin{array}{ll} \mathsf{and}(\mathsf{true},\mathsf{x}) \to \mathsf{x} & \mathsf{or}(\mathsf{true},\mathsf{x}) \to \mathsf{true} \\ \mathsf{and}(\mathsf{false},\mathsf{x}) \to \mathsf{false} & \mathsf{or}(\mathsf{false},\mathsf{x}) \to \mathsf{x} \end{array}$$

*To evaluate a term* $\mathsf{and}(\mathsf{t},\mathsf{s})$ ($\mathsf{or}(\mathsf{t},\mathsf{s})$)*, according to these rules, if the first argument reduces to* $\mathsf{false}(\mathsf{true})$*, any reduction performed on the second argument is useless and does not contribute to computing the intended value.*

*By using the replacement map $\mu(\mathsf{and}) = \mu(\mathsf{or}) = \{1\}$, the evaluation of redexes that are different from the first argument of the input term can be delayed.*

Another case of study arises in the realm of lists.

**Example 4** *The following rules define the standard "projection" operators* head *and* tail *on lists:*

$$\begin{array}{l} \mathsf{head}(\mathsf{x} :: \mathsf{y}) \to \mathsf{x} \\ \mathsf{tail}(\mathsf{x} :: \mathsf{y}) \to \mathsf{y} \end{array}$$

*The first function only requires the evaluation of the head of the list and the second one evaluates the rest of the list. These points are only of interest* *if cons (i.e., ::) does not evaluate its arguments systematically. Therefore, programming languages have been conceived where cons does not evaluate all arguments (see [FW76, HM76, Rea93]). This is easily achieved, for example, by defining $\mu(::) = \emptyset$.*

### 3.3.2 Manipulation of Infinite Data Structures

The lazy *cons* in Example 4 is a suitable tool for computing with infinite data structures [FW76, Rea93], as we illustrate in the following.

**Example 5** *Consider the following program, which selects one element from an infinite list.*

$$\begin{array}{ll} \mathsf{sel}(0, \mathsf{x} :: \mathsf{y}) \to \mathsf{x} & \qquad \mathsf{from}(\mathsf{x}) \to \mathsf{x} :: \mathsf{from}(\mathsf{s}(\mathsf{x})) \\ \mathsf{sel}(\mathsf{s}(\mathsf{x}), \mathsf{y} :: \mathsf{z}) \to \mathsf{sel}(\mathsf{x}, \mathsf{z}) & \end{array}$$

*We define $\mu(::) = \{1\}$ and $\mu(f) = \mathbb{N}^+_{ar(f)}$ for any other operator $f$. Let us consider the following derivation:*

$$\begin{array}{ll} \mathsf{sel}(\mathsf{s}(\mathsf{s}(0)), \underline{\mathsf{from}(0)}) & \hookrightarrow \underline{\mathsf{sel}(\mathsf{s}(\mathsf{s}(0)), 0 :: \mathsf{from}(\mathsf{s}(0)))} \\ & \hookrightarrow \mathsf{sel}(\mathsf{s}(0), \underline{\mathsf{from}(\mathsf{s}(0))}) \\ & \hookrightarrow \underline{\mathsf{sel}(\mathsf{s}(0), \mathsf{s}(0) :: \mathsf{from}(\mathsf{s}(\mathsf{s}(0))))} \\ & \hookrightarrow \mathsf{sel}(0, \underline{\mathsf{from}(\mathsf{s}(\mathsf{s}(0)))}) \\ & \hookrightarrow \underline{\mathsf{sel}(0, \mathsf{s}(\mathsf{s}(0)) :: \mathsf{from}(\mathsf{s}(\mathsf{s}(\mathsf{s}(0)))))} \\ & \hookrightarrow \mathsf{s}(\mathsf{s}(0)) \end{array}$$

*This avoids progressing into nontermination even if the* from*-rule is not terminating, thus improving termination as in lazy reduction strategies. In fact, termination of $\hookrightarrow$ for this TRS can be formally established (see below).*

### 3.3.3 Safe Computations

The concept of *safe computation* is the operational counterpart of the semantic concept of *strictness* [Lal93, Vui74]. Given a Scott domain[3] $D$, a function $f : D^k \to D$ is strict in the $i$-th argument if $f(x_1, \ldots, x_{i-1}, \perp, \ldots, x_k) = \perp$. By interpreting (as usual) the bottom element of a domain as a nonterminating computation, the strictness information can be used to define a computation strategy. If we evaluate a strict argument, then the *nontermination of that evaluation will not prevent a value from being produced* [Lal93].

[3]Any Scott domain has a least element which we denote as $\perp$; see [Sto77].

A reduction strategy that always evaluates on safe contexts is called a *safe* strategy. A context $C[\ ]_u$ is safe if $C[\bot]_u = \bot$. By defining the replacement map $\mu(f)$ as the set of *strict* arguments of $f$, for each symbol $f$ in the signature, it is immediate to see that any replacing context $C[\ ]_u$, with $u \in O^{\mu}(C)$, is a safe context. Then any context-sensitive derivation turns into a safe derivation. Strictness information for the functions can be eventually obtained from some kind of strictness analysis [Myc80, Wad87].

However, we note that this fact does not mean that the evaluation is complete whenever it is possible. This subject is addressed in the section below.

### 3.3.4 Mechanization of Inference Systems

Context-sensitive rewriting can be formulated as an inference system (see [Luc95]) by considering each rewrite rule $l \to r$ as a scheme of the axiom:

$$\frac{}{l \to r}$$

and introducing as many context-passing rules [Lal93] as replacing indices $i \in \mu(f)$ for each $k$-ary function symbol $f$ in the signature. That is, for each replacing argument position $i$, we define the scheme of the rule:

$$\frac{t \to s}{f(t_1, \ldots, t_{i-1}, t, \ldots, t_k) \to f(t_1, \ldots, t_{i-1}, s, \ldots, t_k)}$$

Different computational systems have been formalized using inference systems whose axioms and rules match these patterns, including weak $\beta$-reduction, call-by-name reduction strategies of $\lambda$-calculus, and $\pi$-calculus. Therefore, we can use *csr* as a suitable mechanization of the execution of processes of these systems. By treating these computational systems in a rewriting framework, we are able to exploit the existing rewriting machinery (reduction machines, graph reduction, etc.) and get efficient implementations of these systems on real engines. In particular, graph reduction offers great improvements in saving space and in removing redundant reductions when parallel reduction strategies are considered [GKM87, Mar90]. Aside from this, all the results of computational properties on *csr* are also applicable.

## 4 Confluence of Context-Sensitive Rewriting

A binary relation $R \subseteq A \times A$ on a set $A$ is *confluent* [Hue80] 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 $c\ R^*d$. Analogously, $R$ is said to be *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 $c\ R^*d$. Also, $R$ is said to be *strongly confluent* if, for all $a, b, c \in A$, $a\ R\ b \wedge a\ R\ c \Rightarrow b\ R^*d \wedge c\ R^\epsilon d$ for some $d \in A$, with $R^\epsilon$ is the reflexive closure of $R$, with $R^\epsilon = R \cup \{(a, a) \mid a \in A\}$.

An element $\bar{a} \in A$ is said to be an $R$-normal form if there exists no $b$ such that $\bar{a}\ R\ b$. We say that $\bar{a}$ is an $R$-normal form of $a$, if $\bar{a}$ is an $R$-normal form and $a\ R^*\bar{a}$. We say that $R$ is terminating [Der87, Hue80] if and only if there is no infinite sequence $a_1\ R\ a_2\ R\ a_3 \ldots$. In a terminating relation, each element $a \in A$ has at least a normal form. In a confluent terminating relation, the normal form exists and it is unique.

We need to distinguish the computational properties of the standard rewrite relation and the analogous ones in *csr*. Therefore we will speak of $\mu$-confluence and $\mu$-termination for the confluence and termination properties of $\mu$-rewriting, i.e., the *csr* which uses the replacement map $\mu$.

Because we prove confluence of *csr* by using Newman's lemma, "a terminating, locally confluent relation is confluent," we briefly recall the existing methods to prove termination of *csr* or, more accurately, *$\mu$-termination* for a given replacement map $\mu$ [Luc96b, Zan97].

### 4.1 Proving $\mu$-Termination

We prove termination of *csr* by using standard methods in rewriting [Luc96b].

**Theorem 1 ([Luc96b])** *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, and $\mu$ be a $\Sigma$-map. If $\mathcal{R}$ terminates, then $\mathcal{R}$ $\mu$-terminates.*

For example, the TRSs in Examples 1, 3, and 4 prove to be $\mu$-terminating: the instances of these rules can be oriented from left to right by means of a simplification ordering [Der87]. Many other TRSs are $\mu$-terminating despite the fact that this simple technique does not apply.

**Example 6** *Consider the nonterminating TRS $\mathcal{R}$:*

$$\begin{array}{ll} \mathsf{first}(0, \mathsf{x}) \to [] & \mathsf{from}(\mathsf{x}) \to \mathsf{x} :: \mathsf{from}(\mathsf{s}(\mathsf{x})) \\ \mathsf{first}(\mathsf{s}(\mathsf{x}), \mathsf{y} :: \mathsf{z}) \to \mathsf{y} :: \mathsf{first}(\mathsf{x}, \mathsf{z}) & \end{array}$$

*If we fix $\mu$ to be $\mu(::) = \mu(\mathsf{from}) = \emptyset$, $\mu(\mathsf{s}) = \{1\}$, and $\mu(\mathsf{first}) = \{1, 2\}$, then it is easy to verify that any $\mu$-rewriting sequence from a given term $t$ using $\mathcal{R}$ will eventually terminate.*

A formal proof of $\mu$-termination for the TRS in Example 6 can be given by using standard methods also [Luc96b]. The proof uses a transformed program $\mathcal{R}^\mu$. The rules in $\mathcal{R}^\mu$ are obtained from the rules in $\mathcal{R}$ by recursively removing from the lhs and rhs of each rule the nonreplacing immediate subterms of each term. Since the symbols of the original signature lose arguments, we use new symbols from a new signature $\Sigma^\mu$: $f_\mu \in \Sigma^\mu$ if and only if $f \in \Sigma$. The arity of each $f_\mu \in \Sigma^\mu$ is $ar_{\Sigma^\mu}(f_\mu) = |\mu(f)|$. In the following theorem, by a *$\mu$-conservative* TRS we mean a TRS that verifies $Var^\mu(r) \subseteq Var^\mu(l)$ for all rule $l \to r$, i.e., every $\mu$-replacing variable in the rhs is also $\mu$-replacing in the lhs. Then we have the following theorem.

**Theorem 2 ([Luc96b])** *Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu$ be a $\Sigma$-map such that $\mathcal{R}$ is $\mu$-conservative. If $\mathcal{R}^\mu$ terminates, then $\mathcal{R}$ $\mu$-terminates.*

If $\mathcal{R}$ is not $\mu$-conservative, then $\mathcal{R}^\mu$ will have extra variables in some rhs. Hence it is not a standard TRS, and the available methods for proving termination do not apply. For instance, $\mathcal{R}$ in Example 6 is $\mu$-conservative (with $\mu$ as given in the example). We can ensure that $\mathcal{R}$ $\mu$-terminates, because we can prove that $\mathcal{R}^\mu$:

$$\begin{array}{ll} \mathsf{first}_\mu(\mathsf{0}_\mu, \mathsf{x}) \to []_\mu & \mathsf{from}_\mu \to ::_\mu \\ \mathsf{first}_\mu(\mathsf{s}_\mu(\mathsf{x}), ::_\mu) \to ::_\mu & \end{array}$$

terminates. For more details and a discussion on the limitations of the method, refer to [Luc96b].

TRSs which are not $\mu$-conservative can be proven $\mu$-terminating by using the methods in [Zan97]. For instance, the TRS $\mathcal{R}$ in Example 5 is not $\mu$-conservative (being $\mu$ as given in the example). In fact, $\mathcal{R}^\mu$ is:

$$\begin{array}{ll} \mathsf{sel}_\mu(\mathsf{0}, ::_\mu(\mathsf{x})) \to \mathsf{x} & \mathsf{from}_\mu(\mathsf{x}) \to ::_\mu(\mathsf{x}) \\ \mathsf{sel}_\mu(\mathsf{s}_\mu(\mathsf{x}), ::_\mu(\mathsf{y})) \to \mathsf{sel}_\mu(\mathsf{x}, \mathsf{z}) & \end{array}$$

which has an extra variable $\mathsf{z}$ in the rhs of the second $\mathsf{sel}_\mu$ rule. In [Zan97], a method for proving $\mu$-termination by transformation of the TRS is also given, but the nonreplacing occurrences are marked, rather than removed. Then, from a TRS $\mathcal{R}$ and a replacement map $\mu$, we obtain $\Psi(\mathcal{R}, \mu)$, which is always a TRS. For instance, in the previous case, we obtain the following $\Psi(\mathcal{R},\mu)$:

$$\begin{array}{ll}
\mathsf{sel}(0,\mathsf{x}::\mathsf{y})\to\mathsf{x} & \mathsf{from}(\mathsf{x})\to\mathsf{x}::\overline{\mathsf{from}}(\mathsf{s}(\mathsf{x}))\\
\mathsf{sel}(\mathsf{s}(\mathsf{x}),\mathsf{y}::\mathsf{z})\to\mathsf{sel}(\mathsf{x},\mathsf{a}(\mathsf{z})) & \\
\mathsf{a}(\overline{\mathsf{from}}(\mathsf{x}))\to\mathsf{from}(\mathsf{x}) & \mathsf{from}(\mathsf{x})\to\overline{\mathsf{from}}(\mathsf{x})\\
\mathsf{a}(\mathsf{x})\to\mathsf{x} &
\end{array}$$

which can be proved terminating by using recursive path ordering (see [Der87]). Then, we have the following result.

**Theorem 3 ([Zan97])** *A TRS $\mathcal{R}$ is $\mu$-terminating if $\Psi(\mathcal{R},\mu)$ is terminating.*

However, as remarked in [Zan97], this method does not subsume the previous one. For instance, the TRS $\mathcal{R}$:

$$\mathsf{f}(\mathsf{x})\to\mathsf{g}(\mathsf{h}(\mathsf{f}(\mathsf{x})))$$

with $\mu(\mathsf{f})=\mu(\mathsf{h})=\{1\}$ and $\mu(\mathsf{g})=\varnothing$ is $\mu$-terminating because $\mathcal{R}^\mu$:

$$\mathsf{f}_\mu(\mathsf{x})\to\mathsf{g}_\mu$$

is easily proved terminating. However, $\Psi(\mathcal{R},\mu)$:

$$\begin{array}{l}
\mathsf{f}(\mathsf{x})\to\mathsf{g}(\overline{\mathsf{h}}(\mathsf{f}(\mathsf{x})))\\
\mathsf{a}(\overline{\mathsf{h}}(\mathsf{x}))\to\mathsf{h}(\mathsf{x})\\
\mathsf{h}(\mathsf{x})\to\overline{\mathsf{h}}(\mathsf{x})\\
\mathsf{a}(\mathsf{x})\to\mathsf{x}
\end{array}$$

is not terminating owing to the first rule.

## 4.2 Local Confluence

The *diamond lemma* (Newman, [HL91, New42]) establishes that any terminating, locally confluent relation is confluent. In this section, we center our discussion on analyzing local $\mu$-confluence, to obtain results on $\mu$-confluence for $\mu$-terminating TRSs.

In standard rewriting, a terminating TRS with nonoverlapping rules is confluent [Hue80]. In *csr*, the fact that the rules do not overlap does not imply $\mu$-confluence, as the following example illustrates.

**Example 7** *Consider the following nonoverlapping TRS $\mathcal{R}$.*

$$\begin{array}{l} \mathsf{f(x)} \to \mathsf{g(x,x)} \\ \mathsf{h(0)} \to \mathsf{0} \end{array}$$

*If we define $\mu(\mathsf{f}) = \{1\}$ and $\mu(\mathsf{g}) = \{1\}$, then we have the following $\mu$-rewriting chains leading to two different $\hookrightarrow$-normal forms for the considered input term:*

$$\begin{array}{l} \underline{\mathsf{f(h(0))}} \hookrightarrow \mathsf{g(}\underline{\mathsf{h(0)}}\mathsf{,h(0))} \hookrightarrow \mathsf{g(0,h(0))} \\ \mathsf{f(}\underline{\mathsf{h(0)}}\mathsf{)} \hookrightarrow \underline{\mathsf{f(0)}} \hookrightarrow \mathsf{g(0,0)} \end{array}$$

The following definition initiates the description of the class of TRSs that we can guarantee to be locally $\mu$-confluent. The main idea is that if there is any variable occurrence in the lhs of a rewrite rule which satisfies the replacement condition, then any other occurrence of this variable in the rule must be a replacing occurrence too.

**Definition 5 (TRS with Left Homogeneous Replacing Variables)** *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, and $\mu$ be a $\Sigma$-map. Consider a rule $\alpha : l \to r \in R$. We say that $\alpha$ has left homogeneous $\mu$-replacing variables if, for all $x \in Var^\mu(l)$, $O_x(l) = O^\mu_x(l)$ and $O_x(r) = O^\mu_x(r)$. The TRS $\mathcal{R}$ has left homogeneous $\mu$-replacing variables if all rules in $R$ have left homogeneous $\mu$-replacing variables.*

As usual, we drop references to the replacement map $\mu$ when no confusion can arise. For example, the TRSs in Examples 1, 3, and 4 have left homogeneous replacing variables. The TRS in Example 5 has no left homogeneous replacing variables (e.g., the replacement map that is given in the example).

This restriction constrains the canonical nature of a TRS. As a counterpart, it is more difficult for the rules to $\mu$-overlap. First, we introduce the concept of $\mu$-overlapping terms.

**Definition 6 ( $\mu$-Overlapping Terms)** *Let $t, s \in \mathcal{T}(\Sigma, V)$, and $\mu$ be a $\Sigma$-map. The term $s$ $\mu$-overlaps $t$ at the occurrence $u$, if $u \in O^\mu_\Sigma(t)$ and $t|_u =^? s$. Terms $t$ and $s$ are not $\mu$-overlapping if neither $s$ $\mu$-overlaps $t$ nor $t$ $\mu$-overlaps $s$.*

Now, a suitable notion of non-$\mu$-overlapping TRSs can be given.

