

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN
UNIVERSIDAD POLITÉCNICA DE VALENCIA

P.O. Box: 22012 E-46071 Valencia (SPAIN)



Informe Técnico / Technical Report

Ref. No.: DSIC-II/20/03

Pages: 47

Title: Termination of programs with strategy annotations

Author(s): Salvador Lucas

Date: 11/09/03

Keywords: CafeOBJ, Maude, OBJ, program annotations, programming languages, programmable strategies, term rewriting, termination.

Termination of programs with strategy annotations*

Salvador Lucas

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

Abstract

Strategy annotations have been used in a number of programming languages and rewriting-based systems (e.g., OBJ2, OBJ3, CafeOBJ, Maude, JITty, μ CRL, ...) for improving the termination behavior and avoiding useless computations. Regarding the ability of these annotations to improve the termination behavior of programs, the possibility of forbidding reductions on some arguments of function symbols play a crucial role. From an abstract point of view, in term rewriting this feature has been modeled as the so-called context-sensitive rewriting (*CSR*). We show that *CSR* provides a framework for analyzing termination of programs running under control of strategy annotations. Thus, *CSR* and its associated techniques for ensuring termination provide a useful tool for the programmers and designers of software systems aimed at supporting the practical use of such programming languages and rewriting-based systems.

Keywords: CafeOBJ, Maude, OBJ, program annotations, programming languages, programmable strategies, term rewriting, termination.

1 Introduction

Most computational systems whose operational principle is based on reducing expressions (e.g., functional, algebraic, and equational programming languages as well as theorem provers based on rewriting techniques) incorporate a predefined reduction strategy which is used to break down the non-determinism which is inherent to reduction relations. Thus, every program will be executed according to that strategy. Eventually, this can rise problems, as each kind of strategy only behaves properly (i.e., it is normalizing, optimal, etc.) for particular classes of programs.

For this reason, the designers of programming languages have developed some features and language constructs aimed at giving the user more flexible control of the program execution. For instance, *syntactic annotations*

*This paper is a revised and extended version of [Luc01b]. Work partially supported by CICYT TIC2001-2705-C03-01.

(which are associated to arguments of symbols) have been used in programming languages such as Clean [NSEP92], Haskell [HPW92], Lisp [McC60], Maude [CDEL⁺02, CDEL⁺03], OBJ2 [FGJM85], OBJ3 [GWMFJ00], CafeOBJ [FN97], etc., to improve the termination and efficiency of computations. Lazy languages (e.g., Haskell, Clean) interpret them as *strictness annotations* in order to become ‘more eager’ and efficient. Eager languages (e.g., Lisp, Maude, OBJ2, OBJ3, CafeOBJ) use them as *replacement restrictions* to become ‘more lazy’ thus (hopefully) avoiding nontermination. We focus on these ones. For instance, [FW76] studied implementations of Lisp where the list constructor operator (*cons*) did not evaluate its arguments during certain stages of the computation. Also, algebraic languages, such as Maude, OBJ2, OBJ3, and CafeOBJ, admit the *explicit* specification of a particular class of *strategy annotations*, which (basically) are lists of integers associated to function symbols which specify the ordering in which the arguments are (eventually) evaluated in function calls. This very simple strategy language provides quite a powerful way to control the program execution. Due to its simplicity, such strategy annotations also provide a simple interface for understanding and eventually modifying the execution of programs.

Strategy annotations have been given different formats and operational interpretations. We consider two kinds of strategy annotations: *E*-strategies and Van de Pol’s (or *Just-in-time*) strategies.

1. *E*-strategies [Eke98, GWMFJ00], which are lists of non-negative integers associated to function symbols to specify the ordering in which the arguments are (eventually) evaluated in function calls: when considering a function call $f(t_1, \dots, t_k)$, only the arguments whose indices are present as *positive* integers in the local strategy $(i_1 \dots i_n)$ for f are evaluated (following the specified ordering). If 0 is found, a reduction step on the whole term $f(t_1, \dots, t_k)$ is attempted. *E*-strategies are used in, e.g., Maude, OBJ2, OBJ3, and CafeOBJ¹. We especially note that, even when the user provides no annotation for a given symbol, the (CafeOBJ, Maude, OBJ2, OBJ3) interpreter automatically assigns a *default* strategy annotation. For instance, OBJ2 (and OBJ3) *built-in* conditional operator has the following (implicit) strategy annotation (see [FGJM85], Section 4.4 and [GWMFJ00], Section 2.4.4 and Appendix D.3:

```
op if_then_else_fi : Bool Univ Univ -> Univ [strat (1 0)]
```

which says to evaluate the first argument until it is reduced, and then apply rules at the top (indicated by ‘0’). Note the use of the ‘polymorphic’ sort `Univ` (`Universal` in [GWMFJ00], Appendix D.3) for the second and third arguments.

The presence of such ‘true’ *replacement restrictions* (e.g., by forbidding replacements in the second and third arguments of `if_then_else_fi`) is

¹CafeOBJ also permits negative integers in *E*-strategies. Their use is intended to capture the notion of *demanded evaluation* [OF00, NO01]. See [AEGL02] for a discussion about the use and properties of these strategy annotations. The results in this papers applies to CafeOBJ programs whose *E*-strategies do not include negative indices.

often invoked to justify that Maude and *OBJ* programs are able to *avoid nontermination* ([GWMFJ00], Section 2.4.4).

Example 1 Consider the following OBJ3 program which implements the subtraction of natural numbers by comparing their values and using the first argument to recursively proceed in that way:

```
obj DIFF is
  sort Nat .
  op 0 : -> Nat .
  ops p s : Nat -> Nat .
  op _<=_ : Nat Nat -> Bool .
  op _-_ : Nat Nat -> Nat .
  vars M N : Nat .
  eq p(s(N)) = N .
  eq 0 <= N = true .
  eq s(M) <= 0 = false .
  eq s(M) <= s(N) = M <= N .
  eq M - N = if M <= N then 0 else s(p(M) - N) fi .
endo
```

where `Bool`, `true`, `false`, and `if_then_else_fi` are predefined in the ‘prelude’ of OBJ3. Viewed as a term rewriting system, this program is not terminating (due to the rule defining ‘-’). The program does not specify any *E*-strategy, but due to the implicit strategy annotation (1 0) for the built-in conditional operator, the program is terminating when executed using an OBJ3 interpreter (this can be formally proved, see Example 16 below). In contrast, the execution of the program using a Maude interpreter is not terminating (we use version 2.0 of Maude interpreter²):

```
Maude> red p(0) - 0 .
reduce in DIFF : p(0) - 0 .
Segmentation fault (core dumped)
```

As we show in Example 9 below, this is due to the following *E*-strategy which is (implicitly) used in Maude (see [CDEL⁺02, page 195]):

```
op if_then_else_fi : Bool Nat Nat -> Nat [strat (1 0 2 3 0)]
```

This example shows that having a precise knowledge of the real *E*-strategy which is used to run the program is essential to understand its execution. Example 1 also shows that the use of strategy annotations can benefit the termination behavior. The following example shows that, when looking

²Available at <http://maude.cs.uiuc.edu/download>.

for a better termination behavior, other semantic properties can also be altered. In particular, the evaluation of a given expression could stop before reaching a normal form.

Example 2 *Consider the following OBJ program:*

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

We need the explicit strategy annotations for symbols `s` and `cons` to achieve a terminating behavior of the program (proved in Example 11 below). The evaluation of `fst(s(0),from(add(0,0)))` in Maude yields:

```
Maude> red fst(s(0),from(add(0,0))) .
reduce in FIRST_FROM_LENGTH : fst(s(0), from(add(0, 0))) .
rewrites: 3 in 0ms cpu (0ms real) (~ rewrites/second)
result LNat: cons(0, fst(0, from(s(0))))
```

Example 2 shows that the use of E -strategies that forbid reductions on some arguments can jeopardize correctness and completeness of computations (regarding, e.g., the usual semantics of normal forms). This problem is not addressed here and we refer the reader to [Luc03b] for a discussion on these issues. Also, the evaluation of the same expression can yield different values when using different interpreters with the *same* program!

Example 3 *In contrast to the computation obtained in Example 2, the evaluation using the OBJ3 interpreter yields:*

```
OBJ> red fst(s(0),from(add(0,0))) .
reduce in FIRST_FROM_LENGTH : fst(s(0),from(add(0,0)))
rewrites: 3
result Nat: cons(0,fst(0,from(s(add(0,0))))))
```

This is because the default E-strategy for symbol from in Maude is (1 0) whereas the default E-strategy in OBJ3 is (0 1 0), see Examples 8 and 10 below.

This phenomenon can be thought of as a problem of ‘portability’ of OBJ programs between different platforms (e.g., OBJ3 and Maude interpreters).

2. Van de Pol’s strategies [Pol01] are a refinement of the *E*-strategies: instead of using ‘0’ to indicate that the application of *rules* must be attempted in some stage of computation, van de Pol permits to exactly specify what a *rule* should eventually be applied. Van de Pol’s style of strategy annotations is the basis of the JITty system [Pol02] which has been integrated in the μ CRL tool set [BFGL⁺01].

Example 4 *The following JITty program*

```
signature
  true(0)      if(3)
  false(0)
rules
  if1([x,y], if(true,x,y), x)
  if2([x,y], if(false,x,y), y)
  if3([x,y], if(x,y,y), y)
strategies
  if([1,if1,if2,2,3,if3])
end
```

indicates that, after evaluating the first argument of a call to if, only the first and second rules (labelled if1 and if2, respectively) can be attempted. If they fail, then the second and third arguments must be evaluated and finally only the third rule (with label if3) is considered. As we show below (see Section 5), this feature makes a sharp difference regarding E-strategies.

Despite its simplicity and their use in a number of programming languages and systems, computational properties of programs using strategy annotations

have hardly been studied before. Thus, there is a lack of formal techniques to analyze how a particular (explicit or implicit) choice of strategy annotations modifies the behavior of such programs. Since term rewriting systems (TRSs [BN98]) provide a suitable computational model for programs written in these programming languages, in this paper we investigate termination of rewriting computations controlled by strategy annotations. The main achievements of this paper are the following:

Foundational aspects. We show that context-sensitive rewriting (*CSR* [Luc98]), a simple restriction of rewriting that only permits reductions on selected arguments of functions, provides a suitable framework for describing and analyzing computations with programs using strategy annotations. In *CSR*, we consider TRSs \mathcal{R} supplied with a *replacement map* μ , i.e., a mapping from function symbols f to sets of positive integers satisfying that $\mu(f) \subseteq \{1, \dots, k\}$ if k is the arity of f . Programs with strategy annotations can be modeled as rewrite systems supplied with a replacement map. Computations of such programs are context-sensitive rewritings of the corresponding TRS.

Termination. We formally relate termination of programs with strategy annotations and termination of *CSR*. In general, proving termination of *CSR* suffices to ensure termination of the corresponding program. For the *E*-strategies, we even give conditions ensuring that termination of *innermost CSR* provides a correct and complete characterization of termination of *Maude* and *OBJ** programs. We also show that, in spite of their syntactic similarity, termination of *E*-strategies and termination of Van de Pol's strategies are quite different problems.

Proving termination of programs with strategy annotations. Termination of *CSR* has been studied in a number of papers [BLR02, FR99, GL02a, GL02b, GM99, GM03a, GM03b, Luc96, Luc01b, Luc03a, Luc02c, SX98, Zan97]. We, then, have now a number of techniques for proving termination of programs using strategy annotations. In particular, this is the case for *Maude* and *OBJ** programs whose termination analysis can now be more accurately addressed by using the previous techniques. We discuss the use of such techniques to develop practical tools for proving termination of programs with strategy annotations.

Section 2 gives some preliminary definitions. Section 3 introduces *CSR*. Section 4 introduces *E*-strategies and connects termination of rewriting under *E*-strategies and (innermost) termination of *CSR*. Section 5 deals with van de Pol's strategies. Section 6 discusses how to prove termination of programs with strategy annotations in practice. Section 7 concludes. For the sake of readability, proofs of theorems are given in Appendix A.

2 Preliminaries

Let us first introduce the main notations used in the paper. For full definitions we refer to [BN98].

Binary relations. Let $R \subseteq A \times A$ be a binary relation on a set A . We denote the transitive closure of R by R^+ and its reflexive and transitive closure by R^* . A finite R -sequence is a sequence a_1, a_2, \dots, a_n of elements taken from A such that $a_i R a_{i+1}$ for $1 \leq i < n$; we say that such a sequence begins in a_1 and ends in a_n . We say that R is *confluent* if, for every $a, b, c \in A$, whenever $a R^* b$ and $a R^* c$, there exists $d \in A$ such that $b R^* d$ and $c R^* d$. An element $a \in A$ is said to be an R -normal form if there exists no b such that $a R b$; otherwise, a is called R -reducible. We say that b is an R -normal form of a (written $a R^! b$) if b is an R -normal form and $a R^* b$. We say that R is *terminating* iff there is no infinite sequence $a_1 R a_2 R a_3 \dots$.

Terms and positions. Throughout the paper, \mathcal{X} denotes a countable set of variables and \mathcal{F} denotes a set of function symbols $\{\mathbf{f}, \mathbf{g}, \dots\}$, each having a fixed arity given by a function $ar : \mathcal{F} \rightarrow \mathbb{N}$. We denote the set of terms by $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A k -tuple t_1, \dots, t_k of terms is written \bar{t} . The number k of elements of the tuple \bar{t} will be clarified by the context. $\mathcal{V}ar(t)$ is the set of variables in t .

Terms are viewed as labelled trees in the usual way. Positions p, q, \dots are represented by chains of positive natural numbers which are used to address subterms of t . We denote the empty chain by Λ . We denote the length of a chain p as $|p|$. If p is a position, and Q is a set of positions, $p.Q$ is the set $\{p.q \mid q \in Q\}$. Positions are ordered by the standard prefix ordering: $p \leq q$ iff $\exists q'$ such that $q = p.q'$; $p \parallel q$ means $p \not\leq q$ and $q \not\leq p$. 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 with s . We denote the set of positions of a term t by $\mathcal{P}os(t)$. Given terms t and s , $\mathcal{P}os_s(t)$ denotes the set of positions of s in t , i.e., $p \in \mathcal{P}os_s(t)$ iff $t|_p = s$. 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 variable occurrences. A term is said to be linear if it has no multiple occurrences of a single variable. The symbol labelling the root of t is denoted as $root(t)$.

Term rewriting systems. A rewrite rule is an ordered pair (l, r) , written $l \rightarrow r$, with $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side (*lhs*) of the rule is l and the right-hand side (*rhs*) is r . A TRS is a pair $\mathcal{R} = (\mathcal{F}, R)$ where R is a set of rewrite rules. $L(\mathcal{R})$ denotes the *lhs*'s of \mathcal{R} . Given a substitution σ , an instance $\sigma(l)$ of a *lhs* l of a rule is a redex. The set of redex positions in t is $\mathcal{P}os_{\mathcal{R}}(t) = \{p \in \mathcal{P}os(t) \mid \exists l \in L(\mathcal{R}) : t|_p = \sigma(l)\}$.

