Improving On-Demand Strategy Annotations*

M. Alpuente[1], S. Escobar[1], B. Gramlich[2], and S. Lucas[1]

[1] DSIC, UPV, Camino de Vera s/n, E-46022 Valencia, Spain.
{alpuente,sescobar,slucas}@dsic.upv.es
[2] AG Theoretische Informatik und Logik, Institut für Computersprachen,
TU Wien, Favoritenstr. 9, E185/2 A-1040 Wien, Austria.
gramlich@logic.at

Abstract. In functional languages such as OBJ*, CafeOBJ, and Maude, symbols are given strategy annotations which specify (the order in) which subterms are evaluated. Syntactically, they are given either as lists of natural numbers or as lists of integers associated to function symbols whose (absolute) values refer to the arguments of the corresponding symbol. A positive index enables the evaluation of an argument whereas a negative index means "evaluate on-demand". While strategy annotations containing only natural numbers have been implemented and received some recent investigation endeavor (regarding, e.g., termination and completeness), fully general annotations (also called *on-demand* strategy annotations), which have been proposed to support laziness in OBJ-like languages, are disappointingly under-explored to date. In this paper, we first point out a number of problems of current proposals for handling on-demand strategy annotations. Then, we propose a solution to these problems which is based on a suitable extension of the E-evaluation strategy of OBJ-like languages (that only considers annotations given as natural numbers) to on-demand strategy annotations. Our strategy incorporates a better treatment of demandness and also exhibits good computational properties; in particular, we show how to use it for computing (head-)normal forms. We also introduce a transformation for proving termination of the new evaluation strategy by using standard techniques.

Keywords: Declarative programming, demandness, lazy evaluation, OBJ, on-demand strategy annotations

1 Introduction

Eager rewriting-based programming languages such as Lisp, OBJ*, CafeOBJ, ELAN, or Maude evaluate expressions by innermost rewriting. Since nontermination is a known problem of innermost reduction, *syntactic annotations* (generally specified as sequences of integers associated to function arguments, called *local strategies*) have been used in OBJ2 [FGJM85], OBJ3 [GWM+00], CafeOBJ [FN97], and Maude [CELM96] to improve efficiency and (hopefully) avoid nontermination. Local strategies are used in OBJ programs[3] for guiding the *evaluation strategy* (abbr. E-strategy): when considering a function call $f(t_1, \ldots, t_k)$,

* Work partially supported by CICYT TIC2001-2705-C03-01, Acciones Integradas HI2000-0161, HA2001-0059, HU2001-0019, and Generalitat Valenciana GV01-424.

[3] As in [GWM+00], by OBJ we mean OBJ2, OBJ3, CafeOBJ, or Maude.

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. The limits of using only positive annotations regarding correctness and completeness of computations are discussed in [AEL02,Luc01,Luc02b,OF00,NO01]: the obvious problem is that the absence of some indices in the local strategies can have a negative impact in the ability of such strategies to compute normal forms.

Example 1. Consider the following OBJ program (borrowed from [NO01]):

```
obj Ex1 is
  sorts Nat LNat .
  op 0    : -> Nat .
  op s    : Nat -> Nat        [strat (1)] .
  op nil  : -> LNat .
  op cons : Nat LNat -> LNat  [strat (1)] .
  op 2nd  : LNat -> Nat       [strat (1 0)] .
  op from : Nat -> LNat       [strat (1 0)] .
  vars X Y : Nat . var Z : LNat .
  eq 2nd(cons(X,cons(Y,Z))) = Y .
  eq from(X) = cons(X,from(s(X))) .
endo
```

The OBJ evaluation of `2nd(from(0))` is given by the sequence:

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

The evaluation stops here since reductions on the second argument of `cons` are disallowed. Note that we cannot apply the rule defining `2nd` because the subterm `from(s(0))` should be further reduced. Thus, a further step is *demanded* (by the rule of `2nd`) in order to obtain the desired outcome:

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

Now, we do *not* need to reduce the second argument of the inner occurrence of `cons` anymore, since reducing at the root position yields the final value:

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

Therefore, the rather intuitive notion of demanded evaluation of an argument of a function call (see [AL02] for a survey discussing this topic) arises as a possible solution to this problem. In [OF00,NO01], *negative* indices are proposed to indicate those arguments that should be evaluated only 'on-demand', where the 'demand' is an attempt to match an argument term with the left-hand side of a rewrite rule [Eke98,GWM$^+$00,OF00]. For instance, in [NO01] the authors suggest `(1 -2)` as the "apt" local strategy for `cons` in Example 1. The inspiration for the local strategies of OBJ comes from *lazy rewriting* (*LR*) [FKW00], a demand-driven technique where syntactic annotations allow the eager evaluation of function arguments, whereas the default strategy is lazy. However, the extended, on-demand E-strategy of [OF00,NO01] presents a number of drawbacks, which we formally address in the paper. The following example illustrates that the notion of demandness which is formalized in [NO01] needs to be refined to be entirely satisfactory in practice.

Example 2. Consider the following OBJ program:

```
obj Ex2 is
  sorts Nat LNat .
  op 0    : -> Nat .
  op s    : Nat -> Nat        [strat (1)] .
  op nil  : -> LNat .
  op cons : Nat LNat -> LNat  [strat (1)] .
  op from : Nat -> LNat       [strat (1 0)] .
  op length : LNat -> Nat     [strat (0)] .
  op length' : LNat -> Nat    [strat (-1 0)] .
  var X : Nat . var Z : LNat .
  eq from(X) = cons(X,from(s(X))) .
  eq length(nil) = 0 .
  eq length(cons(X,Z)) = s(length'(Z)) .
  eq length'(Z) = length(Z) .
endo
```

The expression `length'(from(0))` is rewritten (in one step) to `length(from(0))`. No evaluation is demanded on the argument of `length'` for enabling this step and no further evaluation on `length(from(0))` should be performed due to the local strategy `(0)` of `length`. However, the annotation `(-1 0)` of function `length'` is treated in such a way that the on-demand evaluation of the expression `length'(from(0))` yields an infinite sequence (whether[4] we use the operational model[5] in [OF00] or whether we use [NO01]). This is because the negative annotations are implemented as marks on terms which can (unproperly) initiate reductions later on; see Example 3 below.

In this paper, after some preliminaries in Section 2, in Section 3 we recall the current proposals for dealing with on-demand E-strategy annotations in OBJ languages and discuss some drawbacks regarding the treatment of demandness. In Section 4 we (re-)formulate the computational model of *on-demand strategy annotations* by handling demandness in a different way. We demonstrate that the new on-demand strategy outperforms the original one. We also show that our definition behaves better than lazy rewriting (LR) regarding the ability to compute (head-)normal forms. Section 5 introduces a transformation which can be used to formally prove termination of programs that use our computational model for implementing arbitrary strategy annotations. Section 6 concludes and summarizes our contributions.

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 transitive closure by R^+ and its reflexive and transitive closure by R^* [BN98]. An element $a \in A$ is an R-normal form, if there

[4] Actually, the operational models in [OF00] and [NO01] differ and deliver different computations, see Example 4 below.

[5] Negative annotations are (syntactically) accepted in current OBJ implementations, namely OBJ3 and CafeOBJ, but they have no effect over the computations.

exists 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 $a\ R^* b$. 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 function $ar : \mathcal{F} \to \mathbb{N}$. We denote the set of terms built from $\mathcal{F}$ and $\mathcal{X}$ by $\mathcal{T}(\mathcal{F}, \mathcal{X})$. 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 $p, q, \ldots \in \mathbb{N}^*_+$ are represented by chains of positive natural numbers used to address subterms of t. 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 $\mathcal{P}os_{\mathcal{F}}(t)$, and $\mathcal{P}os_{\mathcal{X}}(t)$ are the positions of variables. 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. $\leq$. The subterm at position p of t is denoted as $t|_p$, and $t[s]_p$ is the term t with the subterm at position p replaced by s. The symbol labelling the root of t is denoted as $root(t)$.

A rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side (*lhs*) of the rule is l and r is the right-hand side (*rhs*). A TRS is a pair $\mathcal{R} = (\mathcal{F}, R)$ where R is a set of rewrite rules. $L(\mathcal{R})$ 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 $\mathcal{R} = (\mathcal{F}, R)$, we take $\mathcal{F}$ as the disjoint union $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ of symbols $c \in \mathcal{C}$, called *constructors*, and symbols $f \in \mathcal{D}$, called *defined functions*, where $\mathcal{D} = \{root(l) \mid l \to r \in R\}$ and $\mathcal{C} = \mathcal{F} - \mathcal{D}$. An instance $\sigma(l)$ of a *lhs* $l \in L(\mathcal{R})$ by any substitution σ is called a redex. A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ rewrites to s (at position p), written $t \stackrel{p}{\to}_{\mathcal{R}} s$ (or just $t \to s$), if $t|_p = \sigma(l)$ and $s = t[\sigma(r)]_p$, for $l \to r \in R$, $p \in \mathcal{P}os(t)$ and substitution σ.

