Lazy Rewriting and Context-Sensitive Rewriting

Salvador Lucas\textsuperscript{1,2}

\textit{DSIC}

\textit{Universidad Politécnica de Valencia, Spain}

Abstract

Lazy rewriting (LR) is intended to improve the termination behavior of TRSs. This is attempted by restricting reductions for selected arguments of functions. Similarly, context-sensitive rewriting (CSR) forbids any reduction on those arguments. We show that LR and CSR coincide under certain conditions. On the basis of this result, we also describe a transformation which permits us to prove termination of LR as termination of CSR for the transformed system. Since there are a number of different techniques for proving termination of CSR, this provides a formal framework for proving termination of lazy rewriting.

1 Introduction

Syntactic annotations (which are associated to arguments of symbols) have been used in programming languages such as Lisp, Haskell, Clean, OBJ2, OBJ3, CafeOBJ, Maude, etc., to improve the termination and efficiency of computations. Lazy languages (e.g., Haskell, Clean) interpret them as strictness annotations in order to become ‘more eager’ and efficient. Eager languages (e.g., Lisp, OBJ2, OBJ3, CafeOBJ, Maude) use them as replacement restrictions to become ‘more lazy’ thus (hopefully) avoiding nontermination. For instance, [FW76] studied implementations of Lisp where the list constructor operator (\texttt{cons}) did not evaluate its arguments during certain stages of the computation. Also, algebraic languages, such as OBJ2 [FGJM85], OBJ3 [GWMFJ00], CafeOBJ [FN97], or Maude [CELM96], admit the explicit specification of strategy annotations as sequences of integers in parentheses. They are interpreted as replacement restrictions that constrain an underlying eager evaluation strategy: an argument $t_i$ of a function call $f(t_1, \ldots, t_k)$ whose

\textsuperscript{1} Work partially supported by CICYT TIC2001-2705-C03-01, Acciones Integradas H1 2000-0161, HA 2001-0059, HU 2001-0019, and Generalitat Valenciana GV01-424.

\textsuperscript{2} Email: slucas@dsic.upv.es, WWW: http://www.dsic.upv.es/users/elp/slucas.html

©2002 Published by Elsevier Science B. V.
index $i \in \{1, \ldots, k\}$ does not occur in the strategy annotation $(i_1 \ i_2 \ \ldots \ i_n)$ (where $i_1, i_2, \ldots, i_n \in \{0, 1, \ldots, k\}$) associated to the function symbol $f$ is not considered for evaluation. Moreover, even the application of rules at the top must also be explicitly indicated by means of ‘0’ [Eke98]. The presence of such ‘true’ replacement restrictions is often invoked to justify that OBJ programs\(^3\) are able to avoid nontermination despite their (underlying) eager semantics ([GWMFJ00], Section 2.4.4).

**Example 1.1** The following OBJ3 program:

```plaintext
obj EXAMPLE is
  sorts Sort .
  op 0    :-> Sort .
  op s    : Sort -> Sort .
  op cons : Sort Sort -> Sort [strat (1 0)] .
  op inf : Sort -> Sort .
  op nth : Sort Sort -> Sort .
  var X Y L : Sort .
  eq nth(0,cons(X,L)) = X .
  eq nth(s(X),cons(X,L)) = nth(X,L) .
  eq inf(X) = cons(X,inf(s(X))) .
  endo
```

specifies an *explicit* strategy annotation $(1 \ 0)$ for the list constructor `cons` which disables reductions on the second argument\(^4\). In this way, the evaluation of expression `nth(s(0),inf(0))` always finishes and produces the term `s(0)`, even if the ‘infinite list’ `inf(0)` is a part of the expression.

Context-sensitive rewriting (CSR [Luc98]) provides a suitable framework for proving termination of OBJ programs using such strategy annotations (see [Luc01a,Luc01b]). In CSR, a mapping $\mu : \mathcal{F} \rightarrow \mathcal{P}(\mathbb{N})$ is called a replacement map if $\mu(f) \subseteq \{1, \ldots, k\}$ holds for each $k$-ary symbol $f$ of the signature $\mathcal{F}$. Replacement maps are used to discriminate the argument positions on which replacements are allowed. In this way, a rewriting restriction is obtained (see Section 3). Terminating TRSs are $\mu$-terminating (i.e., no term initiates an infinite sequence of CSR under $\mu$). However, CSR can achieve termination, by pruning (all) infinite rewrite sequences. Several methods have been developed to formally prove termination\(^5\) of CSR [BLR02,FR99,GL02,GM99,GM02,Luc96,SX98,Zan97], see [GM02,Luc02c] for a comparison of most of these techniques. For instance, the TRS that corresponds to the OBJ3 program of Example 1.1 can be proved terminating with regard to CSR (see Example 3.2 below). According to [Luc01a,Luc01b], such

---

\(^3\) As in [GWMFJ00], by OBJ we mean OBJ2, OBJ3, CafeOBJ, or Maude.

\(^4\) The other symbols are given a default strategy annotation (see [GWMFJ00]).

\(^5\) See [http://www.dsic.upv.es/users/elp/schemas/muterm](http://www.dsic.upv.es/users/elp/schemas/muterm) for a tool (MU-TERM 1.0) which implements most of these methods.
a proof actually ensures termination of the OBJ3 program.

Using rewriting restrictions may give rise to incomplete computations. For instance, the normal forms of some terms could be unreachable by restricted computation.

**Example 1.2** The following CafeOBJ program (borrowed from [NO01]):

```plaintext
mod! TEST {
    [T]
    op 0    : -> T
    op s   : T -> T       {strat: (1)}
    op cons : T T -> T    {strat: (1)}
    op 2nd : T -> T       {strat: (1 0)}
    op from : T -> T      {strat: (1 0)}
    vars X Y Z : T
    eq 2nd(cons(X,cons(Y,Z))) = Y .
    eq from(X) = cons(X,from(s(X))) .
}
```

specifies a strategy annotation (1) for the list constructor cons that makes the program terminating; however, evaluating \(2nd(from(s(0)))\) into \(s(0)\):

\[
\begin{align*}
2nd(from(0)) & \rightarrow 2nd(cons(0,from(s(0)))) \\
& \rightarrow 2nd(cons(0,cons(s(0),from(s(s(0))))) ) \\
& \rightarrow s(0)
\end{align*}
\]

is not possible. The reason is that reductions on the second argument of cons are disallowed; hence, the second reduction step is no longer possible. On the other hand the evaluation is possible using a local strategy such as \((1\ 2)\), but the following infinite reduction sequence is obtained:

\[
\begin{align*}
2nd(from(0)) & \rightarrow 2nd(cons(0,from(s(0)))) \\
& \rightarrow 2nd(cons(0,cons(s(0),from(s(s(0)))))) \\
& \rightarrow \ldots
\end{align*}
\]

Example 1.2 shows the limits of the current interpretation of syntactic annotations in OBJ programs (that can be given using the CSR framework). Fokkink et al.'s lazy *graph* rewriting [FKW00] provides a different (more liberal) operational model for using syntactic replacement restrictions specified by a replacement map \(\mu\). In Section 4, we adapt Fokkink et al.'s framework to lazy *term* rewriting (LR). Indeed, lazy rewriting is also intended to 'improve the termination behavior of TRSs' [FKW00]. For instance, with lazy rewriting, we can compute the value of \(2nd(from(0))\) (using the replacement restrictions that correspond to the strategy annotation of Example 1.2) without jeopardizing nontermination. Although reductions are (in principle) disallowed on non-replacing arguments of symbols, they are still possible if they can eventually contribute to the application of a rule on a replacing position of the
term.