A term t rewrites to s (at position p), written $t \xrightarrow{p}_{\mathcal{R}} s$ (or just $t \xrightarrow{p} s$, or $t \rightarrow_{\mathcal{R}} s$, or even $t \rightarrow s$) if $t|_p = \sigma(l)$ and $s = t[\sigma(r)]_p$, for some rule $l \rightarrow r \in R$, $p \in \mathcal{P}os(t)$ and substitution σ . The one-step rewrite relation for \mathcal{R} is \rightarrow . A finite \rightarrow -sequence is called a rewrite sequence. If $t \rightarrow^* s$, then s is a reduct of t .

In this paper, the \rightarrow -normal forms (resp. \rightarrow -reducible terms) are called normal forms (resp. reducible terms). A TRS is terminating if \rightarrow is terminating.

We say that t *innermost* rewrites to s , written $t \xrightarrow{i} s$, if $t \xrightarrow{p} s$ and $p \in \text{maximal}_{\leq}(\text{Pos}_{\mathcal{R}}(t))$. A TRS is *innermost* terminating if \xrightarrow{i} is terminating.

3 Context-sensitive rewriting

Given a signature \mathcal{F} , a mapping $\mu : \mathcal{F} \rightarrow \mathcal{P}(\mathbb{N})$ is a *replacement map* (or \mathcal{F} -map) if for all $f \in \mathcal{F}$, $\mu(f) \subseteq \{1, \dots, \text{ar}(f)\}$. The replacement map μ determines the *argument* positions which can be reduced for each symbol in \mathcal{F} [Luc98]. The set of all \mathcal{F} -maps is $M_{\mathcal{F}}$. When considering a TRS $\mathcal{R} = (\mathcal{F}, R)$, we also write $M_{\mathcal{R}}$ rather than $M_{\mathcal{F}}$. An ordering \sqsubseteq is defined on $M_{\mathcal{F}}$, the set of all \mathcal{F} -maps: $\mu \sqsubseteq \mu'$ if for all $f \in \mathcal{F}$, $\mu(f) \subseteq \mu'(f)$. Thus, $\mu \sqsubseteq \mu'$ means that μ considers less positions than μ' for reduction, i.e., μ is ‘more restrictive’ than (or equally restrictive to) μ' . Given $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $\mu \in M_{\mathcal{F}}$, the set of μ -*replacing* positions $\text{Pos}^{\mu}(t)$ of t is:

$$\text{Pos}^{\mu}(t) = \{\Lambda\},$$

if $t \in \mathcal{X}$ and

$$\text{Pos}^{\mu}(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(\text{root}(t))} i.\text{Pos}^{\mu}(t|_i)$$

if $t \notin \mathcal{X}$. By abuse, the occurrence of subterm $t|_p$ at position p is called replacing if $p \in \text{Pos}^{\mu}(t)$. In context-sensitive rewriting (*CSR*), we rewrite subterms at *replacing* positions: t μ -rewrites to s , written

$$t \xrightarrow{p}_{\mathcal{R}(\mu)} s$$

(or simply $t \xrightarrow{\mathcal{R}(\mu)} s$, $t \xrightarrow{\mu} s$ or $t \xrightarrow{\hookrightarrow} s$) if $t \xrightarrow{p}_{\mathcal{R}} s$ and $p \in \text{Pos}^{\mu}(t)$. The $\xrightarrow{\mu}$ -normal forms are called μ -normal forms.

Example 5 Consider the TRS \mathcal{R} :

$$\begin{array}{llll} \text{add}(0, x) & \rightarrow x & \text{fst}(0, x) & \rightarrow \text{nil} \\ \text{add}(s(x), y) & \rightarrow s(\text{add}(x, y)) & \text{fst}(s(x), \text{cons}(y, z)) & \rightarrow \text{cons}(y, \text{fst}(x, z)) \\ \text{len}(\text{nil}) & \rightarrow 0 & \text{from}(x) & \rightarrow \text{cons}(x, \text{from}(s(x))) \\ \text{len}(\text{cons}(x, y)) & \rightarrow s(\text{len}(y)) & & \end{array}$$

together with the replacement map

$$\mu(s) = \emptyset, \mu(\text{cons}) = \mu(\text{from}) = \mu(\text{len}) = \{1\} \quad \text{and} \quad \mu(\text{add}) = \mu(\text{fst}) = \{1, 2\}.$$

The following rewriting sequence is allowed with CSR under μ (we underline the redex which is contracted in each μ -rewriting step):

$$\begin{array}{l} \text{fst}(s(0), \text{from}(\text{add}(0, 0))) \xrightarrow{\hookrightarrow} \text{fst}(s(0), \text{cons}(\text{add}(0, 0), \text{from}(s(\text{add}(0, 0)))))) \\ \quad \xrightarrow{\hookrightarrow} \text{cons}(\text{add}(0, 0), \text{fst}(0, \text{from}(s(\text{add}(0, 0)))))) \\ \quad \xrightarrow{\hookrightarrow} \text{cons}(0, \text{fst}(0, \text{from}(s(\text{add}(0, 0)))))) \end{array}$$

However, the infinite (meaningless) rewrite sequence

$$\begin{aligned}
& \text{fst}(\text{s}(0), \text{from}(\text{add}(0,0))) \rightarrow \text{fst}(\text{s}(0), \text{cons}(\text{add}(0,0), \text{from}(\text{s}(\text{add}(0,0)))) \\
& \rightarrow \text{fst}(\text{s}(0), \text{cons}(\text{add}(0,0), \text{cons}(\text{s}(\text{add}(0,0)), \text{from}(\text{s}(\text{s}(\text{add}(0,0)))))) \\
& \rightarrow \dots
\end{aligned}$$

is avoided since $\mu(\text{cons}) = \{1\}$ (the second argument of ‘cons’ cannot be rewritten). For instance, the second reduction step is not allowed with CSR.

A TRS \mathcal{R} is μ -terminating if \hookrightarrow_μ is terminating. With *innermost* context-sensitive rewriting $\overset{i}{\hookrightarrow}_\mu$, we only contract *maximal* positions (w.r.t. \leq) of replacing redexes: $t \overset{i}{\hookrightarrow}_\mu s$ if $t \xrightarrow{\mathcal{R}} s$ and $p \in \text{maximal}_{\leq}(\text{Pos}_{\mathcal{R}}^\mu(t))$. We say that \mathcal{R} is *innermost* μ -terminating if $\overset{i}{\hookrightarrow}_\mu$ is terminating. Termination of innermost CSR has been studied in [GM03a, Luc01b].

4 E -strategies

A local strategy (or E -strategy) for a k -ary symbol $f \in \mathcal{F}$ is a sequence $(i_1 \cdots i_n)$ of non-negative integers i_1, \dots, i_n taken from $\{0, 1, \dots, k\}$. In Maude and OBJ*, they are given as sequences of numbers in parentheses (see Example 2). A mapping φ that associates a local strategy $\varphi(f)$ to every $f \in \mathcal{F}$ is called an E -strategy map for \mathcal{F} [NO01]; when a TRS $\mathcal{R} = (\mathcal{F}, R)$ is considered, we better say that φ is an E -strategy map for \mathcal{R} .

As remarked above, symbols without an explicit local strategy are given a *default* one whose concrete shape depends on the language considered:

1. In Maude, the default local strategy associated to a k -ary symbol f , is $(1\ 2\ \cdots\ k\ 0)$, see [Eke98, CDEL⁺02].
2. In OBJ3, the default local strategy associated to a k -ary symbol f ‘is determined from its equations by requiring that all argument places that contain a non-variable term in some rule are evaluated before equations are applied at the top’ and ‘all arguments are reduced in some order’ and ‘either the operator has no rules or the strategy ends with a final zero’ [GWMFJ00, Section 2.4.4].
3. In CafeOBJ, the default strategy is computed as follows: ‘for each argument, evaluate the argument before the whole term, if there is a rewrite rule that defines the operator such that, in the place of the argument, a non-variable term appears’ [NSF99, Section 7.4.2]. In other words, the OBJ3 definition has been adopted.

Example 6 *The following table shows the E -strategies that are associated to symbols of programs DIFF and FIRST_FROM_LENGTH in Examples 1 and 2 using OBJ3 and Maude interpreters.*

DIFF			FIRST_FROM_LENGTH		
<i>Symbol</i>	Maude	OBJ3	<i>Symbol</i>	Maude	OBJ3
0	(0)	(0)	0	(0)	(0)
p	(1 0)	(1 0)	s	(1 0)	(1 0)
s	(1 0)	(1 0)	nil	(0)	(0)
<=	(1 2 0)	(1 2 0)	cons	(1 0)	(1 0)
-	(1 2 0)	(0 1 2 0)	fst	(1 2 0)	(1 2 0)
if_then_else_fi	(1 0 2 3 0)	(1 0)	from	(1 0)	(0 1 0)
			add	(1 2 0)	(1 0 2 0)
			len	(1 0)	(1 0)

As shown in Examples 1 and 2, the different settings for default strategies, make sharp differences regarding the behaviour of programs, when executed using different interpreters. In particular, Example 1 illustrates this contrast regarding termination.

4.1 Rewriting under E -strategies

Nagaya describes the operational semantics of term rewriting under E -strategy maps as follows [Nag99]: 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 natural numbers not exceeding $n \in \mathbb{N}$. We use the signature $\mathcal{F}_{\mathcal{L}} = \{f_L \mid f \in \mathcal{F} \wedge L \in \mathcal{L}_{ar(f)}\}$ and labelled variables $\mathcal{X}_{\mathcal{L}} = \{x_{nil} \mid x \in \mathcal{X}\}$. The set of labelled terms is, then, $\mathcal{T}(\mathcal{F}_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}})$.

An E -strategy map φ for \mathcal{F} is extended to a mapping from $\mathcal{T}(\mathcal{F}, \mathcal{X})$ to $\mathcal{T}(\mathcal{F}_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}})$ as follows:

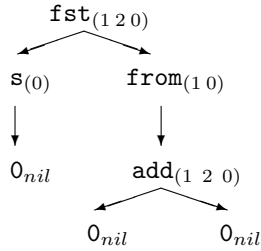
$$\varphi(t) = \begin{cases} x_{nil} & \text{if } t = x \in \mathcal{X} \\ f_{\varphi(f)}(\varphi(t_1), \dots, \varphi(t_k)) & \text{if } t = f(t_1, \dots, t_k) \end{cases}$$

The mapping $erase : \mathcal{T}(\mathcal{F}_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}}) \rightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$ removes labellings from symbols in the obvious way.

Example 7 Consider the term $t = \text{fst}(\text{s}(0), \text{from}(\text{add}(0, 0)))$ of Example 2 and the E -strategy map φ given by $\varphi(\text{fst}) = (1\ 2\ 0)$ and $\varphi(\text{from}) = \varphi(\text{add}) = (1\ 0)$ and $\varphi(\text{s}) = (0)$ (and, obviously, $\varphi(0) = \text{nil}$). Then, $\varphi(t)$ is

$$\text{fst}_{(1\ 2\ 0)}(\text{s}_{(0)}(0_{nil}), \text{from}_{(1\ 0)}(\text{add}_{(1\ 2\ 0)}(0_{nil}, 0_{nil})))$$

Graphically, $\varphi(t)$ is depicted as follows:



The binary relation \rightarrow_φ on $\mathcal{T}(\mathcal{F}_L, \mathcal{X}_L) \times \mathbb{N}_+^*$ (i.e., pairs $\langle t, p \rangle$ of labelled terms t and positions p) is [Nag99, NO01]: $\langle t, p \rangle \rightarrow_\varphi \langle s, q \rangle$ 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. $t|_p = f_{i:L}(t_1, \dots, t_k)$ with $i > 0$, $s = t[f_L(t_1, \dots, t_k)]_p$ and $q = p.i$; or
3. $t|_p = f_{0:L}(t_1, \dots, t_k)$, $erase(t|_p)$ is not a redex, $s = t[f_L(t_1, \dots, t_k)]_p$, $q = p$; or
4. $t|_p = f_{0:L}(t_1, \dots, t_k) = \sigma(l')$, $erase(l') = l$, $s = t[\sigma(\varphi(r))]_p$ for some $l \rightarrow r \in R$ and substitution σ , $q = p$.

Given pairs $\langle t, p \rangle$ and $\langle s, q \rangle$, if $\langle t, p \rangle \rightarrow_\varphi \langle s, q \rangle$ using one of the first three (*traversal*) steps above, we write $\langle t, p \rangle \xrightarrow{T}_\varphi \langle s, q \rangle$. On the other hand, if the last (rewriting) step is used, we write $\langle t, p \rangle \xrightarrow{R}_\varphi \langle s, q \rangle$. This defines two auxiliary reduction relations \xrightarrow{T}_φ and \xrightarrow{R}_φ on pairs of labelled terms and positions which we use later. Obviously, $\rightarrow_\varphi = \xrightarrow{T}_\varphi \cup \xrightarrow{R}_\varphi$.

Given a TRS $\mathcal{R} = (\mathcal{F}, R)$ and a E -strategy map φ for \mathcal{F} , the evaluation mapping $eval_\varphi : \mathcal{T}(\mathcal{F}, \mathcal{X}) \rightarrow \mathcal{P}(\mathcal{T}(\mathcal{F}, \mathcal{X}))$ is defined as [Nag99, NO01]

$$eval_\varphi(t) = \{erase(s) \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \mid \langle \varphi(t), \Lambda \rangle \rightarrow_\varphi^! \langle s, \Lambda \rangle\}.$$

Remark 1 *Computation of $eval_\varphi$ in Maude, OBJ*, and CafeOBJ is implemented by means of a command **red** (or **reduce**) which is available in existing interpreters (see the examples above).*