3 Rewriting with strategy annotations

A local strategy for a k-ary symbol $f \in \mathcal{F}$ is a sequence $\varphi(f)$ of integers taken from $\{-k, \ldots, -1, 0, 1, \ldots, k\}$ which are given in parentheses. We define an ordering $\sqsubseteq$ between sequences of integers as: $nil \sqsubseteq L$, for all sequence L, $(i_1\ i_2\ \cdots\ i_m) \sqsubseteq (j_1\ j_2\ \cdots\ j_n)$ if $i_1 = j_1$ and $(i_2\ \cdots\ i_m) \sqsubseteq (j_2\ \cdots\ j_n)$, or $(i_1\ i_2\ \cdots\ i_m) \sqsubseteq (j_1\ j_2\ \cdots\ j_n)$ if $i_1 \neq j_1$ and $(i_1\ i_2\ \cdots\ i_m) \sqsubseteq (j_2\ \cdots\ j_n)$.

A mapping φ that associates a local strategy $\varphi(f)$ to every $f \in \mathcal{F}$ is called an E-strategy map [Nag99,NO01]. An ordering $\sqsubseteq$ between strategy maps is defined: $\varphi \sqsubseteq \varphi'$ if for all $f \in \mathcal{F}$, $\varphi(f) \sqsubseteq \varphi'(f)$. Roughly speaking, $\varphi \sqsubseteq \varphi'$ if for all $f \in \mathcal{F}$, $\varphi'(f)$ is $\varphi(f)$ where some additional indices have been included.

Semantics of rewriting under a given E-evaluation map φ is usually given by means of a mapping $eval_\varphi : \mathcal{T}(\mathcal{F}, \mathcal{X}) \to \mathcal{P}(\mathcal{T}(\mathcal{F}, \mathcal{X}))$ from terms to the set of its computed values (technically E-normal forms).

Rewriting with positive E-strategy maps. Nagaya describes the mapping $eval_\varphi$ for *positive* E-strategy maps φ (i.e., E-strategy maps where negative indices are *not* allowed) by using a reduction relation on pairs $\langle t, p \rangle$ of labelled terms t and positions p [Nag99]. Let $\mathbb{L}$ be the set of all lists consisting of integers and $\mathbb{L}_n$ be the set of all lists of integers whose absolute value does not exceed $n \in \mathbb{N}$. Given an E-strategy map φ for $\mathcal{F}$, we use the signature $\mathcal{F}_\varphi^{\mathbb{N}} = \{f_L \mid f \in \mathcal{F}, L \in \mathbb{L}_{ar(f)}\ (L \sqsubseteq \varphi(f))\}$ and labelled variables $\mathcal{X}_\varphi^{\mathbb{N}} = \{x_{nil} \mid x \in \mathcal{X}\}$. An E-strategy map φ for $\mathcal{F}$ is extended to a mapping from $\mathcal{T}(\mathcal{F}, \mathcal{X})$ to $\mathcal{T}(\mathcal{F}_\varphi^{\mathbb{N}}, \mathcal{X}_\varphi^{\mathbb{N}})$ by introducing the local strategy associated to each symbol as a subscript of the symbol. The mapping $\mathit{erase} : \mathcal{T}(\mathcal{F}_\varphi^{\mathbb{N}}, \mathcal{X}_\varphi^{\mathbb{N}}) \to \mathcal{T}(\mathcal{F}, \mathcal{X})$ removes labellings from symbols in the obvious way. Then, given a TRS $\mathcal{R} = (\mathcal{F}, R)$ and a positive E-strategy map φ for $\mathcal{F}$, eval_φ is defined as $\mathit{eval}_\varphi(t) = \{\mathit{erase}(s) \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \mid \langle \varphi(t), \Lambda\rangle \overset{\mathbb{N}}{\to}{}^{!}_{\varphi} \langle s, \Lambda\rangle\}$, where the binary relation $\overset{\mathbb{N}}{\to}_\varphi$ on $\mathcal{T}(\mathcal{F}_\varphi^{\mathbb{N}}, \mathcal{X}_\varphi^{\mathbb{N}}) \times \mathbb{N}_+^*$ behaves as follows: given $\langle t, p\rangle$, where $t \in \mathcal{T}(\mathcal{F}_\varphi^{\mathbb{N}}, \mathcal{X}_\varphi^{\mathbb{N}})$ and $p \in \mathcal{P}os(t)$, if a positive index $i > 0$ is found in the list labelling the symbol at $t|_p$, then the index is removed from the list, the "target position" is moved from p to $p.i$, and the subterm $t|_{p.i}$ is next considered. If 0 is found, then the evaluation of $t|_p$ is attempted: if possible, a rewriting step is performed; otherwise, the 0 is removed from the list. In both cases, the evaluation continues at the same position p (see Definition 6.1.3 of [Nag99] or Definition 2.2 of [NO01]).

The on-demand evaluation strategy. Ogata and Futatsugi [OF00] have provided an operational description of the *on-demand evaluation strategy* eval_φ where negative integers are also allowed in local strategies. Nakamura and Ogata [NO01] have described the corresponding evaluation mapping eval_φ by using a reduction relation. We recall here the latter one since the former one is not appropriate for our objectives in this paper.

Given an E-strategy map φ, we use the signature[6] $\mathcal{F}_\varphi = \{f_L^b \mid f \in \mathcal{F} \wedge L \in \mathbb{L}_{ar(f)}.(L \sqsubseteq \varphi(f)) \wedge b \in \{0,1\}\}$ and labelled variables $\mathcal{X}_\varphi = \{x_{nil}^0 \mid x \in \mathcal{X}\}$. An on-demand flag $b = 1$ indicates that the term may be reduced if demanded. An E-strategy map φ for $\mathcal{F}$ is extended to a mapping from $\mathcal{T}(\mathcal{F}, \mathcal{X})$ to $\mathcal{T}(\mathcal{F}_\varphi, \mathcal{X}_\varphi)$ as follows: $\varphi(t) = \begin{cases} x_{nil}^0 & \text{if } t = x \in \mathcal{X} \\ f_{\varphi(f)}^0(\varphi(t_1), \ldots, \varphi(t_k)) & \text{if } t = f(t_1, \ldots, t_k) \end{cases}$. On the other hand, the mapping $\mathit{erase} : \mathcal{T}(\mathcal{F}_\varphi, \mathcal{X}_\varphi) \to \mathcal{T}(\mathcal{F}, \mathcal{X})$ removes labellings from symbols in the obvious way. The (partial) function $\mathit{flag} : \mathcal{T}(\mathcal{F}_\varphi, \mathcal{X}_\varphi) \times \mathbb{N}_+^* \to \{0,1\}$ returns the flag of the function symbol at a position of the term: $\mathit{flag}(t,p) = b$ if $\mathit{root}(t|_p) = f_L^b$. The map $\mathit{up} : \mathcal{T}(\mathcal{F}_\varphi, \mathcal{X}_\varphi) \to \mathcal{T}(\mathcal{F}_\varphi, \mathcal{X}_\varphi)$ (resp. $\mathit{dn} : \mathcal{T}(\mathcal{F}_\varphi, \mathcal{X}_\varphi) \to \mathcal{T}(\mathcal{F}_\varphi, \mathcal{X}_\varphi)$) raises (lowers) the on-demand flag of each function symbol in a term, i.e. $\mathit{up}(x_{nil}^0) = \mathit{dn}(x_{nil}^0) = x_{nil}^0$, $\mathit{up}(f_L^b(t_1, \ldots, t_k)) = f_L^1(\mathit{up}(t_1), \ldots, \mathit{up}(t_k))$, and $\mathit{dn}(f_L^b(t_1, \ldots, t_k)) = f_L^0(\mathit{dn}(t_1), \ldots, \mathit{dn}(t_k))$.

When it is being examined whether a term t matches the left-hand side l of a rule, a top-to-bottom and left-to-right pattern matching is performed. Let

[6] Note that Nakamura and Ogata's definition uses $\mathcal{F}_{\mathbb{L}}$ and $\mathcal{X}_{\mathbb{L}}$ instead of $\mathcal{F}_\varphi$ and $\mathcal{X}_\varphi$, where the restriction to $L \sqsubseteq \varphi(f)$ is not considered. However, using terms over $\mathcal{F}_\varphi$ does not entail loss of generality; furthermore, it actually provides a more accurate framework for formalizing and studying the strategy, since they are the only class of terms involved in the computations.