**Definition 7 (Non-$\mu$-Overlapping TRS)** *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, and $\mu$ be a $\Sigma$-map. Given two rules $\alpha_1 : l_1 \to r_1, \alpha_2 : l_2 \to r_2 \in R$ such that $l_1$ and $l_2$ have no common variable (otherwise rename rules), $\alpha_1$ $\mu$-overlaps $\alpha_2$, if $l_1$ $\mu$-overlaps $l_2$. The rules $\alpha_1$ and $\alpha_2$ are trivially $\mu$-overlapping if $\alpha_1 = \alpha_2$ and $\alpha_1, \alpha_2$ $\mu$-overlap at $\epsilon$. The TRS $\mathcal{R}$ is non-$\mu$-overlapping if it does not contain nontrivial overlapping rules.*

Now we introduce the adequate notion of the critical pair [Hue80] regarding *csr*.

**Definition 8 ($\mu$-Critical Pair)** *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, and $\mu$ be a $\Sigma$-map. Let $\alpha_1 : l_1 \to r_1$ and $\alpha_2 : l_2 \to r_2$ be rewrite rules that are nontrivially $\mu$-overlapping at the occurrence $u \in O^\mu_\Sigma(l_1)$. Let $t = l_1|_u$ and $t' \equiv t \sqcup l_2$, with $Var(t') \cap Var(l_1) = \emptyset$. The superposition of $\alpha_1$ and $\alpha_2$ determines a $\mu$-critical pair $\langle t_1, t_2 \rangle$ defined by $t_1 = \sigma_1(l_1)[\sigma_2(r_2)]_u$ and $t_2 = \sigma_1(r_1)$, where $\sigma_1$ is the matching of $t'$ and $t$ ($t' = \sigma_1(t)$) and $\sigma_2$ is the matching of $t'$ and $l_2$ ($t' = \sigma_2(l_2)$).*

Note that any $\mu$-critical pair of the TRS is also a standard critical pair. The following example shows how the replacement restrictions can be used to avoid some overlaps which otherwise could lead to nonconvergent reductions.

**Example 8** *Consider the following TRS $\mathcal{R}$:*

$$\begin{array}{l} \mathsf{g}(\mathsf{b}, \mathsf{a}) \to \mathsf{b} \\ \mathsf{a} \to \mathsf{b} \end{array}$$

*and a replacement map $\mu$ such that $\mu(\mathsf{g}) = \{1\}$. This TRS has overlapping rules. However, since $2$ is not a replacing occurrence in $\mathsf{g}(\mathsf{b}, \mathsf{a})$, then there is no $\mu$-overlap. Hence, there is no $\mu$-critical pair in the program.*

**Proposition 11** *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, and $\mu$ be a $\Sigma$-map. Let $t \in \mathcal{T}(\Sigma, V)$, $x \in V$, and $O_x(t) = \{u_i\}_{i=1}^n$, for $n \geq 0$. Let $\sigma$, $\sigma'$ be substitutions such that $\sigma(x) \hookrightarrow \sigma'(x)$ and for all $y \neq x$. $\sigma(y) = \sigma'(y)$. Let $s_0 = \sigma(t)$ and $s_i = s_{i-1}[\sigma'(x)]_{u_i}$, $1 \leq i \leq n$. If $O_x(t) = O^\mu_x(t)$, then $s_i \hookrightarrow^{n-i} \sigma'(t)$, $0 \leq i \leq n$.*

**Proof of Proposition 11** By induction on $n = |O_x(t)|$. If $n = 0$, the result is immediate. If $n > 0$, consider $u_1 \in O_x(t)$. Assume that $s_1 \hookrightarrow^{n-1} \sigma'(t)$. Since $\gamma_t(u_1)$, then by Proposition 2, $\gamma_{\sigma(t)}(u_1)$. By Proposition 6, $\sigma(t)|_{u_1} =$

$\sigma(x)$. Now, since $\sigma(x) \hookrightarrow \sigma'(x)$, by Proposition 8, we have $\sigma(t) = s_0 \hookrightarrow s_1 = \sigma(t)[\sigma'(x)]_{u_1}$. Hence, by Proposition 3(2) and by the induction hypothesis, the conclusion follows.

**Proof of Proposition 11** $\square$

The following proposition follows directly from Proposition 3.7 in Huet [Hue80], since a $\mu$-critical pair is just a critical pair from $\mu$-overlapping rules.

**Proposition 12** *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, and $\mu$ be a $\Sigma$-map. Let $l_1 \to r_1$ and $l_2 \to r_2$ be two rewrite rules in $\mathcal{R}$, $u \in O^\mu(l_1)$, $t = l_1|_u \notin V$, and $\sigma_1, \sigma_2$ be substitutions such that $\sigma_1(t) = \sigma_2(l_2)$. Then there exists a $\mu$-critical pair $\langle t_1, t_2 \rangle$ of $\mathcal{R}$ and a substitution $\theta$ such that $\sigma_1(l_1)[\sigma_2(r_2)]_u = \theta(t_1)$ and $\sigma_1(r_1) = \theta(t_2)$.*

In the following theorem, we write $t \downarrow t'$ to denote the context-sensitive joinability, i.e., $t \downarrow t'$ if there exists $s$ such that $t \hookrightarrow^* s$ and $t' \hookrightarrow^* s$.

**Theorem 4** *Let $\mathcal{R} = (\Sigma, R)$ be a TRS with left homogeneous $\mu$-replacing variables, e.g., a $\Sigma$-map $\mu$. Then, $\mathcal{R}$ is locally $\mu$-confluent if and only if for every $\mu$-critical pair $\langle t_1, t_2 \rangle$ we have $t_1 \downarrow t_2$.*

**Proof of Theorem 4** The proof is similar to the proof of Lemma 3.1 in Huet [Hue80] and adapted to consider the replacement maps in *csr*. Using the notations in Definition 8, we proceed as follows. Let $\langle t_1, t_2 \rangle$ be a $\mu$-critical pair in $\mathcal{R}$.

- ($\Rightarrow$) Since $\gamma_{l_1}(u)$, by Proposition 2, $\sigma_1(l_1) \hookrightarrow t_1$. By stability of $\hookrightarrow$ (Proposition 10), $\sigma_1(l_1) \hookrightarrow t_2$. Since $\hookrightarrow$ is locally confluent, then $t_1 \downarrow t_2$.
- ($\Leftarrow$) Assume that for every $\mu$-critical pair $\langle t_1, t_2 \rangle$ in $\mathcal{R}$, $t_1 \downarrow t_2$. Let $t, t', t''$ be terms such that $t \hookrightarrow t'$ and $t \hookrightarrow t''$, that is, there exist rules $l_1 \to r_1$, $l_2 \to r_2 \in \mathcal{R}$, occurrences $u_1, u_2 \in O^\mu(t)$, and substitutions $\sigma_1, \sigma_2$ such that $t|_{u_1} = \sigma_1(l_1)$, $t|_{u_2} = \sigma_2(l_2)$, and $t' = t[\sigma_1(r_1)]_{u_1}$, $t'' = t[\sigma_2(r_2)]_{u_2}$.

Consider two cases according to the relative positions of the two redexes.

1. $u_1 \parallel u_2$, i.e., the occurrences $u_1$ and $u_2$ are disjoint. By the persistence property and Proposition 3, we have $t'|_{u_2} = \sigma_2(l_2)$, $\gamma_{t'}(u_2)$, $t''|_{u_1} = \sigma_1(l_1)$, and $\gamma_{t''}(u_1)$. By commutativity, $s = t'[\sigma_2(r_2)]_{u_2} = t''[\sigma_1(r_1)]_{u_1}$, and then $t' \hookrightarrow s$ and $t'' \hookrightarrow s$.

2. If $u_1, u_2$ are not disjoint, without loss of generality, we can assume $u_1 \leq u_2$ (the case $u_2 \leq u_1$ is perfectly analogous). Let $u_2 = u_1.v$. By the cancellation property; $\sigma_1(l_1)|_v = \sigma_2(l_2)$, and by distributivity, $t''|_{u_1} = \sigma_1(l_1)[\sigma_2(r_2)]_v$. Since $u_2 = u_1.v$ and $\gamma_t(u_2)$, then by Corollary 2, we get $\gamma_{t|_{u_1}}(v)$ or $\gamma_{\sigma_1(l_1)}(v)$. Therefore, $\sigma_1(l_1) \hookrightarrow \sigma(l_1)[\sigma_2(r_2)]_v$.

Now we have to prove that there exists $s$ such that $\sigma_1(r_1) \hookrightarrow^* s$ and $\sigma_1(l_1)[\sigma_2(r_2)]_v \hookrightarrow^* s$. Then the conclusion $t' \downarrow t''$ follows by Proposition 8. According to Proposition 6, there are two cases to consider:

(a) $v = v_1.v_2$, $l_1|_{v_1} = x \in V$, $\sigma_2(l_2) = \sigma_1(x)|_{v_2}$.
We consider the substitution $\sigma'_1$ defined by $\sigma'_1(x) = \sigma_1(x)[\sigma_2(r_2)]_{v_2}$ and $\sigma'_1(y) = \sigma_1(y)$, $\forall y \neq x$. Then we take the term $s = \sigma'_1(r_1)$. It suffices to show that $\sigma_1(x) \hookrightarrow \sigma'_1(x)$: since $\sigma_1(x)|_{v_2} = \sigma_2(l_2)$, then $\sigma_1(x)$ rewrites at occurrence $v_2$ using $l_2 \to r_2$, i.e., $\sigma_1(x) \to \sigma'_1(x)$. Let us show $\gamma_{\sigma_1(x)}(v_2)$: because, by Proposition 6, $\sigma_1(l_1)|_{v_1} = \sigma_1(x)$ and $\gamma_{\sigma_1(l_1)}(v) \Leftrightarrow \gamma_{\sigma_1(l_1)}(v_1.v_2)$, then, by Corollary 2, $\gamma_{\sigma_1(x)}(v_2)$. Hence $\sigma_1(x) \hookrightarrow \sigma'_1(x)$. Also, by Corollary 1, $\gamma_{\sigma_1(l_1)}(v_1)$, and by Proposition 2, $\gamma_{l_1}(v_1)$ since $v_1 \in O(l_1)$. Hence $l_1|_{v_1} = x$ is a replacing variable, $x \in Var^\mu(l_1)$. Therefore, since $\mathcal{R}$ is a TRS with left homogeneous replacing variables, every other occurrence of $x$ in $l_1$ and $r_1$ satisfies the corresponding replacement condition, and by Proposition 11 we have $\sigma_1(r_1) \hookrightarrow^* \sigma'_1(r_1)$ and $\sigma_1(l_1)[\sigma_2(r_2)]_v \hookrightarrow^* \sigma'_1(l_1)$. By stability of $\hookrightarrow$, we have $l_1 \hookrightarrow r_1 \Rightarrow \sigma'_1(l_1) \hookrightarrow \sigma'_1(r_1)$ and $\sigma_1(l_1)[\sigma_2(r_2)]_v \hookrightarrow^* s$. Recall that $t'|_{u_1} = \sigma_1(r_1)$, $t''|_{u_1} = \sigma_1(l_1)[\sigma_2(r_2)]_v$ and, by Proposition 3, $\gamma_{t'}(u_1)$, $\gamma_{t''}(u_1)$. Then by applying Proposition 8 to context $t[\quad]_{u_1}$, $t \hookrightarrow t' \wedge t \hookrightarrow t'' \Rightarrow t' \downarrow t''$.

(b) $\exists l \notin V$ such that $l = l_1|_v$ and $\sigma_2(l_2) = \sigma_1(l)$.
Since $\sigma_1(l_1)$ $\mu$-rewrites to $\sigma_1(l_1)[\sigma_2(r_2)]_v$ using $l_2 \to r_2$ and $\gamma_{\sigma_1(l_1)}(v)$, by Proposition 12, there exists a $\mu$-critical pair $\langle t_1, t_2 \rangle$ and a substitution $\theta$ such that $\sigma_1(l_1)[\sigma_2(r_2)]_v = \theta(t_1)$ and $\sigma(r_1) = \theta(t_2)$. By hypothesis, there exists $t_3$ such that $t_1 \hookrightarrow^* t_3$ and $t_2 \hookrightarrow^* t_3$. Then we take $s = \theta(t_3)$, and the result follows by Proposition 8 and the stability of $\hookrightarrow$.

**Proof of Theorem 4** □

## 4.3 Strong Confluence

By proving strong confluence of a relation, we can also conclude confluence. The procedure can be sketched as follows [Hue80]. Let $R, S \subseteq A \times A$ be relations such that $R^+ = S^+$. If $S$ is strongly confluent, then $R$ is confluent. Clearly, by taking $S = R$, strong confluence of $R$ implies confluence of $R$. However, we have a more interesting application of this result when $R \neq S$.

When we consider orthogonal TRSs, the *parallel moves lemma* is a well-known result in rewriting which can be used to establish strong confluence of *parallel* rewriting, i.e., simultaneous rewriting of redexes at disjoint occurrences. An *elementary multiderivation* simultaneously contracts a set $U$ of disjoint occurrences of redexes (written $t \xrightarrow{U} s$) [HL91]. We write $t \;\,\mathrel{+\!\!\!\!+\!\!\!\!\rightarrow} s$ when the information about the concrete contracted occurrences is not relevant. Given the elementary derivations $A : t \xrightarrow{u} t'$ and $B : t \xrightarrow{v} t''$ (which are also elementary multiderivations if we write $A : t \xrightarrow{\{u\}} t'$ and $B : t \xrightarrow{\{v\}} t''$), the parallel moves lemma defines elementary multiderivations denoted as $B\backslash A : t' \xrightarrow{U} s$ and $A\backslash B : t'' \xrightarrow{W} s$, which *converge* to a common reduct $s$ [HL91]. This proves that $\mathrel{+\!\!\!\!+\!\!\!\!\rightarrow}$ is strongly confluent. Since $\rightarrow^+ = \;\mathrel{+\!\!\!\!+\!\!\!\!\rightarrow}^+$, it easily follows confluence of $\rightarrow$ for orthogonal TRSs.

To generalize this result to *csr*, we analyze the requirements of the parallel moves lemma. First, we consider the standard notion of *residual* by a derivation.

**Definition 9 ([HL91])** *Given an orthogonal TRS $\mathcal{R} = (\Sigma, R)$, and an elementary derivation $A : t \xrightarrow{[u,\alpha]} s$, where $\alpha : l \rightarrow r \in R$, and a redex occurrence $v \in O_{\mathcal{R}}(t)$, the set $v\backslash A$ of residuals of redex $t|_v$ by $A$ is a subset of $O(s)$ as follows:*

$$v\backslash A = \begin{cases} \emptyset & \textit{if } v = u & \\ \{v\} & \textit{if } v \parallel u & (i) \\ \{v\} & \textit{if } v < u & (ii) \\ \{u.w_1.v_1 \mid r|_{w_1} = x\} & \textit{if } v = u.w.v_1 \textit{ and } l|_w = x \in V & (iii) \end{cases}$$

For any nonelementary derivation $A$, we define $v\backslash A$ to be $v\backslash 0 = \{v\}$ (0 is the empty derivation), and $v\backslash(AB) = \bigcup_{w \in v\backslash A} w\backslash B$. The calculus of residuals $u\backslash A$ of a redex occurrence $u$ extends to sets of disjoint redex occurrences as follows: $U\backslash A = \bigcup_{u \in U} u\backslash A$. Given derivations $A, B$, starting from $t$ with $B$ being an elementary multiderivation contracting the set $U \subseteq O_{\mathcal{R}}(t)$, the residual derivation $B\backslash A$ of $B$ by $A$ is the elementary multiderivation contracting the set $U\backslash A$.

**Definition 10** *Given a TRS $\mathcal{R} = (\Sigma, R)$, a $\Sigma$-map $\mu$, and an elementary $\mu$-derivation $A^\mu : t \overset{[u,\alpha]}{\hookrightarrow}_{\mathcal{R}(\mu)} s$, where $\alpha : l \to r \in R$, and $v \in O^\mu_{\mathcal{R}}(t)$, the set $v\backslash A^\mu$ of* residuals *of $t|_v$ by $A^\mu$ is a subset of $O(s)$ as follows: $v\backslash A^\mu = v\backslash A$, where $A : t \overset{[u,\alpha]}{\to} s$.*

Because any $\mu$-derivation is also a derivation, the previous definition is correct. The extension of the notion of mutiderivations to *csr* is also straightforward: $A^\mu : t \overset{U}{\hookrightarrow} s$ is an elementary $\mu$-multiderivation which contracts the disjoint redexes at occurrences $U \subseteq O^\mu_{\mathcal{R}}(t)$, i.e., $\forall u, u' \in U.\ u \neq u' \Rightarrow u \parallel u'$. We also write $t \mathrel{+\!\!\!\!\hookrightarrow} s$ when the information about the contracted redex occurrences is not relevant.

The fact that $v\backslash A \subseteq O_{\mathcal{R}}(s)$ is essential to define the concept of residual derivation, etc. It means that the derivation $A$ (which contracts the redex $t|_u$) left the possibility of reducing the residuals of the redex $t|_v$ unchanged in further reduction steps.

We need to keep the analogous property in $\mu$-derivations $A^\mu : t \overset{u}{\hookrightarrow} s$, i.e., $v \in O^\mu_{\mathcal{R}}(t) \Rightarrow v\backslash A^\mu \subseteq O^\mu_{\mathcal{R}}(s)$. However, a residual of a replacing redex does not need to be a replacing redex.

