Electronic Notes in Theoretical Computer Science 64 (2002)
URL: http://www.elsevier.nl/locate/entcs/volume64.html 21 pages

Lazy Rewriting and Context-Sensitive Rewriting

Salvador Lucas [1,2]

DSIC
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 (*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

[1] Work partially supported by CICYT TIC2001-2705-C03-01, Acciones Integradas HI 2000-0161, HA 2001-0059, HU 2001-0019, and Generalitat Valenciana GV01-424.
[2] Email: slucas@dsic.upv.es, WWW: http://www.dsic.upv.es/users/elp/slucas.html

index $i \in \{1, \ldots, k\}$ does not occur in the strategy annotation $(i_1\ i_2\ \cdots\ i_n)$ (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:

```
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 $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ 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 *μ-terminating* (i.e., no term initiates an infinite sequence of *CSR* under μ). 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/slucas/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]):

```
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)`:

$$\begin{array}{lll}
\texttt{2nd(}\underline{\texttt{from(0)}}\texttt{)} & \to & \texttt{2nd(cons(0,}\underline{\texttt{from(s(0))}}\texttt{))}\\
 & \to & \underline{\texttt{2nd(cons(0,cons(s(0),from(s(s(0))))))}}\\
 & \to & \texttt{s(0)}
\end{array}$$

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:

$$\begin{array}{lll}
\texttt{2nd(}\underline{\texttt{from(0)}}\texttt{)} & \to & \texttt{2nd(cons(0,}\underline{\texttt{from(s(0))}}\texttt{))}\\
 & \to & \texttt{2nd(cons(0,cons(s(0),}\underline{\texttt{from(s(s(0)))}}\texttt{)))}\\
 & \to & \cdots
\end{array}$$

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 μ. 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

$$\texttt{2nd(cons(0,}\underline{\texttt{from(s(0))}}\texttt{))} \to \texttt{2nd(cons(0,cons(s(0),from(s(s(0))))))}$$

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

$$\underline{\texttt{2nd(cons(0,cons(s(0),from(s(s(0))))))}} \to \texttt{s(0)}$$

However, the reduction step

$$\texttt{2nd(cons(0,cons(s(0),}\underline{\texttt{from(s(s(0)))}}\texttt{)))} \to \cdots$$

that potentially 'originates' an infinite rewrite sequence is not allowed, since redex `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 `2nd(cons(0,cons(s(0),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 μ-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 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 $a\,R^!\,b$) if b is an R-normal form and $a\ R^*b$. We say that R is *terminating* iff there is no infinite sequence $a_1\ R\ a_2\ R\ a_3 \cdots$. Throughout the paper, $\mathcal{X}$ denotes a countable set of variables and $\mathcal{F}$ denotes a signature, i.e., a set of function symbols $\{\mathtt{f}, \mathtt{g}, \ldots\}$, 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 Λ. The set of positions of a term t is $\mathcal{P}os(t)$. Positions of non-variable symbols in t are denoted as $\mathcal{P}os_{\mathcal{F}}(t)$, and $\mathcal{P}os_{\mathcal{X}}(t)$ are the positions of variables. The subterm at position p of t is denoted as $t|_p$, and $t[s]_p$ is the term t with the subterm at position p replaced by s. The symbol labelling the root of t is denoted as $root(t)$.

A rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side (*lhs*) of the rule is l and the right-hand side (*rhs*) is r. A TRS is a pair $\mathcal{R} = (\mathcal{F}, R)$ where 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 $t \overset{p}{\to}_{\mathcal{R}} s$ (or just $t \to s$), if $t|_p = \sigma(l)$ and $s = t[\sigma(r)]_p$, for some rule $l \to r \in R$, $p \in \mathcal{P}os(t)$ and substitution σ.

3 Context-sensitive rewriting

A mapping $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ 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 $\sqsubseteq$ on $M_{\mathcal{F}}$, the set of all $\mathcal{F}$-maps, is: $\mu \sqsubseteq \mu'$ if for all $f \in \mathcal{F}$, $\mu(f) \subseteq \mu'(f)$. Thus, $\mu \sqsubseteq \mu'$ means that μ considers fewer positions than μ' (for reduction), i.e., μ is more restrictive than μ'. According to $\sqsubseteq$, $\mu_\bot$ (resp. $\mu_\top$) which is given by $\mu_\bot(f) = \varnothing$ (resp. $\mu_\bot(f) = \{1, \ldots, ar(f)\}$) for all $f \in \mathcal{F}$, is the minimum (maximum) element of $M_{\mathcal{F}}$.

The set of *μ-replacing positions* $\mathcal{P}os^\mu(t)$ of $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is: $\mathcal{P}os^\mu(t) = \{\Lambda\}$, if $t \in \mathcal{X}$ and $\mathcal{P}os^\mu(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(root(t))} i.\mathcal{P}os^\mu(t|_i)$, if $t \notin \mathcal{X}$. The set of *replacing* variables $\mathcal{V}ar^\mu(t)$ of t is $\mathcal{V}ar^\mu(t) = \{x \in \mathcal{V}ar(t) \mid \mathcal{P}os_x(t) \cap \mathcal{P}os^\mu(t) \neq \varnothing\}$. In *context-sensitive rewriting* (*CSR* [Luc98]), we (only) contract *replacing* redexes: t μ-rewrites to s (written $t \hookrightarrow_\mu s$) if $t \overset{p}{\to}_{\mathcal{R}} s$ and $p \in \mathcal{P}os^\mu(t)$.

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

```
2nd(x:y:z) → y
from(x)    → x:from(s(x))
```

and $\mu($`:`$) = \mu($`2nd`$) = \mu($`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:

$$\texttt{2nd(}\underline{\texttt{from(0)}}\texttt{)} \hookrightarrow_\mu \texttt{2nd(0:from(s(0)))}$$

where μ-rewriting stops here since $1.2 \notin \mathcal{P}os^\mu($`2nd(0:from(s(0)))`$)$.