$\mathcal{P}os_{\neq}(t,l) = \{p \in \mathcal{P}os_{\mathcal{F}}(t) \cap \mathcal{P}os_{\mathcal{F}}(l) \mid root(l|_p) \neq root(t|_p)\}$ be the set of (common) positions of non-variable disagreeing symbols of terms t and l. Then, the map $df_l : \mathcal{T}(\mathcal{F},\mathcal{X}) \to \mathbb{N}^*_+ \cup \{\top\}$ returns the first position where the term and the lhs differ (on some non-variable position) or $\top$ if each function symbol of the term coincides with l:

$$df_l(t) = \begin{cases} min_{\leq_{lex}}(\mathcal{P}os_{\neq}(t,l)) & \text{if } \mathcal{P}os_{\neq}(t,l) \neq \varnothing \\ \top & \text{otherwise} \end{cases}$$

where $\leq_{lex}$ is the lexicographic ordering on positions: $p \leq_{lex} q$ iff $p \leq q$ or $p = w.i.p'$, $q = w.j.q'$, $i,j \in \mathbb{N}$, and $i < j$.

Similarly, given a TRS $\mathcal{R}$, the map $DF_{\mathcal{R}} : \mathcal{T}(\mathcal{F},\mathcal{X}) \to \mathbb{N}^*_+ \cup \{\top\}$ returns the first position where the term differ w.r.t. all lhs's:

$$DF_{\mathcal{R}}(t) = \begin{cases} \top & \text{if } df_l(t) = \top \text{ for some } l \to r \in \mathcal{R} \\ max_{<_{lex}}\{df_l(t) \mid l \to r \in \mathcal{R}\} & \text{otherwise} \end{cases}$$

Definition 1. *Given a TRS $\mathcal{R} = (\mathcal{F}, R)$ and an arbitrary E-strategy map φ for $\mathcal{F}$, $eval_\varphi : \mathcal{T}(\mathcal{F},\mathcal{X}) \to \mathcal{P}(\mathcal{T}(\mathcal{F},\mathcal{X}))$ is defined as $eval_\varphi(t) = \{erase(s) \in \mathcal{T}(\mathcal{F},\mathcal{X}) \mid \langle \varphi(t), \Lambda\rangle \to^!_\varphi \langle s, \Lambda\rangle\}$. The binary relation $\to_\varphi$ on $\mathcal{T}(\mathcal{F}_\varphi,\mathcal{X}_\varphi) \times \mathbb{N}^*_+$ is defined as follows ([NO01], Definition 4.4): $\langle t,p\rangle \to_\varphi \langle s,q\rangle$ if and only if $p \in \mathcal{P}os(t)$ and either*

1. *$root(t|_p) = f^b_{nil}$, $s = t$ and $p = q.i$ for some i; or*
2. *$t|_p = f^b_{i:L}(t_1,\ldots,t_k)$, $i > 0$, $s = t[f^b_L(t_1,\ldots,t_k)]_p$ and $q = p.i$; or*
3. *$t|_p = f^b_{-i:L}(t_1,\ldots,t_k)$, $i > 0$, $s = t[f^b_L(t_1,\ldots,up(t_i),\ldots,t_k)]_p$ and $q = p$; or*
4. *$t|_p = f^b_{0:L}(t_1,\ldots,t_k)$, $s = t[t']_p$, $q = p$ where t' is a term such that*
   - *(a) $t' = \theta(\varphi(r))$ if $DF_{\mathcal{R}}(erase(t|_p)) = \top$, $t|_p = \theta(l')$, $erase(l') = l$ and $l \to r \in \mathcal{R}$.*
   - *(b) $t' = f^b_L(t_1,\ldots,t_k)$ if either $DF_{\mathcal{R}}(erase(t|_p)) = \top$ and $erase(t|_p)$ is not a redex, or $DF_{\mathcal{R}}(erase(t|_p)) = \Lambda$, or $DF_{\mathcal{R}}(erase(t|_p)) = p' \neq \Lambda$ and $flag(t,p.p') = 0$;*
   - *(c) $t' = f^b_L(t_1,\ldots,t_i[up(s)]_{p''},\ldots,t_k)$ if $DF_{\mathcal{R}}(erase(t|_p)) = p' = i.p''$, $flag(t,p.p') = 1$, $\langle dn(t|_{p.p'}),\Lambda\rangle \to^!_\varphi \langle s,\Lambda\rangle$, and $DF_{\mathcal{R}}(erase(t|_p[s]_{p'})) = p'$;*
   - *(d) $t' = t|_p[up(s)]_{p'}$ if $DF_{\mathcal{R}}(erase(t|_p)) = p' \neq \Lambda$, $flag(t,p.p') = 1$, $\langle dn(t|_{p.p'}),\Lambda\rangle \to^!_\varphi \langle s,\Lambda\rangle$, and either $p' <_{lex} DF_{\mathcal{R}}(erase(t|_p[s]_{p'}))$ or $DF_{\mathcal{R}}(erase(t|_p[s]_{p'})) = \top$;*

Case 1 means that no more annotations are provided and the evaluation is completed. In case 2, a positive argument index is found and the evaluation goes down to the subterm at such argument. In case 3, the subterm at the argument indicated by the (absolute value of the) negative index is completely marked with on-demand flags. Case 4 considers the attempt to match the term against the left-hand sides of the program rules. Case 4.a applies if the considered (unlabelled) subterm is a redex (which is, then, contracted). If the subterm is not a redex, cases 4.b, 4.c and 4.d are considered possibly involving some on-demand evaluation steps on some subterm. The selected demanded position for term t (w.r.t. program $\mathcal{R}$) is denoted as $DF_{\mathcal{R}}(t)$ (eventually, symbol $\top$ is returned if t matches the left-hand side of some rule of the TRS). According to $DF_{\mathcal{R}}(t)$, case 4.b applies if no demanded evaluation is allowed (or required). Cases 4.c and 4.d apply if the on-demand evaluation of the subterm $t|_{p.p'}$ is required. In both cases, the evaluation is attempted; if it finishes, the evaluation of $t|_p$ continues according to the computed value.

Note that the computational description of on-demand strategy annotations above involves recursive steps. A single reduction step on a (labelled) term t may involve the application of more than one reduction step on subterms of t (as shown by steps 4(c) and 4(d)). 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 generally undecidable. Therefore, associated notions such as normal form (w.r.t. their reduction relation) are also undecidable.

Furthermore, as remarked in our introduction, the notion of demandness formalized in [NO01] needs to be refined to be entirely satisfactory in practice.

Example 3. Following the Example 2. The on-demand evaluation of `length'(from(0))` yields the following infinite sequence:

$$
\begin{array}{l}
\langle \mathtt{length'}^{0}_{(-1\ 0)}(\mathtt{from}^{0}_{(1\ 0)}(\mathtt{0}^{0}_{nil})), \Lambda\rangle \\
\quad \to_{\varphi}\ \langle \mathtt{length'}^{0}_{(0)}(\mathtt{from}^{1}_{(1\ 0)}(\mathtt{0}^{1}_{nil})), \Lambda\rangle\ \to_{\varphi}\ \langle \mathtt{length}^{0}_{(0)}(\mathtt{from}^{1}_{(1\ 0)}(\mathtt{0}^{1}_{nil})), \Lambda\rangle \\
\quad \to^{*}_{\varphi}\ \langle \mathtt{s}^{0}_{(1)}(\mathtt{length'}^{0}_{(-1\ 0)}(\mathtt{from}^{0}_{(1\ 0)}(\mathtt{s}^{0}_{(1)}(\mathtt{0}^{1}_{nil})))), \Lambda\rangle \\
\quad \to_{\varphi}\ \langle \mathtt{s}^{0}_{nil}(\mathtt{length'}^{0}_{(-1\ 0)}(\mathtt{from}^{0}_{(1\ 0)}(\mathtt{s}^{0}_{(1)}(\mathtt{0}^{1}_{nil})))), 1\rangle \\
\quad \to_{\varphi}\ \cdots
\end{array}
$$

Note that within the labelled term $\mathtt{length}^{0}_{(0)}(\mathtt{from}^{1}_{(1\ 0)}(\mathtt{0}^{1}_{nil}))$, the strategy does not recognize that the (activated) on-demand flags on `from` and `0` does *not* come from the local annotation for `length`. That is, the strategy does not maintain any kind of memory about the origin of on-demand flags. Hence, it (unnecessarily) evaluates the argument of `length`. Moreover, at this point, this evaluation does *not* correspond to the 'intended' meaning of the strategy annotations which the programmer may have in mind (since the specific annotation `(0)` for `length` forbids reductions on its argument).

On the other hand, the two existing definitions for the on-demand E-strategy (namely Nakamura and Ogata's [NO01] and Ogata and Futatsugi's [OF00]) sensibly differ. For instance, Nakamura and Ogata select a demanded position for evaluating a given term t by taking the maximum of all positions demanded on t by each rule of the TRS (according to the lexicographic ordering on positions). On the other hand, in Ogata and Futatsugi's selection of demanded positions, the ordering of the rules in the program is very important.