**Example 9** *Consider the orthogonal TRS $\mathcal{R}$ in Example 7. If we define $\mu(\mathsf{f}) = \{1\}$ and $\mu(\mathsf{g}) = \{1\}$, then we have the following $\mu$-derivation:*

$$A^\mu : \underline{\mathsf{f}(\mathsf{h}(\mathsf{0}))} \hookrightarrow \mathsf{g}(\mathsf{h}(\mathsf{0}), \mathsf{h}(\mathsf{0}))$$

*The residuals of the replacing redex $\mathsf{h}(\mathsf{0})$ in $t = \mathsf{f}(\mathsf{h}(\mathsf{0}))$ are $\{1, 2\}$. However, $2 \notin O^\mu(\mathsf{g}(\mathsf{h}(\mathsf{0}), \mathsf{h}(\mathsf{0})))$, and it is not a replacing redex occurrence.*

Note that the problem arises with replacing redex occurrences $v \in O^\mu_{\mathcal{R}}(t)$ that are *below* the occurrence $u \in O^\mu_{\mathcal{R}}(t)$ which the elementary $\mu$-derivation $A^\mu$ contracts. This is because the definition of residual that concerns this case, i.e., (iii) in Definition 9, introduces occurrences $w_1 \in O_V(r)$ to obtain the corresponding residual $u.w_1.v_1$. However, to ensure that $u.w_1.v_1$ is a replacing occurrence of a redex, we must ensure $w_1 \in O^\mu(r)$ as well. The requirement of having left homogeneous $\mu$-replacing variables, which we introduced in Section 4.2, also solves this problem. We note that for left-linear

TRSs (hence for orthogonal TRSs as well), the requirement of left homogeneous $\mu$-replacing variables is simpler: for all $x \in Var^\mu(l)$, $O_x(r) = O^\mu_x(r)$ in every rule $l \to r$. TRSs with left homogeneous $\mu$-replacing variables are expected to fulfill the following: instances of replacing variables in lhs's are replacing in the instantiated rhs's. Hence, when we consider TRSs having left homogeneous replacing variables, residuals of replacing redexes are also replacing.

We also note that orthogonality is essential to ensure that Definition 9 considers all cases. In fact, the case $v > u$, which is treated in part (iii) of the definition, would be incomplete without orthogonality. Moreover, the case (ii), $v < u$, could produce that $v \notin O_{\mathcal{R}}(s)$ without orthogonality. However, in orthogonal TRSs, the reduction of a redex $t|_u$ that is below a redex $t|_v$ does not compromise the possibility of reducing $s|_v$, where $s = t[\sigma(r)]_u$.

We can relax the requirement of orthogonality and still ensure the consistency of Definition 10 by forbidding the replacement of overlapping redexes, using the replacement restrictions. Hence, the generalization of orthogonal TRSs in the presence of a replacement map $\mu$ is as follows:

**Definition 11 ($\mu$-Orthogonal TRS)** *Given a replacement map $\mu$, a left-linear TRS $\mathcal{R}$ is $\mu$-orthogonal if $\mathcal{R}$ is non-$\mu$-overlapping.*

**Example 10** *Consider the TRS $\mathcal{R}$:*

$$\begin{array}{ll} \mathsf{f}(\mathsf{x},\mathsf{y}) \to \mathsf{g}(\mathsf{y},\mathsf{y}) & \mathsf{a} \to \mathsf{c} \\ \mathsf{g}(\mathsf{x},\mathsf{a}) \to \mathsf{f}(\mathsf{x},\mathsf{b}) & \mathsf{b} \to \mathsf{b} \end{array}$$

*and a replacement map $\mu$ such that $\mu(\mathsf{f}) = \mu(\mathsf{g}) = \{1\}$. Then, $\mathcal{R}$ is $\mu$-orthogonal. Note that $\mathcal{R}$ also has left homogeneous $\mu$-replacing variables.*

**Proposition 13** *Let $\mathcal{R} = (\Sigma, R)$ be a $\mu$-orthogonal TRS having left homogeneous $\mu$-replacing variables, e.g., a $\Sigma$-map $\mu$. Let $A^\mu : t \overset{[u,\alpha]}{\hookrightarrow}_{\mathcal{R}(\mu)} s$ be an elementary $\mu$-derivation, where $\alpha : l \to r \in R$, and let $v \in O^\mu_{\mathcal{R}}(t)$ be a replacing redex occurrence. Then $v\backslash A^\mu \subseteq O^\mu_{\mathcal{R}}(s)$.*

**Proof of Proposition 13** By Definition 10, $v\backslash A^\mu = v\backslash A$; therefore, we consider the cases of Definition 9. Recall that $s = t[\sigma(r)]_u$, and $u, v \in O^\mu_{\mathcal{R}}(t)$. Assume that $t \overset{[v,\alpha']}{\to} s'$ for some $\alpha' : l' \to r'$. The case $u = v$ is immediate.

1. $v \parallel u$. Then $v\backslash A^\mu = \{v\}$. Since $v \in O^\mu(t)$, we have $\gamma_t(v)$. Therefore, by Proposition 3(2), $\gamma_{t[\sigma(r)]_u}(v)$, i.e., $\gamma_s(v)$. Hence, $v \in O^\mu(s)$, and moreover, $v \in O^\mu_{\mathcal{R}}(s)$.

2. $v < u$. Similar to the previous point, using Proposition 3(1), we conclude $v \in O^\mu(s)$. Now let us show that $v \in O_{\mathcal{R}}(s)$. By contradiction: if $v \notin O_{\mathcal{R}}(s)$, then the reduction of the redex occurrence $u$ has destroyed the redex. Hence, it must be $u = v.w$ and $w \in O_\Sigma(l')$ (otherwise, if $w \in O_V(l')$ or $w \notin O(l')$, no problem arises). However, since $t|_v = \sigma'(l')$ and $t|_u = \sigma(l) = \sigma'(l')|_w = \sigma'(l'|_w)$, then $\alpha$ and $\alpha'$ overlap. Since $u = v.w \in O^\mu(t)$, by Corollary 2, $w \in O^\mu(\sigma(l'))$. Hence, by Proposition 2, $w \in O^\mu_\Sigma(l')$, and thus $\alpha$ and $\alpha'$ $\mu$-overlap. But this contradicts the $\mu$-orthogonality of the TRS; therefore $v \in O^\mu_{\mathcal{R}}(s)$.

3. If $v > u$, by reasoning as in the previous point, the only possibility is $v = u.w.v_1$ and $l|_w = x \in V$. Consider $v' \in v\backslash A^\mu$; $v' = u.w_1.v_1$ and $r|_{w_1} = x$. Since $v \in O^\mu(t)$, by Corollary 1, $u.w \in O^\mu(t)$. Since $t|_u = \sigma(l)$, by Corollary 2, $w \in O^\mu(\sigma(l))$. Thus, since $w \in O(l)$, by Proposition 2, $w \in O^\mu(l)$, i.e., $x \in Var^\mu(l)$. Hence, because $\mathcal{R}$ has left homogeneous replacing variables, $w_1 \in O^\mu(r)$. Also we have by Corollary 2 that $v_1 \in O^\mu(t|_{u.w})$. Therefore, since $w_1 \in O^\mu(r)$, by Proposition 2, $w_1 \in O^\mu(\sigma(r))$. Since $\sigma(r)|_{w_1} = t|_{u.w} = s|_{u.w_1}$, by Proposition 1, we get $u.w_1.v_1 \in O^\mu(s)$. Hence $v' \in O^\mu_{\mathcal{R}}(s)$.

**Proof of Proposition 13** $\Box$

In this way, the context-sensitive extensions of the notion of "residual" are sound (in particular, the notion of residual derivation $B^\mu\backslash A^\mu$). Indeed, if we cannot ensure $v\backslash A^\mu \subseteq O^\mu_{\mathcal{R}}(s)$, then we cannot ensure that $U\backslash A^\mu \subseteq O^\mu_{\mathcal{R}}(s)$. Because the residual derivation $B^\mu\backslash A^\mu$ is a derivation contracting the set $U\backslash A^\mu$, this would not be well defined for context-sensitive multiderivations, because this set could contain nonreplacing redexes. This means that $B^\mu\backslash A^\mu = B\backslash A$ for all $A^\mu : t \overset{U}{\hookrightarrow} t'$, $U \subseteq O^\mu_{\mathcal{R}}(t)$, $U$ having pairwise disjoint occurrences, and $B^\mu : t \overset{W}{\hookrightarrow} t''$, $W \subseteq O^\mu_{\mathcal{R}}(t)$, with $W$ having pairwise disjoint occurrences.

Next we provide the generalization of the parallel moves lemma for *csr*.

**Lemma 2** *Let $\mathcal{R}$ be a $\mu$-orthogonal TRS, having left homogeneous $\mu$-replacing variables. Let $A^\mu, B^\mu$ be elementary $\mu$-multiderivations starting from a term*

*$t$. Then $B^\mu(A^\mu\backslash B^\mu)$ and $A^\mu(B^\mu\backslash A^\mu)$ are $\mu$-multiderivations starting from $t$ and leading to the same term $s$, and for all $u \in O^\mu_{\mathcal{R}}(t)$, we have $u\backslash B^\mu(A^\mu\backslash B^\mu) = u\backslash A^\mu(B^\mu\backslash A^\mu)$.*

**Proof of Lemma 2** Since $A^\mu = A$, $B^\mu = B$, and $B^\mu\backslash A^\mu = B\backslash A$, and $A^\mu\backslash B^\mu = A\backslash B$, we apply the parallel moves lemma for unrestricted rewriting [HL91].

**Proof of Lemma 2** □

The parallel moves lemma for $\twoheadrightarrow\!\!\!\!\!\!+$ (Lemma 2) shows that $\twoheadrightarrow\!\!\!\!\!\!+$ is strongly confluent.

## 4.4 Confluence

The following theorem allows us to devise a method for checking $\mu$-confluence in a $\mu$-terminating TRS with left homogeneous $\mu$-replacing variables.

**Theorem 5** *Let $\mathcal{R} = (\Sigma, R)$ be a $\mu$-terminating TRS with left homogeneous $\mu$-replacing variables, e.g., a $\Sigma$-map $\mu$. Let us denote by $\bar{t}$ an arbitrary $\mu$-normal form of $t \in \mathcal{T}(\Sigma, V)$. Then, $\mathcal{R}$ is $\mu$-confluent if and only if we have $\bar{t}_1 = \bar{t}_2$ for every $\mu$-critical pair $\langle t_1, t_2\rangle$ of $\mathcal{R}$.*

**Proof of Theorem 5**

- ($\Rightarrow$) For any $\mu$-critical pair $\langle t_1, t_2\rangle$ of $\mathcal{R}$, there exists $t \in \mathcal{T}(\Sigma, V)$ such that $t \hookrightarrow t_1$ and $t \hookrightarrow t_2$. If $\hookrightarrow$ is confluent and terminating, then $t$ admits a unique $\mu$-normal form, $\bar{t}_1 = \bar{t}_2$.
- ($\Leftarrow$) $\bar{t}_1 = \bar{t}_2$ implies $t_1 \downarrow t_2$. Then, by Theorem 4, $\hookrightarrow$ is locally confluent. Since $\hookrightarrow$ is terminating and locally confluent, by the diamond lemma, $\hookrightarrow$ is also confluent.

**Proof of Theorem 5** □

Given a $\mu$-terminating TRS $\mathcal{R}$ with left homogeneous $\mu$-replacing variables, Theorem 5 suggests an effective way for testing $\mu$-confluence. Because $\mathcal{R}$ is $\mu$-terminating, for any $\mu$-critical pair $\langle t_1, t_2\rangle$, we can finitely compute $\bar{t}_1$ and $\bar{t}_2$ and then check whether $\bar{t}_1 = \bar{t}_2$. If we have no $\mu$-critical pairs, checking $\mu$-confluence is easier.

**Corollary 3** *A $\mu$-terminating, non-$\mu$-overlapping TRS with left homogeneous $\mu$-replacing variables is $\mu$-confluent.*

**Example 11** *Consider the TRS $\mathcal{R}$ and replacement map $\mu$ of Example 8. This TRS is not confluent. For example:* $\mathsf{g}(\mathsf{b},\underline{\mathsf{a}}) \to \mathsf{g}(\mathsf{b},\mathsf{b})$, *and* $\underline{\mathsf{g}(\mathsf{b},\mathsf{a})} \to \mathsf{b}$. *Nevertheless, because $\mathcal{R}$ is terminating, by Theorem 1, $\mathcal{R}$ is $\mu$-terminating. Because $\mathcal{R}$ has left homogeneous $\mu$-replacing variables and it has no $\mu$-overlap, by Corollary 3 $\mathcal{R}$ is $\mu$-confluent.*

However, for left-linear TRSs having no $\mu$-critical pairs ($\mu$-orthogonal TRSs), we do not need to restrict ourselves to $\mu$-terminating TRSs, since we have a stronger result. Lemma 2 gives conditions for proving strong confluence of $\mathrel{\|\!\!\!\!\hookrightarrow}$. To conclude confluence of $\hookrightarrow$, we use the fact $\hookrightarrow^+ = \mathrel{\|\!\!\!\!\hookrightarrow}^+$.

**Lemma 3** *Let $\mathcal{R}$ be a TRS and $\mu$ be a replacement map. Then $\hookrightarrow^+ = \mathrel{\|\!\!\!\!\hookrightarrow}^+$.*

**Proof of Lemma 3** $\hookrightarrow^+ \subseteq \mathrel{\|\!\!\!\!\hookrightarrow}^+$ is immediate, since $\hookrightarrow \ \subseteq \mathrel{\|\!\!\!\!\hookrightarrow}$. To prove $\mathrel{\|\!\!\!\!\hookrightarrow}^+ \subseteq \ \hookrightarrow^+$, we proceed by induction on the length $n$ of the multiderivation $t \mathrel{\|\!\!\!\!\hookrightarrow}^+ s = t \overset{U}{\hookrightarrow} t'' \mathrel{\|\!\!\!\!\hookrightarrow}^* s$.

1. If $n = 1$, then $t'' = s$ and $t \mathrel{\|\!\!\!\!\hookrightarrow}^+ s = t \overset{U}{\hookrightarrow} t''$. Thus, we proceed by induction on $|U| \geq 1$.

   (a) If $U = \{u\}$, then $t \overset{\{u\}}{\hookrightarrow} s \Leftrightarrow t \overset{u}{\hookrightarrow} s$.

   (b) If $U = \{u\} \uplus U'$, $U' \neq \emptyset$, then, since for all $u, u' \in U$, $u \neq u' \Rightarrow u \parallel u'$, we can write $t \overset{U}{\hookrightarrow} s \Leftrightarrow t \overset{\{u\}}{\hookrightarrow} t' \overset{U'}{\hookrightarrow} s$. Let us show that this is correct: $t' = t[\sigma(r)]_u$ for some rule $l \to r$. Since occurrences on $U'$ are disjoint from $u$, and we have $\gamma_t(u')$ for all $u' \in U'$, by Proposition 3.2, $U' \subseteq O^\mu(t')$. Reductions in $U'$ are disjoint from reduction at $u$; hence, no redex occurrence in $U'$ is destroyed. Therefore, $U' \subseteq O^\mu_{\mathcal{R}}(t')$, and $t' \overset{U'}{\hookrightarrow} s$. By the induction hypothesis, $t' \hookrightarrow^+ s$, i.e., $t \hookrightarrow^+ s$.

2. If $n > 1$, then we have $t \overset{U}{\hookrightarrow} t'' \mathrel{\|\!\!\!\!\hookrightarrow}^+ s$. We have proven in the previous paragraph that $t \hookrightarrow^+ t''$. Hence, by the induction hypothesis, $t'' \hookrightarrow^+ s$, and $t \hookrightarrow^+ s$.

**Proof of Lemma 3** □

Now we can use the following fact [Hue80]:

**Lemma 4 ([Hue80])** *Let $R, S \subseteq A \times A$ be relations such that $R^+ = S^+$. If $S$ is strongly confluent, then $R$ is confluent.*

Finally, we obtain the desired result.

**Theorem 6** *Let $\mathcal{R}$ be a TRS and $\mu$ be a replacement map such that $\mathcal{R}$ is $\mu$-orthogonal and has left homogeneous $\mu$-replacing variables. Then $\mathcal{R}$ is $\mu$-confluent.*

**Proof of Theorem 6** From Lemma 2, $\mathrel{+\!\!\!\!\!\!\not\!\!\hookrightarrow}$ is strongly confluent. By Lemma 3, $\hookrightarrow^+ = \mathrel{+\!\!\!\!\!\!\not\!\!\hookrightarrow}^+$. By Lemma 4, $\hookrightarrow$ is confluent.

**Proof of Theorem 6** □

For instance, the TRS $\mathcal{R}$ in Example 10 with the replacement map $\mu$ as given there is not $\mu$-terminating. However, since $\mathcal{R}$ is $\mu$-orthogonal and has left homogeneous $\mu$-replacing variables, by Theorem 6, $\mathcal{R}$ is $\mu$-confluent.

We can compare our results with the confluence result of conditional reduction proved in [Mar90]. In [Mar90], Maranget analyzes the use of conditional reduction to perform lazy reductions. Maranget introduces syntactic replacement restrictions by using a domain function *dom*, which essentially coincides with our notion of replacement map. Roughly speaking, conditional reduction coincides with parallel *csr*. In the first section of his paper, Maranget claims that "as in the case of standard TRSs, it is sufficient to check the nonoverlapping condition on the left-hand sides of the reduction rules" to prove confluence of conditional reduction. We have shown that this is not necessarily true (Example 7). Analogously to our method, Maranget's claim is proved by showing strong confluence of conditional reduction. The proof is correct because a strong restriction on the class of TRSs that are considered is also introduced, and this restriction implies that the TRSs have left homogeneous replacing variables. However, the relevance of this additional restriction to ensuring confluence of conditional reduction is not mentioned. In fact, Maranget considers $\mu$-orthogonal TRSs with the following restriction: $V_a(r) \subseteq V_a(l)$ and $V_f(r) \subseteq V_f(l)$ for all rule $l \to r$. Here, $V_a(t)$, the set of allowed variables, can be expressed by $V_a(t) = Var^\mu(t)$. The set of forbidden variables, $V_f(t)$, is given by $V_f(t) = \{x \in Var(t) \mid \exists u \in \widetilde{O^\mu}(t).\ t|_u = x\}$. Such conditions are needed because Maranget deals with graph reduction techniques to implement laziness.

It is not difficult to see that this condition implies that the TRS has left homogeneous replacing variables: since the TRS is left-linear, we only need to show that $x \in Var^\mu(l)$ implies $O_x(r) = O^\mu_x(r)$. Assume that, being $x \in Var^\mu(l)$, there is a nonreplacing occurrence of $x$ in $r$. In this case, $x \in V_f(r)$. Since $V_f(r) \subseteq V_f(l)$, we have that $x \in V_f(l)$, i.e., there is a nonreplacing occurrence of $x$ in $l$. However, due to the left-linearity, this is not possible. Thus, the TRSs considered by Maranget are a subclass of the $\mu$-orthogonal TRSs with left homogeneous $\mu$-replacing variables. Regarding $\mu$-confluence, our results are more general. For instance, the TRS in Example 10 cannot be proven $\mu$-confluent by using Maranget's results. Concerning neededness, we have a similar situation (see [Luc97]). However, different from Maranget's approach, we do not ensure that the subterms are reduced only once (important in lazy reductions). We only ensure neededness of the *cs*-computations.