The $\hookrightarrow_\mu$-normal forms are called μ-normal forms. Note that, except for the trivial case $\mu = \mu_\top$, the set of μ-normal forms strictly includes normal forms of $\mathcal{R}$ (e.g., term `2nd(0:from(s(0)))` in Example 3.1 is a μ-normal form which is not a normal form). A TRS $\mathcal{R}$ is μ-terminating if $\hookrightarrow_\mu$ 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 $\mathcal{R}$:

```
nth(0,x:y)     → x
nth(s(x),y:z)  → nth(x,z)
inf(x)         → x:inf(s(x))
```

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

```
nth(0,x:y)          → x
nth(s(x),y:z)       → nth(x,activate(z))
inf(x)              → x:inf'(s(x))
activate(inf'(x))   → inf(x)
inf(x)              → inf'(x)
activate(x)         → x
```

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

The canonical replacement map $\mu_\mathcal{R}^{can}$ 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 $\mu_\mathcal{R}^{can}$ is easily obtained from $\mathcal{R} = (\mathcal{F}, R)$: for all $f \in \mathcal{F}$ and $i \in \{1, \ldots, ar(f)\}$,

$$i \in \mu_\mathcal{R}^{can}(f) \quad \text{iff} \quad \exists l \in L(\mathcal{R}), p \in \mathcal{P}os_\mathcal{F}(l), (root(l|_p) = f \wedge p.i \in \mathcal{P}os_\mathcal{F}(l))$$

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

Example 3.3 The canonical replacement map $\mu_\mathcal{R}^{can}$ for $\mathcal{R}$ in Example 3.2 is:

$$\mu_\mathcal{R}^{can}(\texttt{:}) = \mu_\mathcal{R}^{can}(\texttt{s}) = \mu_\mathcal{R}^{can}(\texttt{inf}) = \varnothing \quad \text{and} \quad \mu_\mathcal{R}^{can}(\texttt{nth}) = \{1,2\}$$

[7] Since $\mu(c) = \varnothing$ for every constant symbol c, in the remainder of the paper, we only make the replacement map for the other symbols explicit.

Note that μ in Example 3.2 satisfies $\mu^{can}_{\mathcal{R}} \sqsubseteq \mu$, i.e., $\mu \in CM_{\mathcal{R}}$.

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 ℓ 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^ℓ 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 $\mathcal{T}(\mathcal{F} \times \mathcal{L}, \mathcal{X} \times \mathcal{L})$, 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 $root(t|_p) = x^e$ $(= x^\ell)$ or $root(t|_p) = f^e$ $(= f^\ell)$, 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:

$$\begin{tikzcd}
 & \mathtt{2nd}^\ell \arrow[d] & \\
 & :^e \arrow[dl] \arrow[dr] & \\
\mathtt{0}^\ell & & \mathtt{from}^\ell \arrow[d] \\
 & & \mathtt{s}^e \arrow[d] \\
 & & \mathtt{0}^\ell
\end{tikzcd}$$

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 ith argument of f is lazy (Definition 3.1.1 of [FKW00]). Laziness predicate $\Lambda_\mathcal{L}$ can actually be identified with a replacement map μ:

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

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

$label_\mu(f(s_1,\ldots,s_k)) = f^e(label'_{f,1}(s_1),\ldots,label'_{f,k}(s_k))$, if $f \in \mathcal{F}$, where

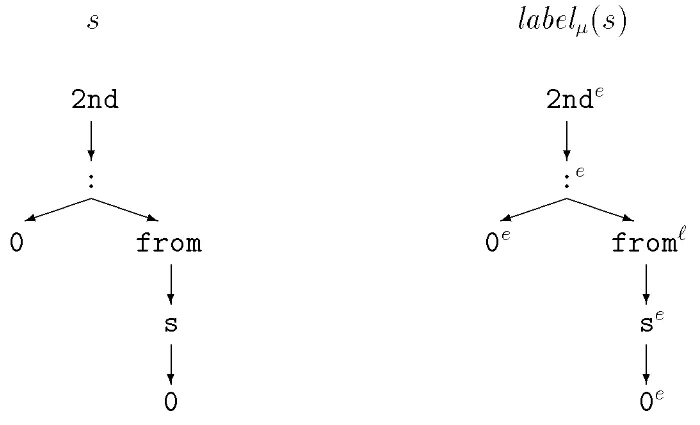
$$label'_{f,i}(x) = \begin{cases} x^e \text{ if } i \in \mu(f) \\ x^\ell \text{ otherwise} \end{cases}$$

$$label'_{f,i}(g(u_1,\ldots,u_m)) = \begin{cases} g^e(label'_{g,1}(u_1),\ldots,label'_{g,m}(u_m)) \text{ if } i \in \mu(f) \\ g^\ell(label'_{g,1}(u_1),\ldots,label'_{g,m}(u_m)) \text{ otherwise} \end{cases}$$

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

$s = \texttt{2nd(0:from(s(0)))}$ is $t = label_\mu(s) = \texttt{2nd}^e\texttt{(0}^e\texttt{:}^e\texttt{from}^\ell\texttt{(s}^e\texttt{(0}^e\texttt{)))}$.

Graphically:



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

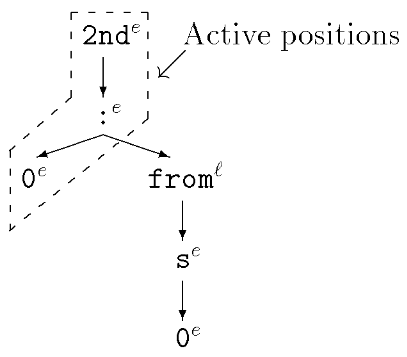
Given $t \in \mathcal{T}(\mathcal{F}_\mathcal{L},\mathcal{X}_\mathcal{L})$, $erase(t)$ is the term in $\mathcal{T}(\mathcal{F},\mathcal{X})$ that (obviously) corresponds to t after removing all labels. Note that $erase \circ label_\mu = id_{\mathcal{T}(\mathcal{F},\mathcal{X})}$, but $label_\mu \circ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

