Termination of on-demand rewriting and termination of OBJ programs*

Salvador Lucas
DSIC, Universidad Politécnica de Valencia
Camino de Vera s/n
E-46022 Valencia, Spain
e.mail: slucas@dsic.upv.es

ABSTRACT

Declarative languages such as OBJ*, CafeOBJ, and Maude use syntactic annotations to introduce replacement restrictions aimed at improving termination or efficiency of computations. Unfortunately, there is a lack of formal techniques for proving such benefits. We show that context-sensitive rewriting and *on-demand rewriting* provide a suitable framework to address this problem. We provide methods to analyze termination of on-demand rewriting and apply them to analyze termination of OBJ*, CafeOBJ, and Maude programs.

Keywords

Program analysis, replacement restrictions, term rewriting, termination

1. INTRODUCTION

Eager rewriting-based programming languages such as Lisp, OBJ*, CafeOBJ, ELAN, or Maude use innermost rewriting to evaluate expressions. A frequent problem here is nontermination. *Syntactic annotations* (i.e., associated to the arguments of symbols) have been used in OBJ2 [8], OBJ3 [11], CafeOBJ [9], or Maude [3] as *replacement restrictions* to (hopefully) avoid nontermination. They are often specified as sequences of integers in parentheses called *local strategies* (see Figure 1).

Local strategies guide the *evaluation strategy* of OBJ programs[1]: when considering a function call $f(t_1, \ldots, t_k)$, only the arguments whose indices are present as *positive* integers in the local strategy for f are evaluated (following the specified ordering). If 0 is found, then the evaluation of f is attempted. *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 [5, 11, 18]. For instance, the local strategy for '`::`' in Example 1 specifies that its second argument is evaluated 'on-demand'. A local strategy such as `(1)` disables the evaluation of `2nd(0::from(s(0)))` into `s(0)`, since reducing the second argument of '`::`' is not allowed. On the other hand, with `(1 2)`, the following infinite reduction sequence is possible:

$$\begin{aligned}\texttt{2nd}(\underline{\texttt{from(0)}}) &\to \texttt{2nd(0::}\underline{\texttt{from(s(0))}}\texttt{)}\\ &\to \texttt{2nd(0::s(0)::}\underline{\texttt{from(s(s(0)))}}\texttt{)}\\ &\to \cdots\end{aligned}$$

The presence of replacement restrictions is often invoked to justify that OBJ programs are able to achieve a *lazy* behavior that avoids nontermination, even though the underlying execution mechanism is innermost rewriting ([11], Section 2.4.4; [18]). For instance, with our CafeOBJ program, the evaluation of `2nd(from(0))` should produce `s(0)` without risk of nontermination. Unfortunately, there is no formal analysis available about how a particular choice of replacement restrictions (like those enumerated for '`::`') modifies termination of OBJ (like) programs.

Term rewriting systems (TRSs) provide a suitable computational model for more sophisticated programming languages [2]. Syntactic replacement restrictions can be associated to symbols f of a signature Σ by means of a *replacement map* $\mu : \Sigma \to \mathcal{P}(\mathbb{N})$ [14] that discriminates the argument positions $\mu(f) \subseteq \{1, \ldots, ar(f)\}$ on which we *can*

The following CafeOBJ program (borrowed from [17]):

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

associates local strategies to functions `s`, `::`, `2nd`, and `from`.

Figure 1: Example of the use of local strategies.

*This work has been partially supported by Spanish CICYT and Conselleria de Cultura y Educació de la Generalitat Valenciana.

[1]As in [11], by OBJ we mean OBJ2, OBJ3, CafeOBJ, or Maude.

perform rewriting steps. Negative integers in local strategies can be managed using an *additional* replacement map μ_D containing their absolute values.

EXAMPLE 1. *The program in Example 1 is associated to (1) signature* {0, s, :, 2nd, from}, *where* 0 *is a constant,* s, 2nd, *and* from *are unary symbols, and* ':' *is a binary (infix) operator; (2) rewrite rules*

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

and (3) μ *and* μ_D *given by:*

$\mu(\texttt{:}) = \mu(\texttt{s}) = \mu(\texttt{2nd}) = \{1\}$,
$\mu(\texttt{from}) = \mu(\texttt{0}) = \varnothing$,
$\mu_D(\texttt{0}) = \mu_D(\texttt{s}) = \mu_D(\texttt{from}) = \mu_D(\texttt{2nd}) = \varnothing$, *and*
$\mu_D(\texttt{:}) = \{2\}$.

Context-sensitive rewriting (*CSR* [14]) provides an operational model for using replacement restrictions specified by a single replacement map μ. In this paper, we prove that *CSR* suffices for modeling computations of OBJ programs specifying local strategies without negative indices. We demonstrate that termination of such OBJ programs can be analyzed as termination of *CSR*. Termination of *CSR* has been studied in [10, 13, 19, 20, 6]. Concerning arbitrary local strategies, we introduce *on-demand rewriting* (*ODR*, using two replacement maps μ and μ_D) as an extension of *CSR*. In [10], Giesl and Middeldorp provide transformations of TRSs that are proved correct (and complete) for proving termination of *CSR*. We extend these transformations to provide correct (and complete) transformations of TRSs that can be used to prove termination of *ODR*. We apply our methods to prove termination of OBJ programs. For instance, we are able to formally prove that the program in Example 1 is terminating.

The paper is organized as follows. Section 2 gives some preliminary definitions. Section 3 introduces *ODR*. Section 4 develops two techniques for proving termination of *ODR*. Section 5 discusses their application to OBJ programs. Section 6 concludes.

2. PRELIMINARIES

Given a set A, $\mathcal{P}(A)$ denotes the set of all subsets of A. Let $R \subseteq A \times A$ be a binary relation on a set A. We denote the reflexive closure of R by $R^=$, its transitive closure by R^+, and its reflexive and transitive closure by R^* [2]. An element $a \in A$ is an R-normal form, if there exists no b such that $a\ R\ b$. We say that b is an R-normal form of a, if b is an R-normal form and $a\ R^* b$. We say that R is *terminating* iff there is no infinite sequence $a_1\ R\ a_2\ R\ a_3 \cdots$. Throughout the paper, $\mathcal{X}$ denotes a countable set of variables and Σ denotes a signature, i.e., a set of function symbols $\{\texttt{f}, \texttt{g}, \ldots\}$, each having a fixed arity given by a function $ar : \Sigma \to \mathbb{N}$. We denote the set of terms built from Σ and $\mathcal{X}$ by $\mathcal{T}(\Sigma, \mathcal{X})$. A term is said to be linear if it has no multiple occurrences of a single variable. Terms are viewed as labelled trees in the usual way. Positions $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms of t. Given positions p, q, we denote its concatenation as $p.q$. Positions are ordered by the standard prefix ordering $\leq$. Given a set of positions P, $minimal_{\leq}(P)$ is the set of minimal positions of P w.r.t. $\leq$. If $p \leq q$, we write q/p to denote the position p' that satisfies $p.p' = q$. If p is a position, and Q is a set of positions, $p.Q$ is the set $\{p.q \mid q \in Q\}$. We denote the empty chain by Λ. By $\mathcal{P}os(t)$ we denote the set of positions of a term t. Positions of non-variable symbols in t are denoted as $\mathcal{P}os_\Sigma(t)$, 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 $root(t)$. The chain of symbols lying on positions above/on $p \in \mathcal{P}os(t)$ is $prefix_t(\Lambda) = root(t)$, $prefix_t(i.p) = root(t).prefix_{t|_i}(p)$. The strict prefix *sprefix* is $sprefix_t(\Lambda) = \Lambda$, $sprefix_t(p.i) = prefix_t(p)$.

A rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\Sigma, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side (*lhs*) of the rule is l and r is the right-hand side (*rhs*). A TRS is a pair $\mathcal{R} = (\Sigma, R)$ 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. Given $\mathcal{R} = (\Sigma, R)$, we take Σ as the disjoint union $\Sigma = \mathcal{C} \uplus \mathcal{F}$ of symbols $c \in \mathcal{C}$, called *constructors* and symbols $f \in \mathcal{F}$, called *defined functions*, where $\mathcal{F} = \{root(l) \mid l \to r \in R\}$ and $\mathcal{C} = \Sigma - \mathcal{F}$. A TRS $\mathcal{R} = (\mathcal{C} \uplus \mathcal{F}, R)$ is a constructor system (CS) if for all $f(l_1, \ldots, l_k) \to r \in R$, $l_i \in \mathcal{T}(\mathcal{C}, \mathcal{X})$, for $1 \leq i \leq k$. A term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ rewrites to s (at position p), written $t \xrightarrow{p}_\mathcal{R} s$ (or just $t \to s$), if $t|_p = \sigma(l)$ and $s = t[\sigma(r)]_p$, for some rule $l \to r \in R$, $p \in \mathcal{P}os(t)$ and substitution σ.

3. ON-DEMAND REWRITING

Given a signature Σ, a mapping $\mu : \Sigma \to \mathcal{P}(\mathbb{N})$ is a *replacement map* (or Σ-map) if for all $f \in \Sigma$, $\mu(f) \subseteq \{1, \ldots, ar(f)\}$ [14]. The inclusion ordering $\subseteq$ on $\mathcal{P}(\mathbb{N})$ extends to an ordering $\sqsubseteq$ on M_Σ, the set of all Σ-maps: $\mu \sqsubseteq \mu'$ if for all $f \in \Sigma$, $\mu(f) \subseteq \mu'(f)$. The lattice $(\mathcal{P}(\mathbb{N}), \subseteq, \varnothing, \mathbb{N}, \cup)$ induces a lattice $(M_\Sigma, \sqsubseteq, \mu_\bot, \mu_\top, \sqcup)$: The minimum (maximum) element is $\mu_\bot$ ($\mu_\top$), given by $\mu_\bot(f) = \varnothing$ ($\mu_\top(f) = \{1, \ldots, ar(f)\}$) for all $f \in \Sigma$. The *lub* $\sqcup$ is given by $(\mu \sqcup \mu')(f) = \mu(f) \cup \mu'(f)$ for all $f \in \Sigma$.