Concerning lazy rewriting in [KW95], Kamperman and Walters give the notion of $\zeta$-confluence. The replacement restrictions introduced by Kamperman and Walters [KW95] use a predicate $\Lambda$ on function symbols and argument positions which is related to our replacement map as follows:[4] $\neg\Lambda(f,i) \Leftrightarrow i \in \mu(f)$ for $f \in \Sigma$ and $1 \leq i \leq ar(f)$. They have the notion of the "eager" path, which essentially coincides with our notion of replacing occurrences. Their notion of a "lazy" path coincides with our nonreplacing occurrences. However, lazy rewriting does not coincide with *csr*. In fact, *csr* is a proper restriction of lazy rewriting. On the other hand, the notion of $\zeta$-confluence is weaker than confluence: the authors use $\zeta$-equality instead of equality to test the joinability of terms. Terms $t$ and $s$ are $\zeta$-equal if, after changing the maximal nonreplacing subterms by a new constant $\zeta$, they yield the same term. Kamperman and Walters only establish that lazy rewriting preserves $\zeta$-confluence. Since $\zeta$-confluence is based on $\zeta$-equality, it is not possible to use it to ensure that the result that is obtained does not depend on the applied rules; this only holds up to $\zeta$-equality, which is somehow weaker.

[4]In the definition of the predicate $\Lambda(f,i)$, the authors allow $i$ to range from 0 to $ar(f)$. However, this possibility is not used subsequently.

## 5 Preserving Meaning in Left-Linear TRSs

We have characterized classes of TRSs whose $\mu$-confluence can be proved. Now we focus on the question of coherence between unrestricted evaluations with a TRS $\mathcal{R}$ and the corresponding context-sensitive computations for a given replacement map $\mu$. The following example motivates the argument.

**Example 12** *Consider the following (canonical) TRS $\mathcal{R}$:*

$$\begin{array}{ll} \mathsf{f}(\mathsf{x},\mathsf{y}) \to \mathsf{g}(\mathsf{x},\mathsf{y}) & \quad \mathsf{a} \to \mathsf{b} \\ \mathsf{g}(\mathsf{b},\mathsf{b}) \to \mathsf{b} & \end{array}$$

*and a replacement map $\mu$ such that $\mu(\mathsf{f}) = \mu(\mathsf{g}) = \{1\}$. Then $\mathcal{R}$ has left homogeneous $\mu$-replacing variables, it is not $\mu$-overlapping, and it is terminating. By Theorem 1 and Corollary 3, $\mathcal{R}$ is $\mu$-confluent. However, we have:*

1. $\underline{\mathsf{f}(\mathsf{b},\mathsf{a})} \hookrightarrow \mathsf{g}(\mathsf{b},\mathsf{a})$ *and* $\mathsf{g}(\mathsf{b},\mathsf{a})$ *is a $\mu$-normal form, but it is not the normal form of* $\mathsf{f}(\mathsf{b},\mathsf{a})$ *because,*

2. $\underline{\mathsf{f}(\mathsf{b},\mathsf{a})} \to \mathsf{g}(\mathsf{b},\underline{\mathsf{a}}) \to \underline{\mathsf{g}(\mathsf{b},\mathsf{b})} \to \mathsf{b}$.

*In other words, in spite of the fact that both relations $\hookrightarrow$ and $\to$ are terminating and confluent, the respective normal forms for the considered input term are different.*

To solve this problem, we first analyze the relation between *csr* and unrestricted rewriting.

### 5.1 Compatibility of the Replacement Restrictions with the Rewriting Process

Because *csr* is a restriction of rewriting, a *cs*-derivation can always be viewed as a rewriting derivation. The main question is the opposite: what are the conditions for ensuring that a rewriting derivation can be viewed as a *cs*-derivation?

The evaluation of a term by term rewriting proceeds by replacing subterms that are instances of the lhs rules of the TRS (i.e., redexes) by the corresponding instances of the rhs rules. Three main aspects should be considered: pattern matching, reduction strategy, and replacement [O'D85].

Pattern matching allows us to detect redexes in the term, the reduction strategy selects the redex to be reduced, and the replacement is finally performed for the selected reduction.

The pattern-matching algorithm, the reduction strategy, and the replacements are usually applied interleaved. For instance, if we consider a rule $f(\tilde{l}) \to r$ and a term $t = f(\tilde{t})$, we have some kind of partial match between $f(\tilde{l})$ and $f(\tilde{t})$, because they are rooted by the same symbol $f$. Perhaps some argument $l_i$, $1 \leq i \leq ar(f)$, also matches an immediate subterm $t_i$ of $t$, i.e., $t_i = \sigma(l_i)$ for some substitution $\sigma$. If we want to "extend" this partial matching, we should eventually reduce (to some extent) other unmatched subterms $t_j$. Therefore, the replacement restrictions should be compatible with this requirement.

**Example 13** *Consider the TRS in Example 1, and the input term* $t = \mathsf{if}(cond, s, s')$*. Note that we have a partial matching between* $t$ *and the rules that define the operation* $\mathsf{if}$*. By considering the* $\mathsf{if}$*-rules, to extend the matching we only need to reduce the condition cond to either* $\mathsf{true}$ *or* $\mathsf{false}$*. If we take* $\mu$ *such that* $1 \notin \mu(\mathsf{if})$*, this reduction is forbidden; then the matching cannot be extended.*

The reduction strategy is expected to select the immediate subterm whose reduction is necessary to allow the matching. If we consider the replacement restrictions introduced by a replacement map $\mu$, we can define a property that a rule's lhs must satisfy to ensure that the pattern matching can be extended (by performing *cs*-reductions) when necessary. This means that the nonreplacing occurrences of an lhs (i.e., those occurrences that we will not further inspect to allow for matching) must be variable occurrences. In this case we have no need for more reductions, because the partial matching for the corresponding subterm is trivially done. For instance, if we return to Example 13, we could reduce either *cond*, or $s$, or $s'$. Since 2 and 3 are variable occurrences of the $\mathsf{if}(\mathsf{true}, \mathsf{x}, \mathsf{y})$ and $\mathsf{if}(\mathsf{false}, \mathsf{x}, \mathsf{y})$ of the $\mathsf{if}$-rules, reductions on $s$ and $s'$ no not improve the matching. Only reducing *cond* does. Thus, it is sensible to impose $\mu(\mathsf{if}) = \{1\}$. This motivates the following definition.

**Definition 12 ($\mu$-Compatible Term)** *Let* $\Sigma$ *be a signature and* $\mu$ *be a* $\Sigma$*-map. A term* $t$ *is* $\mu$*-compatible (written* $\mathsf{comp}_\mu(t)$*) if the nonreplacing occurrences of* $t$ *are variables:* $\widetilde{O^\mu}(t) \subseteq O_V(t)$*. Equivalently,* $t$ *is* $\mu$*-compatible if the nonvariable occurrences are replacing:* $O_\Sigma(t) \subseteq O^\mu(t)$*.*

The property can also be formulated for sets of terms. Thus, we say that a set $T \subseteq \mathcal{T}(\Sigma, V)$ is $\mu$-compatible (written $\mathsf{comp}_\mu(T)$) if and only if $\forall t \in T.\ \mathsf{comp}_\mu(t)$. The predicate $\mathsf{comp}_\mu$ is monotonic, e.g., the order $\sqsubseteq$ in $M_\Sigma$:

**Proposition 14** *Let $\Sigma$ be a signature, $t \in \mathcal{T}(\Sigma, V)$, and $\mu, \mu'$ be $\Sigma$-maps. If $\mu \sqsubseteq \mu'$, then $\mathsf{comp}_\mu(t) \Rightarrow \mathsf{comp}_{\mu'}(t)$.*

**Proof of Proposition 14** $\mathsf{comp}_\mu(t)$ if and only if $O_\Sigma(t) \subseteq O^\mu(t)$. Since $\mu \sqsubseteq \mu'$, $O^\mu(t) \subseteq O^{\mu'}(t)$. Hence $\mathsf{comp}_{\mu'}(t)$.

**Proof of Proposition 14** □

The next proposition establishes that $\mu$-compatibility of a term $t \in \mathcal{T}(\Sigma, V)$ only depends on symbols in $\Sigma(t)$.

**Proposition 15** *Let $\Sigma$ be a signature, $t \in \mathcal{T}(\Sigma, V)$, and $\mu$ be a $\Sigma$-map. Then, $\mathsf{comp}_\mu(t)$ if and only if $\mathsf{comp}_{\mu\downarrow_{\Sigma(t)}}(t)$.*

**Proof of Proposition 15** Since $O_\Sigma(t) = O_{\Sigma(t)}(t)$ and $O^\mu(t) = O^{\mu\downarrow_{\Sigma(t)}}(t)$, the conclusion follows by considering Definition 12.

**Proof of Proposition 15** □

The $\mu$-compatibility is a structural property.

**Proposition 16** *Let $\Sigma$ be a signature, $t \in \mathcal{T}(\Sigma, V)$, and $\mu$ be a $\Sigma$-map. Then, for all $u \in O(t)$, $\mathsf{comp}_\mu(t)$ implies both $\mathsf{comp}_\mu(t[\ ]_u)$ and $\mathsf{comp}_\mu(t|_u)$.*

**Proof of Proposition 16** For $(\mathsf{comp}_\mu(t) \Rightarrow \mathsf{comp}_\mu(t[\ ]_u))$: $\mathsf{comp}_\mu(t) \Leftrightarrow O_\Sigma(t) \subseteq O^\mu(t)$. Since $O_\Sigma(t[\ ]_u) \subseteq O_\Sigma(t)$, by $\mathsf{comp}_\mu(t)$, $u \in O_\Sigma(t[\ ]_u)$ implies $u \in O^\mu(t)$. By Proposition 3(1), $u \in O^\mu(t[\ ]_u)$, and the conclusion follows.

For $(\mathsf{comp}_\mu(t) \Rightarrow \mathsf{comp}_\mu(t|_u))$: here we use $\mathsf{comp}_\mu(t) \Leftrightarrow \widetilde{O^\mu}(t) \subseteq O_V(t)$. Let $v \in \widetilde{O^\mu}(t|_u)$. By Proposition 1, it must be $u.v \in \widetilde{O^\mu}(t)$. Hence, by $\mathsf{comp}_\mu(t)$, $u.v \in O_V(t)$. Therefore, $v \in O_V(t|_u)$, i.e., $\widetilde{O^\mu}(t|_u) \subseteq O_V(t|_u)$. Hence, $\mathsf{comp}_\mu(t|_u)$.

**Proof of Proposition 16** □

We can associate a replacement map $\mu_t$ to any term $t$ which makes $t$ $\mu_t$-compatible.

**Definition 13** *Let $\Sigma$ be a signature, and $t \in \mathcal{T}(\Sigma, V)$. Define the $\Sigma(t)$-map $\mu_t$ to be as follows: if $t \in V$, then $\mu_t = \mu_\perp^\perp$. If $t = f(t_1, \ldots, t_k)$, we define the auxiliary $\{f\}$-map $\mu_t^\epsilon$ to be $\mu_t^\epsilon(f) = \{i \mid t_i \notin V,\ 1 \leq i \leq ar(f)\}$. Then, $\mu_t = \overline{\mu}_t^\epsilon \sqcup \overline{\mu}_{t_1} \sqcup \ldots \sqcup \overline{\mu}_{t_k}$.*

Let us show some examples of calculus of $\mu_t$ for some terms.

**Example 14** *Given the term $t = \mathsf{first}(\mathsf{0}, \mathsf{x})$, $\Sigma(t) = \{\mathsf{0}, \mathsf{first}\}$. We have: $\mu_{\mathsf{first(0,x)}} = \overline{\mu}^\epsilon_{\mathsf{first(0,x)}} \sqcup \overline{\mu}_{\mathsf{0}} \sqcup \overline{\mu}_{\mathsf{x}} = \langle \emptyset, \{1\} \rangle \sqcup \langle \emptyset, \emptyset \rangle \sqcup \langle \emptyset, \emptyset \rangle = \langle \emptyset, \{1\} \rangle$.*
*For the term $t' = \mathsf{first}(\mathsf{s}(\mathsf{x}), \mathsf{y} :: \mathsf{z})$, $\Sigma(t') = \{\mathsf{s}, ::, \mathsf{first}\}$, we get $\mu_{\mathsf{first(s(x),y::z)}} = \overline{\mu}^\epsilon_{\mathsf{first(s(x),y::z)}} \sqcup \overline{\mu}_{\mathsf{s(x)}} \sqcup \overline{\mu}_{\mathsf{y::z}} = \langle \emptyset, \emptyset, \{1,2\} \rangle \sqcup \langle \emptyset, \emptyset, \emptyset \rangle$ $\sqcup \langle \emptyset, \emptyset, \emptyset \rangle = \langle \emptyset, \emptyset, \{1,2\} \rangle$.*

Now we prove that this replacement map characterizes the compatibility of a term. First, we need some previous results.

**Lemma 5** *Let $\Sigma$ be a signature, $t \in \mathcal{T}(\Sigma, V)$, and $f \in \Sigma(t)$. Let $s = f(\tilde{s}) = t|_u$ with $u \in O(t)$. If $i \in \mu_s(f)$, then $i \in \mu_t(f)$.*

**Proof of Lemma 5** By induction on the length of $u$. If $u = \epsilon$, the result is immediate. If $u = j.u'$, and $t = g(\tilde{t})$, then, by the induction hypothesis, $i \in \mu_{t_j}(f)$, $1 \leq j \leq ar(g)$. Since $\mu_t(f) = \mu_t^\epsilon(f) \cup \mu_{t_1}(f) \cup \ldots \cup \mu_{t_{ar(g)}}(f)$, the conclusion follows.

**Proof of Lemma 5** □

**Lemma 6** *Let $\Sigma$ be a signature, $t \in \mathcal{T}(\Sigma, V)$, and $f \in \Sigma(t)$. If $i \in \mu_t(f)$, then there exists $s = f(s_1, \ldots, s_k) = t|_u$ with $u \in O^{\mu_t}(t)$ such that $s_i \notin V$.*

**Proof of Lemma 6** By structural induction. If $t \in \Sigma \cup V$, it is vacuously true. Let $t = g(t_1, \ldots, t_l)$.

1. If $f = g$, we take $u = \epsilon$. Then, if $i \in \mu_t(f)$, we have that $l = k$ and $i \in \mu_t^\epsilon(f) \cup \mu_{t_1}(f) \cup \ldots \cup \mu_{t_k}(f)$. If $i \in \mu_t^\epsilon(f)$, then $t_i \notin V$, and the conclusion follows. If $i \notin \mu_t^\epsilon(f)$, then, $i \in \mu_{t_j}(f)$ for some $t_j$, $1 \leq j \leq k$, and, by the induction hypothesis, there exists a term $s = f(s_1, \ldots, s_k) = t_j|_v$ with $v \in O^{\mu_{t_j}}(t_j)$, which verifies $s_i \notin V$. Because $s_i \notin V$, then $j \in \mu_{t_j}^\epsilon(f)$, and moreover, $j \in \mu_t^\epsilon(f)$; hence $j \in \mu_t(f)$. Because $i \in \mu_t(f)$, $j.v.i \in O^{\mu_t}(t)$ and the conclusion follows.

2. If $f \neq g$, then $\mu^\epsilon_t(f) = \emptyset$ and $\mu_t(f) = \mu_{t_1}(f) \cup \ldots \cup \mu_{t_l}(f)$. By the induction hypothesis, the conclusion follows.

**Proof of Lemma 6** $\Box$

**Proposition 17** *Let $\Sigma$ be a signature, $t \in \mathcal{T}(\Sigma, V)$, and $\mu$ be a $\Sigma(t)$-map. Then, $\mathsf{comp}_\mu(t)$ if and only if $\mu_t \sqsubseteq \mu$.*

**Proof of Proposition 17** For the "if" part, we first prove $\mathsf{comp}_{\mu_t}(t)$, by contradiction. If $\neg\mathsf{comp}_{\mu_t}(t)$, then, $O_\Sigma(t) \not\subseteq O^{\mu_t}(t)$. Hence there is $u \in O_\Sigma(t)$ such that $u \notin O^{\mu_t}(t)$. Therefore, it must be $O_\Sigma(t) \neq \emptyset$, i.e., $t \notin V$. Since $\epsilon \in O^{\mu_t}(t)$ for all $t$, we can write $u = v.w$, where $v \in O^{\mu_t}(t)$ is a maximal replacing occurrence, and $w \neq \epsilon$ (because $u = v.w \notin O^{\mu_t}(t)$). Since $u \in O_\Sigma(t)$, we have $v \in O_\Sigma(t)$, and we can write $t = C[f(\tilde{t})]_v$. Since $w \neq \epsilon$, $w = i.w'$, for some $i$, $1 \leq i \leq ar(f)$, and since $v$ is a maximal replacing occurrence, $i \notin \mu_t(f)$. However, since $u \in O_\Sigma(t)$, and $v.i \leq u$, it must be $v.i \in O_\Sigma(t)$. Hence, if $s = f(\tilde{t})$, we have $i \in \mu_s(f)$. By Lemma 5, $i \in \mu_t(f)$. We get a contradiction. Hence $\mathsf{comp}_{\mu_t}(t)$. Since $\mu_t \sqsubseteq \mu$, by Proposition 14, the conclusion follows.

For the "only if" part, we proceed by contradiction as well. If $\mu_t \not\sqsubseteq \mu$, then by Lemma 1, there exists a symbol $f \in \Sigma(t)$, such that $i \in \mu_t(f)$ and $i \notin \mu(f)$. Thus, by Lemma 6, there is a subterm $s = f(s_1, \ldots, s_k) = t|_u$ with $u \in O^{\mu_t}(t)$ such that $s_i \notin V$. Therefore, because $u.i \notin O^\mu(t)$, then $u.i \in \widetilde{O^\mu}(t)$ but $t|_{u.i} \notin V$. Therefore, $t$ is not $\mu$-compatible.

**Proof of Proposition 17** $\Box$

Therefore, Propositions 15 and 17 entail that the replacement map $\mu_t$ fully characterizes the compatibility of a term $t$, e.g., any replacement map $\mu$.

Analogously, given a set of terms $T \subseteq \mathcal{T}(\Sigma, V)$, we also associate a minimum replacement map $\mu_T = \sqcup_{t \in T} \mu_t$ which makes the set $T$-compatible.