As mentioned above, given $t \in \mathcal{T}(\mathcal{F}_\mathcal{L},\mathcal{X}_\mathcal{L})$, a position $p \in \mathcal{P}os(t)$ is eager (resp. lazy) if $root(t|_p)$ is labelled with e (resp. ℓ). The so called *active* positions of t are defined inductively as follows: Λ is an active position; if $p \in \mathcal{P}os(t)$ is active, then $p.i$ is active for all *eager* positions $p.i$, $1 \leq i \leq ar(root(t|_p))$ 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

$$t = \texttt{2nd}^e\texttt{(0}^e\texttt{:}^e\texttt{from}^\ell\texttt{(s}^e\texttt{(0}^e\texttt{)))}$$

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 $\mathcal{A}ct(t)$ be the set of active positions of a labelled term $t \in \mathcal{T}(\mathcal{F}_\mathcal{L}, \mathcal{X}_\mathcal{L})$. Given $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $\mu \in M_\mathcal{F}$, the set of active positions of $label_\mu(s)$ coincides with $\mathcal{P}os^\mu(s)$.

Proposition 4.4 *Let $\mathcal{F}$ be a signature, $\mu \in M_\mathcal{F}$, and $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Then, $\mathcal{A}ct(label_\mu(s)) = \mathcal{P}os^\mu(s)$.*

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 $l \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ be linear, $t \in \mathcal{T}(\mathcal{F}_\mathcal{L}, \mathcal{X}_\mathcal{L})$, 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) $l = f(l_1, \ldots, l_k)$, $s = f^e(s_1, \ldots, s_k)$ and, for all $i \in \{1, \ldots, k\}$, if $p.i$ is eager, then l_i matches modulo laziness s_i.

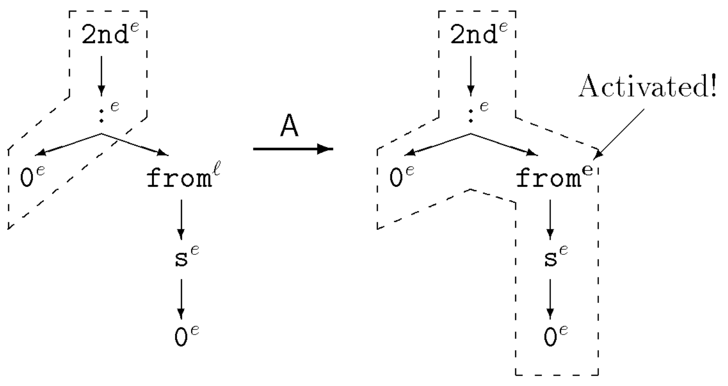
If position $p.i$ of t is lazy and $l_i \notin \mathcal{X}$, then position $p.i$ is called *essential.*

Example 4.6 Consider the TRS $\mathcal{R}$ of Example 3.1. The *lhs* `2nd(x:y:z)` *matches modulo laziness* the labelled term $t =$ `2nd`^{e}`(0`^{e}`:`^{e}`from`$^\ell$`(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 subterm $s = t|_p$ without producing essential positions, then l matches *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 $\overset{\mathsf{A}}{\to}$ between labelled terms is defined as follows. Let p be active in $t \in \mathcal{T}(\mathcal{F}_\mathcal{L}, \mathcal{X}_\mathcal{L})$ and $l \to r \in R$ be such that l matches modulo laziness $t|_p$. Let q be an essential position of t and $t|_q = f^\ell(t_1, \ldots, t_k)$. Then, $t \overset{\mathsf{A}}{\to} t[f^e(t_1, \ldots, t_k)]_q$.

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

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

```
f(c(d,a)) → a
b         → f(c(b,d))
```

Then, we have

$$\mathtt{f}^e(\mathtt{c}^\ell(\mathtt{b}^\ell,\mathtt{d}^\ell)) \overset{\mathsf{A}}{\to} \mathtt{f}^e(\mathtt{c}^e(\mathtt{b}^\ell,\mathtt{d}^\ell)) \overset{\mathsf{A}}{\to} \mathtt{f}^e(\mathtt{c}^e(\mathtt{b}^e,\mathtt{d}^\ell))$$

and $\mathtt{f}^e(\mathtt{c}^e(\mathtt{b}^e,\mathtt{d}^\ell))$ is a $\overset{\mathsf{A}}{\to}$-normal form, since `f(c(d,a))` does not match term $\mathtt{f}^e(\mathtt{c}^e(\mathtt{b}^e,\mathtt{d}^\ell))$ modulo laziness. However,

$$\mathtt{f}^e(\mathtt{c}^\ell(\mathtt{b}^\ell,\mathtt{d}^\ell)) \overset{\mathsf{A}}{\to} \mathtt{f}^e(\mathtt{c}^e(\mathtt{b}^\ell,\mathtt{d}^\ell)) \overset{\mathsf{A}}{\to} \mathtt{f}^e(\mathtt{c}^e(\mathtt{b}^\ell,\mathtt{d}^e))$$

thus leading to a different $\overset{\mathsf{A}}{\to}$-normal form.

Remark 4.9 Note that the activation relation does *not* use the information contained in the replacement map μ. We make this fact explicit by putting no reference to μ in the arrow $\overset{\mathsf{A}}{\to}$ 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 $t, t' \in \mathcal{T}(\mathcal{F}_\mathcal{L}, \mathcal{X}_\mathcal{L})$. If $t \overset{\mathsf{A}}{\to} t'$, then $erase(t) = erase(t')$.*

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, $label_\mu(s)$ is a $\overset{\mathsf{A}}{\to}$-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, $t \in \mathcal{T}(\mathcal{F}_\mathcal{L}, \mathcal{X}_\mathcal{L})$, $p \in \mathcal{P}os(t)$, and $u = t|_p$. If l matches $erase(u)$, then we let the mapping $\sigma_{l,u} : \mathcal{V}ar(l) \to \mathcal{T}(\mathcal{F}_\mathcal{L}, \mathcal{X}_\mathcal{L})$ 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 σ on labelled terms (with variables in $\mathcal{V}ar(l)$) as the homomorphic extension of the following: for all $x \in \mathcal{V}ar(l)$,