A replacement map μ is aimed at specifying the *argument* positions which can be reduced for each symbol in Σ. Accordingly, the set of μ-*replacing* positions $\mathcal{P}os^\mu(t)$ of a term t is: $\mathcal{P}os^\mu(t) = \{\Lambda\}$, if $t \in V$ and $\mathcal{P}os^\mu(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(f)} i.\mathcal{P}os^\mu(t|_i)$, if $root(t) = f$. The following results are used later.

PROPOSITION 1. *[14] Let* $t \in \mathcal{T}(\Sigma, V)$, $p = q.q' \in \mathcal{P}os(t)$. *Then,* $p \in \mathcal{P}os^\mu(t)$ *iff* $q \in \mathcal{P}os^\mu(t) \wedge q' \in \mathcal{P}os^\mu(t|_q)$

PROPOSITION 2. *[14] Let* $t, s \in \mathcal{T}(\Sigma, \mathcal{X})$. *If* $p \in \mathcal{P}os(t) \cap \mathcal{P}os(s)$ *and* $sprefix_t(p) = sprefix_s(p)$, *then* $p \in \mathcal{P}os^\mu(t) \Leftrightarrow p \in \mathcal{P}os^\mu(s)$.

The *non-replacing* positions are $\overline{\mathcal{P}os^\mu(t)} = \mathcal{P}os(t) - \mathcal{P}os^\mu(t)$; we also use $\mathcal{L}azy_\mu(t) = minimal_{\leq}(\overline{\mathcal{P}os^\mu(t)})$ which covers the non-replacing positions of t, i.e., for all $p \in \overline{\mathcal{P}os^\mu(t)}$, there exists $q \in \mathcal{L}azy_\mu(t)$ such that $q \leq p$.

In *context-sensitive rewriting* (*CSR* [14]), we (only) rewrite subterms at *replacing* positions: t μ-rewrites to s, written $t \hookrightarrow_{\mathcal{R}(\mu)} s$ (or simply $t \hookrightarrow_\mu s$ or $t \hookrightarrow s$), if $t \xrightarrow{p}_\mathcal{R} s$ and $p \in \mathcal{P}os^\mu(t)$.

EXAMPLE 2. *Consider* $\mathcal{R}$ *and* μ *as in Example 1. Then we have:*

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

where, since $\mu(\texttt{:}) = \{1\}$, 2nd(0:from(s(0))) *cannot be further* μ*-rewritten.*

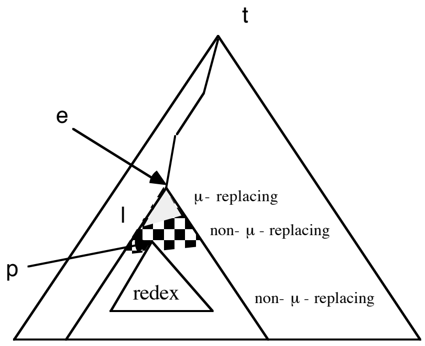

Figure 2: Scheme of a *possible* on-demand step

$\hookrightarrow_\mu$-normal forms are called μ-normal forms. A TRS $\mathcal{R}$ is μ-terminating if $\hookrightarrow_\mu$ is terminating (see [13]).

Given a pair $\langle\mu,\mu_D\rangle$ of replacement maps μ and μ_D, we define *on-demand rewriting* (*ODR*) as an extension of *CSR* (under μ), where we also permit reductions *on-demand* according to μ_D. Given $f \in \Sigma$, indices $j \in \mu_D(f)$ are aimed at enabling reductions on a subterm t_j of a function call $f(t_1,\dots,t_j,\dots,t_k)$ if they can eventually lead to matching a pattern of a rule defining f (i.e., $l \to r \in R$ such that $root(t) = f$). After its formal definition, we explain the notion and give an example.

Definition 1 (On-demand rewriting). *Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu, \mu_D \in M_\Sigma$. Then, $t \overset{p}{\hookrightarrow}_{\langle\mu,\mu_D\rangle} s$ (or simply $t \hookrightarrow_{\langle\mu,\mu_D\rangle} s$), if $t \overset{p}{\to} s$ and either*

1. *$p \in \mathcal{P}os^\mu(t)$, or*
2. *$p \in \mathcal{P}os^{\mu\sqcup\mu_D}(t) - \mathcal{P}os^\mu(t)$ and there exist $e \in \mathcal{P}os^\mu(t)$, $p_1,\dots,p_n \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t)$, $r_1,\dots,r_n,t' \in \mathcal{T}(\Sigma,\mathcal{X})$, $l \to r \in R$, and substitution σ such that*
   - *(1) $e \le p$, $t' = t[r_1]_{p_1}\cdots[r_n]_{p_n}$, $t'|_e = \sigma(l)$ and*
   - *(2) for all $q \in \mathcal{P}os(l)$ s.t. $sprefix_{t|_e}(q) = sprefix_l(q)$, whenever $e.q \le p$, we have that $l|_q \notin \mathcal{X}$.*

Here, $\mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t) = \mathcal{L}azy_\mu(t) \cap \mathcal{P}os^{\mu\sqcup\mu_D}(t)$.

Therefore, given a term t, a rewriting step $t \overset{p}{\to} s$ is *on-demand* (w.r.t. μ and μ_D) if either

- (i) $t \overset{p}{\hookrightarrow}_\mu s$, or
- (ii) $t \overset{p}{\hookrightarrow}_{\mu\sqcup\mu_D} s$ and reducing $t|_p$ may contribute to a future μ-rewriting step at μ-replacing position e, using some rule $l \to r$.

Such a contribution is approximated by checking whether the replacement of some non-μ-replacing maximal subterms of t would eventually make the matching possible (condition 2(1) of Definition 1). On-demand indices in μ_D determine the positions (in $\mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t)$) of the subterms of t that can be refined. Note that p is always covered by some position $p_i \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t)$, i.e., $p_i \le p$ (see Figure 2). On the other hand, the position p on which the rewriting step is performed is constrained to having no variable position of l covering p (condition 2(2) of Definition 1); otherwise, the reduction at $t|_p$ would not improve the matching (see Figure 3).

Example 3. *Consider $\mathcal{R}$, μ, and μ_D as in Example 1. Now we have:*

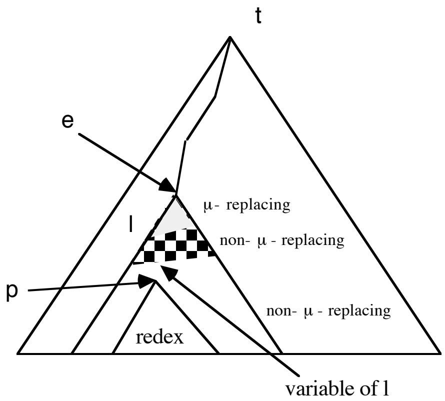

Figure 3: A *forbidden* on-demand step

$$\begin{array}{ll} \texttt{2nd}(\underline{\texttt{from(0)}}) & \hookrightarrow_{\langle\mu,\mu_D\rangle} \texttt{2nd(0:}\underline{\texttt{from(s(0))}}\texttt{)} \\ & \hookrightarrow_{\langle\mu,\mu_D\rangle} \underline{\texttt{2nd(0:s(0):from(s(s(0))))}} \\ & \hookrightarrow_{\langle\mu,\mu_D\rangle} \texttt{s(0)} \end{array}$$

but

$$\begin{array}{l} \texttt{2nd(0:s(0):}\underline{\texttt{from(s(s(0)))}}\texttt{)} \\ \quad \not\hookrightarrow_{\langle\mu,\mu_D\rangle} \texttt{2nd(0:s(0):s(s(0)):from(s(s(0))))} \end{array}$$

since $1.2.2 \in \mathcal{P}os(l)$ and $sprefix_t(1.2.2) = sprefix_l(1.2.2)$, but $l|_{1.2.2} \in \mathcal{X}$ (where $l = \texttt{2nd(x:y:z)}$).

Note that s(0) could also be obtained from 2nd(from(0)) by using *CSR* if we let $\mu(\texttt{:}) = \{1,2\}$. However, as shown in the introduction, we have a risk of nontermination which is avoided by $\hookrightarrow_{\langle\mu,\mu_D\rangle}$, as shown in Example 3.

$\hookrightarrow_{\langle\mu,\mu_D\rangle}$-normal forms are called $\langle\mu,\mu_D\rangle$-normal forms. A TRS $\mathcal{R}$ is $\langle\mu,\mu_D\rangle$-terminating if $\hookrightarrow_{\langle\mu,\mu_D\rangle}$ is terminating. Thus, *ODR* seems to provide better opportunities for improving termination than *CSR*. *ODR* is also able to achieve useful computations (see Section 5.2). In general, *CSR* and *ODR* are related as follows.

Theorem 1. *Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu \in M_\Sigma$. Then, $\hookrightarrow_\mu = \hookrightarrow_{\langle\mu,\mu_\perp\rangle}$.*

Proof. It is a consequence of $\mathcal{P}os^{\mu\sqcup\mu_\perp}(t) - \mathcal{P}os^\mu(t) = \mathcal{P}os^\mu(t) - \mathcal{P}os^\mu(t) = \varnothing$. □

Theorem 2. *Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu, \mu_D \in M_\Sigma$. Then, $\hookrightarrow_\mu \subseteq \hookrightarrow_{\langle\mu,\mu_D\rangle} \subseteq \hookrightarrow_{\mu\sqcup\mu_D}$.*

Proof. Immediate, by definition of *ODR*. □

The definition of *ODR* follows *lazy rewriting* of [12]. The more recent definition of [7] is aimed at dealing with graphs, rather than terms. Moreover, it uses decorations of nodes that incorporate the replacing/non-replacing character of each node inside the syntactic structure of graphs, and reduces such decorated graphs mixing rewriting and *relabelling* steps. Since our purpose is to give a treatment of termination of *ODR* by using the standard techniques for proving termination of term rewriting, we prefer to follow the previous, more standard, approach. Also, in both cases, lazy rewriting is defined by using a single replacement map μ and is not well-suited for our purposes. A more detailed comparison is left as a subject for future work.