**Proposition 18** *Let $\Sigma$ be a signature, $T \subseteq \mathcal{T}(\Sigma, V)$, and $\mu$ be a $\Sigma(T)$-map. Then, $\mathsf{comp}_\mu(T)$ if and only if $\mu_T \sqsubseteq \mu$.*

**Proof of Proposition 18** For the "if" part, if $\mu_T \sqsubseteq \mu$, then by definition of $\mu_T$, for all $t \in T$, $\mu_t \sqsubseteq \mu_T \sqsubseteq \mu$. Hence, by Proposition 17, $\mathsf{comp}_\mu(t)$ for all $t \in T$. Hence, $\mathsf{comp}_\mu(T)$.

For the "only if" part, we proceed by contradiction. Assume $\mu_T \not\sqsubseteq \mu$. Then, there is $t \in T$ such that $\mu_t \not\sqsubseteq \mu$. Otherwise, since $\mu_T$ is the lub of $\{\mu_t \mid t \in T\}$, it should be $\mu_T \sqsubseteq \mu$. However, if $\mu_t \not\sqsubseteq \mu$, by Proposition 17, $\neg\mathsf{comp}_\mu(t)$. Hence, $\neg\mathsf{comp}_\mu(T)$.

**Proof of Proposition 18** □

We trivially note that if $T = T' \cup T''$, then $\mu_T = \mu_{T'} \sqcup \mu_{T''}$.

## 5.2 Context-Sensitive Rewriting vs. Rewriting

The definitions and results in the previous section can be used with TRSs.

**Definition 14 (Canonical Replacement Map)** *Let $\mathcal{R}$ be a TRS. The replacement map $\mu^{com}_{\mathcal{R}}$ defined as $\mu^{com}_{\mathcal{R}} = \mu_{L(\mathcal{R})} = \sqcup_{l \in L(\mathcal{R})} \mu_l$ is the* canonical *replacement map for the TRS $\mathcal{R}$.*

From Proposition 18, $\mu^{com}_{\mathcal{R}}$ is the minimum replacement map that simultaneously make every lhs of the TRS rules compatible terms. Therefore, in the remainder of the paper, we use this fact by assuming that $\mu^{com}_{\mathcal{R}} \sqsubseteq \mu$ means $\mathsf{comp}_\mu(l)$ for all $l \in L(\mathcal{R})$, i.e., $O_\Sigma(l) \subseteq O^\mu(l)$ or $\widetilde{O^\mu}(l) \subseteq O_V(l)$. As we show below, this replacement map has important properties concerning the completeness of context-sensitive computations. Let us show an example of the calculus of the canonical replacement map for a given TRS.

**Example 15** *Let $\Sigma =$ $maths f 0, [], false, true, s, from, ::, and, first, if\}$ and the TRS $\mathcal{R}$:*

$$\begin{array}{ll}
\mathsf{if(true,x,y)} \to \mathsf{x} & \mathsf{and(true,x)} \to \mathsf{x} \\
\mathsf{if(false,x,y)} \to \mathsf{y} & \mathsf{and(false,y)} \to \mathsf{false} \\
\mathsf{0+x} \to \mathsf{x} & \mathsf{first(0,x)} \to [] \\
\mathsf{s(x)+y} \to \mathsf{s(x+y)} & \mathsf{first(s(x),y :: z)} \to \mathsf{y :: first(x,z)} \\
\mathsf{from(x)} \to \mathsf{x :: from(s(x))} &
\end{array}$$

*We obtain the following:*

- $\mu_{\mathsf{if(true,x,y)}}(\mathsf{if}) = \{1\}$ *and* $\mu_{\mathsf{if(true,x,y)}}(\mathsf{true}) = \varnothing$,
- $\mu_{\mathsf{if(false,x,y)}}(\mathsf{if}) = \{1\}$ *and* $\mu_{\mathsf{if(false,x,y)}}(\mathsf{false}) = \varnothing$,
- $\mu_{\mathsf{and(true,x)}}(\mathsf{and}) = \{1\}$ *and* $\mu_{\mathsf{and(true,x)}}(\mathsf{true}) = \varnothing$,

- $\mu_{\mathsf{and(false,x)}}(\mathsf{and}) = \{1\}$ *and* $\mu_{\mathsf{and(false,x)}}(\mathsf{false}) = \emptyset$,
- $\mu_{\mathsf{0+x}}(+) = \{1\}$ *and* $\mu_{\mathsf{0+x}}(\mathsf{0}) = \emptyset$,
- $\mu_{\mathsf{s(x)+y}}(+) = \{1\}$ *and* $\mu_{\mathsf{s(x)+y}}(\mathsf{s}) = \emptyset$,
- $\mu_{\mathsf{first(0,x)}}(\mathsf{first}) = \{1\}$ *and* $\mu_{\mathsf{first(0,x)}}(\mathsf{0}) = \emptyset$,
- $\mu_{\mathsf{first(s(x),y::z)}}(\mathsf{s}) = \mu_{\mathsf{first(s(x),y::z)}}(::) = \emptyset$ *and* $\mu_{\mathsf{first(s(x),y::z)}}(\mathsf{first}) = \{1,2\}$, *and*
- $\mu_{\mathsf{from(x)}}(\mathsf{from}) = \emptyset$.

*Therefore:* $\mu^{com}_{\mathcal{R}}(\mathsf{0}) = \mu^{com}_{\mathcal{R}}(\mathsf{false}) = \mu^{com}_{\mathcal{R}}(\mathsf{true}) = \mu^{com}_{\mathcal{R}}(\mathsf{s}) = \mu^{com}_{\mathcal{R}}(\mathsf{from}) = \mu_{\mathcal{R}}(::) = \emptyset$, $\mu^{com}_{\mathcal{R}}(\mathsf{if}) = \mu^{com}_{\mathcal{R}}(\mathsf{and}) = \mu^{com}_{\mathcal{R}}(+) = \{1\}$ *and* $\mu^{com}_{\mathcal{R}}(\mathsf{first}) = \{1,2\}$.

Now, we investigate the properties of *csr* when considering replacement maps that are greater than or equal to the canonical replacement map $\mu^{com}_{\mathcal{R}}$.

We denote as $|A|$ the length of a multiderivation $A : t \;\#\!\!\!\!\rightarrow^* s$. Given terms $s, s'$, we write $s \prec s'$ to mean that $s$ is a proper subterm of $s'$, i.e., there exists $u \in O(s')$, $u \neq \epsilon$ such that $s = s'|_u$. In the proof of the following proposition, we use a well-founded strict ordering $<$ between multiderivations: let $A : t \;\#\!\!\!\!\rightarrow^* s$ and $A' : t' \;\#\!\!\!\!\rightarrow^* s'$ be two multiderivations. Then, $A < A'$ if and only if $(|A|, s) <_{lex} (|A'|, s')$, i.e., either $|A| < |A'|$, or $|A| = |A'|$ and $s \prec s'$. Roughly speaking, the proposition establishes that every (multi)derivation leading to an instance of a linear term $l$ can be simulated by *csr* under the canonical replacement map (or greater) enriched with $\mu_l$.

**Proposition 19** *Let* $\mathcal{R} = (\Sigma, R)$ *be a left-linear TRS,* $l$ *be a linear term, and* $\mu$ *be a* $\Sigma$*-map such that* $\mu^{com}_{\mathcal{R}} \sqcup \mu_l \sqsubseteq \mu$*. If we have a multiderivation* $A : t \;\#\!\!\!\!\rightarrow^* \sigma(l)$*, for some substitution* $\sigma$*, then there is a substitution* $\theta$ *such that* $t \;\#\!\!\!\!\rightarrow^* \theta(l)$ *and, for all* $x \in Var(l)$*, we have* $B_x : \theta(x) \;\#\!\!\!\!\rightarrow^* \sigma(x)$ *and* $|B_x| \leq |A|$.

**Proof of Proposition 19** By noetherian induction, using the ordering $<$ between multiderivations. The base case (minimal derivations, e.g., $<$) which considers multiderivations $A$ whose length is zero ($|A| = 0$) is immediate.

For the induction step, we let $n = |A| > 0$. Let $A : t \;\#\!\!\!\!\rightarrow^* t' \xrightarrow{U} \sigma(l)$, where the length of $A' : t \;\#\!\!\!\!\rightarrow^* t'$ is $|A'| = n - 1$, and $U \subseteq O_{\mathcal{R}}(t')$ is such that $\forall u, u' \in U$, $u \neq u' \Rightarrow u \parallel u'$. We consider two cases: $\epsilon \in U$ and $\epsilon \notin U$.

1. If $\epsilon \in U$, then, because occurrences in $U$ are mutually disjoint, it must be $U = \{\epsilon\}$. Then $t'$ is a redex, and there is a substitution $\sigma'$ and a rule $l' \to r'$ such that $A : t \mathrel{+\!\!\!\!+\!\!\!\rightarrow}^* \sigma'(l') \overset{\{\epsilon\}}{\to} \sigma'(r') = \sigma(l)$. Since $A' < A$, and $\mu_{\mathcal{R}}^{com} \sqsubseteq \mu$ (since $l' \in L(\mathcal{R})$, and $\mu_{l'} \sqsubseteq \mu_{\mathcal{R}}^{com}$, we ensure the initial assumptions for the induction step) by the induction hypothesis, there is $\theta'$ such that $t \mathrel{+\!\!\!\!+\!\!\!\rightarrow}^* \theta'(l')$ and, for all $x \in Var(l')$, $B'_x : \theta'(x) \mathrel{+\!\!\!\!+\!\!\!\rightarrow}^* \sigma'(x)$, and $|B'_x| \leq |A'|$. Note that $\theta'(l') \overset{\{\epsilon\}}{\hookrightarrow} \theta'(r')$. We define the multiderivation $B' : \theta'(r') \mathrel{+\!\!\!\!+\!\!\!\rightarrow}^* \sigma'(r')$ as follows: let $r' = C[x_1, \ldots, x_m]$, where $x_1, \ldots, x_m$ are all the variables in $Var(r') \subseteq Var(l')$, but with possible repetitions, i.e., $m \geq |Var(r')|$. Let $w_1, \ldots, w_m$ be the (disjoint) occurrences of variables $x_1, \ldots, x_m$ in $r'$: $x_i = r'|_{w_i}$ for $1 \leq i \leq m$.

   First, we write each multiderivation $B'_{x_i} : \theta'(x_i) \mathrel{+\!\!\!\!+\!\!\!\rightarrow}^* \sigma'(x_i)$ as $\theta'(x_i) = s_i^1 \overset{V_i^1}{\to} s_i^2 \to \ldots \to s_i^{p_i} \overset{V_i^{p_i}}{\to} s_i^{p_i+1} = \sigma'(x_i)$ for $1 \leq i \leq m$. Here, $p_i = |B'_{x_i}|$. Then, $B'$ is $B' : r^1 \overset{W_1}{\to} r^2 \to \ldots \to r^p \overset{W_p}{\to} r^{p+1}$, where $p = max\{p_i \mid 1 \leq i \leq m\}$, and:

   - $r^1 = C[s_1^1, \ldots, s_m^1] = C[\theta'(x_1), \ldots, \theta'(x_m)]$;
   - for all $j$, $1 < j \leq p+1$, $r^j = C[r_1^j, \ldots, r_m^j]$ is such that, for all $i$, $1 \leq i \leq m$, $r_i^j = s_i^j$ if $j \leq p_i + 1$, and $r_i^j = r_i^{j-1}$ otherwise; and
   - for all $j$, $1 \leq j \leq p$, let $RedOc(j) = \{i \mid 1 \leq i \leq m \wedge j \leq p_i\}$ in $W_j = \bigcup_{i \in RedOc(j)} w_i.V_i^j$.

   Clearly, $\theta'(r') = r^1$, and $\sigma'(r') = r^{p+1}$. Therefore, since $|B'_{x_i}| \leq |A'|$ and $|A'| = n - 1$, we have $p_i \leq n - 1$ for each $i$, $1 \leq i \leq m$, and the length of the multiderivation $B'$ is $|B'| = p = max\{p_i \mid 1 \leq i \leq m\} < n$. Then, since $\sigma'(r') = \sigma(l)$, we have a multiderivation $B' : \theta'(r') \mathrel{+\!\!\!\!+\!\!\!\rightarrow}^* \sigma(l)$ such that $B' < A$. By the induction hypothesis, $\theta'(r') \mathrel{+\!\!\!\!+\!\!\!\rightarrow}^* \theta(l)$, and for all $x \in Var(l)$, $B_x : \theta(x) \mathrel{+\!\!\!\!+\!\!\!\rightarrow}^* \sigma(x)$ and $|B_x| \leq |B'| < |A|$. Since $t \mathrel{+\!\!\!\!+\!\!\!\rightarrow}^* \theta'(l') \overset{\{\epsilon\}}{\hookrightarrow} \theta'(r') \mathrel{+\!\!\!\!+\!\!\!\rightarrow}^* \theta(l)$, i.e., $t \mathrel{+\!\!\!\!+\!\!\!\rightarrow}^* \theta(l)$, the conclusion follows.

2. If $\epsilon \notin U$, we split the set $U$ into a set $U_{ov} = \{u \in U \mid u \in O_\Sigma(l)\}$ of occurrences in $U$ that "overlap" $l$, and a set $\overline{U}_{ov} = U \backslash U_{ov}$. Then:

   - for all $u \in \overline{U}_{ov}$, we have either $u \notin O(l)$ or $u \in O_V(l)$. We can decompose $\overline{U}_{ov}$ as follows: $\overline{U}_{ov} = \overline{U}_1 \uplus \ldots \uplus \overline{U}_p$ such that, for all $j$, $1 \leq j \leq p$, there exist $y_j \in Var(l)$ and $v_j \in O_V(l)$ such that

$l|_{v_j} = y_j$, and $\overline{U}_j = \{u \in \overline{U}_{ov} \mid \exists w.\ u = v_j.w\} = v_j.W_j$. Let $Y = \{y_1, \ldots, y_p\}$, and

- by considering $U_{ov} = \{u_1, \ldots, u_q\}$, we define $l' = l[z_1, \ldots, z_q]$, where $z_i$ are new, distinct variables, $z_i \notin Var(l)$ and $l|_{u_i} = z_i$ for all $i$, $1 \leq i \leq q$. Let $Z = \{z_1, \ldots, z_q\}$. Note that $l'$ is linear. Moreover, by Propositions 16 and 17, $\mu_{l'} \sqsubseteq \mu_l \sqsubseteq \mu$. Also, note that $Y \subseteq Var(l')$, because occurrences in $U_{ov}$ and $\overline{U}_{ov}$ are mutually disjoint.

Then, we can express $t'$ in $A : t \mathrel{\not\hookrightarrow}^* t' \xrightarrow{U} \sigma(l)$ by using a substitution $\sigma'$, such that $t' = \sigma'(l')$. The substitution $\sigma'$ is defined as follows:

(a) $\sigma'(x) = \sigma(x)$ for all $x \in Var(l') \backslash (Y \cup Z)$,

(b) $\sigma'(y_j) = t'|_{v_j}$ for $1 \leq j \leq p$, therefore, $C_{y_j} : \sigma'(y_j) = t'|_{v_j} \xrightarrow{W_j} \sigma(l)|_{v_j}$. Note that $|C_{y_j}| = 1$, and recall that $W_j = \{w \mid v_j.w \in \overline{U}_j\}$; and

(c) $\sigma'(z_i) = t'|_{u_i}$ for $1 \leq i \leq q$.

Therefore, from items 1 and 2 above, it follows that for all $x \in Var(l[\,])$ ($Z$ is disjoint from $Var(l)$), we have derivations $C_x : \sigma'(x) \mathrel{\not\hookrightarrow}^{\epsilon} \sigma(x)$ such that $|C_x| \leq 1$ (recall that $t \mathrel{\not\hookrightarrow}^{\epsilon} s$ means either $t \mathrel{\not\hookrightarrow} s$ or $t = s$).

Now we write: $A : t \mathrel{\not\hookrightarrow}^* \sigma'(l') \xrightarrow{U} \sigma(l)$. Let $A' : t \mathrel{\not\hookrightarrow}^* \sigma'(l')$. Since $|A'| < |A|$, by the induction hypothesis, there is $\theta'$ such that $t \mathrel{\not\hookrightarrow}^* \theta'(l')$, and for all $x \in Var(l')$, there exists $B'_x : \theta'(x) \mathrel{\not\hookrightarrow}^* \sigma'(x)$ such that $|B'_x| \leq |A'|$. Note that, for all $x \in Var(l[\,])$, we have derivations $B_x : \theta'(x) \mathrel{\not\hookrightarrow}^* \sigma'(x) \mathrel{\not\hookrightarrow}^{\epsilon} \sigma(x)$. Hence, $|B_x| = |B'_x C_x| \leq |B'_x| + 1 \leq |A'| + 1 = |A|$ for each $x \in Var(l[\,])$. For variables $z_i \in Z$, we have derivations $C_{z_i} : \theta'(z_i) \mathrel{\not\hookrightarrow}^* \sigma'(z_i)$ such that $|C_{z_i}| \leq |A'|$. Since $\sigma'(z_i) = t'|_{u_i} \to \sigma(l|_{u_i})$, we have $C'_{z_i} : \theta'(z_i) \mathrel{\not\hookrightarrow}^* \sigma(l|_{u_i})$, and $|C'_{z_i}| \leq |A|$. Since $u_i \neq \epsilon$, we have $\sigma(l|_{u_i}) \prec \sigma(l)$. Hence, $C'_{z_i} < A$. By the induction hypothesis, for each $i$, $1 \leq i \leq q$, there exists a substitution $\theta_i$ such that $\theta'(z_i) \mathrel{\not\hookrightarrow}^* \theta_i(l|_{u_i})$ and, for all $x \in Var(l|_{u_i})$, derivations $B_x : \theta_i(x) \mathrel{\not\hookrightarrow}^* \sigma(x)$ such that $|B_x| \leq |C'_{z_i}| \leq |A|$. From these facts:

- we define $\theta$ to be $\theta = \theta' \cup \theta_1 \cup \ldots \cup \theta_q$. Since $Var(l) = Var(l[\,]) \uplus Var(l|_{u_1}) \uplus \ldots \uplus Var(l|_{u_q})$, and $l$ is left-linear, $\theta$ is well defined, and it covers all bindings for variables in $Var(l)$. Then, we get that each $B_x : \theta(x) \mathrel{\Vert\!\!\!\hookrightarrow}^* \sigma(x)$ verifies $|B_x| \leq |A|$ for all $x \in Var(l)$; and

- we have $t \mathrel{\Vert\!\!\!\hookrightarrow}^* \theta'(l')$. Since $l' = l[z_1, \ldots, z_q]$, we obtain $\theta'(l') = \theta'(l[z_1, \ldots, z_q]) = \theta'(l)[\theta'(z_1), \ldots, \theta'(z_q)]$. Hence, we get the derivation $t \mathrel{\Vert\!\!\!\hookrightarrow}^* \theta'(l)[\theta'(z_1), \ldots, \theta'(z_q)]$. We also have $\theta'(z_i) \mathrel{\Vert\!\!\!\hookrightarrow}^* \theta_i(l|_{u_i})$ for all $i$, $1 \leq i \leq q$. Since $\mu_l \sqsubseteq \mu$, $O_\Sigma(l) \subseteq O^\mu(l)$. By Proposition 2, $O^\mu(l) \subseteq O^\mu(\theta'(l))$. Thus, since $u_i \in O_\Sigma(l)$, for all $i$, $1 \leq i \leq q$, we get $u_i \in O^\mu(\theta'(l))$ and by Propositions 3(2) and 8, $\theta'(l)[\theta'(z_1), \ldots, \theta'(z_q)] \mathrel{\Vert\!\!\!\hookrightarrow}^* \theta'(l)[\theta_1(l|_{u_1}), \ldots, \theta_q(l|_{u_q})]$. By definition of $\theta$, $\theta'(l)[\theta_1(l|_{u_1}), \ldots, \theta_q(l|_{u_q})] = \theta(l)$. Thus, $t \mathrel{\Vert\!\!\!\hookrightarrow}^* \theta(l)$.