Example 8 *The E -strategy map φ which Maude uses for **FIRST_FROM_LENGTH** in Example 2 is given in Example 6. The use of command **red** of the Maude interpreter to evaluate $t = \mathbf{fst}(\mathbf{s}(0), \mathbf{from}(\mathbf{add}(0, 0)))$ can now be more precisely described as the evaluation of*

$$\varphi(t) = \mathbf{fst}_{(1\ 2\ 0)}(\mathbf{s}_{(0)}(\mathbf{0}_{nil}), \mathbf{from}_{(1\ 0)}(\mathbf{add}_{(1\ 2\ 0)}(\mathbf{0}_{nil}, \mathbf{0}_{nil})))$$

using \rightarrow_φ (we underline the redex contracted at each step):

$$\begin{aligned} & \langle \mathbf{fst}_{(1\ 2\ 0)}(\mathbf{s}_{(0)}(\mathbf{0}_{nil}), \mathbf{from}_{(1\ 0)}(\mathbf{add}_{(1\ 2\ 0)}(\mathbf{0}_{nil}, \mathbf{0}_{nil}))), \Lambda \rangle \\ & \rightarrow_\varphi \langle \mathbf{fst}_{(2\ 0)}(\mathbf{s}_{(0)}(\mathbf{0}_{nil}), \mathbf{from}_{(1\ 0)}(\mathbf{add}_{(1\ 2\ 0)}(\mathbf{0}_{nil}, \mathbf{0}_{nil}))), 1 \rangle \\ & \rightarrow_\varphi \langle \mathbf{fst}_{(2\ 0)}(\mathbf{s}_{nil}(\mathbf{0}_{nil}), \mathbf{from}_{(1\ 0)}(\mathbf{add}_{(1\ 2\ 0)}(\mathbf{0}_{nil}, \mathbf{0}_{nil}))), 1 \rangle \\ & \rightarrow_\varphi \langle \mathbf{fst}_{(2\ 0)}(\mathbf{s}_{nil}(\mathbf{0}_{nil}), \mathbf{from}_{(1\ 0)}(\mathbf{add}_{(1\ 2\ 0)}(\mathbf{0}_{nil}, \mathbf{0}_{nil}))), \Lambda \rangle \\ & \rightarrow_\varphi \langle \mathbf{fst}_{(0)}(\mathbf{s}_{nil}(\mathbf{0}_{nil}), \mathbf{from}_{(1\ 0)}(\mathbf{add}_{(1\ 2\ 0)}(\mathbf{0}_{nil}, \mathbf{0}_{nil}))), 2 \rangle \\ & \rightarrow_\varphi \langle \mathbf{fst}_{(0)}(\mathbf{s}_{nil}(\mathbf{0}_{nil}), \mathbf{from}_{(0)}(\mathbf{add}_{(1\ 2\ 0)}(\mathbf{0}_{nil}, \mathbf{0}_{nil}))), 2.1 \rangle \\ & \rightarrow_\varphi \langle \mathbf{fst}_{(0)}(\mathbf{s}_{nil}(\mathbf{0}_{nil}), \mathbf{from}_{(0)}(\mathbf{add}_{(2\ 0)}(\mathbf{0}_{nil}, \mathbf{0}_{nil}))), 2.1.1 \rangle \\ & \rightarrow_\varphi \langle \mathbf{fst}_{(0)}(\mathbf{s}_{nil}(\mathbf{0}_{nil}), \mathbf{from}_{(0)}(\mathbf{add}_{(2\ 0)}(\mathbf{0}_{nil}, \mathbf{0}_{nil}))), 2.1 \rangle \\ & \rightarrow_\varphi \langle \mathbf{fst}_{(0)}(\mathbf{s}_{nil}(\mathbf{0}_{nil}), \mathbf{from}_{(0)}(\mathbf{add}_{(0)}(\mathbf{0}_{nil}, \mathbf{0}_{nil}))), 2.1.2 \rangle \\ & \rightarrow_\varphi \langle \mathbf{fst}_{(0)}(\mathbf{s}_{nil}(\mathbf{0}_{nil}), \mathbf{from}_{(0)}(\mathbf{add}_{(0)}(\mathbf{0}_{nil}, \mathbf{0}_{nil}))), 2.1 \rangle \\ & \rightarrow_\varphi \langle \mathbf{fst}_{(0)}(\mathbf{s}_{nil}(\mathbf{0}_{nil}), \mathbf{from}_{(0)}(\mathbf{0}_{nil})), 2.1 \rangle \\ & \rightarrow_\varphi \langle \mathbf{fst}_{(0)}(\mathbf{s}_{nil}(\mathbf{0}_{nil}), \mathbf{from}_{(0)}(\mathbf{0}_{nil})), 2 \rangle \end{aligned}$$

$$\begin{aligned}
&\rightarrow_{\varphi} \langle \mathbf{fst}_{(0)}(\mathbf{s}_{nil}(0_{nil}), \mathbf{cons}_{(1)}(0_{nil}, \mathbf{from}_{(1\ 0)}(\mathbf{s}_{(1)}(0_{nil}))) \rangle, 2 \rangle \\
&\rightarrow_{\varphi}^+ \langle \mathbf{fst}_{(0)}(\mathbf{s}_{nil}(0_{nil}), \mathbf{cons}_{nil}(0_{nil}, \mathbf{from}_{(1\ 0)}(\mathbf{s}_{(1)}(0_{nil}))) \rangle, \Lambda \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{cons}_{(1)}(0_{nil}, \mathbf{fst}_{(1\ 2\ 0)}(0_{nil}, \mathbf{from}_{(1\ 0)}(\mathbf{s}_{(1)}(0_{nil}))) \rangle, \Lambda \rangle \\
&\rightarrow_{\varphi}^+ \langle \mathbf{cons}_{nil}(0_{nil}, \mathbf{fst}_{(1\ 2\ 0)}(0_{nil}, \mathbf{from}_{(1\ 0)}(\mathbf{s}_{(1)}(0_{nil}))) \rangle, \Lambda \rangle
\end{aligned}$$

where

$$\langle \mathbf{cons}_{nil}(0_{nil}, \mathbf{fst}_{(1\ 2\ 0)}(0_{nil}, \mathbf{from}_{(1\ 0)}(\mathbf{s}_{(1)}(0_{nil}))) \rangle, \Lambda \rangle$$

is a \rightarrow_{φ} -normal form. Then, $\mathbf{cons}(0, \mathbf{fst}(0, \mathbf{from}(\mathbf{s}(0)))) \in \mathit{eval}_{\varphi}(t)$. Note that this corresponds to the evaluation of the expression performed by the Maude interpreter (see Example 2).

Example 9 The failed evaluation of $t = \mathbf{p}(0) - 0$ using the Maude interpreter that we shown in Example 1 can also be more precisely described as the nonterminating evaluation of

$$\varphi(t) = \mathbf{p}_{(1\ 0)}(0_{nil})_{-(1\ 2\ 0)}0_{nil}$$

where $\varphi(\mathbf{if_then_else_fi}) = (1\ 0\ 2\ 3\ 0)$ and $\varphi(f) = (1 \dots k\ 0)$ for every other symbol³:

$$\begin{aligned}
&\langle \mathbf{p}_{(1\ 0)}(0_{nil})_{-(1\ 2\ 0)}0_{nil}, \Lambda \rangle \rightarrow_{\varphi} \langle \mathbf{p}_{(1\ 0)}(0_{nil})_{-(2\ 0)}0_{nil}, 1 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{p}_{(0)}(0_{nil})_{-(2\ 0)}0_{nil}, 1.1 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{p}_{(0)}(0_{nil})_{-(2\ 0)}0_{nil}, 1 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{p}_{nil}(0_{nil})_{-(2\ 0)}0_{nil}, 1 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{p}_{nil}(0_{nil})_{-(2\ 0)}0_{nil}, \Lambda \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{p}_{nil}(0_{nil})_{-(0)}0_{nil}, 2 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{p}_{nil}(0_{nil})_{-(0)}0_{nil}, \Lambda \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(1\ 0\ 2\ 3\ 0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{(1\ 2\ 0)}0_{nil}, 0_{nil}, \mathbf{s}_{(1)}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), \Lambda \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(0\ 2\ 3\ 0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{(1\ 2\ 0)}0_{nil}, 0_{nil}, \mathbf{s}_{(1)}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), 1 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(0\ 2\ 3\ 0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{(2\ 0)}0_{nil}, 0_{nil}, \mathbf{s}_{(1)}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), 1.1 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(0\ 2\ 3\ 0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{(2\ 0)}0_{nil}, 0_{nil}, \mathbf{s}_{(1)}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), 1 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(0\ 2\ 3\ 0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{(0)}0_{nil}, 0_{nil}, \mathbf{s}_{(1)}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), 1.2 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(0\ 2\ 3\ 0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{(0)}0_{nil}, 0_{nil}, \mathbf{s}_{(1)}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), 1 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(0\ 2\ 3\ 0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{nil}0_{nil}, 0_{nil}, \mathbf{s}_{(1)}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), 1 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(0\ 2\ 3\ 0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{nil}0_{nil}, 0_{nil}, \mathbf{s}_{(1)}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), \Lambda \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(2\ 3\ 0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{nil}0_{nil}, 0_{nil}, \mathbf{s}_{(1)}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), \Lambda \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(3\ 0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{nil}0_{nil}, 0_{nil}, \mathbf{s}_{(1)}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), 2 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(3\ 0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{nil}0_{nil}, 0_{nil}, \mathbf{s}_{(1)}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), \Lambda \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{nil}0_{nil}, 0_{nil}, \mathbf{s}_{(1)}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), 3 \rangle \\
&\rightarrow_{\varphi} \langle \mathbf{if}_{(0)}(\mathbf{p}_{nil}(0_{nil}) \leq_{nil}0_{nil}, 0_{nil}, \mathbf{s}_{nil}(\mathbf{p}_{(1\ 0)}(\mathbf{p}_{nil}(0_{nil}))_{-(1\ 2\ 0)}0_{nil})), 3.1 \rangle \\
&\rightarrow_{\varphi} \dots
\end{aligned}$$

In principle, the use of command `red` in Maude and OBJ* interpreters produces a single output expression or no expression at all (i.e., the evaluation does not terminate). We can, though, obtain all possible values associated to an input expression t by explicitly activating such a multiple evaluation.

The following proposition shows that the evaluation of labelled terms using \rightarrow_{φ} can be viewed as first traversing the term as much as possible (according to labels induced by φ) and next contracting a redex.

³For the sake of brevity, we just use $\varphi(c) = (1 \dots k)$ for constructor symbols c .

Proposition 1 *Let \mathcal{R} be a TRS and φ be an E-strategy map for \mathcal{R} . Then,*

$$eval_{\varphi}(t) = \{erase(s) \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \mid \langle \varphi(t), \Lambda \rangle (\xrightarrow{\varphi}^! \circ \xrightarrow{\varphi}^R)^! \langle s, p \rangle\}.$$

4.2 E-strategies and context-sensitive rewriting

We write $e \in L$ to denote that item e appears somewhere within the list L . The following definition shows how to associate a replacement map to a give E-strategy map.

Definition 1 *Given a E-strategy map φ for \mathcal{F} , we define $\mu^{\varphi} \in M_{\mathcal{F}}$ as follows: for all $f \in \mathcal{F}$, $\mu^{\varphi}(f) = \{i > 0 \mid i \in \varphi(f)\}$.*

The following theorem establishes a very close connection between \rightarrow_{φ} -reduction and $\hookrightarrow_{\mu^{\varphi}}$ -reduction: each computation step induced by an E-strategy map φ corresponds to a (possibly empty) μ^{φ} -rewriting step.

Theorem 1 *Let \mathcal{R} be a TRS and φ be an E-strategy map. Let $t \in \mathcal{T}(\mathcal{F}_{\mathcal{L}}, \mathcal{X}_{\mathcal{L}})$, and $p \in \mathcal{Pos}^{\mu^{\varphi}}(erase(t))$ be s.t. $root(t|_p) = f_L$ for some suffix L of $\varphi(f)$. If $\langle t, p \rangle \rightarrow_{\varphi} \langle s, q \rangle$, then $q \in \mathcal{Pos}^{\mu^{\varphi}}(erase(s))$ and*

1. *if $\langle t, p \rangle \xrightarrow{\varphi}^T \langle s, q \rangle$, then $erase(t) = erase(s)$.*
2. *if $\langle t, p \rangle \xrightarrow{\varphi}^R \langle s, q \rangle$, then $erase(t) \hookrightarrow_{\mu^{\varphi}} erase(s)$.*

Example 10 *Consider the \rightarrow_{φ} -evaluation sequence of Example 8. Note that, according to Theorem 1, the evaluation only performs three μ^{φ} -rewriting steps:*

$$\begin{aligned} &fst(s(0), \underline{from(add(0,0))}) \hookrightarrow_{\mu^{\varphi}} fst(s(0), \underline{from(0)}) \\ &\quad \hookrightarrow_{\mu^{\varphi}} \underline{fst(s(0), cons(0, from(s(0))))} \\ &\quad \hookrightarrow_{\mu^{\varphi}} cons(0, \underline{fst(0, from(s(0)))}) \end{aligned}$$

On the other hand, the $\mu^{\varphi'}$ -sequence that correspond to the evaluation of the expression using the OBJ3 interpreter (i.e., the E-strategy map φ' in the column labelled OBJ3 for program FIRST_FROM_LENGTH in Example 6) is:

$$\begin{aligned} &fst(s(0), \underline{from(add(0,0))}) \hookrightarrow_{\mu^{\varphi'}} fst(s(0), \underline{cons(add(0,0), from(s(add(0,0))))}) \\ &\quad \hookrightarrow_{\mu^{\varphi'}} \underline{fst(s(0), cons(0, from(s(add(0,0))))})} \\ &\quad \hookrightarrow_{\mu^{\varphi'}} cons(0, \underline{fst(0, from(s(add(0,0))))}) \end{aligned}$$

4.3 Termination of programs with E-strategies

According to the previous definition of $eval_{\varphi}$, we give the following:

Definition 2 *A TRS \mathcal{R} is φ -terminating if, for all $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, there is no infinite \rightarrow_{φ} -rewrite sequence starting from $\langle \varphi(t), \Lambda \rangle$.*

The following theorem connects termination of CSR and termination programs using E-strategies.

Theorem 2 Let \mathcal{R} be a TRS and φ be an E -strategy map for \mathcal{R} . If \mathcal{R} is μ^φ -terminating, then \mathcal{R} is φ -terminating.

As a consequence of Theorem 2, the existing methods for proving termination of *CSR* [BLR02, FR99, GL02a, GL02b, GM99, GM03b, Luc96, Luc03a, Luc02c, SX98, Zan97] can be used now to prove termination of Maude, OBJ*, and CafeOBJ programs.

Example 11 Termination of program `FIRST_FROM_LENGTH` in Example 2 can be proved as μ^φ -termination of the corresponding TRS \mathcal{R} (given in Example 5). Here, $\mu^\varphi(\mathbf{s}) = \emptyset$, $\mu^\varphi(\mathbf{cons}) = \{1\}$ and $\mu^\varphi(f) = \{1, \dots, ar(f)\}$ for every other symbol f . Now, we can use the contractive transformation of [Luc96] (which just drops the non-replacing subterms and reduces the arity of symbols if necessary) to obtain the TRS $\mathcal{R}_L^{\mu^\varphi}$:

<code>fst(0,Z)</code>	\rightarrow <code>nil</code>	<code>add(0,X)</code>	\rightarrow <code>X</code>
<code>fst(s,cons(Y))</code>	\rightarrow <code>cons(Y)</code>	<code>add(s,Y)</code>	\rightarrow <code>s</code>
<code>len(nil)</code>	\rightarrow <code>0</code>	<code>from(X)</code>	\rightarrow <code>cons(X)</code>
<code>len(cons(X))</code>	\rightarrow <code>s</code>		

Termination of $\mathcal{R}_L^{\mu^\varphi}$ can be proved by using a recursive path orderings (RPO [Der82]) based on precedence

`fst` > `nil`; `from` > `cons`; and `len` > `0, s`

Termination of $\mathcal{R}_L^{\mu^\varphi}$ implies μ^φ -termination of \mathcal{R} ([Luc96, Theorem 4.21]), hence termination of `FIRST_FROM_LENGTH`.

However, termination of *CSR* only approximates φ -termination.

Example 12 Consider the TRS [Gra96]:

<code>f(a)</code>	\rightarrow <code>f(a)</code>
<code>a</code>	\rightarrow <code>b</code>

and let $\varphi(\mathbf{f}) = (1\ 0)$ and $\varphi(\mathbf{a}) = (0)$. This TRS is φ -terminating, but it is not μ^φ -terminating, since we have: $\underline{\mathbf{f(a)}} \hookrightarrow_{\mu^\varphi} \underline{\mathbf{f(a)}} \hookrightarrow_{\mu^\varphi} \dots$.

The point here is that computations under the E -strategy are ‘basically’ innermost. Innermost rewriting computations can be terminating even for nonterminating TRSs. This gives rise the topic of *innermost* termination of rewriting which has been studied in e.g., [AG97, Gra96, Kri00]. For instance, the TRS in Example 12 is nonterminating, but innermost terminating [Gra96].