4. TERMINATION OF ON-DEMAND REWRITING

Recently, Giesl and Middeldorp have introduced a transformation which can be used to prove termination of *CSR* [10]. Basically, their idea is to explicitly *mark* the replacing positions of a term (by using a new symbol `active`), since these positions are the only ones where *CSR* may take place.

Definition 2. *[10] Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu \in M_\Sigma$. The TRS $\mathcal{R}^1_\mu = (\Sigma \cup \{\mathtt{active}, \mathtt{mark}\}, R^1_\mu)$ consists of the following rules in R^1_μ (for all $l \to r \in R$ and $f \in \Sigma$):*

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

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

We give an example of the transformation:

Example 4. *Consider $\mathcal{R}$ and μ as in Example 1. Then, $\mathcal{R}^1_\mu$ is:*

```
active(2nd(x:y:z)) → mark(y)
active(from(x))    → mark(x:from(s(x)))
mark(2nd(x))       → active(2nd(mark(x)))
mark(from(x))      → active(from(x))
mark(x:y)          → active(mark(x):y)
mark(s(x))         → active(s(mark(x)))
mark(0)            → active(0)
active(x)          → x
```

Remark 1. *$\mathcal{R}^1_\mu$ in Example 4 is proved to be terminating by using Arts and Giesl's dependency pairs technique, i.e., by checking that there is no infinite chain of dependency pairs associated to $\mathcal{R}^1_\mu$ [1]. It is not difficult to see that more conventional techniques such as Knuth-Bendix, polynomial, or recursive path orderings [4] do not apply to this example.*

Given replacement maps μ and μ_D, *ODR* permits reductions on positions that are not replacing (according to μ); this is not captured by the transformation of Definition 2. In the following, we *extend* Giesl and Middeldorp's transformation to deal with *ODR*. According to Definition 1, given a term t, we need to consider information about the non-replacing positions of t (more precisely, positions in $\mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t)$) in order to establish the allowed on-demand reduction steps. This is not appropriate for providing a transformation which is based on the shape of the *rules* of the TRS rather than on each concrete term. The following result allows us to overcome this problem.

Theorem 3. *Let $\mathcal{R} = (\Sigma, R)$ be a left-linear TRS and $\mu, \mu_D \in M_\Sigma$. If $t \overset{p}{\hookrightarrow}_{\langle\mu,\mu_D\rangle} s$ and $p \in \mathcal{P}os^{\mu\sqcup\mu_D}(t) - \mathcal{P}os^\mu(t)$, then there exist $e \in \mathcal{P}os^\mu(t)$, $l \to r \in R$, $p_1,\ldots,p_n \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l) - \mathcal{P}os_\mathcal{X}(l)$, $r_1,\ldots,r_n,t' \in \mathcal{T}(\Sigma,\mathcal{X})$, and σ such that $e \leq p$, $t' = t[r_1]_{e.p_1}\cdots[r_n]_{e.p_n}$, $t'|_e = \sigma(l)$ and for all $q \in \mathcal{P}os(l)$ such that $sprefix_{t|_e}(q) = sprefix_l(q)$, if $e.q \leq p$, then $l|_q \notin \mathcal{X}$.*

Proof. We only need to prove that whenever $p'_1,\ldots,p'_m \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t)$, $r'_1,\ldots,r'_m,t'' \in \mathcal{T}(\Sigma,\mathcal{X})$, $l \to r \in R$, and σ' are such that $t'' = t[r'_1]_{p'_1}\cdots[r'_n]_{p'_n}$ and $t''|_e = \sigma'(l)$, then there exist $p_1,\ldots,p_n \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l) - \mathcal{P}os_\mathcal{X}(l)$, $r_1,\ldots,r_n$, $t' \in \mathcal{T}(\Sigma,\mathcal{X})$, and σ such that $t' = t[r_1]_{e.p_1}\cdots[r_n]_{e.p_n}$. First, without losing generality, we can assume that positions $p'_1,\ldots,p'_m$ satisfy $e \leq p'_i$ for $1 \leq i \leq m$, and that, since $e \in \mathcal{P}os^\mu(t)$, by Proposition 1 $p'_i/e \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t|_e)$ for $1 \leq i \leq m$. Since $t''|_e = \sigma'(l)$, by Proposition 2 we have $\mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l) \subseteq \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t|_e)$. If $q \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t|_e) - \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l)$, then, since $t''|_e = \sigma'(l)$ there exists $q' \in \mathcal{P}os_\mathcal{X}(l)$ such that $q' < q$. Therefore, by left-linearity, subterm $t|_{e.q}$ does not disturb the matching of $t|_e$ against l. Therefore, we can just use positions in $\mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l)$ instead of $\mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t|_e)$. Moreover, the same reasoning shows that we de not need to consider positions in $\mathcal{P}os_\mathcal{X}(l)$. Therefore, the conclusion follows. ☐

Unfortunately, left-linearity is essential to ensure Theorem 3.

Example 5. *Consider the TRS*

```
f(x,x,b) → c(a)
a        → b
```

with $\mu(\mathtt{f}) = \{1,2\}$, $\mu(\mathtt{c}) = \varnothing$ and $\mu_D(\mathtt{f}) = \{3\}$, $\mu_D(\mathtt{c}) = \varnothing$. Let $t = \mathtt{f(c(a),c(b),a)}$. We have $\mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t) = \{1.1, 2.1, 3\}$. Therefore, since

$$t' = \mathtt{f(c(a),c(a),b)} = t[\mathtt{a}]_{2.1}[\mathtt{b}]_3$$

matches $\mathtt{f(x,x,b)}$, we have that

$$\mathtt{f(c(a),c(b),\underline{a})} \hookrightarrow_{\langle\mu,\mu_D\rangle} \mathtt{f(c(a),c(b),b)}$$

However, since $\mathcal{L}azy_{\langle\mu,\mu_D\rangle}(\mathtt{f(x,x,b)}) - \mathcal{P}os_\mathcal{X}(\mathtt{f(x,x,b)}) = \{3\}$, there is no s such that $t[s]_3$ matches $\mathtt{f(x,x,b)}$.

We use Theorem 3 to ensure the correctness of our transformations. Consequently, our methods are restricted to left-linear TRSs.

With *ODR*, an *attempt* to match a subterm $t|_e$ (where $e \in \mathcal{P}os^\mu(t)$) against (the *lhs* of) a rule $l \to r$ can 'activate' some position $p > e$ (with $p \in \mathcal{P}os^{\mu\sqcup\mu_D}(t) - \mathcal{P}os^\mu(t)$) for reductions (possibly involving a different rule) that we call (pure) 'on-demand' reductions. Since $e \in \mathcal{P}os^\mu(t)$, we add a rule

$$\mathtt{active}(x) \to \mathtt{demand}(x)$$

where `demand` signals a point under which an on-demand rewriting step may take place: for each rewrite rule $l \to r \in R$ such that $\mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l) - \mathcal{P}os_\mathcal{X}(l) \neq \varnothing$, we add a rule

$$\mathtt{demand}(\tau(l)) \to \mathtt{onDemand}(l[s(l,p_1)]_{p_1}\cdots[s(l,p_n)]_{p_n})$$

where

$$\{p_1,\ldots,p_n\} = \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l) - \mathcal{P}os_\mathcal{X}(l)$$

and

$$\tau(l) = l[X_1]_{p_1}\cdots[X_n]_{p_n},$$

with $X_1,\ldots,X_n$ being fresh variables. On the other hand,

$$s(l,p_i) = \mathtt{mark'}(X_i)$$

if $l|_{p_i} \in \mathcal{T}(\Sigma)$, and

$$s(l,p_i) = \mathtt{mark''}(l_\Omega|_{p_i}, X_i)$$

if $l|_{p_i} \notin \mathcal{T}(\Sigma)$, where Ω is a new constant symbol and l_Ω is l with all its variables replaced by Ω. Symbols `mark'` and `mark''` are intended to 'activate' pure on-demand reductions according to the definition of *ODR*. Similar to `mark`, symbol

For all $l \in L(\mathcal{R})$ such that $\mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l) - \mathcal{P}os_{\mathcal{X}}(l) \neq \varnothing$, $f \in \Sigma$, $t \in TL_{\langle\mu,\mu_D\rangle}(\mathcal{R})$, $g \in \Sigma$ such that $ar(g) > 0$, and $j \in \mu \sqcup \mu_D(f)$

$$\begin{array}{rcl}
\texttt{demand}(\tau(l)) & \to & \texttt{onDemand}(l[s(l,p_1)]_{p_1}\cdots[s(l,p_n)]_{p_n}) \\
\texttt{mark'}(f(x_1,\ldots,x_k)) & \to & \texttt{active}(f([x_1]'_f,\ldots,[x_k]'_f)) \\
\texttt{mark''}(t,f(x_1,\ldots,x_k)) & \to & \texttt{active}(f(u(t,f,1),\ldots,u(t,f,k))) \\
g(x_1,\ldots,\texttt{done}(x_j),\ldots,x_k) & \to & \texttt{done}(g(x_1,\ldots,x_j,\ldots,x_k)) \\
\texttt{onDemand}(\texttt{done}(x)) & \to & \texttt{mark}(x) \\
\texttt{active}(x) & \to & \texttt{demand}(x) \\
\texttt{mark}(x) & \to & \texttt{done}(x)
\end{array}$$

where

1. $\tau(l) = l[X_1]_{p_1}\cdots[X_n]_{p_n}$, where $\{p_1,\ldots,p_n\} = \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l) - \mathcal{P}os_{\mathcal{X}}(l)$ and $X_1,\ldots,X_n$ are fresh variables.
2. For each $1 \leq i \leq n$, $s(l,p_i) = \begin{cases} \texttt{mark'}(X_i) & \text{if } l|_{p_i} \in \mathcal{T}(\Sigma) \\ \texttt{mark''}(l_\Omega|_{p_i},X_i) & \text{otherwise} \end{cases}$.
3. $[x_j]'_f = \texttt{mark'}(x_j)$ if $j \in \mu \sqcup \mu_D(f)$ and $[x_j]'_f = x_j$ if $j \notin \mu \sqcup \mu_D(f)$.
4. $u(t,f,j) = \texttt{mark'}(x_j)$, if $j \in \mu \sqcup \mu_D(f)$ and either $root(t) \neq f$ or $t|_j \in \mathcal{T}(\Sigma)$; $u(t,f,j) = \texttt{mark''}(t|_j,x_j)$, if $j \in \mu \sqcup \mu_D(f)$, $root(t) = f$ and $t|_j$ contains Ω but $t|_j \neq \Omega$; $u(t,f,j) = x_j$, otherwise.