Therefore, the conclusion follows.

**Proof of Proposition 19** $\Box$

To simplify the notation, in the following proposition, $\overset{>\epsilon}{\longrightarrow}{}^*$ denotes a multiderivation that does not consider redexes at the empty occurrence $\epsilon$. The next theorem is a consequence of the previous proposition. It establishes the ability of *csr* to compute the same root symbol that unrestricted rewriting obtains.

**Theorem 7** *Let $\mathcal{R} = (\Sigma, R)$ be a left-linear TRS, and $\mu$ be a $\Sigma$-map such that $\mu^{com}_{\mathcal{R}} \sqsubseteq \mu$. Let $t \in \mathcal{T}(\Sigma, V)$, and let $s$ be a head-normal form. If $t \mathrel{\Vert\!\!\!\hookrightarrow}^* s$, then there exists $s'$ such that $t \mathrel{\Vert\!\!\!\hookrightarrow}^* s'$, $root(s) = root(s')$, and $s' \overset{>\epsilon}{\longrightarrow}{}^* s$.*

**Proof of Theorem 7** If $t$ is an *hnf*, the result is immediate. If $t$ is not an *hnf*, and it reduces to an *hnf*, $s$, there is a derivation $A : t \mathrel{\Vert\!\!\!\hookrightarrow}^* \sigma(l) \overset{\{\epsilon\}}{\to} \sigma(r) \overset{>\epsilon}{\longrightarrow}{}^* s$ for some rule $l \to r$. We write $A' : t \mathrel{\Vert\!\!\!\hookrightarrow}^* \sigma(l)$ and $A'' : \sigma(r) \overset{>\epsilon}{\longrightarrow}{}^* s$. Note that $|A| = |A'| + 1 + |A''|$. We proceed by induction on the length $|A| = n \geq 1$ of the derivation $A$. If $n = 1$, then $t$ is a redex. Since $t = \sigma(l) \hookrightarrow \sigma(r) = s$, we take $s' = \sigma(r) = s$. If $n > 1$, we apply Proposition 19 to the derivation $A'$. Thus, there is $\theta$ such that $t \mathrel{\Vert\!\!\!\hookrightarrow}^* \theta(l)$ and $B_x : \theta(x) \mathrel{\Vert\!\!\!\hookrightarrow}^* \sigma(x)$ for all $x \in Var(l)$ such that $|B_x| \leq |A'|$. Hence, by proceeding as in case 1 in the proof of Proposition 19, we can define a multiderivation $B' : \theta(r) \mathrel{\Vert\!\!\!\hookrightarrow}^* \sigma(r)$ such that $|B'| \leq |A'|$. Hence, the derivation $B : \theta(r) \mathrel{\Vert\!\!\!\hookrightarrow}^* \sigma(r) \overset{>\epsilon}{\longrightarrow}{}^* s$ verifies $|B| = |B'| + |A''| \leq |A'| + |A''| < |A|$. If $r = g(\tilde{r})$, then $root(s) = g$, and by construction, the derivation $B'$ does not reduce an occurrence $\epsilon$ of a redex.

Then we take $s' = \theta(r)$, and, since $s' = \theta(r)\xrightarrow{>\epsilon}{}^*\sigma(r)\xrightarrow{>\epsilon}{}^*s$, $t \mathrel{\not\!\not\rightarrow}^*\theta(l) \overset{\{\epsilon\}}{\hookrightarrow} \theta(r) = s'$, and $root(s') = g$, the conclusion follows. If $r \in V$, then if $\theta(r)$ is an *hnf*, we also have $\theta(r)\xrightarrow{>\epsilon}{}^*\sigma(r)\xrightarrow{>\epsilon}{}^*s$, and we take $s' = \theta(r)$. Clearly, $root(s') = root(s)$. If $\theta(r)$ is not an *hnf*, since the derivation $B$ verifies $|B| < |A|$, by the induction hypothesis there is $s'$ such that $\theta(r) \mathrel{\not\!\not\rightarrow}^* s'$, $s'\xrightarrow{>\epsilon}{}^*s$, and $root(s) = root(s')$. Thus, since $t \mathrel{\not\!\not\rightarrow}^*\theta(l) \overset{\{\epsilon\}}{\hookrightarrow} \theta(r) \mathrel{\not\!\not\rightarrow}^* s'$, the conclusion follows.

**Proof of Theorem 7** $\square$

Left-linearity is required for this proposition, as the following example shows.

**Example 16** *Consider the TRS $\mathcal{R}$:*

$$\begin{array}{ll} \mathsf{f}(\mathsf{x},\mathsf{x}) \to \mathsf{0} & \quad \mathsf{h}(\mathsf{0}) \to \mathsf{0} \\ \mathsf{g}(\mathsf{0},\mathsf{x}) \to \mathsf{0} & \end{array}$$

*This TRS is not left-linear. The term $t = \mathsf{f}(\mathsf{g}(\mathsf{s}(\mathsf{0}),\mathsf{h}(\mathsf{0})),\mathsf{g}(\mathsf{s}(\mathsf{0}),\mathsf{0}))$ is not a head-normal form, since we have:*

$$\mathsf{f}(\mathsf{g}(\mathsf{s}(\mathsf{0}),\underline{\mathsf{h}(\mathsf{0})}),\mathsf{g}(\mathsf{s}(\mathsf{0}),\mathsf{0})) \to \underline{\mathsf{f}(\mathsf{g}(\mathsf{s}(\mathsf{0}),\mathsf{0}),\mathsf{g}(\mathsf{s}(\mathsf{0}),\mathsf{0}))} \to \mathsf{0}$$

*we have $\mu^{com}_{\mathcal{R}}(\mathsf{f}) = \emptyset$, $\mu^{com}_{\mathcal{R}}(\mathsf{g}) = \mu^{com}_{\mathcal{R}}(\mathsf{h}) = \{1\}$. Because* 1.2 *is not a replacing occurrence of the term $t$, $\mathsf{f}(\mathsf{g}(\mathsf{s}(\mathsf{0}),\underline{\mathsf{h}(\mathsf{0})}),\mathsf{g}(\mathsf{s}(\mathsf{0}),\mathsf{0})) \not\hookrightarrow$. This prevents further reductions at the root, unlike the unrestricted derivation.*

Also, Example 12 shows that the condition $\mu^{com}_{\mathcal{R}} \sqsubseteq \mu$ is required to ensure the result. In the example, we had $\mu^{com}_{\mathcal{R}} \parallel \mu$. This prevents the derivation from continuing after the term $\mathsf{g}(\mathsf{b},\mathsf{a})$. Note that the lhs $\mathsf{g}(\mathsf{b},\mathsf{b})$ is not $\mu$-compatible. However, we note that the condition $\mu^{com}_{\mathcal{R}} \sqsubseteq \mu$ is sufficient, but not necessary. For instance, consider the following example.

**Example 17** *Consider the following TRS $\mathcal{R}$ (following Example 12):*

$$\begin{array}{ll} \mathsf{g}(\mathsf{b},\mathsf{b}) \to \mathsf{b} & \quad \mathsf{f}(\mathsf{x},\mathsf{y}) \to \mathsf{g}(\mathsf{x},\mathsf{y}) \\ \mathsf{g}(\mathsf{b},\mathsf{a}) \to \mathsf{b} & \quad \mathsf{a} \to \mathsf{b} \\ \mathsf{g}(\mathsf{b},\mathsf{f}(\mathsf{x},\mathsf{y})) \to \mathsf{b} & \\ \mathsf{g}(\mathsf{b},\mathsf{g}(\mathsf{x},\mathsf{y})) \to \mathsf{b} & \end{array}$$

*Then, it is possible, even if we set $\mu(\mathsf{g}) = \{1\}$, to follow the unrestricted derivation. This is because we obviate the need to extend matching in the second argument of the* g *rules, because we explicitly provide for each possibility of extending matching in this argument. This looks like a variable, but it is not exactly equivalent.*

In the remainder of the paper, we will use Proposition 19 and Theorem 7 concerning derivations, rather than multiderivations. However, since any derivation $t \xrightarrow{u_1} t_2 \to \ldots \to t_n \xrightarrow{u_n} t_{n+1}$ can be considered as a multiderivation $t \overset{\{u_1\}}{\to} t_2 \to \ldots \to t_n \overset{\{u_n\}}{\to} t_{n+1}$, we can consider derivations as being multiderivations. Also, any elementary multiderivation $t \overset{U}{\to} s$ with $U = \{u_1, \ldots, u_n\}$, $n \geq 0$, has an equivalent derivation $t \to^* s$ given by $t = t_1 \xrightarrow{u_1} t_2 \to \ldots \to t_n \xrightarrow{u_n} t_{n+1} = s$ (since the redex occurrences are disjoint, the order of the elementary derivations is not a problem), and thus we can also consider multiderivations as being derivations.

Similar considerations can be made for $\hookrightarrow$ and $\mathrel{+\!\!\!\!\!\!\hookrightarrow}$, by taking into account the corresponding properties of the replacement condition (see Lemma 3).

## 5.3 Meaningful Computations using Context-Sensitive Rewriting

In this section, we use the previous results to establish the ability of *csr* to compute interesting information in functional programming. Here, when we speak about *completeness*, we mean that *csr* is able to compute interesting information (head-normal forms, values, normal forms) using a nontrivial replacement map $\mu$, i.e., $\mu \sqsubset \mu_\top$. Otherwise, the discussion is meaningless, because if $\mu = \mu_\top$, *csr* and unrestricted rewriting coincide. Our results on completeness involve replacement maps $\mu$ which are greater than or equal to the canonical replacement map $\mu_{\mathcal{R}}^{com}$, i.e., $\mu_{\mathcal{R}}^{com} \sqsubseteq \mu$. Of course, we can have $\mu_{\mathcal{R}}^{com} = \mu_\top$ (we show an example at the end of this section). However, we usually have $\mu_{\mathcal{R}}^{com} \sqsubset \mu_\top$. In this situation, it is worthy to speak about completeness of (nontrivial) context-sensitive computations. The most general result concerns head-normal forms.

**Theorem 8** *Let $\mathcal{R} = (\Sigma, R)$ be a left-linear TRS, and $\mu$ be a $\Sigma$-map such that $\mu_{\mathcal{R}}^{com} \sqsubseteq \mu$. Every $\mu$-normal form is a head-normal form.*

**Proof of Theorem 8** By contradiction. Let $t$ be a $\mu$-normal form. If $t$ is not a head-normal form, there is a derivation $t \to^* \sigma(l) \to \sigma(r)$ for some $l \to r \in R$. By Proposition 19, there exists a substitution $\theta$ such that $t \hookrightarrow^* \theta(l)$. Since $t$ is a $\mu$-normal form, it must be $t = \theta(l)$. Hence, $t$ is a redex, and $\theta(l) \hookrightarrow \theta(r)$. Thus, $t$ is not a $\mu$-normal form. Contradiction.

**Proof of Theorem 8** □

From Example 16, we note that left-linearity is needed to ensure Theorem 8. Also, Example 12 shows that the condition $\mu_{\mathcal{R}}^{com} \sqsubseteq \mu$ is not vacuously used.

Given a TRS $\mathcal{R} = (\Sigma, R)$, we consider the signature $\Sigma$ as the disjoint union $\Sigma = \mathcal{C} \uplus \mathcal{F}$ of symbols $c \in \mathcal{C}$, called *constructors*, which have no associated rule, and symbols $f \in \mathcal{F}$, called *defined functions* or *operations*, which are defined by some program rule $f(\tilde{l}) \to r \in R$: $\mathcal{F} = \{f \in \Sigma \mid f(\tilde{l}) \to r \in R\}$ and $\mathcal{C} = \Sigma \backslash \mathcal{F}$. Constructor terms (i.e., values) are denoted as $\delta \in \mathcal{T}(\mathcal{C}, V)$. Note that we do not impose any constructor discipline.

When considering such decomposition of the signature, we can formulate some interesting results about completeness of *csr*. Theorem 8 established that, if we perform a $\mu$-normalizing derivation, we also obtain a head-evaluation of the term. However, we would like to ensure head-evaluations using shorter $\mu$-derivations without the need to fully $\mu$-normalize the term. Moreover, since we do not assume confluence, many head symbols for head-normal forms, values, normal forms, etc. are possible. The first result concerns computations leading to constructor head-normal forms.

**Theorem 9** *Let $\mathcal{R} = (\Sigma, R) = (\mathcal{C} \uplus \mathcal{F}, R)$ be a left-linear TRS, and $\mu$ be a $\Sigma$-map such that $\mu_{\mathcal{R}}^{com} \sqsubseteq \mu$. Let $t \in \mathcal{T}(\Sigma, V)$, $x \in V$, and $s = c(\tilde{s})$ for some $c \in \mathcal{C}$. If $t \to^* x$, then $t \hookrightarrow^* x$. If $t \to^* s$, then there exists $s' = c(\widetilde{s'})$ such that $t \hookrightarrow^* s'$ and $s' \overset{>\epsilon}{\longrightarrow}{}^* s$.*

**Proof of Theorem 9** Immediate consequence of Theorem 7.

**Proof of Theorem 9** □

Thus, Theorem 9 provides conditions ensuring that *csr* is complete in derivations leading to constructor head-normal forms (including variables).

For general head-normal forms, we have to restrict the class of TRSs under consideration a bit more. First, we need the following lemma.

**Lemma 7 ([Mid97])** *Let $\mathcal{R}$ be an almost-orthogonal TRS. If $t$ is a head-normal form and $s\overset{>\epsilon}{\longrightarrow}{}^*t$, then $s$ is a head-normal form.*

Now we have the following result.

**Theorem 10** *Let $\mathcal{R} = (\Sigma, R)$ be an almost-orthogonal TRS, and $\mu$ be a $\Sigma$-map such that $\mu^{com}_{\mathcal{R}} \sqsubseteq \mu$. Let $t \in \mathcal{T}(\Sigma, V)$ and $s = f(\tilde{s})$ be a head-normal form. If $t \to^* s$, then there exists a head-normal form $s' = f(\widetilde{s'})$ such that $t \hookrightarrow^* s'$ and $s'\overset{>\epsilon}{\longrightarrow}{}^*s$.*

**Proof of Theorem 10** By Theorem 7, there exists $s'$ such that $t \hookrightarrow^* s'$ and $s'\overset{>\epsilon}{\longrightarrow}{}^*s$. By Lemma 7, $s'$ is a head-normal form.

**Proof of Theorem 10** $\square$

We conjecture that, in fact, this result extends to arbitrary left-linear TRSs. Concerning the notion of $\mu$-orthogonal TRS, as introduced in Section 4.3 (Definition 11), we note that it does not lead to a generalization of Theorem 10. This is because whenever we consider a replacement map $\mu$ such that $\mu^{com}_{\mathcal{R}} \sqsubseteq \mu$, a $\mu$-orthogonal TRS is always orthogonal. This is owing to the fact that, because of $\mu$-compatibility of the lhs of rules in the TRS, all occurrences in $O_\Sigma(l)$, i.e., the occurrences that can introduce overlapping, are replacing occurrences. Hence, non-$\mu$-overlapping is the same as nonoverlapping.

The following theorem is the main result of this section. It gives conditions ensuring that, for left-linear TRSs, *csr* is complete in evaluations, i.e., in reductions leading to values. To obtain this, we need to relax the restrictions imposed by the canonical replacement map $\mu^{com}_{\mathcal{R}}$. Given a set $\mathcal{B} \subseteq \mathcal{C}$, the replacement map $\mu^{\mathcal{B}}_{\mathcal{R}}$ is $\mu^{\mathcal{B}}_{\mathcal{R}} = \mu^{com}_{\mathcal{R}} \sqcup \mu_{\mathcal{B}}$, where $\mu_{\mathcal{B}}(c) = \mathbb{N}^+_{ar(c)}$ for all $c \in \mathcal{B}$, and $\mu_{\mathcal{B}}(f) = \emptyset$ if $f \notin \mathcal{B}$.