Given a TRS $\mathcal{R} = (\mathcal{F}, R)$, we consider \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 $\mathcal{D} = \{\text{root}(l) \mid l \rightarrow r \in R\}$ and $\mathcal{C} = \mathcal{F} - \mathcal{D}$. The following definition allows us to introduce a class of programs whose φ -termination is equivalent to termination of innermost *CSR*.

Definition 3 (Elementary E -strategy map) Given a TRS $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$, we say that an E -strategy map φ is elementary for \mathcal{R} if for all $f \in \mathcal{D}$, $\varphi(f) = (i_1 \dots i_n\ 0)$ and $i_j > 0$ for $1 \leq j \leq n$.

Remark 2 *Elementarity is quite usual in most programs in the literature. Default strategies in Maude are also elementary (except for the `if_then_else-fi` operator). Unfortunately, this is not the case for OBJ3 or CafeOBJ (see Example 6). On the other hand, Eker [Eke98] and Nagaya [Nag99] largely motivate the interest of requiring that 0 be the last index of local strategies associated to defined symbols. More details about the relevance of this requirement can be found in [Luc03b].*

The following theorem shows that, for elementary E -strategies, innermost CSR suffices for modeling evaluations using \rightarrow_φ (recall Proposition 1).

Theorem 3 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and φ be an elementary E -strategy map for \mathcal{R} . Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If*

$$\langle \varphi(t), \Lambda \rangle = \langle t_1, p_1 \rangle \xrightarrow{\text{T}}_{\varphi^*}^* \circ \xrightarrow{\text{R}}_{\varphi} \langle t_2, p_2 \rangle \xrightarrow{\text{T}}_{\varphi^*}^* \circ \xrightarrow{\text{R}}_{\varphi} \cdots \xrightarrow{\text{T}}_{\varphi^*}^* \circ \xrightarrow{\text{R}}_{\varphi} \langle t_n, p_n \rangle,$$

then $t = \text{erase}(t_1) \xrightarrow{i}_{\mu^\varphi} \text{erase}(t_2) \xrightarrow{i}_{\mu^\varphi} \cdots \xrightarrow{i}_{\mu^\varphi} \text{erase}(t_n)$.

Without requiring elementarity of φ , Theorem 3 does not hold.

Example 13 *Consider the following TRS \mathcal{R} :*

$$\begin{array}{ll} \mathbf{f}(\mathbf{b}) \rightarrow \mathbf{c} & \mathbf{h}(\mathbf{c}) \rightarrow \mathbf{g}(\mathbf{f}(\mathbf{a})) \\ \mathbf{g}(\mathbf{x}) \rightarrow \mathbf{h}(\mathbf{x}) & \mathbf{a} \rightarrow \mathbf{b} \end{array}$$

and the E -strategy map φ given by $\varphi(\mathbf{f}) = (0 \ 1)$, $\varphi(\mathbf{g}) = \varphi(\mathbf{h}) = (1 \ 0)$, $\varphi(\mathbf{a}) = (0)$, and $\varphi(\mathbf{b}) = \text{nil}$. Let $t = \mathbf{g}(\mathbf{f}(\mathbf{a}))$. According to Theorem 1, the μ^φ -rewriting steps associated to the evaluation of t are:

$$\mathbf{g}(\mathbf{f}(\underline{\mathbf{a}})) \xrightarrow{i}_{\mu^\varphi} \underline{\mathbf{g}(\mathbf{f}(\mathbf{b}))} \xrightarrow{i}_{\mu^\varphi} \mathbf{h}(\mathbf{f}(\mathbf{b}))$$

Due to redex $\mathbf{f}(\mathbf{b})$, the second μ^φ -rewriting step is not innermost.

As a consequence of Theorem 3, we have the following theorem establishing that proving termination of innermost CSR suffices to prove termination of rewriting controlled by elementary E -strategy maps.

Theorem 4 *Let \mathcal{R} be a TRS and φ be an elementary E -strategy map for \mathcal{R} . If \mathcal{R} is innermost μ -terminating, then \mathcal{R} is φ -terminating.*

For nonelementary E -strategies, Theorem 4 can fail to hold.

Example 14 *Consider the TRS \mathcal{R} of Example 12 and φ given by $\varphi(\mathbf{f}) = (0 \ 1 \ 0)$, $\varphi(\mathbf{a}) = (0)$, and $\varphi(\mathbf{b}) = \text{nil}$. Note that \mathcal{R} is innermost μ^φ -terminating. However, \mathcal{R} is not φ -terminating, since we have:*

$$\langle \underline{\mathbf{f}(0 \ 1 \ 0)}(\underline{\mathbf{a}(0)}), \Lambda \rangle \rightarrow_\varphi \langle \underline{\mathbf{f}(0 \ 1 \ 0)}(\underline{\mathbf{a}(0)}), \Lambda \rangle \rightarrow_\varphi \cdots$$

Since μ -termination implies innermost μ -termination (but not vice versa), analyzing innermost termination of CSR [GL02a, GM03a, Luc01b] provides a more accurate framework for proving termination of TRSs under elementary E -strategies. In fact, we obtain a complete proof method.

Theorem 5 Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and φ be an elementary E -strategy map for \mathcal{R} . If \mathcal{R} is φ -terminating, then \mathcal{R} is innermost μ^φ -terminating.

Without elementarity, φ -termination may not imply innermost μ^φ -termination.

Example 15 Consider \mathcal{R} and φ as in Example 13. Note that \mathcal{R} is not innermost μ^φ -terminating:

$$\underline{\mathbf{h(c)}} \xrightarrow{i}_{\mu^\varphi} \underline{\mathbf{g(f(a))}} \xrightarrow{i}_{\mu^\varphi} \underline{\mathbf{g(f(b))}} \xrightarrow{i}_{\mu^\varphi} \underline{\mathbf{g(c)}} \xrightarrow{i}_{\mu^\varphi} \underline{\mathbf{h(c)}} \xrightarrow{i}_{\mu^\varphi} \dots$$

However, \mathcal{R} is φ -terminating, since, as shown in Example 13, whenever (the labelled version of) the term $\mathbf{g(f(a))}$ is reached, the derivation stops in $\mathbf{h(f(b))}$ without producing $\mathbf{h(c)}$ which is needed to generate the cycle.

In [GL02a, GM03a], termination of CSR and termination of innermost CSR has been compared. A constructor system (CS) is a TRS $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ where for all $f(l_1, \dots, l_k) \rightarrow r \in R$, we have that $l_1, \dots, l_k \in \mathcal{T}(\mathcal{C}, \mathcal{X})$. An orthogonal TRS is a left-linear TRS without overlapping left-hand sides [BN98]. The following result is interesting in our setting.

Theorem 6 [GM03a] Termination of CSR and termination of innermost CSR coincide for orthogonal constructor systems.

This result is interesting in programming since the class of orthogonal constructor systems covers most first order functional programs (as used, e.g., in functional modules of Maude).

Example 16 In [GM03a], Giesl and Middeldorp give a proof of innermost termination of CSR for the orthogonal constructor system \mathcal{R} :

$$\begin{array}{llll} 0 & \leq n & \rightarrow \text{true} & p(0) \rightarrow 0 \\ s(m) \leq 0 & \rightarrow \text{false} & & p(s(n)) \rightarrow n \\ s(m) \leq s(n) & \rightarrow m \leq n & & \\ \text{if}(\text{true}, m, n) & \rightarrow m & & \\ \text{if}(\text{false}, m, n) & \rightarrow n & & \\ m - n & \rightarrow \text{if}(m \leq n, 0, s(p(m) - n)) & & \end{array}$$

together with $\mu(\text{if}) = \{1\}$ and $\mu(f) = \{1, \dots, ar(f)\}$ for every other symbol f . By Theorem 6, \mathcal{R} is μ -terminating. By Theorem 2, the program DIFF in Example 1 is terminating when executed using the OBJ3 interpreter (that uses the E -strategy (1 0) for the conditional operator `if_then_else_fi`).

Note that, even though φ is elementary, we cannot directly use Theorem 4 to prove termination of DIFF. This is because DIFF does not contain the rule $p(0) \rightarrow 0$ which is contained in \mathcal{R} . Since termination of innermost CSR is not preserved under removals of rules (e.g., think of \mathcal{R} in Example 12, which is not innermost terminating anymore if rule $\mathbf{a} \rightarrow \mathbf{b}$ is removed), we need to first prove μ -termination of \mathcal{R} (which is preserved under removal of rules) before proving termination of DIFF.

4.4 Related work

As far as the author knows, only Fissore, Gnaedig and Kirchner [FGK01] have investigated the problem of proving termination of rewriting under E -strategies. In [FGK01], an inductive method is proposed to directly prove (ground) termination of rewriting with positive local strategies without applying any transformation. In [FGK02], the authors describe a software system, CARIBOO, which implements the technique described in [FGK01].

The weak point of Fissore et al.'s technique is that, in order to be able to use their methods, it is necessary to use a different technique to ensure that the constant symbols are terminating (w.r.t. computations guided by the strategies). This is easy if there is no rewrite rule $c \rightarrow r$ associated to any constant symbol c (as in the Examples proposed by the authors in [FGK01]). Note that φ -termination of (a TRS containing) the following TRS \mathcal{R} which generates the list of natural numbers:

$$\begin{aligned} \mathbf{nats} &\rightarrow 0:\mathbf{incr}(\mathbf{nats}) \\ \mathbf{incr}(\mathbf{x}:\mathbf{y}) &\rightarrow \mathbf{s}(\mathbf{x}):\mathbf{incr}(\mathbf{y}) \end{aligned}$$

together with $\varphi(\mathbf{nats}) = (0)$ and $\varphi(\cdot) = \varphi(\mathbf{incr}) = \varphi(\mathbf{s}) = (1\ 0)$, could not be proven in this way. However, the φ -termination of \mathcal{R} is easily proved by using the transformation of [Luc96] (already used in Example 11), since $\mathcal{R}^{\mu^\varphi}$:

$$\begin{aligned} \mathbf{nats} &\rightarrow : (0) \\ \mathbf{incr}(:(\mathbf{x})) &\rightarrow : (\mathbf{s}(\mathbf{x})) \end{aligned}$$

is clearly terminating. Nevertheless, Fissore et al.'s technique can work when our techniques do not.

Example 17 *The TRS [GF01]*

$$\begin{aligned} \mathbf{f}(\mathbf{a},\mathbf{g}(\mathbf{x})) &\rightarrow \mathbf{f}(\mathbf{a},\mathbf{h}(\mathbf{x})) \\ \mathbf{h}(\mathbf{x}) &\rightarrow \mathbf{g}(\mathbf{x}) \end{aligned}$$

terminates with the strategy φ_1 such that $\varphi_1(\mathbf{f}) = (0\ 1\ 2)$ but it does not terminate with $\varphi_2(\mathbf{f}) = (1\ 2\ 0)$ (let $\varphi_1(\mathbf{h}) = \varphi_2(\mathbf{h}) = (0)$). In both cases $\mu^{\varphi_1}(\mathbf{f}) = \mu^{\varphi_2}(\mathbf{f}) = \{1, 2\}$. Thus, we are not able to distinguish them (note that φ_1 is not elementary), whereas CARIBOO is able to obtain a proof of termination⁴.

Note, however, that φ_1 in Example 17 is not a zero-ended E -strategy (e.g., $\varphi_1(\mathbf{f})$ does not end in 0) and would not be accepted (as it is) by some interpreters. For instance, the Maude interpreter accepts this program but it changes $\varphi_1(\mathbf{f})$ into $\varphi'_1(\mathbf{f}) = (0\ 1\ 2\ 0)$, thus introducing non-termination.

5 Van de Pol's strategy annotations

Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS. According to van de Pol [Pol01], a strategy annotation associated to a given symbol $f \in \mathcal{F}$ is a list $\zeta(f)$ whose elements can be either

⁴See <http://www.loria.fr/fissore/Demo/Exemples/LS/contrexCS.html>.

1. a number i with $1 \leq i \leq ar(f)$; or
2. a rule $l \rightarrow r \in R$ such that $root(t) = f$.

In principle, strategy annotations contain no duplicated items. Van de Pol's style of strategy annotations is the basis of the JITty system [Pol02] which has been integrated in the μ CRL tool set [BFGL⁺01].

Example 18 Consider the TRS \mathcal{R} (where the rules have been conveniently labelled [Pol01]):

$$\begin{aligned} \alpha &: \text{if}(\text{true}, \mathbf{x}, \mathbf{y}) \rightarrow \mathbf{x} & \gamma &: \text{if}(\mathbf{b}, \mathbf{x}, \mathbf{x}) \rightarrow \mathbf{x} \\ \beta &: \text{if}(\text{false}, \mathbf{x}, \mathbf{y}) \rightarrow \mathbf{y} \end{aligned}$$

A possible strategy annotation for `if` is $\varsigma(\text{if}) = [1, \alpha, \beta, 2, 3, \gamma]$. The TRS \mathcal{R} and the strategy annotation ς correspond to the JITty program of Example 4.

A strategy annotation ς is *in-time* if for all $f \in \mathcal{F}$, $\alpha : l \rightarrow r \in R$ such that $root(l) = f$, and $i \in \{1, \dots, ar(f)\}$, whenever $\varsigma(f) = L_1 \alpha L_2 i L_3$, then i is not needed for α . Here, index i is needed for a rule $\alpha : l \rightarrow r$ if $l|_i \notin \mathcal{X}$ or $l|_i \in \mathcal{X}$ occurs in $l|_j$ for $i \neq j$. For example, the strategy of Example 18 is full and in-time.

Given a strategy annotation, van de Pol describes the rewriting strategy that it specifies. A strategy is seen as a function that, given a term t , yields either some rewrite of t , i.e., a pair (p, s) such that $t \xrightarrow{p} s$, or \perp if no rewrite step has been selected. Given a term t and a strategy annotation ς , $rewr_\varsigma(t)$ indicates the (unique, if any) rewrite step that can be issued on t .

Definition 4 [Pol01] Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, ς be a strategy annotation, and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Then, $rewr_\varsigma(t) = rewr_\varsigma(t, \varsigma(root(t)))$, where

$$\begin{aligned} rewr_\varsigma(t, nil) &= \perp \\ rewr_\varsigma(t, (l \rightarrow r : L)) &= \begin{cases} (\Lambda, \sigma(r)) & \text{if } t = \sigma(l) \text{ for some } \sigma \\ rewr_\varsigma(t, L) & \text{otherwise} \end{cases} \\ rewr_\varsigma(t, (i : L)) &= \begin{cases} (i.p, t[s]_i) & \text{if } rewr_\varsigma(t|_i) = (p, s) \text{ for some } p, s \\ rewr_\varsigma(t, L) & \text{otherwise} \end{cases} \end{aligned}$$

We write $t \xrightarrow{p}_\varsigma s$ (or just $t \rightarrow_\varsigma s$) if $(p, s) = rewr_\varsigma(t) \neq \perp$. Thus, t is a \rightarrow_ς -normal form (or just a ς -normal form) if and only if $rewr_\varsigma(t) = \perp$.

5.1 Van de Pol's annotations and context-sensitive rewriting

Given a strategy annotation ς for \mathcal{F} , we define $\mu^\varsigma \in M_{\mathcal{F}}$ as follows: for all $f \in \mathcal{F}$, $\mu^\varsigma(f) = \{i \in \mathbb{N} \mid i \in \varsigma(f)\}$. The following theorem establishes a very close connection between \rightarrow_ς and $\hookrightarrow_{\mu^\varsigma}$.