**Example 1.3** (Continuing Example 1.2) The reduction step

\[ 2\text{nd}(\text{cons}(0, \text{from}(s(0)))) \rightarrow 2\text{nd}(\text{cons}(0, \text{cons}(s(0), \text{from}(s(s(0)))))) \]

is possible with lazy rewriting. In fact, it actually contributes to making the following (crucial) step possible:

\[ 2\text{nd}(\text{cons}(0, \text{cons}(s(0), \text{from}(s(s(0)))))) \rightarrow s(0) \]

However, the reduction step

\[ 2\text{nd}(\text{cons}(0, \text{cons}(s(0), \text{from}(s(s(0)))))) \rightarrow \ldots \]

that potentially ‘originates’ an infinite rewrite sequence is not allowed, since redex \text{from}(s(s(0))) occurs at a non-replacing position without facilitating the application of a rule (namely, the first rule of the program in Example 1.2) on the (trivially) replacing term \[ 2\text{nd}(\text{cons}(0, \text{cons}(s(0), \text{from}(s(s(0)))))). \]

**Remark 1.4** Note that programs in Examples 1.1 and 1.2 could be given an *optimal* normalizing strategy by using other techniques. For instance, it is not difficult to see that both programs are strongly sequential\(^6\). Since they are also orthogonal, both of them admit a computable normalizing strategy [HL91]. Of course, such a strategy proceeds quite differently from the OBJ evaluation strategy and (in general) cannot be simulated as OBJ computations. However, there can also be OBJ programs that cannot be given a normalizing strategy by using the aforementioned techniques, whereas we can still achieve normalizations on the basis of proving their termination and using program transformation techniques, see [Luc02b] and also [Luc02a].

Unfortunately, no analysis of termination of lazy rewriting is yet available. In Section 5, we show that under certain conditions (namely, that all non-variable subterms of the left-hand sides of rules are \(\mu\)-replacing), CSR and LR coincide. In this case, termination of LR is equivalent to termination of CSR and can be studied using the techniques which have been developed for CSR. In Section 6, for the cases where LR and CSR differ, we provide a transformation which permits proving termination of lazy rewriting as termination of CSR for the transformed system. In this way, we can prove termination of LR by using the techniques for proving termination of CSR. The transformation is available for use within \textsc{mu-term} 1.0, where several transformations for proving termination of CSR have also been implemented.

# 2 Preliminaries

Given a set \( A \), \( \mathcal{P}(A) \) denotes the set of all subsets of \( A \). Given a binary relation \( R \) on a set \( A \), we denote its transitive closure by \( R^+ \) and its reflexive

\(^6\) Indeed, they are *inductively sequential* in the sense of [Ant92]; these TRSs are strongly sequential, see [HLM98].
and transitive closure by $R^*$. An element $a \in A$ is an $R$-normal form, if there is no $b$ such that $a R b$. We say that $b$ is an $R$-normal form of $a$ (written $aR^*b$) if $b$ is an $R$-normal form and $aR^*b$. We say that $R$ is terminating iff there is no infinite sequence $a_1 R a_2 R a_3 \cdots$. Throughout the paper, $\mathcal{X}$ denotes a countable set of variables and $\mathcal{F}$ denotes a signature, i.e., a set of function symbols $\{f, g, \ldots\}$, each having a fixed arity given by a mapping $ar : \mathcal{F} \to \mathbb{N}$. The set of terms built from $\mathcal{F}$ and $\mathcal{X}$ is $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A term is said to be linear if it has no multiple occurrences of a single variable. Terms are viewed as labelled trees in the usual way. Positions $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms of $t$. Given positions $p, q$, we denote its concatenation as $p.q$. If $p$ is a position, and $Q$ is a set of positions, $p.Q = \{p.q \mid q \in Q\}$. We denote the empty chain by $\lambda$. The set of positions of a term $t$ is $\mathcal{P}(t)$. Positions of non-variable symbols in $t$ are denoted as $\mathcal{P}(\mathcal{F}, t)$, and $\mathcal{P}(\mathcal{X}, t)$ are the positions of variables. The subterm at position $p$ of $t$ is denoted as $t[p]$, and $t[s]_p$ is the term $t$ with the subterm at position $p$ replaced by $s$. The symbol labelling the root of $t$ is denoted as $\text{root}(t)$.

A rewrite rule is an ordered pair $(l, r)$, written $l \rightarrow r$, with $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\text{Var}(r) \subseteq \text{Var}(l)$. The left-hand side (lhs) of the rule is $l$ and the right-hand side (rhs) is $r$. A TRS is a pair $\mathcal{R} = (\mathcal{F}, \mathcal{R})$ where $\mathcal{R}$ is a set of rewrite rules. $L(\mathcal{R})$ denotes the set of lhs’s of $\mathcal{R}$. A TRS $\mathcal{R}$ is left-linear if for all $l \in L(\mathcal{R})$, $l$ is a linear term. A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ rewrites to $s$ (at position $p$), written $t \overset{p}{\rightarrow}_\mathcal{R} s$ (or just $t \rightarrow s$), if $t[p] = \sigma(l)$ and $s = t[\sigma(r)]_p$, for some rule $l \rightarrow r \in \mathcal{R}$, $p \in \mathcal{P}(\mathcal{R})$ and substitution $\sigma$.

## 3 Context-sensitive rewriting

A mapping $\mu : \mathcal{F} \rightarrow \mathcal{P}(\mathcal{N})$ is a replacement map (or $\mathcal{F}$-map) if $\mu(f) \subseteq \{1, \ldots, ar(f)\}$ for all $f \in \mathcal{F}$ [Luc98]. The ordering $\subseteq$ on $M_\mathcal{F}$, the set of all $\mathcal{F}$-maps, is: $\mu \subseteq \mu'$ if for all $f \in \mathcal{F}$, $\mu(f) \subseteq \mu'(f)$. Thus, $\mu \subseteq \mu'$ means that $\mu$ considers fewer positions than $\mu'$ (for reduction), i.e., $\mu$ is more restrictive than $\mu'$. According to $\subseteq, \mu_\perp$ (resp. $\mu_\top$) which is given by $\mu_\perp(f) = \emptyset$ (resp. $\mu_\top(f) = \{1, \ldots, ar(f)\}$) for all $f \in \mathcal{F}$, is the minimum (maximum) element of $M_\mathcal{F}$.

The set of $\mu$-replacing positions $\mathcal{P}(\mu, t)$ of $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is: $\mathcal{P}(\mu, t) = \{\lambda\}$, if $t \in \mathcal{X}$ and $\mathcal{P}(\mu, t) = \{\lambda\} \cup \bigcup_{l \in \text{root}(t)} : \mathcal{P}(\mu, l)$, if $t \notin \mathcal{X}$. The set of replacing variables $\text{Var}(\mu, t)$ of $t$ is $\text{Var}(\mu, t) = \{x \in \text{Var}(t) \mid \mathcal{P}(\mu, t) \cap \mathcal{P}(\mu, t) \neq \emptyset\}$. In context-sensitive rewriting (CSR [Luc98]), we (only) contract replacing redexes: $t \overset{p}{\rightarrow}_\mu s \overset{p}{\rightarrow}_\mathcal{R} s$ and $p \notin \mathcal{P}(\mu, t)$.

**Example 3.1** Consider the TRS $\mathcal{R}$:

$2nd(x:y:z) \rightarrow y$

$\text{from}(x) \rightarrow x:\text{from}(s(x))$
and \( \mu(:) = \mu(2\text{nd}) = \mu(\text{from}) = \mu(s) = \{1\} \) that correspond\(^7\) to the CafeOBJ program of Example 1.2 (we use : instead of cons), see [Luc01a] for further details about this correspondence. Then we have:

\[
\text{2nd(from}(0)) \rightarrow_\mu \text{2nd}(0:\text{from}(s(0)))
\]

where \( \rightarrow_\mu \) rewriting stops here since \( 1.2 \notin \text{Pos}_\mu(2\text{nd}(0:\text{from}(s(0)))) \).