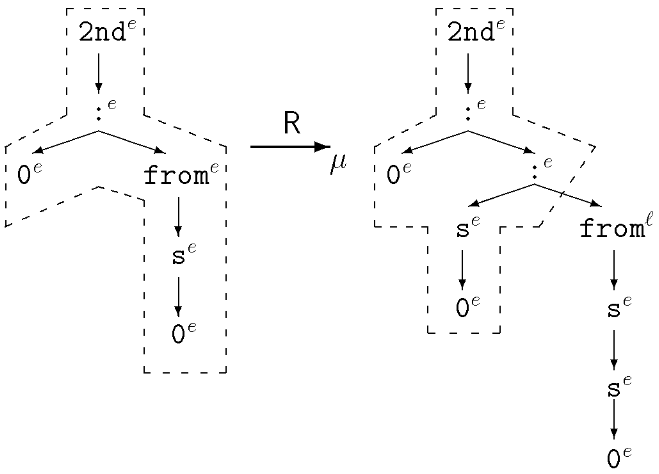
$$\sigma(x^e) = \begin{cases} y^e & \text{if } \sigma_{l,u}(x) = y^\lambda \in \mathcal{X}_\mathcal{L} \\ f^e(t_1, \ldots, t_k) & \text{if } \sigma_{l,u}(x) = f^\lambda(t_1, \ldots, t_k) \end{cases}$$
$$\sigma(x^\ell) = \begin{cases} y^\ell & \text{if } \sigma_{l,u}(x) = y^\lambda \in \mathcal{X}_\mathcal{L}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

for $\lambda \in \{e, \ell\}$. Since we are going to apply such a substitution to the labelled *rhs* $label_\mu(r)$ of a (left-linear) rewrite rule $l \to r$, 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 σ on labelled terms applies to a labelled term t, the labelling that corresponds to the symbol in position q in $\sigma(t)$ is that of q in t, for every variable position $q \in \mathcal{P}os_{\mathcal{X}_\mathcal{L}}(t)$. Thus, we give the following.

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

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



Note that term $\mathtt{2nd}^e(\mathtt{0}^e\mathtt{:}^e\mathtt{s}^e(\mathtt{0}^e)\mathtt{:}^e\mathtt{from}^\ell(\mathtt{s}^e(\mathtt{s}^e(\mathtt{0}^e))))$, which is obtained after this $\overset{\textsf{R}}{\to}_\mu$-step, is a $\overset{\textsf{A}}{\to}$-normal form.

Example 4.14 Consider the TRS

$\mathtt{f(b,x)} \to \mathtt{g(x)}$

and $\mu(\mathtt{f}) = \mu(\mathtt{g}) = \{1\}$. Let $t = \mathtt{f}^e(\mathtt{b}^e,\mathtt{a}^\ell)$; notice that $\mathit{label}_\mu(\mathtt{g(x)}) = \mathtt{g}^e(\mathtt{x}^e)$. Then, $\mathtt{f(b,x)}$ matches $\mathit{erase}(t)$. We have $\sigma_{\mathtt{f(b,x)},t}(\mathtt{x}) = \mathtt{a}^\ell$. We obtain the substitution σ given by $\sigma(\mathtt{x}^e) = \mathtt{a}^e$ and $\sigma(\mathtt{x}^\ell) = \mathtt{a}^\ell$. Then,

$\mathtt{f}^e(\mathtt{b}^e,\mathtt{a}^\ell) \overset{\textsf{R}}{\to}_\mu \mathtt{g}^e(\mathtt{a}^e)$

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

We also note the following obvious fact.

Proposition 4.16 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear TRS, $\mu \in M_{\mathcal{F}}$, and $t, t' \in \mathcal{T}(\mathcal{F}_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}})$. If $t \overset{\textsf{R}}{\to}_\mu t'$, then $\mathit{erase}(t) \to \mathit{erase}(t')$.*

4.3 Lazy term rewriting

The lazy *graph* rewriting as given in Definition 3.2.3 of [FKW00] corresponds to relation $\overset{\textsf{LR}}{\to}_\mu = \overset{\textsf{A}}{\to} \cup \overset{\textsf{R}}{\to}_\mu$ on labelled *terms LR*.

Remark 4.17 Actually, $\overset{\textsf{LR}}{\to}_\mu$ permits reduction steps that are not allowed by Definition 3.2.3 of [FKW00] (but all of them can be simulated as $\overset{\textsf{LR}}{\to}_\mu$-steps). In particular, in the original formulation, rewriting an active position p of a term t (i.e., the application of a $\overset{\textsf{R}}{\to}_\mu$-step at $t|_p$) is allowed *only after the full* *activation of subterms of* $t|_p$ (i.e., after obtaining a $\overset{\mathsf{A}}{\to}$-normal form of $t|_p$). 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 $s \in \mathcal{T}(\mathcal{F},\mathcal{X})$, we are actually interested in $\overset{\mathsf{LR}}{\to}_\mu$-reductions issued from $label_\mu(s)$. 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 $LR\text{-}eval_\mu : \mathcal{T}(\mathcal{F},\mathcal{X}) \to \mathcal{P}(\mathcal{T}(\mathcal{F},\mathcal{X}))$ that obtains the evaluation of a given term by using *LR*:

$$LR\text{-}eval_\mu(s) = \{erase(t) \in \mathcal{T}(\mathcal{F},\mathcal{X}) \mid label_\mu(s) \overset{\mathsf{LR}}{\longrightarrow}{}^{!}_{\mu}\, t\}$$

For *CSR* we can do the same:

$$CSR\text{-}eval_\mu(s) = \{s' \in \mathcal{T}(\mathcal{F},\mathcal{X}) \mid s \hookrightarrow^{!}_{\mu} s'\}.$$

Now we can compare both evaluation mechanisms.