Theorem 7 Let \mathcal{R} be a TRS, ς be a strategy annotation for \mathcal{R} , and $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If $t \rightarrow_\varsigma s$, then $t \hookrightarrow_{\mu^\varsigma} s$.

The main concern of [Pol01] is to achieve normalization using \rightarrow_ζ . Van de Pol provides the following normalization (partial) function:

Definition 5 [Pol01] *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, ζ be a strategy annotation, and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Then, $norm_\zeta(t) = norm_\zeta(t, \zeta(\text{root}(t)))$, where*

$$\begin{aligned} norm_\zeta(t, nil) &= t \\ norm_\zeta(t, (l \rightarrow r : L)) &= \begin{cases} norm_\zeta(\sigma(r)) & \text{if } t = \sigma(l) \text{ for some } \sigma \\ norm_\zeta(t, L) & \text{otherwise} \end{cases} \\ norm_\zeta(t, (i : L)) &= norm_\zeta(t[norm_\zeta(t|_i)]_i, L) \end{aligned}$$

At first sight, one could think that the output of $norm_\zeta(t)$ can be described as a \rightarrow_ζ -rewrite sequence. In general, this is not true.

Example 19 *Consider the TRS:*

$$\begin{array}{l} \mathbf{f}(\mathbf{b}) \rightarrow \mathbf{c} \qquad \mathbf{b} \rightarrow \mathbf{c} \\ \mathbf{a} \rightarrow \mathbf{b} \end{array}$$

together with $\zeta(\mathbf{f}) = [\alpha_{\mathbf{f}}, 1]$, $\zeta(\mathbf{a}) = [\alpha_{\mathbf{a}}]$, $\zeta(\mathbf{b}) = [\alpha_{\mathbf{b}}]$, and $\zeta(\mathbf{c}) = nil$, where α_g is the label of the (only) rule defining symbol g for $g \in \{\mathbf{f}, \mathbf{a}, \mathbf{b}\}$. Then,

$$\begin{aligned} norm_\zeta(\mathbf{f}(\mathbf{a})) &= norm_\zeta(\mathbf{f}(\mathbf{a}), [\alpha_{\mathbf{f}}, 1]) \\ &= norm_\zeta(\mathbf{f}(\mathbf{a}), [1]) \\ &= norm_\zeta(\mathbf{f}(norm_\zeta(\mathbf{a})), nil) \\ &= norm_\zeta(\mathbf{f}(norm_\zeta(\mathbf{a}, [\alpha_{\mathbf{a}}])), nil) \\ &= norm_\zeta(\mathbf{f}(norm_\zeta(\mathbf{b})), nil) \\ &= norm_\zeta(\mathbf{f}(norm_\zeta(\mathbf{b}, [\alpha_{\mathbf{b}}])), nil) \\ &= norm_\zeta(\mathbf{f}(norm_\zeta(\mathbf{c})), nil) \\ &= norm_\zeta(\mathbf{f}(norm_\zeta(\mathbf{c}, nil)), nil) \\ &= norm_\zeta(\mathbf{f}(\mathbf{c}), nil) \\ &= \mathbf{f}(\mathbf{c}) \end{aligned}$$

However, the only possible \rightarrow_ζ -reduction on $\mathbf{f}(\mathbf{a})$ is:

$$\mathbf{f}(\underline{\mathbf{a}}) \rightarrow_\zeta \underline{\mathbf{f}(\mathbf{b})} \rightarrow_\zeta \mathbf{c}$$

Van de Pol proves the following.

Theorem 8 [Pol01] *If ζ is in-time, then $norm_\zeta(t)$ is the last element of the maximal \rightarrow_ζ -reduction sequence starting from t .*

We can give another interesting result regarding the connection between $norm_\zeta$ and CSR (without requiring ‘in-time’).

Theorem 9 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, ζ be a strategy annotation and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If $norm_\zeta(t)$ is defined for t , then $t \hookrightarrow_{\mu\zeta}^* norm_\zeta(t)$.*

5.2 Termination of programs with Van de Pol's strategy annotations

One is tempted to say that a TRS is ς -terminating if \rightarrow_ς is terminating. Of course, according to Theorem 7, we have the following immediate consequence.

Theorem 10 *Let \mathcal{R} be a TRS and ς be a strategy annotation. If \mathcal{R} is μ^ς -terminating, then \rightarrow_ς is terminating.*

Unfortunately, termination of \rightarrow_ς does not imply that $norm_\varsigma(t)$ is defined for all terms t .

Example 20 *Consider the following TRS:*

$$\begin{array}{l} \mathbf{f}(\mathbf{b}) \rightarrow \mathbf{c} \qquad \mathbf{b} \rightarrow \mathbf{f}(\mathbf{a}) \\ \mathbf{a} \rightarrow \mathbf{b} \end{array}$$

together with $\varsigma(\mathbf{f}) = [\alpha_{\mathbf{f}}, 1]$, $\varsigma(\mathbf{a}) = [\alpha_{\mathbf{a}}]$, $\varsigma(\mathbf{b}) = [\alpha_{\mathbf{b}}]$, and $\varsigma(\mathbf{c}) = nil$. Since the only possible \rightarrow_φ -reduction on $\mathbf{f}(\mathbf{a})$ is:

$$\mathbf{f}(\mathbf{a}) \rightarrow_\varsigma \underline{\mathbf{f}(\mathbf{b})} \rightarrow_\varsigma \mathbf{c}$$

it is not difficult to see that \mathcal{R} is \rightarrow_ς -terminating. However, $norm_\varsigma(\mathbf{f}(\mathbf{a}))$ is not defined:

$$\begin{aligned} norm_\varsigma(\mathbf{f}(\mathbf{a})) &= norm_\varsigma(\mathbf{f}(\mathbf{a}), [\alpha_{\mathbf{f}}, 1]) \\ &= norm_\varsigma(\mathbf{f}(\mathbf{a}), [1]) \\ &= norm_\varsigma(\mathbf{f}(norm_\varsigma(\mathbf{a})), nil) \\ &= norm_\varsigma(\mathbf{f}(norm_\varsigma(\mathbf{a}, [\alpha_{\mathbf{a}}])), nil) \\ &= norm_\varsigma(\mathbf{f}(norm_\varsigma(\mathbf{b})), nil) \\ &= norm_\varsigma(\mathbf{f}(norm_\varsigma(\mathbf{b}, [\alpha_{\mathbf{b}}])), nil) \\ &= norm_\varsigma(\mathbf{f}(norm_\varsigma(\mathbf{f}(\mathbf{a}))), nil) \\ &= \dots \end{aligned}$$

Of course, Theorem 8 implies that, for in-time strategy annotations ς , termination of \rightarrow_ς guarantees that $norm_\varsigma(t)$ is defined for all terms t . However, we prefer to give a ‘semantics-oriented’ definition of termination (as done for the E -strategies as well) which does not depend of considering ‘in-time’ strategies. Moreover, since the JITty system actually implements the function $norm_\varsigma$ [Pol01, Section 4], the following definition better corresponds to the real use of van de Pol's strategy annotations.

Definition 6 *A TRS \mathcal{R} is ς -terminating if $norm_\varsigma(t)$ is defined for all $t \in T(\mathcal{F}, \mathcal{X})$.*

Fortunately, termination of CSR is also useful in this setting.

Theorem 11 *Let \mathcal{R} be a TRS and ς be a strategy annotation for \mathcal{R} . If \mathcal{R} is μ^ς -terminating, then \mathcal{R} is ς -terminating.*

Example 21 Consider the following JITty module `check-full-arg` which is part of the installation of the tool⁵.

```
signature

  0(0)
  S(1)
  nth(2)
  cons(2)
  nil(0)
  from(1)

rules

  n1([x,y],nth(0,cons(x,y)),x)
  n2([m,x,y],nth(S(m),cons(x,y)),nth(m,y))

  f1([x],from(x),cons(x,from(S(x))))

default justintime

strategies

  cons([1])

end

stop
```

Viewed as a TRS, this program is not terminating (in fact, this is the only non-terminating TRS among all examples in the JITty installation). However, the TRS \mathcal{R}

$$\begin{aligned} \text{nth}(0, \text{cons}(x, y)) &\rightarrow x \\ \text{nth}(s(m), \text{cons}(x, y)) &\rightarrow \text{nth}(m, y) \\ \text{from}(x) &\rightarrow \text{cons}(x, \text{from}(s(x))) \end{aligned}$$

is μ^ς -terminating where $\mu^\varsigma(\text{cons}) = \{1\}$ and $\mu^\varsigma(f) = \{1, \dots, ar(f)\}$ for all other symbols f . This corresponds to the strategy annotation ς which the module `check-full-arg` specifies. The μ^ς -termination of \mathcal{R} can be automatically proved (see Example 28 below and Appendix C).

In general, ς -termination does not imply μ^ς -termination.

Example 22 Consider the TRS \mathcal{R} :

$$\begin{aligned} \alpha : b &\rightarrow a \\ \beta : b &\rightarrow c(b) \end{aligned}$$

⁵See also <http://homepages.cwi.nl/~vdpol/jitty/examples>.

and ς given by $\varsigma(\mathbf{b}) = [\alpha, \beta]$ and $\varsigma(\mathbf{c}) = [1]$. Then, \mathcal{R} is clearly ς -terminating, but it is not μ^ς -terminating.

Regarding some related work, as far as the author knows, no analysis of termination for van de Pol's strategy annotations has been reported to date.

5.3 Comparing E -strategies and Van de Pol's strategies

Van de Pol's strategy annotations include not only indices of arguments of function symbols but also rules defining function symbols. Occurrences of 0 in E -strategies can be thought of as abstractions of these items. Given a strategy annotation ς , it is immediate to obtain a 'corresponding' E -strategy map φ by replacing, for each $f \in \mathcal{F}$, rule items $l \rightarrow r$ in $\varsigma(f)$ by 0 (and removing contiguous occurrences of 0)

Example 22 shows that ς -termination of a TRS does *not* imply φ -termination for the corresponding E -strategy map, since a local strategy (0) for \mathbf{b} in Example 22 would lead to nonterminating computations with the E -strategy (note that, in contrast to \rightarrow_ς or $norm_\varsigma$, \rightarrow_φ is not deterministic). Moreover, φ -termination does not imply ς -termination either.

Example 23 Consider the following TRS (based on that of Example 18)

$$\begin{array}{ll} \alpha : \text{if}(\text{true}, \mathbf{x}, \mathbf{y}) \rightarrow \mathbf{x} & \gamma : \text{if}(\mathbf{x}, \mathbf{y}, \mathbf{y}) \rightarrow \mathbf{a} \\ \beta : \text{if}(\text{false}, \mathbf{x}, \mathbf{y}) \rightarrow \mathbf{y} & \delta : \mathbf{b} \rightarrow \text{if}(\mathbf{a}, \mathbf{b}, \mathbf{b}) \end{array}$$

and $\varsigma(\text{if}) = [1, \alpha, \beta, 2, 3, \gamma]$, $\varsigma(\mathbf{b}) = [\delta]$. The corresponding E -strategy map φ is $\varphi(\text{if}) = (1 \ 0 \ 2 \ 3 \ 0)$, $\varphi(\mathbf{b}) = (0)$. Then, \mathcal{R} is not ς -terminating (since ς is 'in-time', we just show that \rightarrow_ς is not terminating):

$$\underline{\mathbf{b}} \rightarrow_\varsigma \text{if}(\mathbf{a}, \underline{\mathbf{b}}, \mathbf{b}) \rightarrow_\varsigma \text{if}(\mathbf{a}, \text{if}(\mathbf{a}, \underline{\mathbf{b}}, \mathbf{b}), \mathbf{b}) \rightarrow_\varsigma \dots$$

However, \mathcal{R} is φ -terminating, since we have:

$$\begin{aligned} \langle \underline{\mathbf{b}}_{(0)}, \Lambda \rangle &\rightarrow_\varphi \langle \text{if}_{(1 \ 0 \ 2 \ 3 \ 0)}(\mathbf{a}_{nil}, \mathbf{b}_{(0)}, \mathbf{b}_{(0)}), \Lambda \rangle \\ &\rightarrow_\varphi^+ \langle \underline{\text{if}}_{(0 \ 2 \ 3 \ 0)}(\mathbf{a}_{nil}, \mathbf{b}_{(0)}, \mathbf{b}_{(0)}), \Lambda \rangle \rightarrow_\varphi \langle \mathbf{a}_{nil}, \Lambda \rangle \end{aligned}$$

6 Proving termination of programs in practice

Automatic proofs of termination are always desirable although difficult (and not always possible due to undecidability of termination [HL78]). Well-founded orderings on terms (i.e., orderings $>$ allowing no infinite sequence $t_1 > t_2 > \dots > t_n > \dots$) provide a suitable basis for proving termination in a number of programming languages and computational systems. For instance, dealing with TRSs a proof of termination can be achieved if we are able to find a monotone and stable well-founded ordering $>$ on terms (i.e., a *reduction ordering*) such that $l > r$ for every rule $l \rightarrow r$ of the rewrite system [Der87, Zan03]. Recall that *monotone* means that, whenever $t > s$, we have $f(t_1, \dots, t_{i-1}, t, \dots, t_k) > f(t_1, \dots, t_{i-1}, s, \dots, t_k)$ for all symbols $f \in \mathcal{F}$, $i \in \{1, \dots, ar(f)\}$ and terms

t, s, t_1, \dots, t_k . *Stable* means that, whenever $t > s$, we have $\sigma(t) > \sigma(s)$ for all terms t, s and substitutions σ .

In practice, if we want to implement a tool for proving termination of TRSs, we need to eventually generate well-founded orderings. *Simplification orderings* are those monotone and stable orderings $>$ satisfying the following subterm property: for each term t , $t > s$ for every proper subterm s of t [Der79]. Well-foundedness of simplification orderings is a consequence of the subterm property (when considering finite signatures). If termination of a TRS \mathcal{R} can be proved by using a simplification ordering, then we say that \mathcal{R} is simply terminating. Although simple termination is also undecidable [MG95] it covers most usual automatizable orderings for proving termination of rewriting: e.g., recursive path orderings (RPO [Der82]), Knuth-Bendix orderings (KBO [KB70]), and polynomial orderings (POLY [Lan79]), see [Ste95] for a survey on simplification orderings.

Zanema has shown that termination of *CSR* is fully captured by the so-called μ -reduction orderings, i.e., well-founded, stable orderings $>$ such that, whenever $t > s$, we have $f(t_1, \dots, t_{i-1}, t, \dots, t_k) > f(t_1, \dots, t_{i-1}, s, \dots, t_k)$ for all symbols $f \in \mathcal{F}$, $i \in \mu(f)$ and terms t, s, t_1, \dots, t_k (μ -monotony). Then, a TRS $\mathcal{R} = (\mathcal{F}, R)$ is μ -terminating if and only if there is a μ -reduction ordering $>$ which is *compatible* with the rules of \mathcal{R} , i.e., for all $l \rightarrow r \in R$, $l > r$ ([Zan97, Proposition 1]). Regarding the automatization of proofs of termination of *CSR*, the corresponding notion of simple termination of *CSR* has been investigated in [GL02b]. We let $\mathcal{E}mb^\mu(\mathcal{F}) = (\mathcal{F}, \{f(x_1, \dots, x_k) \rightarrow x_i \mid f \in \mathcal{F}, i \in \mu(f)\})$.