Figure 4: Rules $S^1_{\langle\mu,\mu_D\rangle}$ of transformation $\mathcal{R}^1_{\langle\mu,\mu_D\rangle}$ for proving termination of *ODR* (Definition 3)

`mark'` activates subterms according to *both* μ and μ_D: for each $f \in \Sigma$, we add a rule

$$\texttt{mark'}(f(x_1,\ldots,x_k)) \to \texttt{active}(f([x_1]'_f,\ldots,[x_k]'_f))$$

where $[x_j]'_f = \texttt{mark'}(x_j)$ if $j \in \mu \sqcup \mu_D(f)$; and $[x_j]'_f = x_j$ if $j \notin \mu \sqcup \mu_D(f)$. With `mark''` we use its first argument (which represents a part of a lazy subterm of the *lhs* of the rule $l \to r$ that originates the on-demand reduction) to decide whether it is allowed to get inside the term in the second argument. More precisely, terms t are taken from the following set

$$TL_{\langle\mu,\mu_D\rangle}(\mathcal{R}) = \{l_\Omega|_q \mid l \in L(\mathcal{R}), q \in \mathcal{P}os_\Sigma(l), \exists p \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l), p \leq q\}$$

For instance, $TL_{\langle\mu,\mu_D\rangle}(\mathcal{R}) = \{\Omega\texttt{:}\Omega\}$ for $\mathcal{R},\mu$, and μ_D as in Example 1. Then, for each $t \in TL_{\langle\mu,\mu_D\rangle}(\mathcal{R})$ and $f \in \Sigma$, we add a rule

$$\texttt{mark''}(t,f(x_1,\ldots,x_k)) \to \texttt{active}(f(u(t,f,1),\ldots,u(t,f,k)))$$

where $u(t,f,j) = \texttt{mark'}(x_j)$ if $j \in \mu \sqcup \mu_D(f)$ and either $root(t) \neq f$ or $t|_j \in \mathcal{T}(\Sigma)$; $u(t,f,j) = \texttt{mark''}(t|_j,x_j)$, if $j \in \mu \sqcup \mu_D(f)$, $root(t) = f$, and $t|_j$ contains Ω but $t|_j \neq \Omega$. Finally, $u(t,f,j) = x_j$ in any other case.

After an 'on-demand' reduction step (that introduces a symbol `mark`), we should allow the activation of new on-demand reductions. Thus, we add a rule

$$\texttt{mark}(x) \to \texttt{done}(x)$$

to 'inform' that a reduction step has been performed 'on-demand', and rules

$$f(x_1,\ldots,\texttt{done}(x_j),\ldots,x_k) \to \texttt{done}(f(x_1,\ldots,x_j,\ldots,x_k))$$

for each $f \in \Sigma$ such that $ar(f) > 0$, and $j \in \mu \sqcup \mu_D(f)$, to propagate symbol `done` upwards. We could not follow *ODR* derivations with the transformed TRS without pushing up `mark` via `done`. On the other hand, propagating `mark` (itself) upwards would produce a non-terminating TRS (due to the other rules of `mark`). We also add:

$$\texttt{onDemand}(\texttt{done}(x)) \to \texttt{mark}(x)$$

Definition 3 (First transformation). *Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu,\mu_D \in M_\Sigma$. We let $\mathcal{R}^1_{\langle\mu,\mu_D\rangle}$ be $(\Sigma \cup \{\Omega, \texttt{active}, \texttt{done}, \texttt{mark}, \texttt{mark'}, \texttt{mark''}, \texttt{demand}, \texttt{onDemand}\}, R^1_\mu \cup S^1_{\langle\mu,\mu_D\rangle})$, where R^1_μ is given as in Definition 2 and rules in $S^1_{\langle\mu,\mu_D\rangle}$ are defined as in Figure 4.*

Example 6. *(continuing Example 4) Let μ_D be as in Example 1. The rules of $\mathcal{R}^1_{\langle\mu,\mu_D\rangle}$ consist of the rules of $\mathcal{R}^1_\mu$ in Example 4 plus the following ones:*

```
demand(2nd(x:X))   → onDemand(2nd(x:mark''(Ω:Ω,X)))
mark'(2nd(x))      → active(2nd(mark'(x)))
mark'(from(x))     → active(from(x))
mark'(x:y)         → active(mark'(x):mark'(y))
mark'(s(x))        → active(s(mark'(x)))
mark'(0)           → active(0)
mark''(Ω:Ω,2nd(x))   → active(2nd(mark'(x)))
mark''(Ω:Ω,from(x))  → active(from(x))
mark''(Ω:Ω,x:y)      → active(x:y)
mark''(Ω:Ω,s(x))     → active(s(x))
mark''(Ω:Ω,0)        → active(0)
2nd(done(x))       → done(2nd(x))
from(done(x))      → done(from(x))
done(x):y          → done(x:y)
x:done(y)          → done(x:y)
s(done(x))         → done(s(x))
onDemand(done(x))  → mark(x)
active(x)          → demand(x)
mark(x)            → done(x)
```

The following theorem establishes the *correctness* of the transformation of Definition 3 for proving termination of *ODR*.

Theorem 4. *Let $\mathcal{R} = (\Sigma, R)$ be a left-linear TRS and $\mu,\mu_D \in M_\Sigma$. If $\mathcal{R}^1_{\langle\mu,\mu_D\rangle}$ is terminating, then $\mathcal{R}$ is $\langle\mu,\mu_D\rangle$-terminating.*

Termination of TRS $\mathcal{R}^1_{\langle\mu,\mu_D\rangle}$ of Example 6 can also be (tediously) proved by checking that there exists no infinite chain of dependency pairs associated to $\mathcal{R}^1_{\langle\mu,\mu_D\rangle}$. Thus, TRS $\mathcal{R}$ of Example 1 is $\langle\mu,\mu_D\rangle$-terminating.

For all $l \to r \in R$, $f \in \Sigma$ such that $k = ar(f) > 0$, $i \in \mu(f)$, and constants $c \in \Sigma$) in R^2_μ:

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

Figure 5: Rules for transformation $\mathcal{R}^2_\mu$ of Giesl and Middeldorp (Definition 4)

4.1 A complete transformation

Giesl and Middeldorp noticed that the transformation of Definition 2 is incomplete, i.e., there exist TRSs $\mathcal{R}$ and replacement maps μ such that $\mathcal{R}$ is μ-terminating but $\mathcal{R}^1_\mu$ is not terminating (see Example 1 in [10]). Since *CSR* is a case of *ODR* and our transformation is an extension of that of Definition 2, the transformation of Definition 3 inherits its incompleteness. Nevertheless Giesl and Middeldorp also provided a correct and complete transformation to deal with termination of *CSR*. Basically their idea is to permit *a single* (context-sensitive) reduction step each time. They achieve this by using new symbols f' for each (non-constant) symbol $f \in \Sigma$ and shifting a single symbol `active` to (non-deterministically) reach the replacing position where the redex is placed. The application of a rewrite rule changes `active` into `mark` which is propagated upwards through the term, in order to be replaced by a new symbol `active` that enables new reduction steps. After checking that no 'strange' symbols remain uncontrolled (using a symbol `proper` such that `proper`(t) reduces to `ok`(t) if and only if t is a ground term of the original signature), a rule `top(ok(`x`))` $\to$ `top(active(`x`))` enables a new reduction step (see [10] for a more detailed explanation).

Definition 4. *[10] Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu \in M_\Sigma$. The TRS $\mathcal{R}^2_\mu = (\Sigma \cup \{f' \mid f \in \Sigma \wedge ar(f) > 0\} \cup \{\mathtt{active}, \mathtt{mark}, \mathtt{ok}, \mathtt{proper}, \mathtt{top}\}, R^2_\mu)$ consists of the rules in Figure 5.*

This transformation is more involved than the previous one as the following example (that can be compared with Example 4) shows:

Example 7. *Consider $\mathcal{R}$ and μ as in Example 1. Then, $\mathcal{R}^2_\mu$ is:*

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

2nd'(mark(x))      → mark(2nd(x))
from'(mark(x))     → mark(from(x))
s'(mark(x))        → mark(s(x))
mark(x):'y         → mark(x:y)

proper(2nd(x))     → 2nd(proper(x))
proper(from(x))    → from(proper(x))
proper(x:y)        → proper(x):proper(y)
proper(s(x))       → s(proper(x))
proper(0)          → ok(0)