Example 4. Consider the `OBJ` program of Example 2 with the strategy `(1 0)` for `length` together with the function `geq` defined by the following module:

```
obj Ex3 is
  protecting Ex2 .
  sorts Bool .
  op true  : -> Bool .
  op false : -> Bool .
  op geq   : Nat Nat -> Nat    [strat (-1 -2 0)] .
  vars X Y : Nat .
  eq geq(s(X),s(Y)) = geq(X,Y) .
  eq geq(X,0) = true .
endo
```

Consider the expression `geq(length(from(0)),length(nil))`. According to Ogata and Futatsugi's definition of on-demand E-strategy, an infinite reduction sequence is started since position 1 is selected as demanded and, thus, its (non-terminating) evaluation attempted. However, Nakamura and Ogata's definition of on-demand E-strategy selects position 2 as demanded and, after the evaluation, the second rule is applied, thus obtaining `true`.

We claim that it is possible to provide a more practical framework for implementing and studying OBJ computations, which may integrate the most interesting features of modern evaluation strategies with on-demand syntactic annotations. This is made more precise from now on.

4 Improving rewriting under on-demand strategy annotations

As discussed at the end of the previous section, the existing operational models for arbitrary strategy annotations have a number of drawbacks: (1) the one-step reduction relation is, in general, undecidable; (2) the implementation of demandness by using negative annotations (via the marking of terms with flag 0 or flag 1) allows evaluation steps that shouldn't be possible, since (3) it does not properly keep track of the origin of the marks (loss of memory, see Example 3). Here, we want to introduce a further consideration which can be used for improving the previous definitions. Let us show it up by means of an example.

Example 5. Consider the OBJ program of Example 4 together with the following function `lt`:

```
obj Ex4 is
  protecting Ex3 .
  op lt : Nat Nat -> Nat    [strat (-1 -2 0)] .
  vars X Y : Nat .
  eq lt(0,s(Y)) = true .
  eq lt(s(X),s(Y)) = lt(X,Y) .
endo
```

Consider the expression t = `lt(length(from(0)),0)`, which is a *head-normal form* since no possible evaluation could lead the expression to match the left-hand side of a rule. Neither Nakamura and Ogata's nor Ogata and Futatsugi's formulations are able to avoid evaluations on t. Nevertheless, by exploiting the standard distinction between constructor and defined symbols of a signature in the presence of a TRS, it is easy to detect that no rule could ever be matched. Indeed, `0` is a constructor symbol in the input term t and, hence, it cannot be reduced for improving the matching of t against the left-hand side of the rule for `lt`. See [AFJV97,AL02,MR92] for a more detailed motivation and formal discussion of the use of these ideas for defining and using demand-driven strategies.

In the following, we propose a rectified and refined definition of the on-demand E-strategy which takes into account all previous considerations.

Given a E-strategy map φ, we use the signature[7] $\mathcal{F}^\sharp_\varphi = \cup\{f_{L_1|L_2}, \overline{f}_{L_1|L_2} \mid f \in \mathcal{F} \wedge L_1, L_2 \in \mathbb{L}_{ar(f)}.(L_1{+}{+}L_2 \sqsubseteq \varphi(f))\}$ and labelled variables $\mathcal{X}^\sharp_\varphi = \{x_{nil|nil} \mid x \in \mathcal{X}\}$ for marking ordinary terms $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ as terms $t \in \mathcal{T}(\mathcal{F}^\sharp_\varphi, \mathcal{X}^\sharp_\varphi)$. Overlining the root symbol of a subterm means that no evaluation is required for that subterm and the control goes back to the parent; the auxiliary list L_1 in the subscript $L_1 \mid L_2$ is interpreted as a kind of memory of previously considered annotations. We use $f^\sharp$ to denote f or $\overline{f}$ for a symbol $f \in \mathcal{F}$. We define the list of activable indices of a labelled symbol $f^\sharp_{L_1|L_2}$ as $\mathit{activable}(f^\sharp_{L_1|L_2}) = \begin{cases} L_1 \text{ if } L_1 \neq nil \\ L_2 \text{ if } L_1 = nil \end{cases}$. The operator φ is extended to a mapping from $\mathcal{T}(\mathcal{F},\mathcal{X})$ to $\mathcal{T}(\mathcal{F}^\sharp_\varphi, \mathcal{X}^\sharp_\varphi)$ as follows: $\varphi(t) = \begin{cases} x_{nil|nil} & \text{if } t = x \in \mathcal{X} \\ f_{nil|\varphi(f)}(\varphi(t_1), \ldots, \varphi(t_k)) & \text{if } t = f(t_1, \ldots, t_k) \end{cases}$. Also, the operator $\mathit{erase} : \mathcal{T}(\mathcal{F}^\sharp_\varphi, \mathcal{X}^\sharp_\varphi) \to \mathcal{T}(\mathcal{F},\mathcal{X})$ removes labellings from terms.

We define the set of demanded positions of t w.r.t. l (a lhs of a rule defining $\mathit{root}(t)$), i.e. the set of (positions of) maximal disagreeing subterms as:

$$DP_l(t) = \begin{cases} \mathit{minimal}_\leq(\mathcal{P}os_{\neq}(t,l)) & \text{if } \mathit{minimal}_\leq(\mathcal{P}os_{\neq}(t,l)) \subseteq \mathcal{P}os_{\mathcal{D}}(t) \\ \varnothing & \text{otherwise} \end{cases}$$

and the set of demanded positions of t w.r.t. TRS $\mathcal{R}$ as $DP_{\mathcal{R}}(t) = \cup\{DP_l(t) \mid l \to r \in \mathcal{R} \wedge \mathit{root}(t) = \mathit{root}(l)\}$. Note that the problem described in Example 5 is solved (along the lines of [MR92]) by restricting the attention to disagreeing positions that correspond to defined symbols (by using $\mathcal{P}os_{\mathcal{D}}(t)$).

Example 6. Continuing Example 5. Let t_1 = `lt(length(from(0)),0)`, t_2 = `lt(length(from(0)),length(nil))`, and t_3 = `lt(0,length(nil))`. We have $DP_{\mathcal{R}}(t_1) = \varnothing$, $DP_{\mathcal{R}}(t_2) = \{1,2\}$, and $DP_{\mathcal{R}}(t_3) = \{2\}$.

We define the set of positive positions of a term $s \in \mathcal{T}(\mathcal{F}^\sharp_\varphi, \mathcal{X}^\sharp_\varphi)$ as $\mathcal{P}os_P(s) = \{\Lambda\} \cup \{i.\mathcal{P}os_P(s|_i) \mid i > 0$ and $\mathit{activable}(\mathit{root}(s))$ contains $i\}$ and the set of activable positions as $\mathcal{P}os_A(s) = \{\Lambda\} \cup \{i.\mathcal{P}os_A(s|_i) \mid i > 0$ and $\mathit{activable}(\mathit{root}(s))$ contains i or $-i\}$. Given a term $s \in \mathcal{T}(\mathcal{F}^\sharp_\varphi, \mathcal{X}^\sharp_\varphi)$, the total ordering $\leq_s$ between activable positions of s is defined as (1) $\Lambda \leq_s p$ for all $p \in \mathcal{P}os_A(s)$; (2) if $i.p, i.q \in \mathcal{P}os_A(s)$ and $p \leq_{s|_i} q$, then $i.p \leq_s i.q$; and (3) if $i.p, j.q \in \mathcal{P}os_A(s)$, $i \neq j$, and i (or $-i$) appears before j (or $-j$) in $\mathit{activable}(\mathit{root}(s))$, then $i.p \leq_s j.q$. The ordering $\leq_s$ allows us to choose a position from the set of all demanded (and activable) positions in s, which is consistent with user's annotations (see

[7] The function ++ defines the concatenation of two sequences of integers.

$min_{\leq_s}$ below). We define the set $OD_{\mathcal{R}}(s)$ of on-demand positions of a term $s \in \mathcal{T}(\mathcal{F}^\sharp_\varphi, \mathcal{X}^\sharp_\varphi)$ w.r.t. TRS $\mathcal{R}$ as follows:

if $DP_{\mathcal{R}}(erase(s)) \cap \mathcal{P}os_P(s) \neq \varnothing$ or $DP_{\mathcal{R}}(erase(s)) \cap \mathcal{P}os_A(s) = \varnothing$
then $OD_{\mathcal{R}}(s) = \varnothing$ else $OD_{\mathcal{R}}(s) = \{min_{\leq_s}(DP_{\mathcal{R}}(erase(s)) \cap \mathcal{P}os_A(s))\}$

Example 7. Continuing Example 2. For $t_1 = \mathtt{length}_{nil|(0)}(\mathtt{from}_{nil|(1\ 0)}(\mathtt{0}_{nil|nil}))$, we have $DP_{\mathcal{R}}(erase(t_1)) = \{1\}$ but $OD_{\mathcal{R}}(t_1) = \varnothing$. Let us consider $t_2 = \mathtt{length}_{(-1)|(0)}(\mathtt{from}_{nil|(1\ 0)}(\mathtt{0}_{nil|nil}))$. We have $OD_{\mathcal{R}}(t_2) = \{1\}$. Finally, for $t_3 = \mathtt{length}_{(1)|(0)}(\mathtt{from}_{nil|(1\ 0)}(\mathtt{0}_{nil|nil}))$, we have $OD_{\mathcal{R}}(t_3) = \varnothing$.