Definition 7 (Simple termination of *CSR* [GL02b]) *Let \mathcal{R} be a TRS and $\mu \in M_{\mathcal{R}}$. Then, \mathcal{R} is simply μ -terminating if $\mathcal{R} \cup \mathcal{E}mb^\mu(\mathcal{F})$ is μ -terminating.*

As for standard rewriting, Definition 7 actually provides a unifying framework for proving termination of *CSR* by using standard simplification orderings via the existing methods based on transformations (see Section 6.1 below), and also covers CSRPO, a very recent proposal that extends the recursive path ordering (RPO) to context-sensitive terms [BLR02] and the use of polynomial interpretations to generate orderings which can be used to directly prove (simple) termination of *CSR* [GL02b] (see Section 6.2).

On the other hand, a proof of *non-simple* μ -termination for a TRS \mathcal{R} immediately gives us the following interesting information: we cannot hope to prove termination of \mathcal{R} by using the previous orderings. Thus, the use of transformations becomes necessary.

In the following sections, we discuss the practic use of these techniques for proving termination of programs with strategy annotations.

6.1 Transformations for proving termination of *CSR*

In [FR99, GM99, GM03b, Luc96, Zan97] different transformations Θ from TRSs \mathcal{R} and replacement maps μ into TRSs \mathcal{R}_Θ^μ have been described (see also [GL02b, GM03a, GM03b, Luc02c] for comparisons among them). All these transformations are correct, in the sense that, if we are able to prove *termination* of \mathcal{R}_Θ^μ ,

(using the standard methods, see [Der87, Zan03]), then the μ -termination of \mathcal{R} is guaranteed (see Example 11 above).

We have developed a tool, MUTERM [Luc03a], which implements all these transformations using a common environment that eases their combination and comparison. MUTERM does not provide any method for proving termination of the transformed systems. We rather provide interfaces for using already existing tools for proving termination of rewriting, namely the CiME 2.0 system [CM96, CMMU03] AProVE [GTSF03], etc. MUTERM is available at <http://www.dsic.upv.es/~slucas/csr/termination/muterm>.

This combination permits automatic proofs of termination of Maude or OBJ programs.

Example 24 *We can use MUTERM to read program FIRST_FROM_LENGTH in Example 2, transforming it into the TRS $\mathcal{R}_L^{\mu^\varphi}$ of Example 11, and finally (automatically) proving termination of $\mathcal{R}_L^{\mu^\varphi}$ by using CiME 2.0. We exemplify the process in Appendix B.*

Finally, regarding the ‘dissuasive’ sense of a proof of non-simple μ -termination for a TRS \mathcal{R} , we have the following fact which can be used as a suitable guide in any attempt to prove termination of a program by using these transformations:

Any proof of simple termination of \mathcal{R}_Θ^μ for any transformation Θ described in [FR99, GM03b, Luc96, Zan97] implies that \mathcal{R} is simply μ -terminating [GL02b, Theorems 2, 5, 6, 7, and 8].

Therefore, if we are able to show that \mathcal{R} is not simply μ -terminating, we should not try to prove termination of \mathcal{R}_Θ^μ by just using simplification orderings (RPO, KBO, POLY, ...).

Example 25 *Consider the program DIFF in Example 1. We know that the program is terminating (when executed in an OBJ3 interpreter which uses E-strategy (1 0) for the if_then_else_fi operator), see Example 16. However, this program is not simply μ^φ -terminating (for μ^φ as in Example 16):*

$$\begin{aligned} \underline{s(0) - 0} &\leftrightarrow \text{if } s(0) \leq 0 \text{ then } 0 \text{ else } s(p(s(0)) - 0) \text{ fi} \\ &\leftrightarrow \underline{\text{if false then } 0 \text{ else } s(p(s(0)) - 0) \text{ fi}} \\ &\leftrightarrow \underline{s(p(s(0)) - 0)} \\ &\xrightarrow{\mathcal{E}_{mb\mu^\varphi}(\mathcal{F})} \underline{p(s(0)) - 0} \\ &\xrightarrow{\mathcal{E}_{mb\mu^\varphi}(\mathcal{F})} \underline{s(0) - 0} \\ &\leftrightarrow \dots \end{aligned}$$

In fact, the proof reported by Giesl and Middeldorp [GM02, Appendix A.2] is based on the analysis of the dependency pairs [AG00] generated by the system obtained after applying their transformation [GM03a].

6.2 Direct methods for proving termination of CSR

Most of aforementioned transformations use new symbols and rules that introduce a loss of structure and intuition, due to the encoding of the context-

sensitive control in the original system by such new elements. They can unnecessarily complicate the proofs of termination and make standard techniques for easing such proofs (e.g., modular approaches) infeasible. In fact, despite the fact that some completeness results (see [GM03a, GM03b, Luc02c]) support the use of some of these transformations as theoretically able to cope with termination of *CSR* in all cases (i.e., they are, not only correct, but also complete), from a more practical point of view some problems arise with them (see [BLR02, Luc02c]). This state of things motivated, very recently, some work devoted to the *direct* analysis of termination by defining suitable orderings for proving termination of *CSR* [Bor03, BLR02, GL02b].

As an example of the use of such direct techniques, let us consider polynomial orderings for *CSR*. Zantema showed that μ -reduction orderings can be defined by means of (well-founded) μ -monotone \mathcal{F} -algebras. An (ordered) \mathcal{F} -algebra, is a triple $(A, \mathcal{F}_A, >)$, where A is a set, \mathcal{F}_A is a set of mappings $f_A : A^k \rightarrow A$ for each $f \in \mathcal{F}$ where $k = ar(f)$, and $>$ is a (strict) ordering on A . Given $f \in \mathcal{F}$, we say that $f_A : A^k \rightarrow A$ is μ -monotone if for all $a, b \in A$ such that $a > b$, we have that

$$f_A(a_1, \dots, a_{i-1}, a, \dots, a_k) > f_A(a_1, \dots, a_{i-1}, b, \dots, a_k)$$

for all $a_1, \dots, a_k \in A$ and $i \in \mu(f)$. An \mathcal{F} -algebra $\mathcal{A} = (A, \mathcal{F}_A, >)$ is μ -monotone if for all $f \in \mathcal{F}$, f_A is μ -monotone. An \mathcal{F} -algebra $\mathcal{A} = (A, \mathcal{F}_A, >)$ is well-founded if $>$ is well-founded. For a given valuation mapping $\alpha : \mathcal{X} \rightarrow A$, the evaluation mapping $[\alpha] : \mathcal{T}(\mathcal{F}, \mathcal{X}) \rightarrow A$ is inductively defined by $[\alpha](x) = \alpha(x)$ if $x \in \mathcal{X}$ and $[\alpha](f(t_1, \dots, t_k)) = f_A([\alpha](t_1), \dots, [\alpha](t_k))$ for $x \in \mathcal{X}$, $f \in \mathcal{F}$, $t_1, \dots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. We write $t >_A s$ if and only if $[\alpha](t) > [\alpha](s)$ for all $\alpha : \mathcal{X} \rightarrow A$. Then, $>_A$ is a μ -reduction ordering on terms for every well-founded μ -monotone \mathcal{F} -algebra $(A, \mathcal{F}_A, >)$ ([Zan97, Proposition 2]). Of course, as for unrestricted rewriting, we can use polynomial interpretations fulfilling these conditions as suitable \mathcal{F} -algebras to induce μ -reduction orderings.

Example 26 *The following TRS \mathcal{R} can be used to approximate $\pi^2/6$ [Luc02a, Example 2 & Example 38]:*

$$\begin{array}{llll} \text{sqr}(0) & \rightarrow 0 & 0 + x & \rightarrow x \\ \text{sqr}(s(x)) & \rightarrow s(\text{sqr}(x) + \text{dbl}(x)) & s(x) + y & \rightarrow s(x+y) \\ \text{dbl}(0) & \rightarrow 0 & \text{fst}(0, x) & \rightarrow \text{nil} \\ \text{dbl}(s(x)) & \rightarrow s(\text{dbl}(x)) & \text{fst}(s(x), y : z) & \rightarrow y : \text{fst}(x, z) \\ \text{terms}(n) & \rightarrow \text{recip}(\text{sqr}(n)) : \text{terms}(s(n)) & & \end{array}$$

Let $\mu(s) = \emptyset$, $\mu(\cdot) = \{1\}$, and $\mu(f) = \{1, \dots, k\}$ for any other k -ary symbol f . The μ -termination of \mathcal{R} can be proved with the following (linear) polynomial interpretation over the natural numbers:

$$\begin{array}{ll} [\text{sqr}](x) & = x + 1 & [\text{terms}](x) & = x + 2 \\ [0] & = 0 & [\text{recip}](x) & = x \\ [s](x) & = 0 & x [:] y & = x \\ x [+] y & = x + y + 1 & [\text{fst}](x, y) & = x + y + 1 \\ [\text{dbl}](x) & = x + 1 & [\text{nil}] & = 0 \end{array}$$

which fulfills the conditions given in [GL02b, Section 5.2] to induce a μ^φ -reduction ordering. Thus, we can use it to directly prove termination of the program by comparing the left- and right-hand sides of rules:

$$\begin{array}{lll}
[\text{sqr}(0)] & = & 1 > 0 & = & [0] \\
[\text{sqr}(s(x))] & = & 1 > 0 & = & [s(\text{sqr}(x)+\text{dbl}(x))] \\
[\text{dbl}(0)] & = & 1 > 0 & = & [0] \\
[\text{dbl}(s(x))] & = & 1 > 0 & = & [s(s(\text{dbl}(x)))] \\
[\text{terms}(n)] & = & n + 2 > n + 1 & = & [\text{recip}(\text{sqr}(n)) : \text{terms}(s(n))] \\
[0 + x] & = & x + 1 > x & = & [0] \\
[s(x) + y] & = & y + 1 > 0 & = & [s(x+y)] \\
[\text{fst}(0,x)] & = & x + 1 > 0 & = & [\text{nil}] \\
[\text{fst}(s(x),y:z)] & = & y + 1 > y & = & [y:\text{fst}(x,z)]
\end{array}$$

Another interesting application of polynomial orderings comes with the following example:

Example 27 *The following TRS \mathcal{R} can be used to compute the quotient of two natural numbers [GM03b, Example 49]:*

$$\begin{array}{lll}
0 - y & \rightarrow & 0 & 0 \div s(y) & \rightarrow & 0 \\
s(x) - s(y) & \rightarrow & x - y & s(x) \div s(y) & \rightarrow & \text{if}(x \geq y, s((x-y) \div s(y)), 0) \\
x \geq 0 & \rightarrow & \text{true} & \text{if}(\text{true}, x, y) & \rightarrow & x \\
0 \geq s(y) & \rightarrow & \text{false} & \text{if}(\text{false}, x, y) & \rightarrow & y \\
s(x) \geq s(y) & \rightarrow & x \geq y & & &
\end{array}$$

Viewed as a TRS, \mathcal{R} is not simply terminating. As in [GM03b, Example 49], let $\mu(\text{if}) = \mu(\div) = \mu(s) = \{1\}$ and $\mu(f) = \emptyset$ for any other symbol f . The μ -termination of \mathcal{R} can be proved by using the following polynomial interpretation over the natural numbers:

$$\begin{array}{lll}
x[-]y & = & x + 1 & [\text{true}] & = & 0 \\
[0] & = & 0 & [\text{false}] & = & 1 \\
[s](x) & = & x + 3 & x[\div]y & = & x^2 + x + 1 \\
x[\geq]y & = & x + 2 & [\text{if}](x, y, z) & = & xz + x + y + 1
\end{array}$$

which can be used to compare left- and right-hand sides of rules:

$$\begin{array}{lll}
[0 - y] & = & 1 > 0 & = & [0] \\
[s(x) - s(y)] & = & x + 4 > x + 1 & = & [x - y] \\
[x \geq 0] & = & x + 2 > 0 & = & [\text{true}] \\
[0 \geq s(y)] & = & 2 > 1 & = & [\text{false}] \\
[s(x) \geq s(y)] & = & x + 5 > x + 2 & = & [x \geq y] \\
[0 \div s(y)] & = & 1 > 0 & = & [0] \\
[s(x) \div s(y)] & = & x^2 + 7x + 13 > x^2 + 4x + 9 & = & [\text{if}(x \geq y, s((x-y) \div s(y)), 0)] \\
[\text{if}(\text{true}, x, y)] & = & x + 1 > x & = & [x] \\
[\text{if}(\text{false}, x, y)] & = & x + y + 2 > y & = & [y]
\end{array}$$

This means that \mathcal{R} is simply μ -terminating.

Nowadays, there is no implementation of a system which is able to automatically generate these polynomials yet. However, it is already possible to use *CiME 2.0* to (try to) solve the corresponding Diophantine constraints that are obtained when the conditions in [GL02b] are imposed to a polynomial interpretation with indeterminate coefficients. This is analogous to the procedure used for proving polynomial termination of TRSs [Lan79] but the set of constraints is, of course, different due to the lack of certain monotonicity restrictions. Solving the set of Diophantine constraints in such indeterminates would implicitly yield a proof of μ -termination of the TRS and also permits to retrieve the concrete polynomial interpretation which can be used to do it. The implementation of this procedure is a subject of future (ongoing) work.

Example 28 *Consider the JITy module `check-full-arg` in Example 21. Termination of the module can be proved by using the CSRPO of [BLR02]. In [BLR02, Example 7], the CSRPO is used to prove μ -termination of a TRS which strictly contains the rules of `check-full-arg` (with the same replacement restrictions for the shared symbols). Thus, termination of `check-full-arg` easily follows.*

As shown in Example 25, though, program `DIFF` is *not* simply μ^φ -terminating. Since both CSRPO and the polynomial orderings for *CSR* of [GL02b] are simplification orderings for *CSR* ([GL02b, Theorems 9 and 11⁶]), no current direct method applies to `DIFF`.

7 Conclusions and future work

Syntactic annotations have been used in `OBJ2`, `OBJ3`, `Maude`, and `CafeOBJ` for many years (as *E*-strategies). However, only recently (but quite intensively) has been addressed the formal analysis of computations with `OBJ`'s local strategies (see, for instance, [AEGL02, AEL02, Eke98, FGK01, Luc02b, Luc01a, Nag99, NO01, OF97, OF00, Pol01]). Regarding termination, though, only [FGK01] discusses a technique for proving termination of *E*-strategies. No analysis of termination for Van de Pol's strategies has been reported to date.

In this paper, we have investigated the problem of proving termination of programs which execute under control of strategy annotations. More concretely, we have investigated how to prove termination of rewriting under *E*-strategies and under van de Pol's strategy annotations. We summarize our contributions as follows:

Foundational aspects. We have proved that context-sensitive rewriting provide an appropriate abstract model for describing computations using *E*-strategies (Theorems 1 and 3) and Van de Pol's strategy annotations (Theorems 7 and 9). Thus, *CSR* provides a good framework to analyze

⁶The fact that Theorem 11 trivially holds for the considered kind of polynomial μ -orderings was pointed out by Aart Middeldorp.

computations of programs using these strategy annotations (e.g., *Maude*, *OBJ2*, *OBJ3*, *CafeOBJ*, and *JITty* programs).