2nd(ok(x))         → ok(2nd(x))
from(ok(x))        → ok(from(x))
ok(x):ok(y)        → ok(x:y)
s(ok(x))           → ok(s(x))

top(mark(x))       → top(proper(x))
top(ok(x))         → top(active(x))
```

Extending this transformation to *ODR* is not difficult now. Again, we add a rule

$$\mathtt{active}(x) \to \mathtt{demand}(x)$$

to activate on-demand computations. The semantics of symbol `demand` is similar to the semantics above: for each $l \to r \in R$ and $p \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l) - \mathcal{P}os_{\mathcal{X}}(l)$, we add a rule

$$\mathtt{demand}(\tau(l)) \to \mathtt{onDemand}(\tau(l)[s(l,p)]_p)$$

where $s(l,p) = \mathtt{active'}(X)$ if $l|_p \in \mathcal{T}(\Sigma)$, and $s(l,p) = \mathtt{active''}(l_\Omega|_p, X)$ if $l|_p \notin \mathcal{T}(\Sigma)$. In both cases, $X = \tau(l)|_p$. Similar to Definition 4, symbols `active'` activate subterms according to *both* μ and μ_D:

$$\mathtt{active'}(f(x_1,\ldots,x_j,\ldots,x_k)) \to f'(x_1,\ldots,\mathtt{active'}(x_j),\ldots,x_k)$$

if $f \in \Sigma$ is such that $ar(f) > 0$ and $j \in \mu \sqcup \mu_D(f)$. With `active''`, we use term t to know whether it is allowed to get inside a term: for each $t \in TL_{\langle\mu,\mu_D\rangle}(\mathcal{R})$, $f \in \Sigma$ such that $ar(f) > 0$, and $j \in \mu \sqcup \mu_D(f)$, we add a rule

$$\mathtt{active''}(t, f(x_1,\ldots,x_j,\ldots,x_k)) \to f'(x_1,\ldots,u(t,f,j),\ldots,x_k)$$

where $u(t,f,j) = \mathtt{active'}(x_j)$ if either $root(t) \neq f$ or $t|_j \in \mathcal{T}(\Sigma)$, and $u(t,f,j) = \mathtt{active''}(t|_j, x_j)$, if $root(t) = f$ and $t|_j$ contains Ω but $t|_j \neq \Omega$.

In order to perform reductions below the on-demand position, for each $l \to r$ in the original TRS $\mathcal{R}$, we need to add rules

$$
\begin{array}{rcl}
\mathtt{active'}(l) & \to & \mathtt{mark}(r) \\
\mathtt{active''}(\mathtt{X}, l) & \to & \mathtt{mark}(r)
\end{array}
$$

to enable reductions activated by `active'` and `active''`. The symbol `mark` should be pushed up through replacing indices in $\mu \sqcup \mu_D$ rather than just μ. Thus, for each $f \in \Sigma$ such that $ar(f) > 0$, and $j \in \mu \sqcup \mu_D(f)$, we complete the rules of Definition 4 with rules

$$f'(x_1,\ldots,\mathtt{mark}(x_j),\ldots,x_k) \to \mathtt{mark}(f(x_1,\ldots,x_j,\ldots,x_k))$$

For all $l \to r \in R$, $p \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l) - \mathcal{P}os_{\mathcal{X}}(l)$, $t \in TL_{\langle\mu,\mu_D\rangle}(\mathcal{R})$, $f \in \Sigma$ such that $ar(f) > 0$, and $j \in \mu \sqcup \mu_D(f)$:

$$\begin{array}{rcl}
\texttt{active'}(l) & \to & \texttt{mark}(r) \\
\texttt{active''}(\texttt{X},l) & \to & \texttt{mark}(r) \\
\texttt{demand}(\tau(l)) & \to & \texttt{onDemand}(\tau(l)[s(l,p)]_p) \\
\texttt{active'}(f(x_1,\ldots,x_j,\ldots,x_k)) & \to & f'(x_1,\ldots,\texttt{active'}(x_j),\ldots,x_k) \\
\texttt{active''}(t,f(x_1,\ldots,x_j,\ldots,x_k)) & \to & f'(x_1,\ldots,u(t,f,j),\ldots,x_k) \\
f'(x_1,\ldots,\texttt{mark}(x_j),\ldots,x_k) & \to & \texttt{mark}(f(x_1,\ldots,x_j,\ldots,x_k)) \\
\texttt{onDemand}(\tau(l)[\texttt{mark}(X)]_p) & \to & \texttt{mark}(\tau(l)) \\
\texttt{active}(x) & \to & \texttt{demand}(x)
\end{array}$$

where $\tau(l)$ is as in Definition 3, $X = \tau(l)|_p$, and

1. $s(l,p) = \texttt{active'}(X)$ if $l|_p \in \mathcal{T}(\Sigma)$ and $s(l,p) = \texttt{active''}(l_\Omega|_p, X)$, otherwise. In both cases, $X = \tau(l)|_p$.
2. $u(t,f,j) = \texttt{active'}(x_j)$ if either $root(t) \neq f$ or $t|_j \in \mathcal{T}(\Sigma)$, and $u(t,f,j) = \texttt{active''}(t|_j, x_j)$, if $root(t) = f$ and $t|_j$ contains Ω but $t|_j \neq \Omega$.

Figure 6: Rules $S^2_{\langle\mu,\mu_D\rangle}$ of transformation $\mathcal{R}^2_{\langle\mu,\mu_D\rangle}$ for proving termination of *ODR* (Definition 5)

We also add rules

$$\texttt{onDemand}(\tau(l)[\texttt{mark}(X)]_p) \to \texttt{mark}(\tau(l))$$

for each $l \to r \in R$ and $p \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(l) - \mathcal{P}os_{\mathcal{X}}(l)$, where $X = \tau(l)|_p$.

Definition 5 (Second transformation). *Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu, \mu_D \in M_\Sigma$. We let $\mathcal{R}^2_{\langle\mu,\mu_D\rangle}$ be $(\Sigma \cup \{f' \mid f \in \Sigma \wedge ar(f) > 0\} \cup \{\Omega, \texttt{active}, \texttt{active'}, \texttt{active''}, \texttt{demand}, \texttt{mark}, \texttt{ok}, \texttt{onDemand}, \texttt{proper}, \texttt{top}\}, R^2_\mu \cup S^2_{\langle\mu,\mu_D\rangle})$, where R^2_μ is given as in Definition 4 and rules in $S^2_{\langle\mu,\mu_D\rangle}$ are defined as in Figure 6.*

Example 8. *(continuing Example 7) Let μ_D be as in Example 1. The rules of $\mathcal{R}^2_{\langle\mu,\mu_D\rangle}$ consist of the rules of $\mathcal{R}^2_\mu$ in Example 7 plus the following ones:*