**Theorem 11** *Let $\mathcal{R} = (\Sigma, R) = (\mathcal{C} \uplus \mathcal{F}, R)$ be a left-linear TRS, $\mathcal{B} \subseteq \mathcal{C}$, and $\mu$ be a $\Sigma$-map such that $\mu^{\mathcal{B}}_{\mathcal{R}} \sqsubseteq \mu$. For all $t \in \mathcal{T}(\Sigma, V)$ and $\delta \in \mathcal{T}(\mathcal{B}, V)$, we have $t \to^* \delta$ if and only if $t \hookrightarrow^* \delta$.*

**Proof of Theorem 11** The "if" part is immediate, since $\hookrightarrow \subseteq \to$. For the "only if" part, we proceed by structural induction on $\delta$. If $\delta \in \mathcal{B} \cup V$, then by Theorem 9, $t \hookrightarrow^* s'$, and $s'\overset{>\epsilon}{\longrightarrow}{}^*\delta$. Since $\delta \in \mathcal{B} \cup V$, $s' = \delta$. If

$\delta = c(\tilde{\delta})$ and $c \in \mathcal{B}$, then we decompose $t \to^* \delta$ as $t \to^* c(\tilde{t}) \to^* \delta$. Since $\mu^{com}_{\mathcal{R}} \sqsubseteq \mu^{\mathcal{B}}_{\mathcal{R}} \sqsubseteq \mu$, by Theorem 9 there exists $c(\tilde{s})$ such that $t \hookrightarrow^*_{\mathcal{R}(\mu)} c(\tilde{s})$ and $c(\tilde{s}) \overset{>\epsilon}{\longrightarrow}{}^* c(\tilde{t})$. Thus, $c(\tilde{s}) \overset{>\epsilon}{\longrightarrow}{}^* c(\tilde{\delta})$, and moreover, $s_i \to^*_{\mathcal{R}} \delta_i$, $1 \leq i \leq ar(c)$. By the induction hypothesis, $s_i \hookrightarrow^* \delta_i$. Since $\mu^{\mathcal{B}}_{\mathcal{R}} \sqsubseteq \mu$, $i \in \mu(c)$ for $1 \leq i \leq ar(c)$. Thus, by Proposition 8, the conclusion follows.

**Proof of Theorem 11** □

To make practical use of Theorem 11, we need to know $\mathcal{B}$ before starting the $\mu$-evaluation in order to establish $\mu^{\mathcal{B}}_{\mathcal{R}}$ and, hence, a replacement map $\mu$ such that $\mu^{\mathcal{B}}_{\mathcal{R}} \sqsubseteq \mu$. This is not difficult if we have some typing information available. For instance, if we deal with typed (or sorted) TRSs, as is usual in functional programming, the $\mathsf{sel}$ function symbol in Example 5 would have the following type: $\mathsf{sel}$:*Nat* ×*NatList* → *Nat*. Then we could ensure that any well-typed term $t = \mathsf{sel}(\mathsf{num}, \mathsf{list})$, where $\mathsf{num}$ and $\mathsf{list}$ are *Nat* and *NatList* terms, respectively, must be evaluated to a *Nat* term. Thus, the type of function indicates what constructor symbols should be in $\mathcal{B}$. For instance, $\mathsf{0}$ and $\mathsf{s}$ are the only constructors of *Nat*.

Theorem 11 formally justifies the evaluation of $\mathsf{sel}(\mathsf{s}(\mathsf{s}(\mathsf{0})), \mathsf{from}(\mathsf{0}))$, using the TRS $\mathcal{R}$ in Example 5. The replacement map $\mu$ given in this example satisfies $\mu^{\mathcal{B}}_{\mathcal{R}} \sqsubseteq \mu$, if we take $\mathcal{B} = \{\mathsf{0}, \mathsf{s}\}$. As remarked in the example, $\mathcal{R}$ is $\mu$-terminating (see Section 4.1). This means that we can evaluate function calls leading to natural numbers (which are the values that can be built from $\mathcal{B}$) without risk of infinite derivations, and at the same time preserve completeness.

In general, it is not possible to extend Theorem 11 to computations leading to general normal forms, as the following example shows.

**Example 18** *Consider the TRS:*

$$\begin{array}{ll} \mathsf{f}(\mathsf{x}) \to \mathsf{g}(\mathsf{x}, \mathsf{x}) & \mathsf{h}(\mathsf{0}) \to \mathsf{0} \\ \mathsf{g}(\mathsf{0}, \mathsf{x}) \to \mathsf{0} & \mathsf{h}(\mathsf{s}(\mathsf{0})) \to \mathsf{0} \end{array}$$

*Take* $\mu(\mathsf{f}) = \mu(\mathsf{g}) = \mu(\mathsf{h}) = \mu(\mathsf{s}) = \{1\}$. *The TRS and* $\mu$ *fulfill the conditions in Theorem 11. Now* $\mathsf{g}(\mathsf{s}(\mathsf{0}), \underline{\mathsf{h}(\mathsf{0})}) \to \mathsf{g}(\mathsf{s}(\mathsf{0}), \mathsf{0})$, *which is a normal form. However,* $\mathsf{g}(\mathsf{s}(\mathsf{0}), \mathsf{h}(\mathsf{0}))$ *is already in* $\mu$*-normal form. Note that* $\mathsf{g}(\mathsf{s}(\mathsf{0}), \mathsf{0}) \notin \mathcal{T}(\mathcal{C}, V)$.

However, we have some oportunities in special cases.

**Theorem 12** *Let $\mathcal{R} = (\Sigma, R)$ be a left-linear TRS, $t \in \mathcal{T}(\Sigma, V)$, and $s \in \mathcal{T}(\Sigma, \emptyset)$ such that $s$ is a normal form. Let $\mu$ be a $\Sigma$-map such that $\mu_{\mathcal{R}}^{com} \sqcup \mu_s \sqsubseteq \mu$. Then, $t \rightarrow^* s$ if and only if $t \hookrightarrow^* s$.*

**Proof of Theorem 12** The "if" part is immediate, since $\hookrightarrow \subseteq \rightarrow$. For the "only if" part, since $t \rightarrow^* s$ and $s$ is ground, we can write $t \rightarrow^* \varepsilon(s)$, where $\varepsilon$ is the identity substitution. By Proposition 19, there exists a substitution $\theta$ such that $t \hookrightarrow^* \theta(s)$. Since $s$ is ground, $\theta(s) = s$.

**Proof of Theorem 12** □

For instance, in Example 18, by taking $\mu(\mathsf{g}) = \{1, 2\}$ we are able to obtain the normalization of the input term $t = \mathsf{g}(\mathsf{s}(\mathsf{0}), \mathsf{h}(\mathsf{0}))$ (in fact, $\mu = \mu_\top$ in this particular case). However, this result can be hard to use in practice, because given an input term $t$, we usually do not know the shape of a normal form $s$ of $t$. Thus, we cannot easily establish either groundness of $s$ (but if $t$ is ground, then $s$ is trivially ground) or the replacement map $\mu_s$.

It is worthy to note here that we do not need $\mu$-confluence to ensure completeness of $\mu$-evaluations. Moreover, confluence of the TRS suffices to ensure that all $\mu$-derivations starting from a term $t$ eventually converge to its value $\delta \in \mathcal{T}(\mathcal{B}, V)$ (if it exists and $\mu_{\mathcal{R}}^{\mathcal{B}} \sqsubseteq \mu$). This is easy to see: if $t$ evaluates to $\delta$ and we have $t \hookrightarrow^* t'$ and $t \hookrightarrow^* t''$, confluence ensures that both $t'$ and $t''$ evaluate to $\delta$. Hence, by Theorem 11, $t' \hookrightarrow^* \delta$ and $t'' \hookrightarrow^* \delta$ as well. For instance, the TRS $\mathcal{R}$ of Example 5 has no left homogeneous $\mu$-replacing variables (being $\mu$ as given in the example). Thus, the results in Section 4 do not apply, and we cannot ensure $\mu$-confluence. However, since $\mathcal{R}$ is confluent and $\mu$-terminating, every context-sensitive evaluation of a given term yields the same result. In this way, we obtain a kind of $\mu$-confluence property that restricts to derivations issuing from terms that have a value. Note that we do not impose that $\mathcal{R}$ has left homogeneous $\mu$-replacing variables.

This suggests that having left homogeneous $\mu$-replacing variables is not a necessary condition for ensuring $\mu$-confluence of a TRS $\mathcal{R} = (\Sigma, R) = (\mathcal{C} \uplus \mathcal{F}, R)$. In fact, if we consider terminating completely defined TRSs (for which every normal form is a value), Theorem 11 entails that confluence of the TRS implies $\mu$-confluence whenever $\mu$ satisfies $\mu_{\mathcal{R}}^{\mathcal{C}} \sqsubseteq \mu$. For instance:

**Example 19** *Consider the terminating, completely defined TRS $\mathcal{R}$:*

$$
\begin{array}{ll}
\mathsf{f}(\mathsf{x}) \rightarrow \mathsf{g}(\mathsf{x}, \mathsf{x}) & \mathsf{h}(\mathsf{0}) \rightarrow \mathsf{0} \\
\mathsf{g}(\mathsf{0}, \mathsf{x}) \rightarrow \mathsf{0} & \mathsf{h}(\mathsf{s}(\mathsf{x})) \rightarrow \mathsf{0} \\
\mathsf{g}(\mathsf{s}(\mathsf{x}), \mathsf{y}) \rightarrow \mathsf{s}(\mathsf{x}) &
\end{array}
$$

*Take $\mu(\mathsf{f}) = \mu(\mathsf{g}) = \mu(\mathsf{h}) = \mu(\mathsf{s}) = \{1\}$; then $\mathcal{R}$ has no left homogeneous replacing variables. However, being $\mathcal{C} = \{\mathsf{0}, \mathsf{s}\}$, we have $\mu_{\mathcal{R}}^{\mathcal{C}} \sqsubseteq \mu$, and Theorem 11 implies $\mu$-confluence of $\mathcal{R}$.*

In [KW95], the results on completeness of lazy rewriting with respect to normalization are only given up to $\zeta$-equality (see Section 4.4), i.e., the authors ensure that given a term $t$, there is a normal form, e.g., lazy rewriting, that is $\zeta$-equal to a normal form of $t$. Therefore, our results on completeness for computing normal forms are more accurate (when we restrict ourselves to values). Also, the results in [KW95] are restricted to left-linear, confluent TRSs. We do not require confluence. Completeness results in [Mar90] also concern normal forms, but they are given for a very restrictive class of TRSs (see Section 4.4). No results on completeness of computations leading to any notion of head-normal forms are given in these works.

In both [KW95, Mar90], no indication is given about how to define the replacement restrictions to obtain the completeness results, or how the completeness of the restricted computations depends on the concrete imposed restrictions. This is because they do not provide a means for explicitly comparing different replacement restrictions (as it is, for example, the ordering $\sqsubseteq$ between replacement maps).

To end this section, a final remark. We note that the canonical replacement map $\mu_{\mathcal{R}}^{com}$ gives a measure of the improvements that can be expected from using *csr* instead of unrestricted rewriting. If $\mu_{\mathcal{R}}^{com}$ is equal to $\mu_{\top}$, we cannot introduce any improvement by using *csr* (at least if we want to ensure the minimum completeness properties for the *cs*-computations). For instance, in the following TRS $\mathcal{R}$ for the definition of the *parallel or* [O'D85]:

$$\begin{array}{ll} \mathsf{or}(\mathsf{true}, \mathsf{x}) \to \mathsf{true} & \mathsf{or}(\mathsf{false}, \mathsf{false}) \to \mathsf{false} \\ \mathsf{or}(\mathsf{x}, \mathsf{true}) \to \mathsf{true} & \end{array}$$

we have $\mu_{\mathcal{R}}^{com} = \mu_{\top}$. Hence, *csr* and unrestricted rewriting coincide if we want to ensure some kind of completeness. We can compare this situation with the one already analyzed for the shortcut version of this operator, in Example 3. There, we had $\mu_{\mathcal{R}}^{com} \sqsubset \mu_{\top}$, and using *csr* instead of unrestricted rewriting is better.

# 6 Context-Sensitive Narrowing

Functional logic languages integrate the most interesting features of pure logic and functional languages in a unified framework. Logical variables, partial data structures, and the search for solutions that stem from logic programming, are available in the integrated language [Han95, Han94]. Nested expressions, higher order functions, and the possibility of benefitting from the deterministic nature of functions also become available from the functional component.

To work with logic variables, we need to provide a new operational mechanism that is different from rewriting and is able to instantiate them. This mechanism is narrowing. To bring the advantages that *csr* offers in functional programming to functional logic languages, let us consider the idea of limiting narrowing by means of a similar kind of replacement restriction, expressed by a replacement map.

**Definition 15 (Context-Sensitive Narrowing)** *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, and $\mu$ be a $\Sigma$-map. A term $t$ $\mu$-narrows to $s$, written $t \overset{\mu}{\leadsto}_{[u,\alpha,\sigma]} s$ (or just $t \overset{\mu}{\leadsto}_{\sigma} s$) if and only if $t \leadsto_{[u,\alpha,\sigma]} s$ in $\mathcal{R}$ and $u \in O^\mu(t)$.*

Note that, analogously to the case of *csr*, we have that $\overset{\mu}{\leadsto} \subseteq \leadsto$. Thus, context-sensitive narrowing (*csn*) explores a smaller search space than unrestricted narrowing. Let us consider an example.

**Example 20** *Consider the TRS and replacement map of Example 1. The (unnecessary) narrowing step*

$$\mathsf{if}(\mathsf{and}(\mathsf{x}, \mathsf{false}), \underline{\mathsf{y} + \mathsf{s}(0)}, 0) \leadsto_{\{\mathsf{y}/0\}} \mathsf{if}(\mathsf{and}(\mathsf{x}, \mathsf{false}), \mathsf{s}(0), 0)$$

*is avoided with csn, because the occurrence* $2$ *of the input term* $\mathsf{if}(\mathsf{and}(\mathsf{x}, \mathsf{false}), \mathsf{y} + \mathsf{s}(0), 0)$ *is not replacing.*

We give a result similar to Theorem 11 for *csn*. It relies on the following theorem, which expresses the correspondence between *csr* and *csn*. In fact, it looks like the well-known result for unrestricted narrowing that is due to Hullot [Hul80]. By a normalized substitution, we mean a substitution $\sigma$ such that $\sigma(x)$ is a normal form, for all $x \in V$.

**Theorem 13** *Let $\mathcal{R}$ be a TRS, and $\mu$ be a $\Sigma$-map. Let $t \in \mathcal{T}(\Sigma, V)$, and $W$ be a finite set of variables containing $Var(t)$. Let $\eta$ be a normalized substitution with $\mathcal{D}om(\eta) \subseteq Var(t)$. Consider the following $\mu$-rewriting derivation issuing from $\eta(t)$: $\eta(t) = s_0 \hookrightarrow_{[u_0,\alpha_0]} s_1 \hookrightarrow_{[u_1,\alpha_1]} s_2 \hookrightarrow \dots s_{n-1} \hookrightarrow_{[u_{n-1},\alpha_{n-1}]} s_n$*

1. *there exists an associated $\mu$-narrowing derivation issuing from $t$:*

$$t = t_0 \overset{\mu}{\rightsquigarrow}_{[u_0,\alpha_0,\sigma_0]} t_1 \overset{\mu}{\rightsquigarrow}_{[u_1,\alpha_1,\sigma_1]} t_2 \overset{\mu}{\rightsquigarrow} \ldots t_{n-1} \overset{\mu}{\rightsquigarrow}_{[u_{n-1},\alpha_{n-1},\sigma_{n-1}]} t_n$$

2. *for each $i$, $0 \leq i \leq n$, there exists a substitution $\eta_i$ and a finite set of variables $W_i$ such that:*
   - $\mathcal{D}om(\eta_i) \subseteq W_i$,
   - *$\eta_i$ is normalized,*
   - $\eta\downarrow_W = (\eta_i\theta_i)\downarrow_W$, *and*
   - $\eta_i(t_i) = s_i$,

   *where $\theta_0 = \epsilon$ and $\theta_{i+1} = \sigma_{i+1}\theta_i$. Conversely, for each $\mu$-narrowing derivation 2 and substitution $\eta$ such that $\theta_n \leq \eta[\![W]\!]$, there exists a corresponding $\mu$-rewriting derivation 1.*

**Proof of Theorem 13** The proof is basically the same as that of Theorem 1 in Hullot [Hul80], because each reduction step takes the same occurrence as the corresponding narrowing step, and vice versa. Because the replacement restrictions when considering narrowing derivations are the same as in rewriting and they only depend on the considered occurrence, the proof is still valid.

**Proof of Theorem 13** □

When $\mu = \mu_\top$, Theorem 13 boils down to the standard result due to Hullot (see [Hul80]). In the setting of this theorem, normalized substitutions are important. To ensure that the computed substitutions are normalized, here we consider constructor-based TRSs (CB-TRSs). When considering CB-TRSs, the rules of a TRS $\mathcal{R} = (\Sigma, R) = (\mathcal{C} \uplus \mathcal{F}, R)$ satisfy the following requirement: for all $l \to r \in R$, $l = f(\tilde{\delta})$, where $f \in \mathcal{F}$ and $\tilde{\delta} \in \mathcal{T}(\mathcal{C}, V)^{ar(f)}$. The following result is auxiliary.

**Lemma 8** *Let $\mathcal{R} = (\Sigma, R)$ be a left-linear CB-TRS. Let $t, s \in \mathcal{T}(\Sigma, V)$, such that $t \rightsquigarrow_\theta^* s$. Then $\theta\downarrow_{Var(t)}$ is a constructor substitution.*

We note that for CB-TRSs, constructor substitutions are normalized. Now we present the main result of this section.

**Theorem 14** *Let $\mathcal{R} = (\Sigma, R) = (\mathcal{C} \uplus \mathcal{F}, R)$ be a left-linear CB-TRS, $\mathcal{B} \subseteq \mathcal{C}$, and $\mu$ be a $\Sigma$-map such that $\mu^{\mathcal{B}}_{\mathcal{R}} \sqsubseteq \mu$. Let $t \in \mathcal{T}(\Sigma, V)$, and $\delta \in \mathcal{T}(\mathcal{B}, V)$. Then $t \rightsquigarrow^*_{\theta'} \delta$ if and only if $t \overset{\mu}{\rightsquigarrow}{}^*_{\theta} \delta$, and $\theta \leq \theta'[\![Var(t)]\!]$.*

**Proof of Theorem 14** The "if" part is immediate, since $\overset{\mu}{\rightsquigarrow} \subseteq \rightsquigarrow$, and we get $\theta = \theta'$. For the "only if" part, by Lemma 8 and Theorem 13 (taking $\eta = \theta'\downarrow_{Var(t)}$, $\mu = \mu_\top$ and considering that, since $\eta = \theta'\downarrow_{Var(t)}$, we have $\eta_n = \epsilon$ and then $\eta_n(\delta) = \delta$), we get $t \rightsquigarrow^*_{\theta'} \delta \Rightarrow \theta'(t) \to^* \delta$. Now, from Theorem 11, $\theta'(t) \to^* \delta \Leftrightarrow \theta'(t) \hookrightarrow^* \delta$. Finally, from Theorem 13, $\theta'(t) \hookrightarrow^* \delta \Rightarrow t \overset{\mu}{\rightsquigarrow}{}^*_{\theta} \delta$ and $\theta \leq \theta'[\![Var(t)]\!]$.