Analysis of termination. We have proved that termination of *CSR* is a sufficient condition to prove termination of *Maude*, *OBJ2*, *OBJ3*, and *CafeOBJ* programs (Theorem 2), although not necessary in the general case (Example 12). For the class of *Maude*, *OBJ2*, *OBJ3*, and *CafeOBJ* programs using elementary strategies, the analysis of termination of innermost *CSR* provides a correct and complete characterization of termination of these programs (Theorems 4 and 5). Regarding Van de Pol’s strategy annotations and *JITty* programs, termination of *CSR* also suffices to prove termination (Theorems 10 and 11). Both kinds of strategy annotations are different regarding their termination behavior, as discussed in Section 5.3, where we have shown that, in general, these problems are not comparable (see Examples 22 and 23).

Proofs of termination in practice. Fortunately, a number of techniques have been developed for proving termination of (innermost) *CSR* [BLR02, FR99, GL02a, GL02b, GM99, GM03a, GM03b, Luc96, Luc01b, Luc03a, Luc02c, SX98, Zan97]. Also, there is a tool, *MUTERM* [Luc03a], which can be used to automatically prove termination of *CSR* by using transformations together with any available tool for proving termination of rewriting such as *CiME* [CMMU03] and *AProVE* [GTSF03]. The tool accepts programs in different formats (also as simple *Maude* and *OBJ** programs). This provides a first practical framework to prove termination of programs with strategy annotations (see Examples 21 and 24 and Appendices B and C). The use of direct methods (polynomial orderings, *CSRPO*, etc.) is also possible (Examples 26, 27, and 28) and there are good perspectives to implement them in the near future.

Other eager programming languages such as *ELAN* [BKKMR98] incorporate the specification of syntactic replacement restrictions as an ingredient of the definition of more complex rewriting strategies which can be used to guide the evaluation of expressions (see [Luc02a]). The results in this paper can be helpful to prove termination of such kind of *ELAN* programs.

7.1 Future work

Although proving termination of TRSs that admit the specification of strategy annotations is closer to termination of *Maude* and *OBJ** programs than termination of rewriting, more work has to be done to obtain more powerful tools for proving termination of such programs. The reason is that the such programming languages permit a lot of sophisticated features such as:

1. the specification of commutative and/or associative (AC) operators,
2. the use of *modules* for writing large programs,

3. the use of equational theories together with the rewrite rules (equational rewriting),
4. the use of sorts and sorted rewriting,
5. the *priority* of applying some rules according to (some) strategy annotations. This can be explicit, as in van de Pol's approach, but also implicit (e.g., the order of appearance of rules in the text of the program).

Regarding 1 above, termination of AC-*CSR* has already been studied in [FR99, GM03b]. Regarding 2, modularity of (innermost, simple) termination of *CSR* has been studied in [GL02a, GL02b]. More work remains to be done to cope with the other features. On the other hand, programs of the most recent version of *Maude*, *Maude* 2.0 [CDEL⁺03], also permits the use of conditional rules whose conditional part permit equational conditions, membership tests, and pattern matching operations in the framework of membership equational logic [BJM00] and Generalized Rewrite Theories [BM03]. Dealing with this kind of rules regarding proofs of termination is also a subject for future work. In this sense, there is an ongoing collaboration with the *Maude* group lead by José Meseguer to use these and other results to provide the *Maude* interpreter with new verification tools.

Finally, a number of programming languages (e.g., ELAN [BKKMR98], *Stratego* [Vis99], and also *Maude*, via the so-called *internal strategies* [CDEL⁺02]) permit very general strategies to be *explicitly specified* as part of the program using a given *strategy language*. Unfortunately, analyzing how such programs actually behave is not easy. In fact, the analysis of computational properties (e.g., termination) of programs supplied with programmable strategies is a challenging and open problem. This is also important for the practical use of these languages, since such specification of program control is a kind of 'non-declarative' feature which needs to be supported with appropriate analysis techniques.

References

- [AEGL02] M. Alpuente, S. Escobar, B. Gramlich, and S. Lucas. Improving On-Demand Strategy Annotations. In Matthias Baaz and Andrei Voronkov, editors, *Proc. 9th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR'02*, LNAI 2514:1-18, Springer-Verlag, Berlin, 2002.
- [AEL02] M. Alpuente, S. Escobar, and S. Lucas. Correct and complete (positive) strategy annotations for OBJ. In F. Gadducci and U. Montanari, editors, *Proc. of the 4th International Workshop on Rewriting Logic and its Applications, WRLA'02, Electronic Notes in Theoretical Computer Science*, volume 71. Elsevier Sciences, to appear 2002.
- [AG97] T. Arts and J. Giesl. Proving Innermost Normalisation Automatically. In H. Comon, editor, *Proc. of 8th International Conference on Rewriting Techniques and Applications, RTA'97*, LNCS 1232:157-171, Springer-Verlag, Berlin, 1997.

- [AG00] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs *Theoretical Computer Science*, 236:133-178, 2000.
- [BFGL⁺01] S. Blom, W. Fokkink, J. Groote, I. van Langevelde, B. Lissner, and J. van de Pol. μ CRL: a toolset for analysing algebraic specifications. In G. Berry, H. Comon, A. Finkel, editors, *13th International Conference on Computer Aided Verification, CAV'01*, LNCS 2102:250-254, Springer-Verlag, Berlin, 2001.
- [BJM00] A. Bouhoula, J.-P. Jouannaud, and J. Meseguer. Specification and proof in membership equational logic. *Theoretical Computer Science*, 236:35-132, 2000.
- [BKKMR98] P. Borovanský, C. Kirchner, H. Kirchner, P.-E. Moreau, and C. Ringesisen. An Overview of ELAN. In C. Kirchner and H. Kirchner, editors, *Proc. of 2nd International Workshop on Rewriting Logic and its Applications, WRLA'98*, Electronic Notes in Computer Science, 15(1998):1-16, 1998.
- [BLR02] C. Borralleras, S. Lucas, and A. Rubio. Recursive Path Orderings can be Context-Sensitive. In A. Voronkov, editor *Proc. of 18th International Conference on Automated Deduction, CADE'02*, LNAI 2392:314-331, Springer-Verlag, Berlin, 2002.
- [BM03] R. Bruni and J. Meseguer. Generalized Rewrite Theories. In J.C.M. Baeten, J.K. Lenstra, J. Parrow, and G.J. Woeginger (editors), *Proc. of 30th International Colloquium on Automata, Languages and Programming, ICALP'03*, LNCS 2719:252-266, Springer-Verlag, Berlin, 2003.
- [BN98] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
- [Bor03] C. Borralleras. Ordering-based methods for proving termination automatically. PhD Thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, May 2003.
- [CDEL⁺02] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J.F. Quesada. Maude: specification and programming in rewriting logic. *Theoretical Computer Science* 285(2):187-243, 2002.
- [CDEL⁺03] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. The Maude 2.0 System. In R. Nieuwenhuis, editor, *Proc. of 14th International Conference on Rewriting Techniques and Applications, RTA'03*, LNCS 2706:76-87, Springer-Verlag, Berlin, 2003.
- [CM96] E. Contejean and C. Marché. CiME: Completion Modulo E. In H. Ganzinger, editor, *Proc. of 7th International Conference of Rewriting Techniques and Applications, RTA'96*, LNCS 1103:416-419, Springer-Verlag, Berlin, 1996. Available at <http://cime.lri.fr>.
- [CMMU03] E. Contejean and C. Marché, B. Monate and X. Urbain. Proving termination of rewriting with CiME. In A. Rubio, editor, *Proc. of 6th International Workshop on Termination, WST'03*, pages 71-73, Technical Report DSIC II/15/03, Valencia, Spain, 2003. Available at <http://cime.lri.fr>.
- [Der79] N. Dershowitz. A note on simplification orderings. *Information Processing Letters*, 9(5):212-215, 1979.

- [Der82] N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.
- [Der87] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69-115, 1987.
- [Eke98] S. Eker. Term Rewriting with Operator Evaluation Strategies. In C. Kirchner and H. Kirchner, editors, *Proc. of 2nd International Workshop on Rewriting Logic and its Applications, WRLA '98*, Electronic Notes in Computer Science, 15(1998):1-20, 1998.
- [FGJM85] K. Futatsugi, J. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Conference Record of the 12th Annual ACM Symposium on Principles of Programming Languages, POPL '85*, pages 52-66, ACM Press, 1985.
- [FGK01] O. Fissore, I. Gnaedig, and H. Kirchner. Induction for termination with local strategies. *Electronic Notes in Theoretical Computer Science*, volume 58(2), 2001.
- [FGK02] O. Fissore, I. Gnaedig, and H. Kirchner. CARIBOO: An Induction Based Proof Tool for Termination with Strategies - System Presentation -. In C. Kirchner, editor, *Proc. of 4th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP '02*, pages 50-61, ACM Press, New York, 2002.
- [FN97] K. Futatsugi and A. Nakagawa. An Overview of CAFE Specification Environment – An algebraic approach for creating, verifying, and maintaining formal specification over networks –. In *Proc. of 1st International Conference on Formal Engineering Methods*, 1997.
- [FR99] M.C.F. Ferreira and A.L. Ribeiro. Context-Sensitive AC-Rewriting. In P. Narendran and M. Rusinowitch, editors, *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA '99*, LNCS 1631:286-300, Springer-Verlag, Berlin, 1999.
- [FW76] D.P. Friedman and D.S. Wise. CONS should not evaluate its arguments. In S. Michaelson and R. Milner, editors, *Automata, Languages and Programming*, pages 257-284, Edinburgh University Press, 1976.
- [GF01] I. Gnaedig and O. Fissore. Personal communication. July 2001.
- [GL01] B. Gramlich and S. Lucas (editors). *Reduction Strategies in Rewriting and Programming, WRS '01*. Proc. of the 1st International Workshop. *Electronic Notes in Theoretical Computer Science*, volume 57, 2001.
- [GL02a] 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*, pages 50-61, ACM Press, New York, 2002.
- [GL02b] B. Gramlich and S. Lucas. Simple termination of context-sensitive rewriting. In B. Fischer and E. Visser, editors, *Proc. of 3rd ACM SIGPLAN Workshop on Rule-Based Programming, RULE '02*, pages 29-41, ACM Press, New York, 2002.
- [GM99] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. In P. Narendran and M. Rusinowitch, editors, *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA '99*, LNCS 1631:271-285, Springer-Verlag, Berlin, 1999.

- [GM02] J. Giesl and A. Middeldorp. Termination of Innermost Context-Sensitive Rewriting. Technical Report AIB-2002-04, RWTH Aachen, Germany, 2002.
- [GM03a] J. Giesl and A. Middeldorp. Termination of Innermost Context-Sensitive Rewriting. In *Proc. of 6th International Conference on Developments in Language Theory, DLT'02*, LNCS 2450:231-244, Springer-Verlag, Berlin, 2003.
- [GM03b] J. Giesl and A. Middeldorp. Transformation Techniques for Context-Sensitive Rewrite Systems. *Journal of Functional Programming*, to appear, 2003.
- [Gra96] B. Gramlich. On Proving Termination by Innermost Termination. In H. Ganzinger, editor, *Proc. of 7th International Conference on Rewriting Techniques and Applications, RTA '96*, LNCS 1103:97-107, Springer-Verlag, Berlin, 1996.
- [GWMFJ00] J.A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouanaud. Introducing OBJ. In J. Goguen and G. Malcolm, editors, *Software Engineering with OBJ: algebraic specification in action*, Kluwer, 2000.
- [GTSF03] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. AProVE: A System for Proving Termination. In A. Rubio, editor, *Proc. of 6th International Workshop on Termination, WST'03*, pages 68-70, Technical Report DSIC II/15/03, Valencia, Spain, 2003. Available at <http://www-i2.informatik.rwth-aachen.de/AProVE>.
- [HL78] G. Huet and D.S. Lankford. On the uniform halting problem for term rewriting systems. Research Report 283, INRIA, March 1978.
- [HPW92] P. Hudak, S.J. Peyton-Jones, and P. Wadler. Report on the Functional Programming Language Haskell: a non-strict, purely functional language. *Sigplan Notices*, 27(5):1-164, 1992.
- [KB70] D.E. Knuth and P.E. Bendix. Simple Word Problems in Universal Algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263-297. Pergamon Press, 1970.
- [Kri00] M.R.K. Krishna Rao. Some characteristics of strong innermost normalization. *Theoretical Computer Science* 239:141-164, 2000.
- [Lan79] D.S. Lankford. On proving term rewriting systems are noetherian. Technical Report, Louisiana Technological University, Ruston, LA, 1979.
- [Luc96] S. Lucas. Termination of context-sensitive rewriting by rewriting. In F. Meyer auf der Heide and B. Monien, editors, *Proc. of 23rd. International Colloquium on Automata, Languages and Programming, ICALP'96*, LNCS 1099:122-133, Springer-Verlag, Berlin, 1996.
- [Luc98] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1):1-61, January 1998.
- [Luc01a] S. Lucas. Termination of on-demand rewriting and termination of OBJ programs. In *Proc. of 3rd International Conference on Principles and Practice of Declarative Programming, PPDP'01*, pages 82-93, ACM Press, 2001.

- [Luc01b] S. Lucas. Termination of Rewriting With Strategy Annotations. In R. Nieuwenhuis and A. Voronkov, editors, *Proc. of 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'01*, LNAI 2250:669-684, Springer-Verlag, Berlin, 2001.
- [Luc02a] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):293-343, 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.
- [Luc03a] S. Lucas. MU-TERM version 2.0. User's manual. Technical Report DSIC-II/21/03, DSIC, Universidad Politécnica de Valencia, 2003. URL: <http://www.dsic.upv.es/~slucas/csr/termination/muterm>.
- [Luc03b] S. Lucas. Semantics of programs with strategy annotations. Technical Report DSIC-II/08/03, DSIC, Universidad Politécnica de Valencia, 2003.
- [McC60] J. McCarthy. Recursive Functions of Symbolic Expressions and their Computations by Machine, Part I. *Communications of the ACM*, 3(4):184-195, 1960.
- [MG95] A. Middeldorp and B. Gramlich. Simple Termination is Difficult. *Applicable Algebra in Engineering, Communication and Computing*, 6:115-128, 1995.
- [Nag99] T. Nagaya. Reduction Strategies for Term Rewriting Systems. PhD Thesis, School of Information Science, Japan Advanced Institute of Science and Technology, March 1999.
- [NO01] M. Nakamura and K. Ogata. The evaluation strategy for head normal form with and without on-demand flags. In K. Futatsugi, editor, *Proc. of 3rd International Workshop on Rewriting Logic and its Applications, WRLA'00*, *Electronic Notes in Theoretical Computer Science*, volume 36, 17 pages, 2001.
- [NSF99] A.T. Nakagawa, T. Sawada, and K. Futatsugi. CafeOBJ User's Manual -ver 1.4-. Available at <http://www.ldl.jaist.ac.jp/Research/CafeOBJ/>.
- [NSEP92] E.G.J.M.H. Nöcker, J.E.W. Smetsers, M.C.J.D. van Eekelen, and M.J. Plasmeijer. Concurrent Clean. In E.H.L. Aarts, J. Leeuwen, and M. Rem, editors, *Proc. of Parallel Architectures and Languages Europe, PARLE'91*, LNCS 506:202-219, Springer-Verlag, Berlin, 1992.
- [OF97] K. Ogata and K. Futatsugi. Implementation of Term Rewritings with the Evaluation Strategy. In H. Glaser and P. Hartel, editors, *Proc of 9th International Symposium on Programming Languages, Implementations, Logics and Programs, PLILP'97*, LNCS 1292:225-239, Springer-Verlag, Berlin, 1997.
- [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.