```
active'(2nd(x:y:z))        → mark(y)
active'(from(x))           → mark(x:from(s(x)))
active''(X,2nd(x:y:z))     → mark(y)
active''(X,from(x))        → mark(x:from(s(x)))
demand(2nd(x:X)) → onDemand(2nd(x:active''(Ω:Ω,X)))
active'(2nd(x))  → 2nd'(active'(x))
active'(from(x)) → from'(x)
active'(x:y)     → active'(x):'y
active'(x:y)     → x:'active'(y)
active'(s(x))    → s'(active'(x))
active''(Ω:Ω,2nd(x))       → 2nd'(active'(x))
active''(Ω:Ω,from(x))      → from'(x)
active''(Ω:Ω,s(x))         → s'(active'(x))
x:'mark(y)                 → mark(x:y)
onDemand(2nd(x:mark(X)))   → mark(2nd(x:X))
active(x)                  → demand(x)
```

The following theorems establish correctness and completeness of our transformation w.r.t. proofs of $\langle\mu,\mu_D\rangle$-termination.

Theorem 5. *Let $\mathcal{R} = (\Sigma, R)$ be a left-linear TRS and $\mu, \mu_D \in M_\Sigma$. If $\mathcal{R}^2_{\langle\mu,\mu_D\rangle}$ is terminating, then $\mathcal{R}$ is $\langle\mu,\mu_D\rangle$-terminating.*

Theorem 6. *Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu, \mu_D \in M_\Sigma$. If $\mathcal{R}$ is $\langle\mu,\mu_D\rangle$-terminating, then $\mathcal{R}^2_{\langle\mu,\mu_D\rangle}$ is terminating.*

5. APPLICATION TO OBJ PROGRAMS

Algebraic languages, such as OBJ2 [8], OBJ3 [11], CafeOBJ [9], or Maude [3] admit the specification of *local strategies* associated to function symbols. A local strategy for a k-ary symbol $f \in \Sigma$ is a sequence $\varphi(f)$ of integers taken from $\{-k,\ldots,-1,0,1,\ldots,k\}$ which are given in parentheses (see Figure 1). Symbols without an explicit local strategy are given a *default* one whose concrete shape depends on the considered language[2]. A mapping φ that associates a local strategy $\varphi(f)$ to every $f \in \Sigma$ is called an E-strategy map [16, 17]. Given φ and $f \in \Sigma$, in OBJ2 or Maude, only nonnegative numbers are allowed in $\varphi(f)$; in OBJ3 and CafeOBJ, arbitrary integers can be used. Local strategies serve to *completely guide* the *evaluation strategy* or *E-strategy* (see Section 1).

Given an E-strategy map φ for Σ, we associate a pair $\langle\mu^\varphi, \mu^\varphi_D\rangle$ of replacement maps $\mu^\varphi, \mu^\varphi_D \in M_\Sigma$ as follows: for each $f \in \Sigma$, $\mu^\varphi(f) = \{i > 0 \mid i \in \varphi(f)\}$ and $\mu^\varphi_D(f) = \{-i \mid i \in \varphi(f) \wedge i < 0\}$, where notation $i \in \varphi(f)$ means that index i appears somewhere within the list $\varphi(f)$. Note that we drop null indices $0 \in \varphi(f)$ and that, since we use *sets* of indices rather than *lists*, we lose the information concerning repeated indices and the *order* of reductions. This is because, in our setting, we are mainly concerned with the analysis of (nondeterministic) reduction *relations* rather than with (deterministic) reduction *strategies*. Given φ, if $\mu^\varphi_D = \mu_\perp$, we say that φ is a *positive* E-strategy map. We will drop superscripts φ from μ^φ and μ^φ_D if no confusion arises.

5.1 Programs with positive E-strategy maps

Semantics of OBJ programs under a given E-evaluation map φ is usually given by means of a recursive evaluation function $eval_\varphi$ rather than specifying the concrete rewrite steps leading to computed values [5, 18]. Nakamura and Ogata describe the evaluation strategy $eval_\varphi$ for positive E-strategy maps by using a reduction relation on pairs $\langle t, p\rangle$ of labelled terms t and positions p [17]. Their formalization is based on Nagaya's [16]. Let $\mathcal{L}$ be the set of all lists consisting of natural numbers. By $\mathcal{L}_n$ we denote the set of all lists of

[2]For instance, in Maude, the default local strategy associated to a k-ary symbol f, is $(1,2,\ldots,k,0)$, see [5].

natural numbers not exceding $n \in \mathbb{N}$. We use the signature $\Sigma_{\mathcal{L}} = \{f_L \mid f \in \Sigma \wedge L \in \mathcal{L}_{ar(f)}\}$ and labelled variables $\mathcal{X}_{\mathcal{L}} = \{x_{nil} \mid x \in \mathcal{X}\}$. An E-strategy map φ for Σ is extended to a mapping from $\mathcal{T}(\Sigma, \mathcal{X})$ to $\mathcal{T}(\Sigma_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}})$ as follows:

$$\varphi(t) = \begin{cases} x_{nil} & \text{if } t = x \in \mathcal{X} \\ f_{\varphi(f)}(\varphi(t_1), \ldots, \varphi(t_k)) & \text{if } t = f(t_1, \ldots, t_k) \end{cases}$$

On the other hand, the mapping $erase : \mathcal{T}(\Sigma_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}}) \to \mathcal{T}(\Sigma, \mathcal{X})$ removes labellings from symbols in the obvious way.

Then, given a TRS $\mathcal{R} = (\Sigma, R)$ and a positive E-strategy map φ for Σ, $eval_\varphi : \mathcal{T}(\Sigma, \mathcal{X}) \to \mathcal{P}(\mathcal{T}(\Sigma, \mathcal{X}))$ is defined as $eval_\varphi(t) = \{erase(s) \in \mathcal{T}(\Sigma, \mathcal{X}) \mid \langle \varphi(t), \Lambda\rangle \to_\varphi^* \langle s, \Lambda\rangle \in \mathsf{NF}_{\to_\varphi}\}$, where $\mathsf{NF}_{\to_\varphi}$ is the set of normal forms of the binary relation $\to_\varphi$ on $\mathcal{T}(\Sigma_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}}) \times \mathbb{N}_+^*$ defined as follows ([17], Definition 2.2; [16], Definition 6.1.3): $\langle t, p\rangle \to_\varphi \langle s, q\rangle$ if and only if $p \in \mathcal{P}os(t)$ and either

1. $root(t|_p) = f_{nil}$, $s = t$ and $p = q.i$ for some i; or
2. $t|_p = f_{i:L}(t_1, \ldots, t_k)$ with $i > 0$, $s = t[f_L(t_1, \ldots, t_k)]_p$ and $q = p.i$; or
3. $t|_p = f_{0:L}(t_1, \ldots, t_k)$, $erase(t|_p)$ is not a redex, $s = t[f_L(t_1, \ldots, t_k)]_p$, $q = p$; or
4. $t|_p = f_{0:L}(t_1, \ldots, t_k) = \sigma(l')$, $erase(l') = l$, $s = t[\sigma(\varphi(r))]_p$ for some $l \to r \in R$ and substitution σ, $q = p$.

Note that the first three cases yield completely *deterministic* steps, whereas the non-determinism of the fourth one (only) comes from the possibility of choosing different rules that apply to the same redex.

The following theorem establishes a very close connection between $\to_\varphi$-reduction and $\hookrightarrow_\mu$-reduction.

THEOREM 7. *Let $\mathcal{R}$ be a TRS and φ be a positive E-strategy map. Let $t \in \mathcal{T}(\Sigma_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}})$, and $p \in \mathcal{P}os^\mu(erase(t))$. If $\langle t, p\rangle \to_\varphi \langle s, q\rangle$, then $q \in \mathcal{P}os^\mu(erase(s))$ and $erase(t) \hookrightarrow_\mu^= erase(s)$.*

PROOF. We consider the four cases of the definition of $\to_\varphi$.

1. For the first case, we have $t = s$ and $p = q.i$; since the set of μ-replacing positions is closed under prefix and $t = s$, we have $q \in \mathcal{P}os^\mu(erase(s))$ and $erase(t) = erase(s)$.
2. For the second case, $t|_p = f_{i:L}(t_1, \ldots, t_k)$ with $i > 0$, $s = t[f_L(t_1, \ldots, t_k)]_p$ and $q = p.i$. Then, since $p \in \mathcal{P}os^\mu(erase(t))$ and, by definition of μ, $i \in \mu(f)$, it follows that $p.i \in \mathcal{P}os^\mu(erase(t))$. Since $erase(t) = erase(s)$, we have $q \in \mathcal{P}os^\mu(erase(s))$,
3. For the third case, we have $erase(t) = erase(s)$ and $q = p \in \mathcal{P}os^\mu(erase(s))$.
4. For the last case, since $p \in \mathcal{P}os^\mu(t)$, we have that $erase(t) \overset{p}{\hookrightarrow}_\mu erase(s)$ and $q = p$. Since we have that $sprefix_{erase(t)}(p) = sprefix_{erase(s)}(p)$, by Proposition 2, we conclude $q \in \mathcal{P}os^\mu(erase(s))$.

□

As a consequence of Theorem 7, computations with OBJ programs that use positive E-strategy maps can be completely modeled as context-sensitive rewriting computations.

5.1.1 Termination

Let P be an OBJ program with positive E-strategy map φ, and $\mathcal{R}_{\mathsf{P}}$ and μ be the TRS and replacement map associated to it. According to the previous definition of $eval_\varphi$, we can say that P is terminating if, for all $t \in \mathcal{T}(\Sigma, \mathcal{X})$, there is no infinite $\to_\varphi$-rewrite sequence starting from $\langle \varphi(t), \Lambda\rangle$. Therefore, we have the following.

THEOREM 8. *An OBJ program P with positive E-strategy map φ is terminating if $\mathcal{R}_{\mathsf{P}}$ is μ-terminating.*

PROOF. Assume that P is not terminating. Then, there exist a term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ and an infinite sequence A : $\langle \varphi(t), \Lambda\rangle = \langle t_1, p_1\rangle \to_\varphi \cdots \to_\varphi \langle t_n, p_n\rangle \to_\varphi \cdots$. Since $p_1 = \Lambda \in \mathcal{P}os^\mu(t)$, by Theorem 7 there exists an infinite sequence $erase(t_1) \hookrightarrow_\mu^= \cdots \hookrightarrow_\mu^= erase(t_n) \hookrightarrow_\mu^= \cdots$. Since lists associated to E-strategy maps are finite, the number of reduction steps that corresponds to items (1) to (3) of the definition of $\to_\varphi$ is also finite. Hence, since $\mathcal{R}$ is μ-terminating, we can assume that $erase(t_n)$ is a μ-normal form (for some positive integer n) and that $erase(t_m) = erase(t_n)$ for all $m \geq n$. Therefore, the only $\to_\varphi$ reduction steps on terms t_m for $m \geq n$ correspond to items (1) to (3) of the definition of $\to_\varphi$. Since the number of such reduction steps is finite, this contradicts that A is an infinite $\to_\varphi$-reduction sequence. □

EXAMPLE 9. *The following* OBJ3 *program:*

```
obj EXAMPLE is
  sorts Sort .
  op 0    : -> Sort .
  op s    : Sort -> Sort .
  op _::_ : Sort Sort -> Sort [strat: (1 0)] .
  op from : Sort -> Sort .
  op sel  : Sort Sort -> Sort .
  var X Y L : Sort .
  eq sel(s(X),Y::L) = sel(X,L) .
  eq sel(0,X::L) = X .
  eq from(X) = X::from(s(X)) .
endo
```

specifies an explicit *(positive) local strategy for the list constructor '`::`'. By using the methods developed in [20], it is possible to prove that the TRS associated to this program is μ-terminating (for μ obtained from the E-strategy map of the program). Hence, we can say that the program is* terminating.

A more interesting (and complicated) example is the OBJ3 program for computing with prime numbers using the Sieve of Erasthothenes, that exemplifies how laziness is obtained with OBJ programs, in Appendix C.5 of [11]. A similar version of such a program (represented as a TRS with a replacement map) was used by Giesl and Middeldorp in [10] to exemplify the (successful) application of their methods to prove termination of *CSR*.

5.1.2 Computing (head-)normal forms

As remarked in [11, 18], computations with programs that use replacement restrictions may not directly compute normal forms (i.e., terms without redexes), but rather normal forms w.r.t. the reduction relation that is used for implementing computations (e.g., $\to_\varphi$). Terms returned by $eval_\varphi$

are called *E-normal forms* (*ENF*s). We prove that they are, in fact, μ-normal forms, under the following restriction on φ: for all $f \in \mathcal{F}$ (i.e., for *defined* symbols), $\varphi(f)$ ends in 0 (see [5, 16] for a discussion about the interest of such a requirement).

THEOREM 9. *Let $\mathcal{R} = (\mathcal{C} \uplus \mathcal{F}, R)$ be a TRS and φ be a positive E-strategy map such that for all $f \in \mathcal{F}$, $\varphi(f)$ ends in 0. If $s \in eval_\varphi(t)$, then s is a μ-normal form.*

PROOF. Analogous to the proof of Theorem 6.1.12 of [16]. As we restrict our result to μ-normal forms, we do not need to use Nagaya's additional hypothesis that $\varphi(f)$ contains all indices $1, \ldots, ar(f)$. $\square$

Theorem 9 does not hold for programs whose defined symbols have local strategies ended with positive indices.

EXAMPLE 10. *Consider the following TRS (from Example 6.1.5 of [16]):*