Given a term $t \in \mathcal{T}(\mathcal{F}^\sharp_\varphi, \mathcal{X}^\sharp_\varphi)$ and position $p \in \mathcal{P}os(t)$, $mark(t,p)$ is the term s with all symbols above p (except the root) marked as non-evaluable, in symbols $\mathcal{P}os(s) = \mathcal{P}os(t)$ and $\forall q \in \mathcal{P}os(t)$, if $\Lambda < q < p$ and $root(t|_q) = f_{L_1|L_2}$, then $root(s|_q) = \overline{f_{L_1|L_2}}$, otherwise $root(s|_q) = root(t|_q)$.

Example 8. Consider the program of Example 1 and the term $t = \mathtt{2nd}_{(1)|(0)}(\mathtt{cons}_{(1\ -2)|nil}(\mathtt{0}_{nil|nil}, \mathtt{from}_{nil|(1\ 0)}(\mathtt{s}_{nil|(1)}(\mathtt{0}_{nil|nil}))))$. We have that $mark(t, 1.2) = \mathtt{2nd}_{(1)|(0)}(\overline{\mathtt{cons}}_{(1\ -2)|nil}(\mathtt{0}_{nil|nil}, \mathtt{from}_{nil|(1\ 0)}(\mathtt{s}_{nil|(1)}(\mathtt{0}_{nil|nil}))))$.

We formulate a binary relation $\overset{\sharp}{\to}_\varphi$ on the set $\mathcal{T}(\mathcal{F}^\sharp_\varphi, \mathcal{X}^\sharp_\varphi) \times \mathbb{N}^*_+$, such that a single reduction step on a (labelled) term t does not involve the application of recursive reduction steps on t. In the following definition, the symbol @ denotes appending an element at the end of a list.

Definition 2. *Given a TRS $\mathcal{R} = (\mathcal{F}, R)$ and an arbitrary E-strategy map φ for $\mathcal{F}$, $eval_\varphi : \mathcal{T}(\mathcal{F}, \mathcal{X}) \to \mathcal{P}(\mathcal{T}(\mathcal{F}, \mathcal{X}))$ is defined as $eval_\varphi(t) = \{erase(s) \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \mid \langle \varphi(t), \Lambda\rangle \overset{\sharp}{\to}^!_\varphi \langle s, \Lambda\rangle\}$. The binary relation $\overset{\sharp}{\to}_\varphi$ on $\mathcal{T}(\mathcal{F}^\sharp_\varphi, \mathcal{X}^\sharp_\varphi) \times \mathbb{N}^*_+$ is defined as follows: $\langle t, p\rangle \overset{\sharp}{\to}_\varphi \langle s, q\rangle$ if and only if $p \in \mathcal{P}os(t)$ and either*

1. *$t|_p = f_{L|nil}(t_1, \ldots, t_k)$, $s = t$ and $p = q.i$ for some i; or*
2. *$t|_p = f_{L_1|i:L_2}(t_1, \ldots, t_k)$, $i > 0$, $s = t[f_{L_1@i|L_2}(t_1, \ldots, t_k)]_p$ and $q = p.i$; or*
3. *$t|_p = f_{L_1|-i:L_2}(t_1, \ldots, t_k)$, $i > 0$, $s = t[f_{L_1@-i|L_2}(t_1, \ldots, t_k)]_p$ and $q = p$; or*
4. *$t|_p = f_{L_1|0:L_2}(t_1, \ldots, t_k) = \sigma(l')$, $erase(l') = l$, $s = t[\sigma(\varphi(r))]_p$ for some $l \to r \in R$ and substitution σ, $q = p$; or*
5. *$t|_p = f_{L_1|0:L_2}(t_1, \ldots, t_k)$, $erase(t|_p)$ is not a redex, $OD_{\mathcal{R}}(t|_p) = \varnothing$, $s = t[f_{L_1|L_2}(t_1, \ldots, t_k)]_p$, and $q = p$; or*
6. *$t|_p = f_{L_1|0:L_2}(t_1, \ldots, t_k)$, $erase(t|_p)$ is not a redex, $OD_{\mathcal{R}}(t|_p) = \{p'\}$, $s = t[mark(t|_p, p')]_p$, $q = p.p'$; or*
7. *$t|_p = \overline{f}_{L_1|L_2}(t_1, \ldots, t_k)$, $s = t[f_{L_1|L_2}(t_1, \ldots, t_k)]_p$ and $p = q.i$ for some i.*

Cases 1 and 2 of Definition 2 essentially correspond to cases 1 and 2 of Nakamura and Ogata's definition; that is, (1) no more annotations are provided and the evaluation is completed, or (2) a positive argument index is provided and the evaluation goes down to the subterm at such argument (note that the index is stored). Case 3 only stores the negative index for further use. Cases 4, 5, and 6 consider the attempt to match the term against the left-hand sides of the rules of the program. Case 4 applies if the considered (unlabelled) subterm is a redex (which is, then, contracted). If the subterm is not a redex, cases 5 and 6 are considered (possibly involving some on-demand evaluation). We use the lists of indices labelling the symbols for fixing the concrete positions on which we are able to allow on-demand evaluations; in particular, the first (memoizing) list is crucial for achieving this (by means of the function *activable* and the order $\leq_s$ used in the definition of the set $OD_{\mathcal{R}}(s)$ of on-demand positions of a term s). Case 5 applies if no demanded evaluation is allowed (or required). Case 6 applies if the on-demanded evaluation of the subterm $t|_{p.p'}$ is required. In this case, the symbols lying on the path from $t|_p$ to $t|_{p.p'}$ (excluding the ending ones) are overlined. Then, the evaluation process continues on $t|_{p.p'}$. Once the evaluation of $t|_{p.p'}$ has finished, the only possibility is the repeated (but possibly empty) application of steps issued according to the last case 7 which implements the return of the evaluation process back to position p (which originated the on-demand evaluation).

Example 9. Following the Examples 2 and 3. The on-demand evaluation of `length'(from(0))` under the refined on-demand strategy is the following:

$$\begin{array}{l}\langle \texttt{length'}_{nil|(-1\ 0)}(\texttt{from}_{nil|(1\ 0)}(\texttt{0}_{nil|nil})), \Lambda\rangle \\ \quad \stackrel{\sharp}{\to}_\varphi\ \langle \texttt{length'}_{(-1)|(0)}(\texttt{from}_{nil|(1\ 0)}(\texttt{0}_{nil|nil})), \Lambda\rangle \\ \quad \stackrel{\sharp}{\to}_\varphi\ \langle \texttt{length}_{nil|(0)}(\texttt{from}_{nil|(1\ 0)}(\texttt{0}_{nil|nil})), \Lambda\rangle \\ \quad \stackrel{\sharp}{\to}_\varphi\ \langle \texttt{length}_{nil|nil}(\texttt{from}_{nil|(1\ 0)}(\texttt{0}_{nil|nil})), \Lambda\rangle \end{array}$$

Therefore, we obtain `length(from(0))` as the computed value of the evaluation since, in the third step, the set of demanded positions $OD_{\mathcal{R}}$ is empty (see Example 7 above).

4.1 Properties of the refined on-demand strategy

The following theorem shows that, for positive strategy annotations, each reduction step with $\stackrel{\sharp}{\to}_\varphi$ exactly corresponds to the original Nagaya's $\stackrel{\mathbb{N}}{\to}_\varphi$ [Nag99,NO01]. For an E-strategy map φ and a term $t \in \mathcal{T}(\mathcal{F}^\sharp_\varphi, \mathcal{X}^\sharp_\varphi)$, we define *positive* : $\mathcal{T}(\mathcal{F}^\sharp_\varphi, \mathcal{X}^\sharp_\varphi) \to \mathcal{T}(\mathcal{F}^\mathbb{N}_\varphi, \mathcal{X}^\mathbb{N}_\varphi)$ as $positive(x_{nil|nil}) = x_{nil}$ for $x \in \mathcal{X}$ and $positive(f^\sharp_{L_1|L_2}(t_1, \ldots, t_n)) = f_{L'_2}(positive(t_1), \ldots, positive(t_n))$ where L'_2 is L_2 without negative indices.