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

$$\begin{array}{l} label_\mu(s) = \mathtt{2nd}^e\mathtt{(from}^e\mathtt{(0}^e\mathtt{))} \overset{\mathsf{R}}{\to}_\mu \mathtt{2nd}^e\mathtt{(0}^e\mathtt{:}^e\mathtt{from}^\ell\mathtt{(s}^e\mathtt{(0}^e\mathtt{)))} \\ \overset{\mathsf{A}}{\to} \mathtt{2nd}^e\mathtt{(0}^e\mathtt{:}^e\mathtt{from}^e\mathtt{(s}^e\mathtt{(0}^e\mathtt{)))} \\ \overset{\mathsf{R}}{\to}_\mu \mathtt{2nd}^e\mathtt{(0}^e\mathtt{:}^e\mathtt{s}^e\mathtt{(0}^e\mathtt{):}^e\mathtt{from}^\ell\mathtt{(s}^e\mathtt{(s}^e\mathtt{(0}^e\mathtt{))))} \\ \overset{\mathsf{R}}{\to}_\mu \mathtt{s}^e\mathtt{(0}^e\mathtt{)} \end{array}$$

Therefore,

$$\mathtt{s(0)} \in LR\text{-}eval_\mu(\mathtt{2nd(from(0))})$$

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

$$\mathtt{s(0)} \notin CSR\text{-}eval_\mu(\mathtt{2nd(from(0))}) = \{\mathtt{2nd(0:from(s(0)))}\}.$$

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 $s \in \mathcal{T}(\mathcal{F},\mathcal{X})$, there is no infinite $\overset{\mathsf{LR}}{\to}_\mu$-rewrite sequence starting from $label_\mu(s)$.

5 Lazy rewriting and context-sensitive rewriting

The following connection between $\overset{\mathsf{R}}{\to}_\mu$ and *CSR* is interesting.

Proposition 5.1 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear TRS, $\mu \in M_{\mathcal{F}}$, $s \in \mathcal{T}(\mathcal{F},\mathcal{X})$ and $t \in \mathcal{T}(\mathcal{F}_{\mathcal{L}},\mathcal{X}_{\mathcal{L}})$. Then, $label_\mu(s) \overset{\mathsf{R}}{\to}_\mu t$ if and only if $\exists s' \in \mathcal{T}(\mathcal{F},\mathcal{X}), s \hookrightarrow_\mu s'$ and $t = label_\mu(s')$.*

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 $s, s' \in \mathcal{T}(\mathcal{F},\mathcal{X})$. Then, $s \hookrightarrow_\mu s'$ if and only if $label_\mu(s) \overset{\mathsf{LR}}{\to}_\mu label_\mu(s')$.*

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, $\mu \in CM_{\mathcal{R}}$, $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, and $t \in \mathcal{T}(\mathcal{F}_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}})$. Then, $label_\mu(s) \overset{\mathsf{LR}}{\to}_\mu t$ if and only if $\exists s' \in \mathcal{T}(\mathcal{F}, \mathcal{X}), s \hookrightarrow_\mu s'$ and $t = label_\mu(s')$.*

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-$eval_\mu =$ CSR-$eval_\mu$.

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 μ-terminating if and only if $\mathcal{R}$ is* LR(μ)*-terminating.*

Example 5.6 Consider $\mathcal{R}$ and μ as in Example 3.2. Fokkink et al. use this TRS and replacement map μ (more precisely, the corresponding laziness predicate $\Lambda_{\mathcal{L}}$) 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 μ-termination coincide. Since $\mathcal{R}$ is μ-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 μ').

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 μ-replacing, i.e., to achieve $\mu^{can}_{\mathcal{R}} \sqsubseteq \mu$. 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 $l \to r \in R$, $p \in \mathcal{P}os_{\mathcal{F}}(l)$, $root(l|_p) = f$, and $i \notin \mu(f)$ be such that $p.i \in \mathcal{P}os_{\mathcal{F}}(l)$. 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 $ar(f') = ar(f)$ such that $\mu'(f') = \mu(f) \cup \{i\}$ and $\mu'(g) = \mu(g)$ for all $g \in \mathcal{F}$. On the other hand,

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

where $l' = l[f'(l|_{p.1}, \ldots, l|_{p.k})]_p$ if $ar(f) = k$ and x is a new variable.

Example 6.1 Consider $\mathcal{R}$ as in Example 4.8 and $\mu = \mu_\perp$. Then, $\mathcal{R}'$ is:

$$\begin{array}{lll} \mathtt{f}_1\mathtt{(c(d,a))} \to \mathtt{a} & & \mathtt{b} \to \mathtt{f(c(b,d))} \\ \mathtt{f(x)} \to \mathtt{f}_1\mathtt{(x)} & & \end{array}$$

and $\mu'(\mathtt{f}) = \mu'(\mathtt{c}) = \varnothing$, $\mu'(\mathtt{f}_1) = \{1\}$.

The transformation proceeds like this (starting now from $\mathcal{R}'$ and μ') until $\mathcal{R}^\#$ and $\mu^\#$ are obtained such that $\mu^{can}_{\mathcal{R}^\#} \sqsubseteq \mu^\#$. In particular, if $\mu^{can}_{\mathcal{R}} \sqsubseteq \mu$, then $\mathcal{R}^\# = \mathcal{R}$ and $\mu^\# = \mu$.

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

$$\begin{array}{lll}
\mathtt{f_1(c_2(d,a))} \to \mathtt{a} & \quad & \mathtt{f_1(c(x,y))} \to \mathtt{f_1(c_3(x,y))} \\
\mathtt{f_1(c_1(d,x))} \to \mathtt{f_1(c_2(d,x))} & & \mathtt{f(x)} \to \mathtt{f_1(x)} \\
\mathtt{f_1(c_3(x,a))} \to \mathtt{f_1(c_1(x,a))} & & \mathtt{b} \to \mathtt{f(c(b,d))}
\end{array}$$

and $\mu^\#$ is given by $\mu^\#(\mathtt{f}) = \mu^\#(\mathtt{c}) = \varnothing$, $\mu^\#(\mathtt{f_1}) = \mu^\#(\mathtt{c_1}) = \{1\}$, $\mu^\#(\mathtt{c_2}) = \{1,2\}$, and $\mu^\#(\mathtt{c_3}) = \{2\}$. Notice that $\mu^{can}_{\mathcal{R}^\#} \sqsubseteq \mu^\#$.