```
g(b,x) → b        b → a
f(a)   → c
```

and the strategy map φ defined by $\varphi(\mathtt{f}) = (0, 1)$, $\varphi(\mathtt{g}) = (1, 0)$ and $\varphi(\mathtt{a}) = \varphi(\mathtt{c}) = nil$ and $\varphi(\mathtt{b}) = (0)$. Consider the term $t =$ `g(f(b),b)`. Then we have that $eval_\varphi(t) =$ {`g(f(a),b)`}. However, `g(f(a),b)` *is not a μ-normal form, since we have:*

$$\mathtt{g}(\underline{\mathtt{f(a)}},\mathtt{b}) \hookrightarrow_\mu \mathtt{g(c,b)}$$

which is a μ-normal form (note that $2 \notin \mu(\mathtt{g})$).

According to Theorem 7, the formalization of $\rightarrow_\varphi$ above could be seen as the *description* of a context-sensitive rewriting *strategy* [15] which is *specified* by means of local strategies.

Computations yielding partially evaluated results can also be used to achieve full evaluations: if a term t has a (possibly infinite) normal form, it can eventually be computed by first obtaining a head-normal form $s = f(s_1, \ldots, s_k)$ of t (i.e., a term that cannot be reduced to a redex), and then recursively normalizing s_i for $1 \leq i \leq k$. Therefore, a quite elementary requirement for a restriction of rewriting is that it is (at least) able to compute head-normal forms of an input term (whenever they exist). With *CSR* we have such a useful property whenever the replacement map μ is less restrictive than the canonical replacement map $\mu^{can}_{\mathcal{R}}$ which, for a given TRS $\mathcal{R} = (\Sigma, R)$, is the most restrictive replacement map (in M_Σ) ensuring that the non-variable subterms of the left-hand sides of the rules of $\mathcal{R}$ are replacing, i.e., the minimum Σ-map μ such that $\forall l \in L(\mathcal{R}), \mathcal{P}os_\Sigma(l) \subseteq \mathcal{P}os^\mu(l)$ [14, 15]. Then, we have the following.

THEOREM 10. *[14] Let $\mathcal{R} = (\Sigma, R)$ be a left-linear TRS and $\mu \in M_\Sigma$ be such that $\mu^{can}_{\mathcal{R}} \sqsubseteq \mu$. Every μ-normal form is a head-normal form.*

Conditions on φ for ensuring that $eval_\varphi$ yields head-normal forms (i.e., terms that do not reduce to a redex) and normal forms using positive E-strategy maps have been studied in [16, 17]. Theorems 9 and 10 can be used to ensure that the E-strategy is able to compute head-normal forms.

COROLLARY 1. *Let* P *be an* OBJ *program with positive E-strategy map φ such that $\mathcal{R}_{\mathsf{P}}$ is left-linear and for all $f \in \mathcal{F}$, $\varphi(f)$ ends in 0 and $\mu^{can}_{\mathcal{R}_{\mathsf{P}}} \sqsubseteq \mu$. If $s \in eval_\varphi(t)$, then s is a head-normal form.*

This result is also given in [17] (Theorem 3.6) with the same conditions we have imposed here[3]; however, we are able to see this as an immediate consequence of computational properties of *CSR*.

For instance, E-normal forms of the OBJ3 program in Example 9 are head-normal forms[4]. By using the results in [14], we can also prove that a term such as `sel(s(0),from(0))` evaluates to the *value* `s(0)` with the OBJ3 program in Example 9 (without risk of infinite computations). Note that, in contrast to [16, 17], we do *not* need to assume that each local strategy $\varphi(f)$ for each $f \in \Sigma$ contains all indices $1, \ldots, ar(f)$ (but we need to consider some type information).

5.2 Programs with arbitrary E-strategy maps

Ogata and Futatsugi have provided an operational description of the *on-demand evaluation strategy* used in OBJ3 and CafeOBJ, where negative integers are allowed in local strategies [18]. Nakamura and Ogata [17] have described the corresponding evaluation mapping $eval_\varphi$ by using a reduction relation. Unfortunately, such a description is recursive and the definition of a single reduction step on a (labelled) term t may involve the application of more than one reduction step on subterms of t (see Definition 4.4 of [17]). In fact, the definition of a single rewriting step may depend on the possibility of evaluating some arguments of the considered function call. This implies that the one-step reduction relation proposed by Nakamura and Ogata is not even decidable (in general). Therefore, associated notions such as normal form (w.r.t. their reduction relation) are also undecidable. We believe that this is not the best framework for implementing and studying OBJ computations. In fact, in Section 6 of [17] (in contrast to the situation for positive E-strategy maps, see [16, 17]), Nakamura and Ogata show the difficulties in formulating conditions to achieve complete normalizations with arbitrary E-strategy maps. They conclude by saying that a 'meta-operation' that uses partially evaluated results (e.g., E-normal forms) would be necessary for obtaining normal forms. However, their E-normal forms would not be decidable in general and such a meta-operation becomes unfeasible in practice.

Normal forms can be obtained by successively computing μ-normal forms and shifting computations to maximal non-replacing subterms when a μ-normal form has been obtained (normalization via μ-normalization, [15]). Hence, we need to *decide* whether a term is a μ-normal form. Normal forms w.r.t. *CSR*, or *ODR*, are clearly decidable for finite TRSs. On the other hand, Proposition 4 below justifies that this 'layered' approach to normalization is possible with *ODR*. The proof uses the following immediate result.

PROPOSITION 3. *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, $\mu, \mu_D \in M_\Sigma$, and $C, t, s \in \mathcal{T}(\Sigma, \mathcal{X})$. If $p \in \mathcal{P}os^\mu(C)$, and $t \hookrightarrow_{\langle \mu, \mu_D \rangle} s$, then $C[t]_p \hookrightarrow_{\langle \mu, \mu_D \rangle} C[s]_p$.*

PROPOSITION 4. *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, $\mu, \mu_D \in M_\Sigma$, and $t \in \mathcal{T}(\Sigma, \mathcal{X})$ be a $\langle \mu, \mu_D \rangle$-normal form. If $p \in \mathcal{P}os^\mu(t)$, then $t|_p$ is a $\langle \mu, \mu_D \rangle$-normal form.*

[3] See `http://www.jaist.ac.jp/ masaki-n/correction.html` for a *corrigendum* of the published version of the paper that concerns this theorem.

[4] Actually, this depends on the form of default strategies for OBJ3 programs. In [11], they are described in such a way that are compatible with the canonical replacement map of the program.

PROOF. By contradiction. If $t|_p$ is not a $\langle\mu,\mu_D\rangle$-normal form, then $t|_p \hookrightarrow_{\langle\mu,\mu_D\rangle} s$ for some $s \in \mathcal{T}(\Sigma,\mathcal{X})$. Then, by Proposition 3, t is not a $\langle\mu,\mu_D\rangle$-normal form. $\Box$

Proposition 4 (or Proposition 3) does not hold if $p \notin \mathcal{P}os^{\mu}(t)$.

EXAMPLE 11. *Consider the TRS $\mathcal{R}$:*

```
f(g(x,a)) → a     g(a,b) → g(b,a)     c → a
```

Let $\mu(\mathtt{f}) = \varnothing$, $\mu(\mathtt{g}) = \{1,2\}$, $\mu_D(\mathtt{f}) = \{1\}$, and $\mu_D(\mathtt{g}) = \varnothing$. Term $t = \mathtt{f(g(c,b))}$ is a $\langle\mu,\mu_D\rangle$-normal form: first, no μ-replacing position of t corresponds to a redex. On the other hand,

$$\mathtt{f(g(\underline{c},b))} \not\hookrightarrow_{\langle\mu,\mu_D\rangle} \mathtt{f(g(a,b))}$$

since $1.1 \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t)$, but $\mathtt{f(g(c,a))}|_{1.1} \in \mathcal{X}$. However, $t|_1 = \mathtt{g(c,b)}$ is not *a $\langle\mu,\mu_D\rangle$-normal form: since $1 \in \mathcal{P}os^{\mu}(\mathtt{g(c,b)})$, we have:*

$$\mathtt{g(\underline{c},b)} \hookrightarrow_{\langle\mu,\mu_D\rangle} \mathtt{g(a,b)}$$

Note that $1 \notin \mathcal{P}os^{\mu}(t)$.

Proposition 4 suggests the following *normalization via $\langle\mu,\mu_D\rangle$-normalization* procedure (that can be thought of as the 'meta-operation' needed to obtain normal forms): given a $\langle\mu,\mu_D\rangle$-normal form s obtained from t, we split s as $s = C[s_1,\dots,s_n]$ where $C[\,]$ is the maximal replacing context of s, i.e., the part of s whose positions are μ-replacing in s (see [15]). Then, we continue the evaluation of t by (recursively) normalizing $s_1,\dots,s_n$. Note that this normalization procedure does not go back to previously $\langle\mu,\mu_D\rangle$-normalized prefixes of terms, i.e., after moving computations into terms $s_1,\dots,s_n$, $C[\,]$ remains untouched. In order to guarantee correctness of this procedure (i.e., that we actually obtain a normal form at the end), we need to ensure that restricted normal forms are head-normal forms. As Theorem 10 shows, this is true for μ-normal forms, whenever $\mu^{can}_{\mathcal{R}} \sqsubseteq \mu$ and $\mathcal{R}$ is left-linear. With *ODR*, we have a similar result.

THEOREM 11. *Let $\mathcal{R} = (\Sigma, R)$ be a left-linear CS and $\mu,\mu_D \in M_\Sigma$ be such that $\mu^{can}_{\mathcal{R}} \sqsubseteq \mu \sqcup \mu_D$. Every $\langle\mu,\mu_D\rangle$-normal form is a head-normal form.*

PROOF. Let $t \in \mathcal{T}(\Sigma,\mathcal{X})$. We proceed by structural induction. If t is a variable, it is obvious. If t is a constant c, then, since $\Lambda \in \mathcal{P}os^{\mu}(t)$, it must be $c \in \mathcal{C}$ which is a head-normal form. If t is not a head-normal form, then there exists a derivation $A : t \xrightarrow{>\Lambda}{}^{*} \sigma(l)$ for some $l \to r \in R$ and substitution σ. Therefore, $root(t) = root(l)$. By Proposition 4, $t|_q$ is a $\langle\mu,\mu_D\rangle$-normal form for all $q \in \mathcal{P}os^{\mu}(t)$ and, by the induction hypothesis, root-stable. Therefore, we can write $\sigma(l) = t[r_1]_{p_1}\cdots[r_n]_{p_n}$ for $\{p_1,\dots,p_n\} = \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t)$, where $r_i = \sigma(l)|_{p_i}$ for $1 \le i \le n$.

Since t is a $\langle\mu,\mu_D\rangle$-normal form, it is not a redex. By left-linearity of $\mathcal{R}$, if t is not a redex, there exists a minimal $q \in \mathcal{P}os(t) \cap \mathcal{P}os_\Sigma(l)$ such that $q \ne \Lambda$, $\mathit{sprefix}_t(q) = \mathit{sprefix}_l(q)$, $root(t|_q) \ne root(l|_q)$. Since $\mathcal{R}$ is a CS, $t|_q$ is not a head-normal form; hence $root(t|_q) \in \mathcal{F}$. By the induction hypothesis, $t|_q$ is not a $\langle\mu,\mu_D\rangle$-normal form. Hence, by Proposition 4, $q \notin \mathcal{P}os^{\mu}(t)$ and there exists a redex position p in $t|_q$ such that $p \in \mathcal{P}os^{\mu\sqcup\mu_D}(t|_q)$. By Proposition 1, $q.p \notin \mathcal{P}os^{\mu}(t)$. Since $q \in \mathcal{P}os_\Sigma(l)$, and $\mu^{can}_{\mathcal{R}} \sqsubseteq \mu \sqcup \mu_D$, it follows that $q \in \mathcal{P}os^{\mu\sqcup\mu_D}(l)$ and by Proposition 2, $q \in \mathcal{P}os^{\mu\sqcup\mu_D}(t)$. Hence, by Proposition 1, $q.p \in \mathcal{P}os^{\mu\sqcup\mu_D}(t)$, i.e., $t|_{q.p}$ is a $\mu \sqcup \mu_D$-replacing redex of t. Since $l|_q \notin \mathcal{X}$, for all $q' \le q$, $l|_{q'} \notin \mathcal{X}$. Thus, $t \hookrightarrow_{\langle\mu,\mu_D\rangle} t[s]_{q'.p}$ for some s, which contradicts that t is a $\langle\mu,\mu_D\rangle$-normal form. $\Box$

In general, Theorem 11 does not hold for TRSs that are not CSs.

EXAMPLE 12. *Consider $\mathcal{R}$ and t as in Example 11. Remember that $t = \mathtt{f(g(c,b))}$ is a $\langle\mu,\mu_D\rangle$-normal form. However, t is not a head-normal form, since we have:*

$$\mathtt{f(g(\underline{c},b))} \to \mathtt{f(\underline{g(a,b)})} \to \mathtt{f(g(b,a))}$$

and $\mathtt{f(g(b,a))}$ is a redex.

This problem was already noticed in [17] for the on-demand evaluation strategy of CafeOBJ. Left-linearity cannot be removed from Theorem 11 as the following example shows.

EXAMPLE 13. *Consider the following TRS $\mathcal{R}$:*