The \( \rightarrow_\mu \)-normal forms are called \( \mu \)-normal forms. Note that, except for the trivial case \( \mu = \mu_T \), the set of \( \mu \)-normal forms strictly includes normal forms of \( \mathcal{R} \) (e.g., term \( 2\text{nd}(0:\text{from}(s(0))) \) in Example 3.1 is a \( \mu \)-normal form which is not a normal form). A TRS \( \mathcal{R} \) is \( \mu \)-terminating if \( \rightarrow_\mu \) is terminating. As mentioned in the introduction, a number of techniques can be used to prove termination of CSR as termination of a transformed TRS.

**Example 3.2** The TRS \( \mathcal{R} \):

\[
\begin{align*}
\text{nth}(0,x,y) & \rightarrow x \\
\text{nth}(s(x),y,z) & \rightarrow \text{nth}(x,z) \\
\text{inf}(x) & \rightarrow x:\text{inf}(s(x))
\end{align*}
\]

with \( \mu(:) = \mu(s) = \mu(\text{inf}) = \{1\} \) and \( \mu(\text{nth}) = \{1,2\} \) correspond to the OBJ3 program in Example 1.1. Using Zantema’s transformation [Zan97], we obtain the following TRS \( \mathcal{R}_{\mathcal{E}}^{\mu} \):

\[
\begin{align*}
\text{nth}(0,x,y) & \rightarrow x \\
\text{nth}(s(x),y,z) & \rightarrow \text{nth}(x,\text{activate}(z)) \\
\text{inf}(x) & \rightarrow x:\text{inf}'(s(x)) \\
\text{activate}(\text{inf}'(x)) & \rightarrow \text{inf}(x) \\
\text{inf}(x) & \rightarrow \text{inf}'(x) \\
\text{activate}(x) & \rightarrow x
\end{align*}
\]

where \( \text{activate} \) and \( \text{inf}' \) are new symbols introduced by the transformation. This TRS is terminating (use a recursive path ordering based on the precedence \( \text{nth} > \text{activate} > \text{inf} :: :\text{nil} \) and \( \text{inf} :: :\text{inf}' :: :s, \) and giving \( \text{nth} \) the usual lexicographic status). Hence, \( \mathcal{R} \) is \( \mu \)-terminating.

The canonical replacement map \( \mu_{\mathcal{E}}^{\text{can}} \) is the most restrictive replacement map which ensures that the non-variable subterms of the left-hand sides of the rules of \( \mathcal{R} \) are replacing. Note that \( \mu_{\mathcal{E}}^{\text{can}} \) is easily obtained from \( \mathcal{R} = (\mathcal{F}, R) \): for all \( f \in \mathcal{F} \) and \( i \in \{1,\ldots,\text{ar}(f)\} \),

\[
i \in \mu_{\mathcal{E}}^{\text{can}}(f) \iff \exists l \in \text{L}(\mathcal{R}), p \in \text{Pos}_\mathcal{F}(l), (\text{root}(l)_p) = f \land p,i \in \text{Pos}_\mathcal{F}(l)
\]

Let \( CM_{\mathcal{R}} = \{ \mu \in M_{\mathcal{F}} \mid \mu_{\mathcal{R}}^{\text{can}} \subseteq \mu \} \) be the set of replacement maps which are less restrictive than or equally restrictive to \( \mu_{\mathcal{R}}^{\text{can}} \).

**Example 3.3** The canonical replacement map \( \mu_{\mathcal{R}}^{\text{can}} \) for \( \mathcal{R} \) in Example 3.2 is:

\[
\begin{align*}
\mu_{\mathcal{R}}^{\text{can}}(:) & = \mu_{\mathcal{R}}^{\text{can}}(s) = \mu_{\mathcal{R}}^{\text{can}}(\text{inf}) = \emptyset \quad \text{and} \quad \mu_{\mathcal{R}}^{\text{can}}(\text{nth}) = \{1,2\}
\end{align*}
\]

\(^7\)Since \( \mu(c) = \emptyset \) for every constant symbol \( c \), in the remainder of the paper, we only make the replacement map for the other symbols explicit.
Note that $\mu$ in Example 3.2 satisfies $\mu^{an}_R \subseteq \mu$, i.e., $\mu \in CM_R$.

4 Lazy rewriting

In lazy graph rewriting [FKW00], reductions are issued on labelled graphs. We adapt the framework to lazy term rewriting on labelled terms. Following [FKW00], we are going to distinguish between eager and lazy positions of terms. Thus, we label each node (or position) of a term $t$ using $e$ for eager positions or $\ell$ for lazy ones: Let $\mathcal{F}$ be a signature and $\mathcal{L} = \{e, \ell\}$; then, $\mathcal{F} \times \mathcal{L}$ is a new signature of labelled symbols. The labelling of a symbol $f \in \mathcal{F}$ is denoted $f^e$ or $f^\ell$ rather than $\langle f, e \rangle$ or $\langle f, \ell \rangle$. The arities are naturally extended: $ar(f^e) = ar(f^\ell) = ar(f)$ for all $f \in \mathcal{F}$. Then, labelled terms are terms in $\mathcal{T}(\mathcal{F}, \mathcal{X} \times \mathcal{L})$, which we denote as $\mathcal{T}(\mathcal{F}_\mathcal{L}, \mathcal{X}_\mathcal{L})$. Given $t \in \mathcal{T}(\mathcal{F}_\mathcal{L}, \mathcal{X}_\mathcal{L})$ and $p \in \mathcal{P}os(t)$, if $\text{root}(t|_p) = x^e$ ($= x^\ell$) or $\text{root}(t|_p) = f^e$ ($= f^\ell$), then we say that $p$ is an eager (resp. lazy) position of $t$.

Example 4.1 Consider the signature $\mathcal{F}$ of the TRS in Example 3.1 and the following labelled term:

Thus, 1 and 1.2.1 are eager positions; positions $\Lambda, 1.1, 1.2$, and 1.2.1.1 are lazy.

Fokkink et al. use the notion of lazy signature, i.e., a signature $\mathcal{F}$ supplied with a laziness predicate $\Lambda_\mathcal{L}$ on $\mathcal{F} \times \mathbb{N}$ that holds for $(f, i)$ if and only if $1 \leq i \leq ar(f)$ and the $i$th argument of $f$ is lazy (Definition 3.1.1 of [FKW00]). Laziness predicate $\Lambda_\mathcal{L}$ can actually be identified with a replacement map $\mu$:

$$\forall f \in \mathcal{F}, i \in \{1, \ldots, ar(f)\}, (i \in \mu(f) \Leftrightarrow \neg\Lambda_\mathcal{L}(f, i))$$

In the following, we use $\mu$ instead of $\Lambda_\mathcal{L}$. Given $\mu \in M_\mathcal{F}$, the mapping $\text{label}_\mu : \mathcal{T}(\mathcal{F}, \mathcal{X}) \to \mathcal{T}(\mathcal{F}_\mathcal{L}, \mathcal{X}_\mathcal{L})$ provides the following intended labelling of a term: given $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, the topmost position $\Lambda$ of $\text{label}_\mu(s)$ is always eager; given a position $p \in \mathcal{P}os(\text{label}_\mu(s))$ and $i \in \{1, \ldots, ar(\text{root}(s|_p))\}$, position $p,i$ of $\text{label}_\mu(s)$ is lazy if and only $i \notin \mu(\text{root}(s|_p))$; otherwise, it is eager (Definition 3.1.2 of [FKW00]). Formally, $\text{label}_\mu(x) = x^e$, if $x \in \mathcal{X}$, and
\[ \text{label}_\mu(f(s_1, \ldots, s_k)) = f^\epsilon(\text{label}_{f,1}^\epsilon(s_1), \ldots, \text{label}_{f,k}^\epsilon(s_k)), \text{ if } f \in \mathcal{F}, \text{ where} \]

\[
\text{label}_{f,i}^\epsilon(x) = \begin{cases} 
x^\epsilon & \text{if } i \in \mu(f) \\
x^\ell & \text{otherwise}
\end{cases}
\]

\[ \text{label}_{f,i}^\epsilon(g(u_1, \ldots, u_m)) = \begin{cases} 
g^\epsilon(\text{label}_{g,1}^\epsilon(u_1), \ldots, \text{label}_{g,m}^\epsilon(u_m)) & \text{if } i \in \mu(f) \\
g^\ell(\text{label}_{g,1}^\epsilon(u_1), \ldots, \text{label}_{g,m}^\epsilon(u_m)) & \text{otherwise}
\end{cases} \]