Theorem 1. *Let $\mathcal{R}$ be a TRS and φ be a positive E-strategy map. Let $t, s \in \mathcal{T}(\mathcal{F}^\sharp_\varphi, \mathcal{X}^\sharp_\varphi)$ and $p \in \mathcal{P}os(t)$. Then, $\langle t, p\rangle \stackrel{\sharp}{\to}_\varphi \langle s, q\rangle$ if and only if $\langle positive(t), p\rangle \stackrel{\mathbb{N}}{\to}_\varphi \langle positive(s), q\rangle$.*

Sometimes, it is interesting to disregard from the ordering of indices in local strategies. Then, we use *replacement maps.* 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]. Let $M_\mathcal{F}$ be the set of all $\mathcal{F}$-maps. The ordering $\sqsubseteq$ on $M_\mathcal{F}$, the set of all $\mathcal{F}$-maps, is: $\mu \sqsubseteq \mu'$ if for all $f \in \mathcal{F}$, $\mu(f) \subseteq \mu'(f)$. Let $\mu^{can}_\mathcal{R}$ be the *canonical* replacement map, i.e. *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*, which is easily obtained from $\mathcal{R}$: $\forall f \in \mathcal{F}$, $i \in \{1, \ldots, ar(f)\}$, $i \in \mu_{\mathcal{R}}^{can}(f)$ iff $\exists l \in L(\mathcal{R}), p \in \mathcal{P}os_{\mathcal{F}}(l), (root(l|_p) = f \wedge p.i \in \mathcal{P}os_{\mathcal{F}}(l))$. 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 than or equally restrictive to $\mu_{\mathcal{R}}^{can}$.

Given an E-strategy map φ, we let μ^{φ} to be the following replacement map $\mu^{\varphi}(f) = \{|i| \mid i \in \varphi(f) \wedge i \neq 0\}$. We say that φ is a *canonical* E-strategy map (and, by abuse, we write $\varphi \in CM_{\mathcal{R}}$) if $\mu^{\varphi} \in CM_{\mathcal{R}}$. Given an E-strategy map, we let $\varphi^{\mathbb{N}}$ to be the E-strategy map obtained after taking away all negative indices for each symbol $f \in \mathcal{F}$. Note that $\varphi^{\mathbb{N}} \sqsubseteq \varphi$, for all E-strategy map φ. We show that, for E-strategy maps φ whose positive part $\varphi^{\mathbb{N}}$ is canonical, extra negative annotations can be completely disregarded. This means that negative annotations are only useful if the positive indices do not collect all indices in the canonical replacement map of the TRS.

Theorem 2. *Let* $\mathcal{R}$ *be a TRS and* φ *be an* E*-strategy map such that* $\varphi^{\mathbb{N}} \in CM_{\mathcal{R}}$. *Let* $t, s \in \mathcal{T}(\mathcal{F}_{\varphi}^{\sharp}, \mathcal{X}_{\varphi}^{\sharp})$ *and* $p \in \mathcal{P}os(t)$. *Then,* $\langle t, p\rangle \stackrel{\sharp}{\to}_{\varphi} \langle s, q\rangle$ *if and only if* $\langle positive(t), p\rangle \stackrel{\mathbb{N}}{\to}_{\varphi^{\mathbb{N}}} \langle positive(s), q\rangle$.

Example 9 shows that evaluating terms by using on-demand strategy annotations can lead to terms which are not even head-normal forms. The following result establishes conditions ensuring that the normal forms of our refined on-demand strategy are ordinary head–normal forms (w.r.t. the TRS). A TRS $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ 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 $1 \leq i \leq k$.

Theorem 3. *Let* $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ *be a left-linear CS,* $\varphi \in CM_{\mathcal{R}}$ *and* $\varphi(f)$ *ends in* 0 *for all* $f \in \mathcal{D}$. *Let* $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. *If* $s \in eval_{\varphi}(t)$, *then* s *is a head-normal form of* t.

Left-linearity and CS conditions cannot be dropped (see [Luc01] for examples that also apply to our setting).

Theorem 3 suggests the following *normalization via* φ*-normalization procedure* to obtain normal forms of a term t: given an E-strategy map φ and $s = f(s_1, \ldots, s_k) \in eval_{\varphi}(t)$, we continue the evaluation of t by (recursively) normalizing $s_1, \ldots, s_k$ using $eval_{\varphi}$. It is not difficult to see that confluence and φ-termination of the TRS guarantee that this procedure actually describes a normalizing strategy (see [Luc01,Luc02a]).

In the next section, we show that our on-demand strategy improves *lazy rewriting*, a popular demand-driven technique to perform lazy functional computations which inspired the development of local strategies in OBJ.

4.2 Comparison with lazy rewriting

In lazy rewriting [FKW00,Luc02b], reductions are issued on a different kind of *labelled terms*. Nodes (or positions) of a term t are labelled with e for the so-called *eager* positions or ℓ for so-called *lazy* ones: Let $\mathcal{F}$ be a signature and $\mathcal{L} = \{e, \ell\}$; then, $\mathcal{F} \times \mathcal{L}$ (or $\mathcal{F}_{\mathcal{L}}$) 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 $\langle f, \ell\rangle$. Labelled terms are terms in $\mathcal{T}(\mathcal{F}_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}})$. 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.

Given a replacement map $\mu \in M_{\mathcal{F}}$ and $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $label_\mu(s)$ denotes the following *intended* labelling of s: the topmost position Λ of $label_\mu(s)$ is 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.

Example 10. Consider the replacement map μ given by $\mu(\mathtt{2nd}) = \mu(\mathtt{from}) = \mu(\mathtt{cons}) = \mu(\mathtt{s}) = \{1\}$. Then, the labelling of $s = \mathtt{2nd(cons(0,from(s(0))))}$ is $t = label_\mu(s) = \mathtt{2nd}^e\mathtt{(cons}^e\mathtt{(0}^e\mathtt{,from}^\ell\mathtt{(s}^e\mathtt{(0}^e\mathtt{))))}$. Thus, $\Lambda, 1, 1.1, 1.2.1$, and 1.2.1.1 are eager positions; 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 labels.

As remarked 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 those positions which are always reachable from the root of the term via a path of eager positions. For instance, positions $\Lambda, 1$, and 1.1 are active in term t of Example 10; positions 1.2.1 and 1.2.1.1 are eager but *not* active, since position 1.2 below is lazy in t. Let $\mathcal{A}ct(t)$ be the set of active positions of a labelled term $t \in \mathcal{T}(\mathcal{F}_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}})$. In lazy rewriting, *the set of active nodes may increase as reduction of labelled terms proceeds.* Each lazy reduction step on labelled terms may have two different effects:

1. changing the "activation" status of a given position within a term, or
2. performing a rewriting step (always on an active position).

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 3 (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 $l \in \mathcal{X}$, or $l = f(l_1, \ldots, l_k)$, $s = f^e(s_1, \ldots, s_k)$ and, for all $i \in \{1, \ldots, k\}$, if $p.i$ is eager, then l_i matches modulo laziness s_i. If position $p.i$ is lazy and $l_i \notin \mathcal{X}$, then position $p.i$ is called* essential.

If p is an active position in $t \in \mathcal{T}(\mathcal{F}_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}})$ and $l \to r$ is a rewrite rule of a left-linear TRS $\mathcal{R}$ such that l matches modulo laziness $t|_p$ giving rise to an essential position q of t and $t|_q = f^\ell(t_1, \ldots, t_k)$, then we write $t \xrightarrow{\mathsf{A}} t[f^e(t_1, \ldots, t_k)]_q$ for denoting the activation of position p.

Lazy rewriting reduces active positions: let p be an active position of $t \in \mathcal{T}(\mathcal{F}_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}})$, $u = t|_p$ and $l \to r$ be a rule of a left-linear TRS $\mathcal{R}$ such that l

matches $erase(u)$ using substitution σ. Then, $t \stackrel{\mathsf{R}}{\to}_\mu s$, where s is obtained from t by replacing $t|_p$ in t by $label_\mu(r)$ with all its variables instantiated according to σ but preserving the label of the variable in $label_\mu(r)$, see [Luc02b] for a formal definition.