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:

$$\begin{array}{lll}
\mathtt{f(c'(d,a))} \to \mathtt{a} & \quad & \mathtt{b} \to \mathtt{f(c(b,d))} \\
\mathtt{f(c(x,a))} \to \mathtt{f(c'(x,a))} & &
\end{array}$$

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

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 $t = \mathtt{f(c(b,d))}$, we have:

$$\begin{array}{l}
\mathit{label}_\mu(t) = \mathtt{f}^e(\mathtt{c}^\ell(\mathtt{b}^\ell,\mathtt{d}^\ell)) \overset{\mathsf{A}}{\to} \mathtt{f}^e(\mathtt{c}^e(\mathtt{b}^\ell,\mathtt{d}^\ell)) \overset{\mathsf{A}}{\to} \mathtt{f}^e(\mathtt{c}^e(\underline{\mathtt{b}^e},\mathtt{d}^\ell)) \\
\qquad \overset{\mathsf{R}}{\to}_\mu \mathtt{f}^e(\mathtt{c}^e(\mathtt{f}^e(\mathtt{c}^\ell(\mathtt{b}^\ell,\mathtt{d}^\ell)),\mathtt{d}^\ell)) \overset{\mathsf{A}}{\longrightarrow}{}^{+} \mathtt{f}^e(\mathtt{c}^e(\mathtt{f}^e(\mathtt{c}^e(\underline{\mathtt{b}^e},\mathtt{d}^\ell)),\mathtt{d}^\ell)) \overset{\mathsf{R}}{\to}_\mu \cdots
\end{array}$$

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

$$\underline{\mathtt{f}^e(\mathtt{c}^\ell(\mathtt{b}^\ell,\mathtt{d}^\ell))} \overset{\mathsf{R}}{\to}_{\mu^\#} \underline{\mathtt{f}^e_1(\mathtt{c}^e(\mathtt{b}^\ell,\mathtt{d}^\ell))} \overset{\mathsf{R}}{\to}_{\mu^\#} \mathtt{f}^e_1(\mathtt{c}^e_3(\mathtt{b}^\ell,\mathtt{d}^e))$$

The lazy subterm b^ℓ *cannot* be activated; $\mathtt{f}^e_1(\mathtt{c}^e_3(\mathtt{b}^\ell,\mathtt{d}^e))$ is a $\overset{\mathsf{R}}{\to}_{\mu^\#}$-normal form. Moreover, since $\mathtt{f}^e_1(\mathtt{c}^e_3(\mathtt{b}^\ell,\mathtt{d}^e)) = \mathit{label}_\mu(\mathtt{f_1(c_3(b,d))})$ and $\mu^\# \in CM_{\mathcal{R}^\#}$, by Proposition 4.11 is a $\overset{\mathsf{A}}{\to}$-normal form, hence a $\overset{\mathsf{LR}}{\to}_{\mu^\#}$-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 MU-TERM 1.0); termination of the transformed TRS can be automatically proved using C*i*ME 2.0 system (available at `http://cime.lri.fr`) if the 'dependency pairs criterion' (see [AG00]) has been previously activated.

$l \to r \in R$ and $p \in \mathcal{P}os(l)$, we let

$$\mathcal{I}(l,p) = \{i \in \{1,\ldots,ar(root(l|_p))\} - \mu(root(l|_p)) \mid p.i \in \mathcal{P}os_{\mathcal{F}}(l)\}$$

Assume that $\mathcal{I}(l,p) = \{i_1,\ldots,i_n\}$ for some $n > 0$ (i.e., $\mathcal{I}(l,p) \neq \varnothing$) and let $f = root(l|_p)$. Then, $\mathcal{R}^\diamond = (\mathcal{F}^\diamond, R^\diamond)$ and $\mu^\diamond \in M_{\mathcal{F}^\diamond}$ are as follows: $\mathcal{F}^\diamond = \mathcal{F} \cup \{f_j \mid 1 \leq j \leq n\}$, where each f_j is a new symbol of arity $ar(f_j) = ar(f)$. We let $\mu^\diamond(f_j) = \mu(f) \cup \{i_j\}$ for $1 \leq j \leq n$, and $\mu^\diamond(g) = \mu(g)$ for all $g \in \mathcal{F}$. On the other hand,

$$R^\diamond = R - \{l \to r\} \cup \{l'_j \to r, l[x]_{p.i_j} \to l'_j[x]_{p.i_j} \mid 1 \leq j \leq n\}$$

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 $\mathcal{R}^\diamond$ 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 *lhs* `f(c(d,a))` $\to$ `a` is considered, we now obtain:

$$\begin{array}{lll@{\qquad}lll}
\mathtt{f(c'(d,a))} & \to & \mathtt{a} & \mathtt{f(c(d,x))} & \to & \mathtt{f(c''(d,x))}\\
\mathtt{f(c''(d,a))} & \to & \mathtt{a} & \mathtt{b} & \to & \mathtt{f(c(b,d))}\\
\mathtt{f(c(x,a))} & \to & \mathtt{f(c'(x,a))} & & &
\end{array}$$

and $\mu^\diamond(\mathtt{f}) = \mu^\diamond(\mathtt{c}) = \varnothing$, $\mu^\diamond(\mathtt{c'}) = \{1\}$, $\mu^\diamond(\mathtt{c''}) = \{2\}$.

Again, the transformation proceeds like this (now starting from $\mathcal{R}^\diamond$ and $\mu^\diamond$) until $\mathcal{R}^\natural = (\mathcal{F}^\natural, R^\natural)$ and $\mu^\natural$ are obtained such that $\mu^{can}_{\mathcal{R}^\natural} \sqsubseteq \mu^\natural$. If $\mu \in CM_{\mathcal{R}}$, then $\mathcal{R}^\natural = \mathcal{R}$ and $\mu^\natural = \mu$.