**Example 4.2** Consider \( \mathcal{R} \) and \( \mu \) as in Example 3.1. Then, the intended labelling of term

\[ s = 2\text{nd}(0:\text{from}(s(0))) \quad \text{is} \quad t = \text{label}_\mu(s) = 2\text{nd}^\epsilon(0^\epsilon:\text{from}^\epsilon(s^\epsilon(0^\epsilon))). \]

Graphically:

Here, \( 1, 1.1, 1.2.1, \) and \( 1.2.1.1 \) are eager positions of \( t \); position \( 1.2 \) is lazy.

Given \( t \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{X}_\mathcal{C}), \text{erase}(t) \) is the term in \( \mathcal{T}(\mathcal{F}, \mathcal{X}) \) that (obviously) corresponds to \( t \) after removing all labels. Note that \( \text{erase} \circ \text{label}_\mu = \text{id}_{\mathcal{T}(\mathcal{F}, \mathcal{X})} \), but \( \text{label}_\mu \circ \text{erase} \neq \text{id}_{\mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{X}_\mathcal{C})} \).

As mentioned above, given \( t \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{X}_\mathcal{C}), \) a position \( p \in \mathcal{P}os(t) \) is eager (resp. lazy) if \( \text{root}(t|_p) \) is labelled with \( \epsilon \) (resp. \( \ell \)). The so called active positions of \( t \) are defined inductively as follows: \( \Lambda \) is an active position; if \( p \in \mathcal{P}os(t) \) is active, then \( p.i \) is active for all eager positions \( p.i, 1 \leq i \leq ar(\text{root}(t|_p)) \) of \( t \) (Definition 3.1.3 of [FKW00]). Active positions are always reachable from the root of the term via a path of eager positions. Eager positions do not necessarily satisfy this.

**Example 4.3** (continuing Example 4.2) Positions \( \Lambda, 1, \) and \( 1.1 \) are active in

\[ t = 2\text{nd}^\epsilon(0^\epsilon:\text{from}^\epsilon(s^\epsilon(0^\epsilon))) \]

Positions \( 1.2.1 \) and \( 1.2.1.1 \) of \( t \) are eager but not active, since position \( 1.2 \) is lazy in \( t \). Graphically:
Let $\text{Act}(t)$ be the set of active positions of a labelled term $t \in \text{T}(\mathcal{F}, \mathcal{X}, \mathcal{C})$. Given $s \in \text{T}(\mathcal{F}, \mathcal{X})$ and $\mu \in \mathcal{M}_\mathcal{F}$, the set of active positions of $\text{label}_\mu(s)$ coincides with $\text{Pos}^\mu(s)$.

**Proposition 4.4** Let $\mathcal{F}$ be a signature, $\mu \in \mathcal{M}_\mathcal{F}$, and $s \in \text{T}(\mathcal{F}, \mathcal{X})$. Then, $\text{Act}(\text{label}_\mu(s)) = \text{Pos}^\mu(s)$.

An important feature of lazy rewriting on labelled terms is that

*the set of active nodes may increase as reduction of labelled terms proceeds.*

Each lazy rewriting step on labelled terms may have two different effects:

(i) changing the status (active or not) of a given position within a labelled term, or

(ii) performing a rewriting step (always on an active position).

In the following, we formally describe them by using two different binary relations on labelled terms.

### 4.1 Activating positions for reduction

The *activation* status of a lazy position immediately below an active position within a (labelled) term can be modified if the position is ‘essential’, i.e., ‘its contraction may lead to new redexes at active nodes’ [FKW00].

**Definition 4.5** [Matching modulo laziness [FKW00]] Let $l \in \text{T}(\mathcal{F}, \mathcal{X})$ be linear, $t \in \text{T}(\mathcal{F}, \mathcal{X}, \mathcal{C})$, and $p$ be an active position of $t$. Then, $l$ matches modulo laziness $s = t|_p$ if either

(i) $l \in \mathcal{X}$, or

(ii) $l = f(l_1, \ldots, l_k)$, $s = f^c(s_1, \ldots, s_k)$ and, for all $i \in \{1, \ldots, k\}$, if $p.i$ is eager, then $l_i$ matches modulo laziness $s_i$.

If position $p.i$ of $t$ is lazy and $l_i \not\in \mathcal{X}$, then position $p.i$ is called essential.

**Example 4.6** Consider the TRS $\mathcal{R}$ of Example 3.1. The $\text{lhs}$ $\text{2nd}(x:y:z)$ matches modulo laziness the labelled term $t = \text{2nd}^e(0^e:\text{from}^e(s^e(0^e)))$. According to Definition 4.5, position 1.2 of $t$ becomes essential.
Note that if $l \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ matches modulo laziness an active labelled sub-term $s = t|_p$ without producing essential positions, then $l$ matches $\text{erase}(s)$ in the usual sense. Changes in ‘activity’ of positions are formalized by the following.

**Definition 4.7** Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear TRS. The activation relation $\bar{\rightarrow}$ between labelled terms is defined as follows. Let $p$ be active in $t \in \mathcal{T}(\mathcal{F}_L, \mathcal{X}_L)$ and $l \rightarrow r \in R$ be such that $l$ matches modulo laziness $t|_p$. Let $q$ be an essential position of $t$ and $t|_q = f^e(t_1, \ldots, t_k)$. Then, $t \bar{\rightarrow} t[f^e(t_1, \ldots, t_k)]_q$.

Consider the TRS $\mathcal{R}$ in Example 3.1. The following figure shows the activation step that corresponds to term $t$ in Example 4.6.

Note that $\bar{\rightarrow}$ is a terminating relation: only a finite number of relabellings (from lazy to eager) is possible for finite terms. In general, $\bar{\rightarrow}$ is not confluent.

**Example 4.8** Consider the (ground) TRS $\mathcal{R}$:

\[
\begin{align*}
  f(c(d, a)) & \rightarrow a \\
  b & \rightarrow f(c(b, d))
\end{align*}
\]

Then, we have

\[
\begin{align*}
  f^e(c^e(b^e, d^e)) & \bar{\rightarrow} f^e(c^e(b^e, d^e)) \bar{\rightarrow} f^e(c^e(b^e, d^e))
\end{align*}
\]

and $f^e(c^e(b^e, d^e))$ is a $\bar{\rightarrow}$-normal form, since $f(c(d, a))$ does not match term $f^e(c^e(b^e, d^e))$ modulo laziness. However,

\[
\begin{align*}
  f^e(c^e(b^e, d^e)) & \bar{\rightarrow} f^e(c^e(b^e, d^e)) \bar{\rightarrow} f^e(c^e(b^e, d^e))
\end{align*}
\]

thus leading to a different $\bar{\rightarrow}$-normal form.

**Remark 4.9** Note that the activation relation does not use the information contained in the replacement map $\mu$. We make this fact explicit by putting no reference to $\mu$ in the arrow $\bar{\rightarrow}$ which we use to represent it.