Example 11. Consider the program of Example 1 (as a TRS) and the term t of Example 10. The reduction step that corresponds to such term is: $\mathtt{2nd}^e\mathtt{(cons}^e\mathtt{(0}^e\mathtt{,from}^\ell\mathtt{(s}^e\mathtt{(0}^e\mathtt{))))} \stackrel{\mathsf{A}}{\to} \mathtt{2nd}^e\mathtt{(cons}^e\mathtt{(0}^e\mathtt{,from}^e\mathtt{(s}^e\mathtt{(0}^e\mathtt{))))} \stackrel{\mathsf{R}}{\to}_\mu$ $\mathtt{2nd}^e\mathtt{(cons}^e\mathtt{(0}^e\mathtt{,cons}^e\mathtt{(s}^e\mathtt{(0}^e\mathtt{):}^e\mathtt{from}^\ell\mathtt{(s}^e\mathtt{(s}^e\mathtt{(0}^e\mathtt{))))))}$. Note that term $\mathtt{2nd}^e\mathtt{(cons}^e\mathtt{(0}^e\mathtt{,cons}^e\mathtt{(s}^e\mathtt{(0}^e\mathtt{):}^e\mathtt{from}^\ell\mathtt{(s}^e\mathtt{(s}^e\mathtt{(0}^e\mathtt{))))))}$ is a $\stackrel{\mathsf{A}}{\to}$-normal form.

The lazy term rewriting relation on labelled *terms* (LR) is $\stackrel{\mathsf{LR}}{\to}_\mu = \stackrel{\mathsf{A}}{\to} \cup \stackrel{\mathsf{R}}{\to}_\mu$ and the evaluation $LR\text{-}eval_\mu(t)$ of a term $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ using LR is given by $LR\text{-}eval_\mu(t) = \{erase(s) \in \mathcal{T}(\mathcal{F},\mathcal{X}) \mid label_\mu(t) \stackrel{\mathsf{LR}}{\longrightarrow}^!_\mu s\}$. We say that a TRS is $LR(\mu)$-terminating if, for all $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$, there is no infinite $\stackrel{\mathsf{LR}}{\to}_\mu$-rewrite sequence starting from $label_\mu(t)$ [Luc02b].

We show that each evaluation step of our refined on-demand strategy is included into some evaluation steps of lazy rewriting. Given a term $t \in \mathcal{T}(\mathcal{F}^\sharp_\varphi,\mathcal{X}^\sharp_\varphi)$ and $p \in \mathcal{P}os(t)$, we translate the labeling of terms in $\mathcal{T}(\mathcal{F}^\sharp_\varphi,\mathcal{X}^\sharp_\varphi)$ into the labeling of $\mathcal{T}(\mathcal{F}_\mathcal{L},\mathcal{X}_\mathcal{L})$ as follows: $lazy^p_\varphi(t) = \rho''_p(\rho'_t(label_{\mu^{\varphi^{\mathbb{N}}}}(erase(t))))$ where (1) $\rho'_t(f^b(t_1,\ldots,t_n)) = f^e(\rho'_{s_1}(t_1),\ldots,\rho'_{s_n}(t_n))$ if $t = \overline{f}_{L_1|L_2}(s_1,\ldots,s_n)$, (2) $\rho'_t(f^b(t_1,\ldots,t_n)) = f^b(\rho'_{s_1}(t_1),\ldots,\rho'_{s_n}(t_n))$ if $t = f_{L_1|L_2}(s_1,\ldots,s_n)$, and (3) $\rho''_p(s) = s[f^e(s_1,\ldots,s_k)]_p$ for $s|_p = f^b(s_1,\ldots,s_k)$. We define the ordering $\leq_{lazy}$ between terms $\mathcal{T}(\mathcal{F}_\mathcal{L},\mathcal{X}_\mathcal{L})$ by extending the ordering $f^e \leq_{lazy} f^e$ and $f^\ell \leq_{lazy} f^e$, for all $f \in \mathcal{F}$, to terms.

The following theorem shows that each evaluation step of our refined on-demand strategy corresponds to some evaluation steps of lazy rewriting. Also, it shows that lazy rewriting (potentially) activates as many symbols (within a term) as our strategy (we use the ordering $\leq_{lazy}$ for expressing this fact).

Theorem 4. *Let $\mathcal{R}$ be a left-linear TRS and φ be an E-strategy map. Let $t \in \mathcal{T}(\mathcal{F}^\sharp_\varphi,\mathcal{X}^\sharp_\varphi)$, $p \in \mathcal{P}os(t)$ and $\mu = \mu^{\varphi^{\mathbb{N}}}$. If $\langle t,p\rangle \stackrel{\sharp}{\to}_\varphi \langle s,q\rangle$ and $p \in \mathcal{A}ct(lazy^p_\varphi(t))$, then $q \in \mathcal{A}ct(lazy^q_\varphi(s))$, and $lazy^p_\varphi(t) \stackrel{\mathsf{LR}}{\longrightarrow}^*_\mu s'$ for $s' \in \mathcal{T}(\mathcal{F}^\sharp_\varphi,\mathcal{X}^\sharp_\varphi)$ such that $lazy^q_\varphi(s) \leq_{lazy} s'$.*

In general, our strategy is strictly more restrictive than LR as the following example shows.

Example 12. Consider the TRS $\mathcal{R}$ and the E-strategy map φ of program of Example 2. In Example 14 below, we prove that $\mathcal{R}$ is φ–terminating. However, LR enters an infinite reduction sequence starting with the expression $label_{\mu^\varphi}\mathtt{(length'(from(0))}$):

$$\mathtt{length'}^e\mathtt{(from}^\ell\mathtt{(0}^e\mathtt{))} \stackrel{\mathsf{R}}{\to}_\mu \mathtt{length}^e\mathtt{(from}^\ell\mathtt{(0}^e\mathtt{))} \stackrel{\mathsf{A}}{\to} \mathtt{length}^e\mathtt{(from}^e\mathtt{(0}^e\mathtt{))}$$
$$\stackrel{\mathsf{R}}{\to}_\mu \mathtt{length}^e\mathtt{(cons}^e\mathtt{(0}^e\mathtt{,from}^\ell\mathtt{(s}^e\mathtt{(0}^e\mathtt{))))} \stackrel{\mathsf{R}}{\to}_\mu \mathtt{s}^e\mathtt{(length'}^e\mathtt{(from}^\ell\mathtt{(s}^e\mathtt{(0}^e\mathtt{))))} \stackrel{\mathsf{LR}}{\to}_\mu \cdots$$

Note that if no positive annotation is provided for an argument of a symbol, *LR* freely demands on this argument. Then, in contrast to φ (where $\varphi(\texttt{length}) = (0)$) *LR* can evaluate position 1 in the expression `length(from(0))`.

In the following section, we formulate methods for proving termination of our on-demand strategy.

5 Proving termination of programs with negative annotations by transformation

In [Luc02b] a method for proving termination of *LR* as termination of *context-sensitive rewriting* (*CSR* [Luc98]) is described. In contrast to *LR*, context-sensitive rewriting forbids *any* reduction on the arguments not included into $\mu(f)$ for a given function call $f(t_1, \ldots, t_k)$. A TRS $\mathcal{R}$ is μ-terminating if the context-sensitive rewrite relation associated to $\mathcal{R}$ and μ is terminating. The idea of the aforementioned method is simple: given a TRS $\mathcal{R}$ and a replacement map μ, a new TRS $\mathcal{R}'$ and replacement map μ' is obtained in such a way that μ'-termination of $\mathcal{R}'$ implies $LR(\mu)$-termination of $\mathcal{R}$. Fortunately, there is a number of different techniques for proving termination of *CSR* (see [GM02,Luc02c] for recent surveys) which provide a formal framework for proving termination of lazy rewriting. A simple modification of such transformation provides a sound technique for proving φ-termination of TRSs for arbitrary strategy annotations φ by taking into account that only those symbols which have associated a negative index may be activated by demandness. Here, as in [Luc01,Luc02b], by φ-termination of a TRS $\mathcal{R}$ we mean the absence of infinite $\stackrel{\sharp}{\to}_{\varphi}$-sequences of terms starting from $\langle \varphi(t), \Lambda \rangle$.