Example 6.6 Consider $\mathcal{R}$ to be the same as in Example 4.8 and $\mu = \mu_\perp$. Then, $\mathcal{R}^\natural$ is

$$\begin{array}{lll@{\qquad}lll}
\mathtt{f_1(c'_2(d,a))} & \to & \mathtt{a} & \mathtt{f_1(c(x,y))} & \to & \mathtt{f_1(c_3(x,y))}\\
\mathtt{f_1(c_2(x,a))} & \to & \mathtt{f_1(c'_2(x,a))} & \mathtt{f_1(c_4(d,y))} & \to & \mathtt{f_1(c_2(d,y))}\\
\mathtt{f_1(c'_1(d,a))} & \to & \mathtt{a} & \mathtt{f_1(c(x,y))} & \to & \mathtt{f_1(c_4(x,y))}\\
\mathtt{f_1(c_1(d,x))} & \to & \mathtt{f_1(c'_1(d,x))} & \mathtt{f(x)} & \to & \mathtt{f_1(x)}\\
\mathtt{f_1(c_3(x,a))} & \to & \mathtt{f_1(c_1(x,a))} & \mathtt{b} & \to & \mathtt{f(c(b,d))}
\end{array}$$

and $\mu^\natural$ is given by $\mu^\natural(\mathtt{f}) = \mu^\natural(\mathtt{c}) = \varnothing$, $\mu^\natural(\mathtt{f_1}) = \mu^\natural(\mathtt{c_1}) = \mu^\natural(\mathtt{c_4}) = \{1\}$, $\mu^\natural(\mathtt{c'_1}) = \mu^\natural(\mathtt{c'_2}) = \{1,2\}$, and $\mu^\natural(\mathtt{c_2}) = \mu^\natural(\mathtt{c_3}) = \{2\}$. Notice that $\mu^{can}_{\mathcal{R}^\natural} \sqsubseteq \mu^\natural$.

Now, we are able to appropriately simulate every $\overset{\mathsf{LR}}{\to}_\mu$-reduction sequence in $\mathcal{R}$ as a $\overset{\mathsf{LR}}{\to}_{\mu^\natural}$-reduction sequence in $\mathcal{R}^\natural$.

Example 6.7 Consider the term t of Example 6.4. Now we have the following (infinite) $\overset{\mathsf{LR}}{\to}_{\mu^\natural}$-reduction sequence in $\mathcal{R}^\natural$:

$$\begin{array}{l}
label_\mu(t) = \underline{\mathtt{f}^e(\mathtt{c}^\ell(\mathtt{b}^\ell,\mathtt{d}^\ell))} \overset{\mathsf{R}}{\to}_{\mu^\natural} \underline{\mathtt{f}^e_1(\mathtt{c}^e(\mathtt{b}^\ell,\mathtt{d}^\ell))} \overset{\mathsf{R}}{\to}_{\mu^\natural} \mathtt{f}^e_1(\mathtt{c}^e_4(\underline{\mathtt{b}^e},\mathtt{d}^\ell))\\
\quad \overset{\mathsf{R}}{\to}_{\mu^\natural} \mathtt{f}^e_1(\mathtt{c}^e_4(\mathtt{f}^e(\mathtt{c}^\ell(\mathtt{b}^\ell,\mathtt{d}^\ell)),\mathtt{d}^\ell)) \overset{\mathsf{R}}{\longrightarrow}^+_{\mu^\natural} \mathtt{f}^e_1(\mathtt{c}^e_4(\mathtt{f}^e_1(\mathtt{c}^e_4(\underline{\mathtt{b}^e},\mathtt{d}^\ell)),\mathtt{d}^\ell)) \overset{\mathsf{R}}{\to}_{\mu^\natural} \cdots
\end{array}$$

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 $\mathcal{R}'$ implies $LR(\mu)$-termination of $\mathcal{R}$. We say that Θ is *complete* if $LR(\mu)$-termination of $\mathcal{R}$ implies $LR(\mu')$-termination of $\mathcal{R}'$. According to our discussion (and since $\mu^\natural$-termination and $LR(^\natural)$-termination coincide, see Corollary 5.5), we have the following.

Theorem 6.8 (Correctness) *Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear TRS and $\mu \in M_\mathcal{F}$. If $\mathcal{R}^\natural$ is $\mu^\natural$-terminating, then $\mathcal{R}$ is* LR(μ)*-terminating.*

Example 6.9 Consider $\mathcal{R}$ and μ as in Example 3.1. Then, $\mathcal{R}^\natural$ is:

```
2nd(x:'(y:z)) -> y
2nd(x:y)      -> 2nd(x:'y)
from(x)       -> x:from(s(x))
```

and $\mu^\natural$ is given by $\mu^\natural$(`2nd`) $= \mu^\natural$(`:`) $= \mu^\natural$(`from`) $= \{1\}$ and $\mu^\natural$(`:'`) $= \{1,2\}$. In fact, in this case $\mathcal{R}^\natural$ and $\mathcal{R}^\#$ coincide (see Example 6.1 of [Ngu01]). However, using Theorem 6.8, we can prove $LR(\mu)$-termination of $\mathcal{R}$, which was an open problem in [Ngu01]: $\mu^\natural$-termination of $\mathcal{R}^\natural$ is proved by using Zantema's transformation [Zan97]: the TRS

```
2nd(x:'(y:z))       -> y
2nd(x:y)            -> 2nd(x:'activate(y))
from(x)             -> x:from'(s(x))
activate(from'(x))  -> from(x)
from(x)             -> from'(x)
activate(x)         -> x
```

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

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

Conjecture 6.10 (Completeness) *Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear TRS and $\mu \in M_\mathcal{F}$. If $\mathcal{R}$ is* LR(μ)*-terminating, then $\mathcal{R}^\natural$ is $\mu^\natural$-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 μ that are less restrictive than the canonical replacement map $\mu^{can}_\mathcal{R}$, then *CSR* and *LR* coincide for left-linear TRSs $\mathcal{R}$. In this case,

[9] Use the C*i*ME 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 problem 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 corresponding TRS $\mathcal{R}$ is μ-terminating, where $\mathcal{R}$ and μ are the same as in Example 3.1. The proof of μ-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 satisfactory 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

[AG00] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs. *Theoretical Computer Science*, 236:133-178, 2000.