Note the following obvious fact.
**Proposition 4.10** Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear TRS and $t, t' \in \mathcal{T}(\mathcal{F}_L, \mathcal{X}_L)$. If $t \xrightarrow{A} t'$, then $\text{erase}(t) = \text{erase}(t')$.

The following proposition establishes that activating new positions is not possible if the labelled term $t$ is obtained by labelling a term $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ using a replacement map $\mu \in CM_{\mathcal{R}}$.

**Proposition 4.11** Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear TRS, $\mu \in CM_{\mathcal{R}}$, and $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Then, $\text{label}_\mu(s)$ is a $\xrightarrow{A}$-normal form.

### 4.2 Reducing active positions

Lazy rewriting reduces active positions. In the following, we formally describe such a process. Note that, according to Fokkink et al.’s formulation, the $lhs$’s and $rhs$’s of rules of the TRS are not labelled terms; they are unlabelled terms that are used to reduce labelled terms. Therefore, as in Definition 4.5, we have to deal with labelled and unlabelled terms. For this reason, the description of the reduction process is slightly more involved than pure rewriting.

**Definition 4.12** Let $l \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ be a linear term, $t \in \mathcal{T}(\mathcal{F}_L, \mathcal{X}_L)$, $p \in \mathcal{P}os(l)$, and $u = t|_p$. If $l$ matches $\text{erase}(u)$, then we let the mapping $\sigma_{l,u} : \mathcal{V}ar(l) \rightarrow \mathcal{T}(\mathcal{F}_L, \mathcal{X}_L)$ be $\sigma_{l,u}(x) = u|_q$ for all $x \in \mathcal{V}ar(l)$.

From $\sigma_{l,u}$ in Definition 4.12, we obtain a substitution $\sigma$ on labelled terms (with variables in $\mathcal{V}ar(l)$) as the homomorphic extension of the following: for all $x \in \mathcal{V}ar(l)$,

$$
\sigma(x^e) = \begin{cases} 
  y^e & \text{if } \sigma_{l,u}(x) = y^l \in \mathcal{X}_L \\
  f^e(t_1, \ldots, t_k) & \text{if } \sigma_{l,u}(x) = f^l(t_1, \ldots, t_k)
\end{cases}
$$

$$
\sigma(x^f) = \begin{cases} 
  y^f & \text{if } \sigma_{l,u}(x) = y^l \in \mathcal{X}_L \\
  f^f(t_1, \ldots, t_k) & \text{if } \sigma_{l,u}(x) = f^l(t_1, \ldots, t_k)
\end{cases}
$$

for $\lambda \in \{e, f\}$. Since we are going to apply such a substitution to the labelled $rhs$ $\text{label}_\mu(r)$ of a (left-linear) rewrite rule $l \rightarrow r$, and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$, our definition suffices for our purposes (see Definition 4.13).

This is according to Definition 3.2.3 in [FKW00]: when a substitution $\sigma$ on labelled terms applies to a labelled term $t$, the labelling that corresponds to the symbol in position $q$ in $\sigma(l)$ is that of $q$ in $t$, for every variable position $q \in \mathcal{P}os_{\mathcal{X}_L}(t)$. Thus, we give the following.

**Definition 4.13** Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear TRS and $\mu \in M_{\mathcal{X}}$. The relation of active rewriting $\xrightarrow{R}_\mu$ between labelled terms is defined as follows. Let $p$ be an active position of $t \in \mathcal{T}(\mathcal{F}_L, \mathcal{X}_L)$, $u = t|_p$ and $l \rightarrow r \in R$ be such that $l$ matches $\text{erase}(u)$. Let $\sigma_{l,u}$ be the corresponding mapping. Then, $t \xrightarrow{R}_\mu t[\sigma(\text{label}_\mu(r))]|_p$. 

11
The following figure shows the reduction step that corresponds to Example 4.6 after the activation step.

Note that term \( 2nd^e(0^e : s^e(0^e) : \text{from}^e(s^e(s^e(0^e)))) \), which is obtained after this \( R^\mu \)-step, is a \( \Lambda \)-normal form.

**Example 4.14** Consider the TRS
\[
f(b, x) \to g(x)
\]
and \( \mu(f) = \mu(g) = \{1\} \). Let \( t = f^e(b^e, a^e) \); notice that \( \text{label}_\mu(g(x)) = g^e(x^e) \). Then, \( f(b, x) \) matches \( \text{erase}(t) \). We have \( \sigma_{f(b, x)}(x) = a^e \). We obtain the substitution \( \sigma \) given by \( \sigma(x^e) = a^e \) and \( \sigma(x^e) = a^e \). Then,
\[
f^e(b^e, a^e) \overset{R^\mu}{\to} g^e(a^e)
\]

**Remark 4.15** Example 4.14 shows that \( R^\mu \)-steps can also indirectly activate lazy positions after contracting a (labelled) redex. For instance, we can think of the \( R^\mu \)-step on \( t = f^e(b^e, a^e) \) as activating the lazy occurrence of \( a \) in \( t \) when it is reduced into \( g^e(a^e) \).

We also note the following obvious fact.

**Proposition 4.16** Let \( R = (F, R) \) be a left-linear TRS, \( \mu \in M_\mathcal{F} \), and \( t, t' \in \mathcal{T}(F_\mathcal{E}, \mathcal{A}_\mathcal{E}) \). If \( t \overset{R^\mu}{\to} t' \), then \( \text{erase}(t) \to \text{erase}(t') \).

### 4.3 Lazy term rewriting

The lazy graph rewriting as given in Definition 3.2.3 of [FKW00] corresponds to relation \( L^R_\mu = \Lambda \cup R^R_\mu \) on labelled terms \( LR \).

**Remark 4.17** Actually, \( L^R_\mu \) permits reduction steps that are not allowed by Definition 3.2.3 of [FKW00] (but all of them can be simulated as \( LR^R_\mu \)-steps). In particular, in the original formulation, rewriting an active position \( p \) of a term \( t \) (i.e., the application of a \( R^R_\mu \)-step at \( t|_p \)) is allowed only after the full
activation of subterms of \( t|_\mu \) (i.e., after obtaining a \( \overset{A}{\rightarrow} \) normal form of \( t|_\mu \)). This fact is not relevant with respect to the main results of this paper and we do not consider them any further here.

Whenever \( LR \) is used for evaluating an unlabelled term \( s \in T(\mathcal{F}, \mathcal{X}) \), we are actually interested in \( LR|_\mu \)-reductions issued from \( \text{label}_\mu(s) \). As done in [NO01,OF00] for OBJ (like) languages (and which is implicit in [FKW00]), we can define an evaluation semantics, i.e., a mapping \( LR\text{-eval}_\mu : T(\mathcal{F}, \mathcal{X}) \rightarrow \mathcal{P}(T(\mathcal{F}, \mathcal{X})) \) that obtains the evaluation of a given term by using \( LR \):

\[
LR\text{-eval}_\mu(s) = \{ \text{erase}(t) \in T(\mathcal{F}, \mathcal{X}) \mid \text{label}_\mu(s) \overset{LR|_\mu}{\rightarrow} t \}
\]

For CSR we can do the same:

\[
CSR\text{-eval}_\mu(s) = \{ s' \in T(\mathcal{F}, \mathcal{X}) \mid s \overset{1}{\rightarrow}_\mu s' \}
\]

Now we can compare both evaluation mechanisms.