As for the transformation in [Luc02b], the idea is to encode the demandness information expressed by the rules of the TRS $\mathcal{R}$ together with the (negative) annotations of the E-strategy map φ as new symbols and rules (together with the appropriate modification/extension of φ) in such a way that φ-termination is preserved in the new TRS and E-strategy map, but the negative indices are finally suppressed (by removing from the lhs of the rules the parts that introduce on-demand computations). We iterate on these basic transformation steps until obtaining a canonical E-strategy map. In this case, we can stop the transformation and use the existing methods for proving termination of *CSR*. Let φ be an arbitrary E-strategy map. Given $l \to r \in R$ and $p \in \mathcal{P}os(l)$, we let

$$\mathcal{I}(l,p) = \{ i > 0 \mid p.i \in \mathcal{P}os_{\mathcal{F}}(l) \text{ and } -i \in \varphi(root(l|_p)) \}$$

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 $\varphi^\diamond$ 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 $ar(f_j) = ar(f)$, and

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

where $l'_j = l[f_j(l|_{p.1}, \ldots, l|_{p.k})]_p$ if $ar(f) = k$, and x is a new variable. We let $\varphi^\diamond(f_j) = (i_j\ 0)$ for $1 \leq j \leq n$ and $f \in \mathcal{D}$, $\varphi^\diamond(f_j) = (i_j)$ for $1 \leq j \leq n$ and

$f \in \mathcal{C}$, and $\varphi^\diamond(g) = \varphi(g)$ for all $g \in \mathcal{F}.(g \neq f)$. Moreover, we let $\varphi^\diamond(f) = \varphi^{\mathbb{N}}(f)$ if $\mu^{can}_{\mathcal{R}^\diamond}(f) \subseteq \mu^{\varphi^{\mathbb{N}}}(f)$, or $\varphi^\diamond(f) = \varphi(f)$ otherwise. The transformation proceeds in this way (starting from $\mathcal{R}^\diamond$ and $\mu^\diamond$) until obtaining $\mathcal{R}^\natural = (\mathcal{F}^\natural, R^\natural)$ and $\varphi^\natural$ such that $\varphi^\natural = \varphi^{\natural^{\mathbb{N}}}$. If $\varphi = \varphi^{\mathbb{N}}$, then $\mathcal{R}^\natural = \mathcal{R}$ and $\varphi^\natural = \varphi$.

Finally, we can state a sufficient condition for φ-termination as termination of *CSR* for the transformed TRS.

Theorem 5. *Let $\mathcal{R}$ be a TRS, φ be an E-strategy map. If $\mathcal{R}^\natural$ is $\mu^{\varphi^\natural}$-terminating, then $\mathcal{R}$ is φ-terminating.*

Example 13. Consider the TRS $\mathcal{R}$ associated to Example 1 but where $\varphi(\texttt{cons}) = (1\ \ -2)$. Then, $\mathcal{R}^\natural$ is:

```
2nd(cons'(x,cons(y,z)))  → y
2nd(cons(x,y))           → 2nd(cons'(x,y))
from(x)                  → cons(x,from(s(x)))
```

and $\varphi^\natural$ is given by $\varphi^\natural(\texttt{2nd}) = \varphi^\natural(\texttt{from}) = (1\ 0)$, $\varphi^\natural(\texttt{cons}) = (1)$, and $\varphi^\natural(\texttt{cons'}) = (2)$. The $\mu^{\varphi^\natural}$-termination of $\mathcal{R}^\natural$ is proved by using Zantema's transformation for proving termination of *CSR* [Zan97]: the TRS

```
2nd(cons'(x,cons(y,z)))  → y
2nd(cons(x,y))           → 2nd(cons'(x,activate(y)))
from(x)                  → cons(x,from'(s(x)))
activate(from'(x))       → from(x)
from(x)                  → from'(x)
activate(x)              → x
```

which is obtained in this way (where `activate` and `from'` are new symbols introduced by Zantema's transformation) is terminating[8].

Example 14. Consider the TRS $\mathcal{R}$ and the E-strategy map φ that correspond to the OBJ program of Example 2. Our transformation returns the original TRS, i.e., $\mathcal{R}^\natural$ is:

```
from(x)             → cons(x,from(s(x)))
length(nil)         → 0
length(cons(x,z))   → s(length'(z))
length'(z)          → length(z)
```

together with the simplified E-strategy $\varphi^\natural(\texttt{s}) = \varphi^\natural(\texttt{cons}) = (1)$, $\varphi^\natural(\texttt{from}) = (1\ 0)$ and $\varphi^\natural(\texttt{length}) = \varphi^\natural(\texttt{length'}) = (0)$. The $\mu^{\varphi^\natural}$-termination of $\mathcal{R}$ can be automatically proved by splitting up the rules of the program into two modules $\mathcal{R}_1$ (consisting of the rule for `from`) and $\mathcal{R}_2$ (consisting of the rules for `length` and `length'`). The $\mu^{\varphi^\natural}$-termination of $\mathcal{R}_1$ can easily be proved by using Zantema's transformation (in fact, the proof can be extracted from that of Example 13). The $\mu^{\varphi^\natural}$-termination of $\mathcal{R}_2$ is easily proved: in fact, $\mathcal{R}_2$ can be proved terminating (regarding standard rewriting) by using a polynomial ordering[9]. Now, $\mu^{\varphi^\natural}$-termination of $\mathcal{R}$ follows by applying the modularity results of [GL02].

[8] Using the C*i*ME 2.0 system (available at `http://cime.lri.fr`).

[9] C*i*ME 2.0 can also be used for achieving this proof.

6 Conclusions

We have provided a suitable extension of the positive E-evaluation strategy of OBJ-like languages to general (positive as well as negative) annotations. Such an extension is conservative, i.e., programs which only use positive strategy annotations and that are executed under our strategy behave exactly as if they were executed under the standard OBJ evaluation strategy (Theorems 1 and 2). The main contributions of the paper are (a) the definition of a rectified and well-defined approach to demandness via E-strategies (see Examples 2, 3, 9, and 12 for motivation regarding some of the problems detected on the previous proposals), (b) the better computational properties associated to such a new on-demand strategy (Theorems 3 and 4), (c) the definition of techniques for analyzing computational properties such as termination (Theorem 5), and that (d) our approach is (hopefully) better suited for implementation. Our on-demand strategy also improves *lazy rewriting*, a popular, demand-driven technique to perform lazy functional computations which inspired the development of on-demand strategies in OBJ. We conjecture that the *on-demand rewriting* (*ODR*) [Luc01], which extends the context sensitive rewriting of [Luc98] by also considering "negative annotations", can also be compared with our refined on-demand strategy, whereas it does not directly apply to OBJ nor it is comparable to *LR*.

There are still some open problems regarding completeness of our refined on-demand strategy which are out of the scope of this paper (see [AEL02] for a thorough discussion and proposal of implementable solutions when dealing with programs that only use positive annotations).

Acknowledgements. We thank the anonymous referees for their helpful remarks.

References

[AEL02] M. Alpuente, S. Escobar, and S. Lucas. Correct and complete (positive) strategy annotations for OBJ. Electronic Notes in Theoretical Computer Science, volume 71. Elsevier Sciences, to appear 2002.

[AFJV97] M. Alpuente, M. Falaschi, P. Julián, and G. Vidal. Specialization of Lazy Functional Logic Programs. *Sigplan Notices*, 32(12):151–162, ACM Press, New York, 1997.

[AL02] S. Antoy and S. Lucas. Demandness in rewriting and narrowing. Electronic Notes in Theoretical Computer Science, volume 76. Elsevier Sciences, to appear 2002.

[BN98] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[CELM96] M. Clavel, S. Eker, P. Lincoln, and J. Meseguer. Principles of Maude. Electronic Notes in Theoretical Computer Science, volume 4. Elsevier Sciences, 1996.

[Eke98] S. Eker. Term rewriting with operator evaluation strategies. Electronic Notes in Theoretical Computer Science, volume 15. Elsevier Sciences, 1998.

[FGJM85] K. Futatsugi, J. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Proc. 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.

[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*, Pittsburg, USA, 2002. ACM Press, New York. To appear, 2002.

[GM02] Jürgen Giesl and Aart Middeldorp. Transformation techniques for context-sensitive rewrite systems. Aachener Informatik-Berichte (AIBs) 2002-02, RWTH Aachen, 2002.

[GWM+00] J. A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.P. Jouannaud. Introducing OBJ. In Joseph A. Goguen and Grant Malcolm, editors, *Software Engineering with OBJ: algebraic specification in action*. Kluwer, 2000.

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

[Luc01] S. Lucas. Termination of on-demand rewriting and termination of OBJ programs. In Harald Sondergaard, editor, *Proc. 3rd International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP'01)*, pages 82–93, Firenze, Italy, September 2001. ACM Press, New York.

[Luc02a] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, to appear, 2002.

[Luc02b] S. Lucas. Lazy rewriting and context-sensitive rewriting. Electronic Notes in Theoretical Computer Science, volume 64. Elsevier Sciences, to appear, 2002.

[Luc02c] S. Lucas. Termination of (canonical) context-sensitive rewriting. In Sophie Tison, editor, *Proc. 13th International Conference on Rewriting Techniques and Applications, RTA'02*, LNCS 2378:296-310, Springer-Verlag, Berlin, 2002.

[MR92] J.J. Moreno-Navarro and M. Rodríguez-Artalejo. Logic Programming with Functions and Predicates: the Language BABEL. *Journal of Logic Programming*, 12(3):191–223, 1992.

[Nag99] T. Nagaya. *Reduction Strategies for Term Rewriting Systems, PhD Thesis.* School of Information Science, Japan Advanced Institute of Science and Technology, 1999.

[NO01] M. Nakamura and K. Ogata. The evaluation strategy for head normal form with and without on-demand flags. Electronic Notes in Theoretical Computer Science, volume 36. Elsevier Sciences, 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.

[Zan97] 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.