```
f(x,x) → c(x)
a      → b
```

which is not left-linear, and let $\mu(\mathtt{f}) = \varnothing$, $\mu(\mathtt{c}) = \varnothing$, and $\mu_D(\mathtt{f}) = \{1,2\}$. Note that $\mu^{can}_{\mathcal{R}} \sqsubseteq \mu \sqcup \mu_D$. Term $t = \mathtt{f(c(a),c(b))}$ is not root-stable:

$$\mathtt{f(c(\underline{a}),c(b))} \to \underline{\mathtt{f(c(b),c(b))}} \to \mathtt{c(c(b))}$$

Note that $1.1 \in \mathcal{P}os^{\mu\sqcup\mu_D}(t) - \mathcal{P}os^{\mu}(t)$, $1 \in \mathcal{L}azy_{\langle\mu,\mu_D\rangle}(t)$ and $\mathtt{f(c(a),c(b))}[\mathtt{b}]_{1.1}$ is a redex of $\mathtt{f(x,x)} \to \mathtt{a}$. However, since $\mathtt{f(x,x)}|_1 \notin \mathcal{X}$, t is a $\langle\mu,\mu_D\rangle$-normal form.

Finally, using pairs $\langle\mu,\mu_D\rangle$ such that $\mu^{can}_{\mathcal{R}} \not\sqsubseteq \mu \sqcup \mu_D$ does not work even with left-linear CSs.

EXAMPLE 14. *Consider the TRS*

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

and μ,μ_D given by $\mu(\mathtt{f}) = \mu(\mathtt{h}) = \varnothing$, and $\mu_D(\mathtt{f}) = \{1\}$. Let $t = \mathtt{f(a,d)}$. Note that t is a $\langle\mu,\mu_D\rangle$-normal form, but it is not a head-normal form.

Therefore, we propose using *ODR* for implementing OBJ computations. Further work should be done to express the evaluation function for OBJ programs that use arbitrary local strategies as reductions issued by an on-demand reduction *strategy*, i.e., a further restriction of *ODR* where the evaluation order introduced by the list structure of local strategies would be taken into account. This means that *ODR* is only a general framework for expressing OBJ computations (in this sense, the formalization of [17] tried to *exactly* capture what OBJ programs do), *ODR* brings us the advantage of being suitable for providing a standard termination analysis, as we have proved in this paper. Thus, according to our analysis, we could say that if the on-demand reduction relation associated to an OBJ program is terminating, then the OBJ program is terminating. For instance, after our analysis of termination in Section 4, we can say that the CafeOBJ program of Figure 1, which is modeled as a TRS with replacement restrictions in Example 1, is terminating.

6. CONCLUSIONS AND FUTURE WORK

We have introduced *on-demand rewriting* (*ODR*) as a generalization of context-sensitive rewriting (*CSR*) which is inspired by lazy rewriting [12]. *ODR* provides a computational model for TRSs equipped with a pair $\langle\mu,\mu_D\rangle$ of replacement maps, where μ is aimed at restricting reductions, and μ_D is aimed at overcoming replacement restrictions 'on-demand'. We have provided two transformations that can be used to analyze termination of *ODR* as termination of rewriting. Our transformations are correct for left-linear TRSs. Thus, it is possible to analyze the termination of *ODR* by using the standard methods for proving termination of rewriting [4]. Moreover, our second transformation is also *complete* thus showing that proving termination of *ODR* is equivalent to proving termination of rewriting (for left-linear TRSs). We have shown that *CSR* and *ODR* can be used to implement and analyze computations with OBJ programs. *CSR* suffices for modeling computations of OBJ programs that use positive local strategies. Therefore, all results on completeness and termination of computations with *CSR* [6, 10, 13, 14, 15, 19, 20] apply to such OBJ programs. On the other hand, *ODR* can be used with arbitrary OBJ programs. In this case, our results can be used to prove termination of OBJ programs and to ensure completeness of computations aimed at normalizing initial expressions.

Termination of *ODR* (or *CSR*) is not a closed topic. We have based our research on Giesl and Middeldorp's approach to termination of *CSR*. Their techniques are based on marking the replacing subterms that can be rewritten. In spite of the good achievements obtained with this approach (especially by providing a complete analysis of termination of *CSR*), other approaches to termination of *CSR* such as [13, 19, 20] are still promising from the practical point of view. For instance, μ-termination of $\mathcal{R}$ in Example 1 can easily be proved by using the TRS $\mathcal{R}^Z_\mu$ obtained by Zantema's transformation [20], which is based on marking the *non-replacing* arguments of function symbols. According to [20], $\mathcal{R}^Z_\mu$ is:

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

where :', from', and a are new symbols. $\mathcal{R}^Z_\mu$ can be proved terminating by using an *rpo* with precedence $>$ such that

$$\texttt{2nd} > \texttt{a} > \texttt{from} > \texttt{:}, \texttt{from'}, \texttt{s} > \texttt{:'}$$

As noticed in Remark 1, this is *not* possible with Giesl and Middeldorp's techniques. Moreover, the use of an *automatic* tool for proving termination of TRSs such as Contejean and Marché's C*i*ME 2.0 system [5] (that implements polynomial orderings rather than *rpo*'s) shows that, whereas termination of $\mathcal{R}^Z_\mu$ can be proved in a few seconds, the termination of $\mathcal{R}^1_\mu$ in Example 4 could not be proved after three days of execution. Thus, an interesting topic for future work is to extend/adapt Zantema's transformation to *ODR*.

Acknowledgements.

I thank Yoshihito Toyama for pointing [16] out to me. I also thank Masaki Nakamura for his remarks on a previous version of this paper.

[5] Available at `http://www.lri.fr/~demons/cime.html`.

7. REFERENCES

[1] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.

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

[3] M. Claver, S. Eker, P. Lincoln, and J. Meseguer. Principles of Maude. In *Proc. 1st International Workshop on Rewriting Logic and its Applications, RWLW'96*, pages 84–89. Electronic Notes in Theoretical Computer Science, Elsevier Sciences, 1996.

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

[5] S. Eker. Term rewriting with operator evaluation strategies. In *Proc. of 2nd International Workshop on Rewriting Logic and its Applications, WRLA'98*, pages 84–89. Electronic Notes in Theoretical Computer Science, Elsevier Sciences, 1998.

[6] M. Ferreira and A. Ribeiro. Context-sensitive ac-rewriting. In *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA'99*, pages 286–300. Springer-Verlag, LNCS 1631, 1999.

[7] W. Fokkink, J. Kamperman, and P. Walters. Lazy rewriting on eager machinery. *ACM Transactions on Programming Languages and Systems*, 22(1):45–86, 2000.

[8] 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, 1998.

[9] 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.

[10] J. Giesl and A. Middeldorp. Transforming context-sensitive rewrite systems. In *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA'99*, pages 271–285. Springer-Verlag, LNCS 1631, 1999.

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

[12] J. Kamperman and H. Walters. Lazy rewriting and eager machinery. In *Proc. of the 6th International Conference on Rewriting Techniques and Applications, RTA'95*, pages 147–162. Springer-Verlag, LNCS 914, 1995.

[13] S. Lucas. Termination of context-sensitive rewriting by rewriting. In *Proc. of 23rd. International Colloquium on Automata, Languages and Programming, ICALP'96*, pages 122–133. Springer-Verlag, LNCS 1099, 1996.

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

[15] S. Lucas. Context-sensitive rewriting strategies. Technical Report DSIC-II/7/00, 2000.

[16] T. Nagaya. *Reduction Strategies for Term Rewriting*

Systems, PhD Thesis. School of Information Science, Japan Advanced Institute of Science and Technology, 1999.

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

[18] 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.

[19] J. Steinbach and H. Xi. Freezing – termination proofs for classical, context-sensitive and innermost rewriting. Technical report, Institut fuer Informatik, T.U. Muenchen, 1998.

[20] H. Zantema. Termination of context-sensitive rewriting. In *Proc. of 8th International Conference on Rewriting Techniques and Applications, RTA'97*, pages 172–186. Springer-Verlag, LNCS 1232, 1997.