**Proof of Theorem 14** □

As a consequence of Theorem 14, *csn* computes more general solutions than unrestricted narrowing without losing completeness (when we observe successful derivations leading to constructor terms).

**Example 21** *A successful csn derivation for the input expression of Example 20 is:*

$$\mathsf{if}(\mathsf{and}(\mathsf{x}, \mathsf{false}), \mathsf{y} + \mathsf{s}(0), 0) \overset{\mu}{\rightsquigarrow}{}^*_{\{\mathsf{x}/\mathsf{true}\}} 0$$

*The more expensive unrestricted narrowing reduction*

$$\mathsf{if}(\mathsf{and}(\mathsf{x}, \mathsf{false}), \mathsf{y} + \mathsf{s}(0), 0) \rightsquigarrow^*_{\{\mathsf{x}/\mathsf{true}, \mathsf{y}/0\}} 0$$

*is not produced.*

Narrowing can be used to solve equations by instantiating the variables in an equation in such a way that the equation becomes true. An equation is a pair $t \approx t'$ of terms. If we do not restrict ourselves to terminating TRSs, as is usual in functional logic languages with a lazy operational semantics, normal forms may not exist. Therefore, the validity of an equation is defined as a *strict equality* on terms. This is done, for instance, in *BABEL* [MR92], *K-LEAF* [GLMP91], etc, and it is the more appropiate choice in the presence of infinite data structures and computations. Thus, a substitution $\sigma$ is a solution for an equation $t \approx t'$ if and only if $\sigma(t)$ and $\sigma(t')$ are reducible to a same ground constructor term. By defining the symbol $\approx$ as a binary operation symbol, equations can also be interpreted as terms. Therefore, all notions for terms, such as substitution, narrowing, etc., will also be used for equations. Given a CB-TRS $\mathcal{R} = (\mathcal{C} \uplus \mathcal{F}, R)$, the operation $\approx$ is defined by the TRS $\mathcal{S}_\approx = (\mathcal{C} \uplus \{\mathsf{true}\} \uplus \{\approx, \wedge\}, S_\approx)$, where $S_\approx$ is: $S_\approx = \{c \approx c \to \mathsf{true} \mid c \in \mathcal{C}$ and $ar(c) = 0\} \cup \{c(x_1, \ldots, x_k) \approx c(y_1, \ldots, y_k) \to x_1 \approx y_1 \wedge \ldots \wedge x_k \approx y_k \mid c \in \mathcal{C}, k > 0\} \cup \{\mathsf{true} \wedge x \to x\}$, and $\wedge$ is assumed to be a right-associative infix symbol. These are the equality rules of a signature. By adding the equality rules to the rewrite system, equation solving can be done by narrowing equations to "true."

The TRS $\mathcal{S}_\approx$ is left-linear, and $\mu^{com}_{\mathcal{S}_\approx}$ is $\mu^{com}_{\mathcal{S}_\approx}(c) = \emptyset$ for all $c \in \mathcal{C} \uplus \{\mathsf{true}\}$, $\mu(\wedge) = \{1\}$, and $\mu^{com}_{\mathcal{S}_\approx}(\approx) = \{1, 2\}$. Thus, given a left-linear CB-TRS $\mathcal{R} = (\Sigma, R) = (\mathcal{C} \uplus \mathcal{F}, R)$, $\mathcal{B} \subseteq \mathcal{C}$, and a replacement map $\mu$ such that $\mu^{\mathcal{B}}_{\mathcal{R}} \sqsubseteq \mu$, it is obvious that, being $\mathcal{R}_\approx = (\Sigma \cup \{\wedge, \approx\}, R \cup S_\approx)$, we also have $\mu^{com}_{\mathcal{R}} \sqcup \mu^{com}_{\mathcal{S}_\approx} \sqcup \mu_{\mathcal{B}} = \mu^{\mathcal{B}}_{\mathcal{R}_\approx} \sqsubseteq \mu \sqcup \mu^{com}_{\mathcal{S}_\approx}$. Therefore, we just need to add $\mu^{com}_{\mathcal{S}_\approx}$ to $\mu$, and we have no further problems.

**Example 22** *Consider the TRS in Example 20 and the equation*

$$\mathsf{if}(\mathsf{and}(\mathsf{x}, \mathsf{false}), \mathsf{y} + \mathsf{s}(0), 0) \approx 0$$

*We obtain*

$$\begin{array}{rl} & \overset{\mu}{\leadsto}_{\{\mathsf{x}/\mathsf{true}\}} \mathsf{if}(\mathsf{false}, \mathsf{y} + \mathsf{s}(0), 0) \approx 0 \\ \mathsf{if}(\mathsf{and}(\mathsf{x}, \mathsf{false}), \mathsf{y} + \mathsf{s}(0), 0) \approx 0 & \overset{\mu}{\leadsto}_\epsilon 0 \approx 0 \\ & \overset{\mu}{\leadsto}_\epsilon \mathsf{true} \end{array}$$

*and also*

$$\begin{array}{rl} & \overset{\mu}{\leadsto}_{\{\mathsf{x}/\mathsf{false}\}} \mathsf{if}(\mathsf{false}, \mathsf{y} + \mathsf{s}(0), 0) \approx 0 \\ \mathsf{if}(\mathsf{and}(\mathsf{x}, \mathsf{false}), \mathsf{y} + \mathsf{s}(0), 0) \approx 0 & \overset{\mu}{\leadsto}_\epsilon 0 \approx 0 \\ & \overset{\mu}{\leadsto}_\epsilon \mathsf{true} \end{array}$$

*Note that* $\{\mathsf{x}/\mathsf{true}\}$ *and* $\{\mathsf{x}/\mathsf{false}\}$ *are the most general solutions to the initial equation. Unrestricted narrowing would compute an infinite number of solutions corresponding to useless instantiations of the variable* $\mathsf{y}$.

## 7 Conclusions

From a computational point of view, context-sensitive rewriting allows one to achieve more efficient computations in comparison to standard rewriting. Termination is preserved or enhanced by *csr* [Luc96b, Zan97]. Confluence may sometimes be enhanced, but might also be lost. We have established classes of TRSs for which we can prove confluence of *csr*. We have also established syntactic conditions to ensure that computations are kept under *csr*. We have shown how to define a minimum replacement map to achieve head-evaluations, evaluations, and even normalization of input terms. In many cases, the restrictions imposed by such replacement maps are able to preserve completeness and improve termination as well.

**Example 23** *Consider the following TRS from [Zan97]:*

$$\begin{array}{ll} \mathsf{sel}(\mathsf{0}, \mathsf{x} :: \mathsf{y}) \to \mathsf{x} & \mathsf{g}(\mathsf{0}) \to \mathsf{s}(\mathsf{0}) \\ \mathsf{sel}(\mathsf{s}(\mathsf{x}), \mathsf{y} :: \mathsf{z}) \to \mathsf{sel}(\mathsf{x}, \mathsf{z}) & \mathsf{g}(\mathsf{s}(\mathsf{x})) \to \mathsf{s}(\mathsf{s}(\mathsf{g}(\mathsf{x}))) \\ \mathsf{f}(\mathsf{x}) \to \mathsf{x} :: \mathsf{f}(\mathsf{g}(\mathsf{x})) & \end{array}$$

*The term* $\mathsf{f}(\mathsf{0})$ *describes the infinite list of numbers of the shape* $2^k - 1$ *for all* $k \geq 0$*. By using* $\mathsf{sel}$*, we can compute the n-th element of this list. For instance:* $\mathsf{sel}(\mathsf{s}(\mathsf{s}(\mathsf{0})), \mathsf{f}(\mathsf{0}))$*. By considering* $\mu(::) = \mu(\mathsf{s}) = \mu(\mathsf{f}) = \mu(\mathsf{g}) = \{1\}$*, and* $\mu(\mathsf{sel}) = \{1, 2\}$*, in [Zan97], Zantema proves that this TRS is* $\mu$*-terminating. We note that by considering* $\mathcal{B} = \{\mathsf{0}, \mathsf{s}\}$*, we have* $\mu^{\mathcal{B}}_{\mathcal{R}} \sqsubseteq \mu$*. Hence, Theorem 11 ensures the use of csr under the replacement map* $\mu$ *to compute the value of the function call* $\mathsf{sel}(\mathsf{s}(\mathsf{s}(\mathsf{0})), \mathsf{f}(\mathsf{0}))$ *(without getting an infinite derivation).*

We have also introduced context-sensitive narrowing, a kind of narrowing relation based on the definition of restrictions on the replacement of arguments in function calls. Finally, we have proven that *csn* is equivalent to unrestricted narrowing under some syntactic restrictions on the program, while it provides for more efficient computations.

Therefore, we can use the context-sensitive replacement restrictions expressed by a replacement map and the replacement condition as a suitable operational tool for implementing computations with functional and functional logic programs (via *csr* and *csn*, respectively). In fact, the results in Section 5.3 show that *csr* under the canonical replacement map (or greater) is sufficient to perform the reductions that are necessary to obtain meaningful computations in functional programming. Since these results are valid for a broad class of TRSs (left-linear TRSs), we think that the systematic use of such fixed, simple replacement restrictions in rewriting is useful for implementing the execution of functional (and functional logic) programs. In fact, there exist reduction engines that provide means to either enable or disable reductions of subterms of an input term during the evaluation process (e.g., the Rewrite Rule Machine in [GKM87]). These features could be used to efficiently implement the context-sensitive computations once we have shown that they do not damage the evaluation process. However, a more detailed analysis of this problem is a subject of future work.

The results of this paper show that in many cases, there exist natural generalizations of many standard results of rewriting and narrowing to *csr* and *csn*, respectively. This is the case of confluence, termination, the correspondence between narrowing and rewriting ... but also of more involved topics as sequentiality of computations (see [Luc97]). From a theoretic point of view, this suggests that the consideration of this kind of fixed replacement restriction easily fits into the general theory of rewriting by just reformulating some well-known results to explicitly take into account the replacement restrictions. Therefore, more benefits could be gained from a further analysis (in this direction) of still unexplored areas of the theory. For instance, conditional TRSs, modularity, strategies, etc.

**Acknowledgement of support:** This work has been partially supported by CICYT under grant TIC 95-0433-C03-03.

I would like to thank María Alpuente and Germán Vidal for their encouragement and support during the development of this work.

## References

[CPJ85] C. Clack and S. L. Peyton-Jones. Strictness analysis—a practical approach. In J.-P. Jouannaud, editor, *Functional Programming Architecture*, volume 201 of *Lecture Notes in Computer Science*, pages 35–39, Berlin, 1985. Springer-Verlag.

[Der87] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69–115, 1987.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 243–320, Amsterdam, 1990. Elsevier.

[Dur94] I. Durand. Bounded, strongly sequential and forward-branching term rewriting systems. *Journal of Symbolic Computation*, 18:319–352, 1994.

[FW76] D. P. Friedman and D. S. Wise. CONS should not evaluate its arguments. In S. Michaelson and R. Milner, editors, *Automata, Languages and Programming*, pages 257–284, Edinburgh, 1976. Edinburgh University Press.

[GKM87] J. Goguen, C. Kirchner, and J. Meseguer. Concurrent term rewriting as a model of computation. In J. H. Fasel and R. M. Keller, editors, *Graph Reduction. Proceedings of a Workshop*, volume 279 of *Lecture Notes in Computer Science*, pages 53–93, Berlin, 1987. Springer-Verlag.

[GLMP91] E. Giovannetti, G. Levi, C. Moiso, and C. Palamidessi. Kernel leaf: a logic plus functional language. *Journal of Computer and System Sciences*, 42(2):139–185, 1991.

[Han94] M. Hanus. The integration of functions into logic programming: from theory to practice. *Journal of Logic Programming*, 19–20:583–628, 1994.

[Han95] M. Hanus. Functional logic languages: combine search and efficient evaluation. In J. W. Lloyd, editor, *Proceedings of the 1995 International Symposium on Logic Programming, ILPS'95*, pages 625–626, Cambridge, MA, 1995. The MIT Press.

[HL91] G. Huet and J.-J. Lévy. Computations in orthogonal term rewriting systems I, II. In J. L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honour of J. Alan Robinson*, pages 395–414 and 415–443, Cambridge, MA, 1991. The MIT Press.

[HM76] P. Henderson and J. Morris. A lazy evaluator. In *Conference Record of the 3rd Annual ACM Symposium on Principles of Programming Languages, POPL'76*, pages 95–103, New York, 1976. ACM Press.

[Hue80] G. Huet. Confluent reductions: abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27:797–821, 1980.

[Hul80] J. M. Hullot. Canonical forms and unification. In *Proceedings of the 5th International Conference on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334, Berlin, 1980. Springer-Verlag.

[HW87] C. V. Hall and D. S. Wise. Compiling strictness into streams. In *Conference Record of the 14th Annual ACM Symposium on Principles of Programming Languages, POPL'87*, pages 132–143, New York, 1987. ACM Press.

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

[KM91] J. W. Klop and A. Middeldorp. Sequentiality in orthogonal term rewriting systems. *Journal of Symbolic Computation*, 12:161–195, 1991.

[KW95] J. F. Th. Kamperman and H. R. Walters. Lazy rewriting and eager machinery. In H. Hsiang, editor, *Proceedings of the 6th International Conference on Rewriting Techniques and Applications, RTA'95*, volume 914 of *Lecture Notes in Computer Science*, pages 147–162, Berlin, 1995. Springer-Verlag.

[Lal93] R. Lalement. *Computation as Logic*. Masson-Prentice-Hall International, Paris, Hemel Hempstead, 1993.

[Luc95] S. Lucas. Fundamentals of context-sensitive rewriting. In M. Bartŏsek, J. Staudek, and J. Wiedermann, editors, *Proceedings of the XXII Seminar on Current Trends in Theory and Practice of Informatics, SOFSEM'95*, volume 1012 of *Lecture Notes in Computer Science*, pages 405–412, Berlin, 1995. Springer-Verlag.

[Luc96a] S. Lucas. Context-sensitive computations in confluent programs. In H. Kuchen and S. D. Swierstra, editors, *Proceedings of the 8th International Symposium on Programming Languages, Implementations, Logics, and Programs, PLILP'96*, volume 1140 of *Lecture Notes in Computer Science*, pages 408–422, Berlin, 1996. Springer-Verlag.

[Luc96b] S. Lucas. Termination of context-sensitive rewriting by rewriting. In F. Meyer auf der Heide and B. Monien, editors, *Proceedings of the 23rd International Colloquium on Automata, Languages, and Programming, ICALP'96*, volume 1099 of *Lecture Notes in Computer Science*, pages 122–133, Berlin, 1996. Springer-Verlag.

[Luc96c] S. Lucas. Using replacement restrictions in functional-logic languages. In *5th Compulog-Network Area Meeting on Language Design and Semantic Analysis Methods.* Università degli Studi di Pisa, 1996.

[Luc97] S. Lucas. Needed reductions with context-sensitive rewriting. In M. Hanus and K. Meinke, editors, *Proceedings of the 6th International Conference on Algebraic and Logic Programming, ALP'97*, volume 1298 of *Lecture Notes in Computer Science*, pages 129–143, Berlin, 1997. Springer-Verlag.

[Mar90] L. Maranget. Optimal derivations in weak lambda-calculi and in orthogonal term rewriting systems. In *Conference Record of the 18th Annual ACM Symposium on Principles of Programming Languages, POPL'90*, pages 255–269, New York, 1990. ACM Press.

[Mid97] A. Middeldorp. Call by need computations to root-stable form. In *Conference Record of the 24th Annual ACM Symposium on Principles of Programming Languages, POPL'97*, pages 94–105, New York, 1997. ACM Press.

[MR92] J. J. Moreno-Navarro and M. Rodríguez-Artalejo. Logic programming with functions and predicates: the language babel. *Journal of Logic Programming*, 12(3):191–223, 1992.

[Myc80] A. Mycroft. The theory and practice of transforming call-by-need into call-by-value. In *Proceedings of the 4th International Symposium on Programming*, volume 83 of *Lecture Notes in Computer Science*, pages 269–281, Berlin, 1980. Springer-Verlag.

[New42] M. H. A. Newman. On theories with a combinatorial definition of "equivalence." *Annals of Mathematics*, 43:223–243, 1942.

[O'D77] M. J. O'Donnell. *Computing in Systems Described by Equations*, volume 58 of *Lecture Notes in Computer Science*, Berlin, 1977. Springer-Verlag.

[O'D85] M. J. O'Donnell. *Equational Logic as a Programming Language.* Cambridge, MA, 1985. The MIT Press.

[PJ87] S. L. Peyton-Jones. *The Implementation of Functional Programming Languages.* London, 1987. Prentice-Hall International.

[Rea93] C. Reade. *Elements of Functional Programming.* Reading, MA, 1993. Addison-Wesley.

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

[Sto77] J. E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.* Cambridge, MA, 1977. The MIT Press.

[TSvEP93] Y. Toyama, S. Smetsers, M. C. J. D. van Eekelen, and R. Plasmeijer. The functional strategy and transitive term rewriting systems. In M. R. Sleep, M. J. Plasmeijer, and M. C. J. D. van Eekelen, editors, *Term Graph Rewriting—Theory and Practice*, pages 266–275, New York, 1993. John Wiley.

[Vui74] J. Vuillemin. Correct and optimal implementation of recursion in a simple programming language. *Journal of Computer and System Sciences*, 9(3):332–354, 1974.

[Wad87] P. Wadler. Strictness analysis on non-flat domains (by abstract interpretation over finite domains). In S. Abramsky and C. Hankin, editors, *Abstract Interpretation of Declarative Languages*, pages 266–275, 1987. Ellis Horwood Ltd.

[Zan97] H. Zantema. Termination of context-sensitive rewriting. In H. Comon, editor, *Proceedings of the 8th International Conference on Rewriting Techniques and Applications, RTA'97*, volume 1232 of *Lecture Notes in Computer Science*, pages 172–186, Berlin, 1997. Springer-Verlag.