**Example 4.18** Consider \( \mathcal{R} \) and \( \mu \) as in Example 3.1 and \( s = 2\text{nd}(\text{from}(0)) \). We have the following \( LR \)-evaluation sequence:

\[
\begin{align*}
\text{label}_\mu(s) &= 2\text{nd}^\ast(\text{from}^\ast(0^\ast)) \overset{R|_\mu}{\rightarrow} 2\text{nd}^\ast(0^\ast : \text{from}^\ast(s^\ast(0^\ast))) \\
&\overset{A}{\rightarrow} 2\text{nd}^\ast(0^\ast : \text{from}^\ast(s^\ast(0^\ast))) \\
&\overset{R|_\mu}{\rightarrow} 2\text{nd}^\ast(0^\ast : s^\ast(0^\ast) : \text{from}^\ast(s^\ast(s^\ast(0^\ast)))) \\
&\overset{R|_\mu}{\rightarrow} s^\ast(0^\ast)
\end{align*}
\]

Therefore,

\[
s(0) \in LR\text{-eval}_\mu(2\text{nd}(\text{from}(0)))
\]

as desired (this follows the discussion in Example 1.3). In contrast,

\[
s(0) \notin CSR\text{-eval}_\mu(2\text{nd}(\text{from}(0))) = \{ 2\text{nd}(0 : \text{from}(s(0))) \}.
\]

According to this, given \( \mathcal{R} = (\mathcal{F}, R) \) and \( \mu \in M_\mathcal{F} \), we say that \( \mathcal{R} \) is \( LR(\mu) \)-terminating if, for all \( s \in T(\mathcal{F}, \mathcal{X}) \), there is no infinite \( \overset{LR|_\mu}{\rightarrow} \) rewrite sequence starting from \( \text{label}_\mu(s) \).

## 5 Lazy rewriting and context-sensitive rewriting

The following connection between \( \overset{R|_\mu}{\rightarrow} \) and CSR is interesting.

**Proposition 5.1** Let \( \mathcal{R} = (\mathcal{F}, R) \) be a left-linear TRS, \( \mu \in M_\mathcal{F} \), \( s \in T(\mathcal{F}, \mathcal{X}) \) and \( t \in T(\mathcal{F}_C, \mathcal{X}_C) \). Then, \( \text{label}_\mu(s) \overset{R|_\mu}{\rightarrow} t \) if and only if \( \exists s' \in T(\mathcal{F}, \mathcal{X}) \), \( s \overset{\mu}{\rightarrow} s' \) and \( t = \text{label}_\mu(s') \).

The following theorem expresses that CSR can always be seen as a restriction of \( LR \) that only considers ‘canonically labelled’ terms.

**Theorem 5.2** Let \( \mathcal{R} = (\mathcal{F}, R) \) be a left-linear TRS, \( \mu \in M_\mathcal{F} \), and \( s, s' \in T(\mathcal{F}, \mathcal{X}) \). Then, \( s \overset{\mu}{\rightarrow} s' \) if and only if \( \text{label}_\mu(s) \overset{LR|_\mu}{\rightarrow} \text{label}_\mu(s') \).

13
The following theorem expresses that \( LR \) can be simulated by \( CSR \) whenever \( \mu \in CM_\mathcal{R} \).

**Theorem 5.3** Let \( \mathcal{R} = (\mathcal{F},R) \) be a left-linear TRS, \( \mu \in CM_\mathcal{R} \), \( s \in T(\mathcal{F},X) \), and \( t \in T(\mathcal{F}_\mathcal{C},X_\mathcal{C}) \). Then, \( \text{label}_\mu(s) \xrightarrow{LR}_\mu t \) if and only if \( \exists s' \in T(\mathcal{F},X), s \leftrightarrow_\mu s' \) and \( t = \text{label}_\mu(s') \).

In this way, \( CSR \) provides an alternative (simpler) evaluation mechanism. We have:

**Corollary 5.4** Let \( \mathcal{R} \) be a left-linear TRS and \( \mu \in CM_\mathcal{R} \). Then, \( LR\text{-eval}_\mu = CSR\text{-eval}_\mu \).

Example 4.18 shows that this result does not hold if \( \mu \notin CM_\mathcal{R} \). Concerning \( LR(\mu) \)-termination, Theorem 5.3 also has the following consequence.

**Corollary 5.5** Let \( \mathcal{R} \) be a left-linear TRS and \( \mu \in CM_\mathcal{R} \). Then, \( \mathcal{R} \) is \( \mu \)-terminating if and only if \( \mathcal{R} \) is \( LR(\mu) \)-terminating.

**Example 5.6** Consider \( \mathcal{R} \) and \( \mu \) as in Example 3.2. Fokkink et al. use this TRS and replacement map \( \mu \) (more precisely, the corresponding laziness predicate \( \Lambda_\mathcal{C} \)) to motivate lazy rewriting to be (hopefully) able ‘to avoid infinite reductions’ ([FKW00], page 47). Since \( \mu \in CM_\mathcal{R} \) (see Example 3.3), by Corollary 5.5, \( LR(\mu) \)-termination and \( \mu \)-termination coincide. Since \( \mathcal{R} \) is \( \mu \)-terminating (see Example 3.2), Corollary 5.5 proves Fokkink et al.’s claim.

### 6 Proving termination of lazy rewriting

Corollary 5.5 is quite limited regarding proofs of \( LR(\mu) \)-termination. However, it provides the basis for proving \( LR(\mu) \)-termination as termination of \( CSR \) for a transformed TRS (and replacement map \( \mu' \)).

In [Ngu01], a transformation of pairs \( \langle \mathcal{R}, \mu \rangle \) of TRSs and replacement maps is proposed to force non-variable subterms of all left-hand sides of rules in \( \mathcal{R} \) to be \( \mu \)-replacing, i.e., to achieve \( \mu^{\geq 0} \mu \subseteq \mu \). The transformation is as follows (see Section 6.1 of [Ngu01]): let \( \mathcal{R} = (\mathcal{F},R) \) be a TRS and \( \mu \in M_\mathcal{F} \). Let \( l \rightarrow r \in R \), \( p \in \mathcal{P}_\mathcal{F}(l) \), \( \text{root}(l[p]) = f \), and \( i \notin \mu(f) \) be such that \( p,i \in \mathcal{P}_\mathcal{F}(l) \). Then we obtain \( \mathcal{R}' = (\mathcal{F}',R') \) and \( \mu' \in M_\mathcal{F} \) as follows: \( \mathcal{F}' = \mathcal{F} \cup \{ f' \} \), where \( f' \) is a new symbol of arity \( \text{ar}(f') = \text{ar}(f) \) such that \( \mu'(f') = \mu(f) \cup \{ i \} \) and \( \mu'(g) = \mu(g) \) for all \( g \in \mathcal{F} \). On the other hand,

\[ R' = R - \{ l \rightarrow r \} \cup \{ l' \rightarrow r, l[x]_{p,i} \rightarrow l'[x]_{p,i} \} \]

where \( l' = l[f'(l[p],\ldots,l[p,k])]_{p} \) if \( \text{ar}(f) = k \) and \( x \) is a new variable.

**Example 6.1** Consider \( \mathcal{R} \) as in Example 4.8 and \( \mu = \mu_\bot \). Then, \( \mathcal{R}' \) is:

\[
\begin{align*}
f_1(c(d,a)) & \rightarrow a \\
& \quad b \rightarrow f(c(b,d)) \\
& \quad f(x) \rightarrow f_1(x)
\end{align*}
\]

and \( \mu'(f) = \mu'(c) = \varnothing \), \( \mu'(f_1) = \{ 1 \} \).
The transformation proceeds like this (starting now from $\mathcal{R}'$ and $\mu'$) until $\mathcal{R}^\#$ and $\mu^\#$ are obtained such that $\mu_{\mathcal{R}^\#}^{con} \subseteq \mu^\#$. In particular, if $\mu_{\mathcal{R}}^{con} \subseteq \mu$, then $\mathcal{R}^\# = \mathcal{R}$ and $\mu^\# = \mu$.