[Ant92] S. Antoy. Definitional Trees. In H. Kirchner and G. Levi, editors, *Proc. of 3rd International Conference on Algebraic and Logic Programming, ALP'92*, LNCS 632:143-157, Springer-Verlag, Berlin, 1992.

[BLR02] C. Borralleras, S. Lucas, and A. Rubio. Recursive Path Orderings can be Context-Sensitive. In A. Voronkov, editor *Proc. of 18th International Conference on Automated Deduction, CADE'02*, Springer LNAI volume 2392, to appear, 2002.

[CELM96] M. Clavel, S. Eker, P. Lincoln, and J. Meseguer. Principles of Maude. In J. Meseguer, editor, *Proc. 1st International Workshop on Rewriting Logic and its Applications*, Electronic Notes in Theoretical Computer Science, volume 4, 25 pages, Elsevier Sciences, 1996.

[Eke98] S. Eker. Term Rewriting with Operator Evaluation Strategies. In C. Kirchner and H. Kirchner, editors, *Proc. of 2nd International Workshop on Rewriting Logic and its Applications, WRLA'98*, Electronic Notes in Computer Science, 15(1998):1-20, 1998.

[FGJM85] K. Futatsugi, J. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Conference Record of the 12th Annual ACM Symposium on Principles of Programming Languages, POPL'85*, pages 52-66, ACM Press, 1985.

[FKW00] W. Fokkink, J. Kamperman, and P. Walters. Lazy Rewriting on Eager Machinery. *ACM Transactions on Programming Languages and Systems*, 22(1):45-86, 2000.

[FN97] K. Futatsugi and A. Nakagawa. An Overview of CAFE Specification Environment – An algebraic approach for creating, verifying, and maintaining formal specification over networks –. In *Proc. of 1st International Conference on Formal Engineering Methods*, 1997.

[FR99] M.C.F. Ferreira and A.L. Ribeiro. Context-Sensitive AC-Rewriting. In P. Narendran and M. Rusinowitch, editors, *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA'99*, LNCS 1631:286-300, Springer-Verlag, Berlin, 1999.

[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 University Press, 1976.

[GL02] B. Gramlich and S. Lucas. Modular termination of context-sensitive rewriting. In C. Kirchner, editor, *Proc. of 4th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP'02*, ACM Press, New York, *to appear*, 2002.

[GM99] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. In P. Narendran and M. Rusinowitch, editors, *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA'99*, LNCS 1631:271-285, Springer-Verlag, Berlin, 1999.

[GM02] J. Giesl and A. Middeldorp. Transformation Techniques for Context-Sensitive Rewrite Systems. Technical Report AIB-2002-02, Department of Computer Science, RWTH Aachen, 2002.

[GWMFJ00] J.A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducing OBJ. In J. Goguen and G. Malcolm, editors, *Software Engineering with OBJ: algebraic specification in action*, Kluwer, 2000.

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

[HLM98] M. Hanus, S. Lucas, and A. Middeldorp. Strongly sequential and inductively sequential term rewriting systems. *Information Processing Letters*, 67(1):1-8, 1998.

[Luc96] S. Lucas. Termination of context-sensitive rewriting by rewriting. In F. Meyer auf der Heide and B. Monien, editors, *Proc. of 23rd. International Colloquium on Automata, Languages and Programming, ICALP'96*, LNCS 1099:122-133, Springer-Verlag, Berlin, 1996.

[Luc98] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1):1-61, January 1998.

[Luc01a] S. Lucas. Termination of on-demand rewriting and termination of OBJ programs. In *Proc. of 3rd International Conference on Principles and Practice of Declarative Programming, PPDP'01*, pages 82-93, ACM Press, 2001.

[Luc01b] S. Lucas. Termination of Rewriting With Strategy Annotations. In R. Nieuwenhuis and A. Voronkov, editors, *Proc. of 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'01*, LNAI 2250:669-684, Springer-Verlag, Berlin, 2001.

[Luc02a] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, to appear.

[Luc02b] S. Lucas. Context-sensitive rewriting techniques for programs with strategy annotations (tutorial). In U. Montanari, editor, *Proc. of 4th International Workshop on Rewriting Logic and its Applications, WRLA'02, Electronic Notes in Theoretical Computer Science*, volume 71, to appear.

[Luc02c] S. Lucas. Termination of (Canonical) Context-Sensitive Rewriting. In S. Tison, editor, *Proc. of 13th International Conference on Rewriting Techniques and Applications, RTA'02*, LNCS volume 2378, *to appear*, 2002.

[Ngu01] Q.-H. Nguyen. Compact Normalisation Trace via Lazy Rewriting. In B. Gramlich and S. Lucas, editors, *Proc. of International Workshop on Reduction Strategies in Rewriting and Programming, WRS'01, Electronic Notes in Theoretical Computer Science*, volume 57, 2001.

[NO01] M. Nakamura and K. Ogata. The evaluation strategy for head normal form with and without on-demand flags. In K. Futatsugi, editor, *Proc. of 3rd International Workshop on Rewriting Logic and its Applications, WRLA'00, Electronic Notes in Theoretical Computer Science*, volume 36, 17 pages, 2001.

[OF00] K. Ogata and K. Futatsugi. Operational Semantics of Rewriting with the On-demand Evaluation Strategy. In *Proc of 2000 International Symposium on Applied Computing, SAC'00*, pages 756-763, ACM Press, 2000.

[SX98] J. Steinbach and H. Xi. Freezing – Termination Proofs for Classical, Context-Sensitive and Innermost Rewriting. Institut für Informatik, T.U. München, January 1998.

[Zan97] H. Zantema. Termination of Context-Sensitive Rewriting. In H. Comon, editor, *Proc. of 8th International Conference on Rewriting Techniques and Applications, RTA'97*, LNCS 1232:172-186, Springer-Verlag, Berlin, 1997.