**Example 6.2** Continuing Example 6.1, $\mathcal{R}^\#$ is:

\[
\begin{align*}
&f_1(c_2(d,a)) \rightarrow a & f_1(c(x,y)) \rightarrow f_1(c_3(x,y)) \\
&f_1(c_1(d,x)) \rightarrow f_1(c_2(d,x)) & f(x) \rightarrow f_1(x) \\
&f_1(c_0(x,a)) \rightarrow f_1(c_1(x,a)) & b \rightarrow f(c(b,d))
\end{align*}
\]

and $\mu^\#$ is given by $\mu^\#(f) = \mu^\#(c) = \emptyset$, $\mu^\#(f_1) = \mu^\#(c_1) = \{1\}$, $\mu^\#(c_2) = \{1, 2\}$, and $\mu^\#(c_3) = \{2\}$. Notice that $\mu_{\mathcal{R}^\#}^{con} \subseteq \mu^\#$.

**Remark 6.3** Note that the transformation has some ‘non-determinism’ due to the selection of $f$ and $p$ in each step. For instance, a different possibility (among others) for the first step of Example 6.1 is the following:

\[
\begin{align*}
f(c'(d,a)) & \rightarrow a & b \rightarrow f(c(b,d)) \\
f(c(x,a)) & \rightarrow f(c'(x,a))
\end{align*}
\]

and $\mu''(f) = \mu''(c) = \emptyset$, $\mu''(c') = \{1\}$.

Corollary 5.5 suggests using such a transformation for proving $LR(\mu)$-termination of $\mathcal{R}$ as $\mu^\#$-termination of $\mathcal{R}^\#$, provided that the transformation preserves $LR(\mu)$-termination of $\mathcal{R}$. Unfortunately, this is not true.

**Example 6.4** Consider $\mathcal{R}$ as in Example 4.8, $\mu = \mu_\perp$, and $\mathcal{R}^\#$ and $\mu^\#$ as in Example 6.2. Note that $\mathcal{R}$ is not $LR(\mu)$-terminating: for $t = f(c(b,d))$, we have:

\[
\begin{align*}
\text{label}_t(t) &= f^e(c^e(b^f,d^f)) \xrightarrow{A} f^e(c^e(b^f,d^f)) \xrightarrow{A} f^e(c^e(b^f,d^f)) \\
\xrightarrow{R_{\mu}} f^e(c^e(f^e(c^e(b^f,d^f)),d^f)) \xrightarrow{A^+} f^e(c^e(c^e(b^f,d^f)),d^f) \xrightarrow{R_{\mu}} \ldots
\end{align*}
\]

However, $\mathcal{R}^\#$ is $LR(\mu^\#)$-terminating$^8$. The problem is that some activations of lazy subterms are not possible now:

\[
\begin{align*}
f^e(c^e(b^f,d^f)) & \xrightarrow{R_{\mu^\#}} f_1^e(c^e(b^f,d^f)) \xrightarrow{R_{\mu^\#}} f_1^e(c_3^e(b^f,d^f))
\end{align*}
\]

The lazy subterm $b^f$ cannot be activated; $f_1^e(c_3^e(b^f,d^f))$ is an $\xrightarrow{R_{\mu^\#}}$-normal form. Moreover, since $f_1^e(c_3^e(b^f,d^f)) = \text{label}_t(f_1^e(c_3(b,d)))$ and $\mu^\# \in CM_{\mathcal{R}^\#}$, by Proposition 4.11 it is an $\xrightarrow{A}$-normal form, hence a $\xrightarrow{LR_{\mu^\#}}$-normal form.

A simple modification of Nguyen’s transformation provides a sound technique for proving $LR(\mu)$-termination. The trick is to include all possible activations of lazy (problematic) arguments for each considered symbol: given

---

$^8$ This can be formally proved: According to Corollary 5.5, we just need to prove $\mu^\#$-termination of $\mathcal{R}^\#$. Use Giesl and Middeldorp’s transformation described in the last section of [GM99,GM02] (which is available in $\textsc{mu-term}$ 1.0); termination of the transformed TRS can be automatically proved using $\textsc{CiME}$ 2.0 system (available at $\text{http://cime.lri.fr}$) if the ‘dependency pairs criterion’ (see [AG00]) has been previously activated.
l \rightarrow r \in R \text{ and } p \in Pos(l), \text{ we let}
I(l,p) = \{i \in \{1, \ldots, ar(root(l|p))\} \setminus \mu(root(l|p)) \mid p,i \in Pos_F(l)\}

Assume that \( I(l,p) = \{i_1, \ldots, i_n\} \) for some \( n > 0 \) (i.e., \( I(l,p) \neq \emptyset \)) and let \( f = root(l|p) \). Then, \( R^\circ = (F^\circ, R^\circ) \) and \( \mu^\circ \in M_{F^\circ} \) are as follows: \( F^\circ = F \cup \{f_j \mid 1 \leq j \leq n\} \), where each \( f_j \) is a new symbol of arity \( ar(f_j) = ar(f) \).

We let \( \mu^\circ(f_j) = \mu(f) \cup \{i_j\} \) for \( 1 \leq j \leq n \), and \( \mu^\circ(g) = \mu(g) \) for all \( g \in F \).

On the other hand,
\[
R^\circ = R - \{l \rightarrow r \} \cup \{l'_j \rightarrow r, l[x]_{p,i_j} \rightarrow l'_j[x]_{p,i_j} \mid 1 \leq j \leq n\}
\]
where \( l'_j = l[f_j(l|p,1, \ldots, l|p,k)]_p \) if \( ar(f) = k \), and \( x \) is a new variable.

**Example 6.5** Consider \( \mathcal{R} \) as in Example 4.8 and \( \mu = \mu_\perp \). With the new transformation, we could obtain \( R^\circ \) to be the same as the first \( \mathcal{R}' \) obtained in Example 6.1. On the other hand, if symbol \( c \) (rather than \( f \)) of \( \text{lhs} \) \( f(c(d,a)) \rightarrow a \) is considered, we now obtain:

- \( f(c'(d,a)) \rightarrow a \)
- \( f(c'(d,x)) \rightarrow f(c'(d,x)) \)
- \( f(c'(d,a)) \rightarrow a \)
- \( b \rightarrow f(c(b,d)) \)
- \( f(c(x,a)) \rightarrow f(c'(x,a)) \)

and \( \mu^\circ(f) = \mu^\circ(c) = \emptyset, \mu^\circ(c') = \{1\}, \mu^\circ(c'') = \{2\}. \)

Again, the transformation proceeds like this (now starting from \( \mathcal{R}^\circ \) and \( \mu^\circ \)) until \( \mathcal{R}^5 = (F^5, R^5) \) and \( \mu^5 \) are obtained such that \( \mu^\circ_{R^5} \subseteq \mu^5 \). If \( \mu \in CM_\mathcal{R} \), then \( \mathcal{R}^5 = \mathcal{R} \) and \( \mu^5 = \mu \).

**Example 6.6** Consider \( \mathcal{R} \) to be the same as in Example 4.8 and \( \mu = \mu_\perp \).

Then, \( \mathcal{R}^5 \) is:

- \( f_1(c'_2(d,a)) \rightarrow a \)
- \( f_1(c_2(x,a)) \rightarrow f_1(c'_2(x,a)) \)
- \( f_1(c'_1(d,a)) \rightarrow a \)
- \( f_1(c_4(x,y)) \rightarrow f_1(c_4(x,y)) \)
- \( f_1(c_1(d,x)) \rightarrow f_1(c_1'(d,x)) \)
- \( f_1(c_3(x,a)) \rightarrow f_1(c_1(x,a)) \)

and \( \mu^5 \) is given by \( \mu^5(f) = \mu^5(c) = \emptyset, \mu^5(f_1) = \mu^5(c_1) = \mu^5(c_4) = \{1\}, \mu^5(c'_1) = \mu^5(c'_2) = \{1, 2\}, \) and \( \mu^5(c_2) = \mu^5(c_3) = \{2\}. \) Notice that \( \mu^\circ_{R^5} \subseteq \mu^5 \).

Now, we are able to appropriately simulate every \( \vdash^L_\mu \)-reduction sequence in \( \mathcal{R} \) as a \( \vdash^L_\mu \)-reduction sequence in \( \mathcal{R}^5 \).

**Example 6.7** Consider the term \( t \) of Example 6.4. Now we have the following (infinite) \( \vdash^L_\mu \)-reduction sequence in \( \mathcal{R}^5 \):

\[
\text{label}_\mu(t) = f^e(c'(b^e,d^e)) \Rightarrow^R_\mu \ f^e_1(c'((b^e,d^e)),d^e) \Rightarrow^R_\mu \ f^e_1(c'_4(b^e,d^e)) \\
\Rightarrow^R_\mu \ f^e_1(c'_5(f^e(c'(b^e,d^e)),d^e)) \Rightarrow^R_\mu \ f^e_1(c'_5(c'_4(b^e,d^e)),d^e)) \Rightarrow^R_\mu \ ...
\]

We say that a transformation \( \Theta : (\mathcal{R}, \mu) \mapsto (\mathcal{R}', \mu') \) from pairs (TRS, replacement map) into the same kind of pairs is *correct* (regarding \( LR(\mu) \)-
termination) if $LR(\mu')$-termination of $R'$ implies $LR(\mu)$-termination of $R$. We say that $\Theta$ is complete if $LR(\mu)$-termination of $R$ implies $LR(\mu')$-termination of $R'$. According to our discussion (and since $\mu^2$-termination and $LR(\mu)$-termination coincide, see Corollary 5.5), we have the following.

**Theorem 6.8 (Correctness)** Let $R = (F, R)$ be a left-linear TRS and $\mu \in M_F$. If $R^\mu$ is $\mu^2$-terminating, then $R$ is $LR(\mu)$-terminating.

**Example 6.9** Consider $R$ and $\mu$ as in Example 3.1. Then, $R^\mu$ is:

\[
\begin{align*}
2nd(x: '(y: z)) & \rightarrow y \\
2nd(x: y) & \rightarrow 2nd(x: 'y) \\
\text{from}(x) & \rightarrow x: \text{from}(s(x))
\end{align*}
\]

and $\mu^x$ is given by $\mu^x(2nd) = \mu^x(:) = \mu^x(\text{from}) = \{1\}$ and $\mu^x(':') = \{1, 2\}$.

In fact, in this case $R^\mu$ and $R^\#$ coincide (see Example 6.1 of [Ngu01]). However, using Theorem 6.8, we can prove $LR(\mu)$-termination of $R$, which was an open problem in [Ngu01]: $\mu^2$-termination of $R^\mu$ is proved by using Zantema’s transformation [Zan97]: the TRS

\[
\begin{align*}
2nd(x: '(y: z)) & \rightarrow y \\
2nd(x: y) & \rightarrow 2nd(x: '\text{activate}(y)) \\
\text{from}(x) & \rightarrow x: \text{from}'(s(x)) \\
\text{activate}(\text{from}'(x)) & \rightarrow \text{from}(x) \\
\text{from}(x) & \rightarrow \text{from}'(x) \\
\text{activate}(x) & \rightarrow x
\end{align*}
\]

obtained in this way (where $\text{activate}$ and $\text{from}'$ are new symbols introduced by Zantema’s transformation) is terminating\(^9\). Note that, since $\mu \notin CM_R$, Corollary 5.5 does not apply to $R$ and $\mu$.

We conjecture that our transformation is not only correct but also complete.

**Conjecture 6.10 (Completeness)** Let $R = (F, R)$ be a left-linear TRS and $\mu \in M_F$. If $R$ is $LR(\mu)$-terminating, then $R^\mu$ is $\mu^2$-terminating.

Thus, we could say that termination of $LR$ is completely equivalent to termination of $CSR$.

## 7 Conclusions and future work

We have provided an adaptation of lazy graph rewriting of [FKW00] to lazy term rewriting, $LR$. An alternative presentation can be found in [Ngu01]. We believe that our formalization is simpler and closer to [FKW00]. If we use replacement maps $\mu$ that are less restrictive than the canonical replacement map $\mu^{CM}$, then $CSR$ and $LR$ coincide for left-linear TRSs $R$. In this case,

\(^9\) Use the CiME 2.0 system again.
it makes sense to use CSR as it is the simplest one. By looking for better
implementations of LR, [FKW00,Ngu01] pay some attention to developing
transformation techniques to achieve this condition thereby (silently) using
CSR rather than LR. This also allows us to prove termination of LR by
proving termination of CSR for a transformed rewriting system. As far as the
author knows, this is the first proposal of a technique for proving termination
of LR.

We hope that our results may contribute to formally addressing the prob-
lem of specifying more general strategy annotations in OBJ programs (see
[OF00,NO01]): negative annotations have been recently proposed for achieving
the desirable trade-off between termination and completeness discussed in
the introduction (see Examples 1.2 and 1.3). Such negative indices indicate
that the corresponding argument is evaluated ‘on-demand’, where a ‘demand’
is an attempt to match a pattern to the term that occurs in such an argument
position [Eke98,GWMFJ00,OF00]. Note that, according to [Luc01a],
CSR (not LR) is the restriction of rewriting that can be used to model OBJ
computations of programs using positive strategy annotations. For instance,
the CafeOBJ program in Example 1.2 is terminating because the correspond-
ing TRS $\mathcal{R}$ is $\mu$-terminating, where $\mathcal{R}$ and $\mu$ are the same as in Example
3.1. The proof of $\mu$-termination of $\mathcal{R}$ can easily be achieved using Zantema’s
transformation. However, as shown in Example 1.2, in this case, we do not
achieve completeness in evaluations. As discussed in Example 1.2, relaxing
the restrictions on the list constructor by adding a new positive annotation
for the second argument of cons is dangerous. Therefore, no completely satis-
factory behavior can be obtained with positive annotations for the considered
program. For this reason, negative annotations have been proposed.

**Example 7.1** The following version of the CafeOBJ program of Example 1.2
(borrowed from [NO01]):

```
mod TEST {
  [T]
  op 0  : -> T
  op s : T -> T {strat: (1)}
  op cons : T T -> T {strat: (1 -2)}
  op 2nd : T -> T {strat: (1 0)}
  op from : T -> T {strat: (1 0)}
  vars X Y Z : T
  eq 2nd(cons(X,cons(Y,Z))) = Y .
  eq from(X) = cons(X,from(s(X))) .
}
```

associates negative annotations to the operator cons.

Unfortunately, the operational semantics of CafeOBJ programs using strategy
annotations with negative indices has not been related to either CSR or LR
yet. In [Luc01a], we have proposed on-demand rewriting (ODR) as a suitable extension of CSR that can cope with negative annotations. Unfortunately, in contrast to OBJ programs with positive strategy annotations (regarding CSR), it is not clear whether computations of OBJ programs with negative strategy annotations can be appropriately (or easily) expressed using ODR. Thus, despite the fact that [Luc01a] describes a technique for proving termination of ODR, it is not clear that such a technique correctly applies to the CafeOBJ program in Example 7.1.

Also, Fokkink et al.’s lazy rewriting is invoked in [OF00,Ngu01,NO01] as being a kind of ‘underlying’ or ‘inspiring’ mechanism for dealing with the negative indices in strategies annotations. However, no clear connection between lazy rewriting and computations of OBJ programs with negative annotations has yet been established. Therefore, more work remains to be done before applying the LR (or ODR) framework to model such programs.

References