- [Pol01] J. van de Pol. Just-in-time: on Strategy Annotations. In [GL01], pages 39-58.
- [Pol02] J. van de Pol. JITty: A Rewriter with Strategy Annotations. In Sophie Tison, editor, *Proc. 13th International Conference on Rewriting Techniques and Applications, RTA'02*, LNCS 2378:367-370, Springer-Verlag, Berlin, 2002.
- [Ste95] J. Steinbach. Simplification orderings: History of results. *Fundamenta Informaticae*, 24:47–88, 1995.
- [SX98] J. Steinbach and H. Xi. Freezing – Termination Proofs for Classical, Context-Sensitive and Innermost Rewriting. Institut für Informatik, T.U. München, January 1998.
- [Vis99] E. Visser. Strategic Pattern Matching. In P. Narendran and M. Rusinowitch, editors, *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA'99*, LNCS 1631:30-44, Springer-Verlag, Berlin, 1999.
- [Zan97] H. Zantema. Termination of Context-Sensitive Rewriting. In H. Comon, editor, *Proc. of 8th International Conference on Rewriting Techniques and Applications, RTA'97*, LNCS 1232:172-186, Springer-Verlag, Berlin, 1997.
- [Zan03] H. Zantema. Termination. In TeReSe, editor, *Term Rewriting Systems*, chapter 6. Cambridge University Press, 2003.

A Appendix

Proofs of Section 4

Proposition 1 *Let \mathcal{R} be a TRS and φ be an E-strategy map for \mathcal{R} . Then,*

$$\text{eval}_\varphi(t) = \{\text{erase}(s) \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \mid \langle \varphi(t), \Lambda \rangle (\xrightarrow{\varphi}^! \circ \xrightarrow{\varphi}^{\text{R}})^! \langle s, p \rangle\}.$$

PROOF. Due to the mutually excluding cases of the definition of \rightarrow_φ , we know that $\xrightarrow{\varphi}^{\text{T}}$ is deterministic, i.e., if $\langle t, p \rangle \xrightarrow{\varphi}^{\text{T}} \langle s, q \rangle$ and $\langle t, p \rangle \xrightarrow{\varphi}^{\text{T}} \langle s', q' \rangle$, then $s = s'$ and $q = q'$. Moreover, $\xrightarrow{\varphi}^{\text{R}}$ -steps are only possible when no further $\xrightarrow{\varphi}^{\text{T}}$ -step can be performed. Then, we have $\xrightarrow{\varphi}^* = (\xrightarrow{\varphi}^! \circ \xrightarrow{\varphi}^{\text{R}} \circ \xrightarrow{\varphi}^!)^* = (\xrightarrow{\varphi}^! \circ \xrightarrow{\varphi}^{\text{R}})^* \circ \xrightarrow{\varphi}^!$. However, since $\text{erase}(t) = \text{erase}(s)$, whenever $\langle t, p \rangle \xrightarrow{\varphi}^{\text{T}} \langle s, q \rangle$, we can drop the last traversing steps in any $\xrightarrow{\varphi}^!$ -sequence. Hence, the conclusion follows. \square

Proposition 2 [Luc98] *Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $p = q.q' \in \mathcal{P}os(t)$. Then, $p \in \mathcal{P}os^\mu(t)$ iff $q \in \mathcal{P}os^\mu(t) \wedge q' \in \mathcal{P}os^\mu(t|_q)$*

In the following proposition (which is used below), we consider the following: The chain of symbols lying on positions above/on $p \in \mathcal{P}os(t)$ is $\text{prefix}_t(\Lambda) = \text{root}(t)$, $\text{prefix}_t(i.p) = \text{root}(t).\text{prefix}_t(i)$. The strict prefix *sprefix* is $\text{sprefix}_t(\Lambda) = \Lambda$, $\text{sprefix}_t(p.i) = \text{prefix}_t(p)$.

Proposition 3 [Luc98] *Let $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If $p \in \mathcal{P}os(t) \cap \mathcal{P}os(s)$ and $\text{sprefix}_t(p) = \text{sprefix}_s(p)$, then $p \in \mathcal{P}os^\mu(t) \Leftrightarrow p \in \mathcal{P}os^\mu(s)$.*

Theorem 1 *Let \mathcal{R} be a TRS and φ be an E-strategy map. Let $t \in \mathcal{T}(\mathcal{F}_\mathcal{L}, \mathcal{X}_\mathcal{L})$, and $p \in \mathcal{P}os^{\mu^\varphi}(\text{erase}(t))$ be s.t. $\text{root}(t|_p) = f_L$ for some suffix L of $\varphi(f)$. If $\langle t, p \rangle \rightarrow_\varphi \langle s, q \rangle$, then $q \in \mathcal{P}os^{\mu^\varphi}(\text{erase}(s))$ and*

1. if $\langle t, p \rangle \xrightarrow{\varphi}^{\text{T}} \langle s, q \rangle$, then $\text{erase}(t) = \text{erase}(s)$.
2. if $\langle t, p \rangle \xrightarrow{\varphi}^{\text{R}} \langle s, q \rangle$, then $\text{erase}(t) \hookrightarrow_{\mu^\varphi} \text{erase}(s)$.

PROOF.

- Regarding $\xrightarrow{\varphi}^{\text{T}}$ -steps, we consider the three cases that correspond to them:
 1. We have $t = s$ and $p = q.i$. Since $t = s$, by Proposition 2, we have $q \in \mathcal{P}os^{\mu^\varphi}(\text{erase}(s))$ and $\text{erase}(t) = \text{erase}(s)$.
 2. Now, $t|_p = f_{i:L}(t_1, \dots, t_k)$ with $i > 0$, $s = t[f_L(t_1, \dots, t_k)]_p$ and $q = p.i$. By hypothesis, $i : L$ is a suffix of $\varphi(f)$. Thus, $i \in \mu^\varphi(f)$. Since $p \in \mathcal{P}os^{\mu^\varphi}(\text{erase}(t))$ by definition of replacing position, $p.i \in \mathcal{P}os^{\mu^\varphi}(\text{erase}(t))$. Since $\text{erase}(t) = \text{erase}(s)$, we have $q \in \mathcal{P}os^{\mu^\varphi}(\text{erase}(s))$,

3. We have $\text{erase}(t) = \text{erase}(s)$ and $q = p \in \mathcal{Pos}^{\mu^\varphi}(\text{erase}(s))$.

- Regarding $\xrightarrow{\mathbb{R}}_\varphi$ -steps, since $p \in \mathcal{Pos}^{\mu^\varphi}(t)$ and $t|_p = \sigma(l')$ for some $l' \rightarrow r \in R$ such that $\text{erase}(l') = l$, we have that $\text{erase}(t)|_p = \text{erase}(t|_p) = \theta(l)$ where $\theta(x) = \text{erase}(\sigma(x_{\text{nil}}))$ for all $x \in \text{Var}(l)$. Also, $s|_p = \sigma(\varphi(r))$. Then, $\text{erase}(s)|_p = \text{erase}(s|_p) = \text{erase}(\sigma(\varphi(r))) = \theta(r)$. Then, $\text{erase}(t) \xrightarrow{\mathbb{P}}_{\mu^\varphi} \text{erase}(s)$ and $q = p$. Since we have that $\text{prefix}_{\text{erase}(t)}(p) = \text{prefix}_{\text{erase}(s)}(p)$, by Proposition 3, we conclude $q \in \mathcal{Pos}^\mu(\text{erase}(s))$.

□

Lemma 1 *Let \mathcal{R} be a TRS and φ be an E-strategy map for \mathcal{R} . Let $t \in \mathcal{T}(\mathcal{F}_\mathcal{L}, \mathcal{X}_\mathcal{L})$ be such that, for all $u \in \mathcal{Pos}(t)$, $t|_u = f_L$ for some suffix L of $\varphi(f)$. Let $p \in \mathcal{Pos}(t)$. If $\langle t, p \rangle \rightarrow_\varphi \langle s, q \rangle$, then for all $u \in \mathcal{Pos}(s)$, $s|_u = f_L$ for some suffix L of $\varphi(f)$.*

PROOF. We consider the four cases of the definition of \rightarrow_φ . In the first case, the labelling of s is the same than in t . In the second and third cases, the labelling of $t|_p = f_L$ is shortened to a strict suffix of L ; the labellings of other positions in t remain unchanged. The obtained term s clearly satisfies the required property. Regarding the fourth case of the definition, $t|_p$ is replaced by $\varphi(\sigma(r))$ for some substitution σ and rule $l \rightarrow r \in R$ such that $t|_p = \sigma(l')$ and $\text{erase}(l') = l$. Since $\varphi(r)$ and $\sigma(x_{\text{nil}})$ for all $x \in \text{Var}(l)$ obviously satisfy the required property and the labellings of the context surrounding $t|_p$ in t remain unchanged, the property holds for $s = t[\sigma(\varphi(r))]_p$. □

Proposition 4 *Let \mathcal{R} be a TRS and φ be an E-strategy map. Then, \mathcal{R} is φ -terminating if and only if for all $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, there is no infinite $\xrightarrow{\mathbb{T}}_\varphi^* \circ \xrightarrow{\mathbb{R}}_\varphi$ -rewrite sequence starting from $\langle \varphi(t), \Lambda \rangle$.*

PROOF. The ‘only if’ case is obvious. For the ‘if’ case, just note that every \rightarrow_φ -sequence can be written as a (possibly empty) $\xrightarrow{\mathbb{T}}_\varphi^* \circ \xrightarrow{\mathbb{R}}_\varphi$ -sequence followed of a number of pure $\xrightarrow{\mathbb{T}}_\varphi$ -steps. Since $\xrightarrow{\mathbb{T}}_\varphi$ is clearly terminating (it only either goes up in the term -case 1- or shortens the labelling of symbols in a term -cases 2 and 3-, which are clearly terminating operations), regarding the termination behavior, we can dismiss the (always finite) number of ending \rightarrow_φ -steps to just consider the $\xrightarrow{\mathbb{T}}_\varphi^* \circ \xrightarrow{\mathbb{R}}_\varphi$ -steps. □

Theorem 2 *Let \mathcal{R} be a TRS and φ be an E-strategy map for \mathcal{R} . If \mathcal{R} is μ^φ -terminating, then \mathcal{R} is φ -terminating.*

PROOF. Assume that \mathcal{R} is not φ -terminating. According to Proposition 4, there exist a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and an infinite sequence

$$\langle s_1, q_1 \rangle \xrightarrow{\mathbb{T}}_\varphi^* \circ \xrightarrow{\mathbb{R}}_\varphi \langle s_2, q_2 \rangle \xrightarrow{\mathbb{T}}_\varphi^* \circ \xrightarrow{\mathbb{R}}_\varphi \dots \xrightarrow{\mathbb{T}}_\varphi^* \circ \xrightarrow{\mathbb{R}}_\varphi \langle s_n, q_n \rangle \xrightarrow{\mathbb{T}}_\varphi^* \circ \xrightarrow{\mathbb{R}}_\varphi \dots$$

where $\langle \varphi(t), \Lambda \rangle = \langle s_1, q_1 \rangle$. Now, since $\varphi(t)$ satisfies the condition required in Lemma 1, i.e., for all $q \in \mathcal{Pos}(\varphi(t))$, $\varphi(t)|_q = f_L$ for some suffix L of $\varphi(f)$, we can use Lemma 1 and Theorem 1 to conclude the existence of an infinite μ^φ -sequence

$$t = \text{erase}(s_1) \hookrightarrow_{\mu^\varphi} \text{erase}(s_2) \hookrightarrow_{\mu^\varphi} \dots \hookrightarrow_{\mu^\varphi} \text{erase}(s_n) \hookrightarrow_{\mu^\varphi} \dots$$

which contradicts μ^φ -termination of \mathcal{R} . \square

Following [Nag99], given a labelled term t and a position $q \in \mathcal{Pos}(t)$, $\mathfrak{s}(t, q)$ is the sequence L' such that $f_L = \text{root}(t|_q)$ and $\varphi(f) = L'; L$ (where ‘;’ concatenates two lists). Then, we have:

Lemma 2 [Nag99] *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and φ be an E-strategy map for \mathcal{R} . Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If $\langle \varphi(t), \Lambda \rangle \rightarrow_\varphi^* \langle s, p \rangle$ and $p = p_1.i.p_2$, then the last element of $\mathfrak{s}(s, p_1)$ is i .*

The following Proposition (and its proof) is analogous to Lemma 6.1.11 in [Nag99].

Proposition 5 *Let $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and φ be an E-strategy map for \mathcal{R} such that for all $f \in \mathcal{D}$, $\varphi(f)$ ends in 0. Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $\langle \varphi(t), \Lambda \rangle \rightarrow_\varphi^* \langle s, p \rangle$, and $q \in \mathcal{Pos}^\mu(\text{erase}(s))$. Then,*

- (1) *If $q < p$, then $\text{erase}(s|_{q.i})$ is a μ -normal form for every $i \in \mathfrak{s}(s, q)$ which is neither zero nor the last element of $\mathfrak{s}(s, q)$.*
- (2) *Otherwise,*
 - (2.1) *$\text{erase}(s|_{q.i})$ is a μ -normal form for every $i \in \mathfrak{s}(s, q)$ with $i \neq 0$.*
 - (2.2) *If the last element of $\mathfrak{s}(s, q)$ is zero, then $\text{erase}(s|_q)$ is not a redex.*

PROOF. By induction on the length n of $\langle \varphi(t), \Lambda \rangle \rightarrow_\varphi^* \langle s, p \rangle$. If $n = 0$, then $\mathfrak{s}(s, q) = \text{nil}$ for every $q \in \mathcal{Pos}(s)$ (in particular, for every $q \in \mathcal{Pos}^\mu(s)$). Thus, the proposition holds trivially. If $n > 0$, we write $\langle \varphi(t), \Lambda \rangle \rightarrow_\varphi^* \langle s', p' \rangle \rightarrow_\varphi \langle s, p \rangle$, where the length of derivation $\langle \varphi(t), \Lambda \rangle \rightarrow_\varphi^* \langle s', p' \rangle$ is $n - 1$. Now we consider the four cases of the definition of \rightarrow_φ for the last step $\langle s', p' \rangle \rightarrow_\varphi \langle s, p \rangle$.

1. Let $\text{root}(s'|_{p'}) = f_{\text{nil}}$, $s = s'$ and $p' = p.j$ for some $j (\neq 0)$. If $q \neq p$, then (1) and (2) follow from the induction hypothesis. If $q = p$, then by Lemma 2, the last element of $\mathfrak{s}(s', q)$ is j . Since $s = s'$, we have $\mathfrak{s}(s, q) = \mathfrak{s}(s', q)$; hence (2.2) holds as the last element of $\mathfrak{s}(s, q)$ is $j \neq 0$. Let $i \in \mathfrak{s}(s, q)$ with $i \neq 0$. If $i \neq j$, then $i \in \mathfrak{s}(s', q)$ and i is not the last element of $\mathfrak{s}(s', q)$. Thus, by the induction hypothesis, $\text{erase}(s'|_{q.i})$ is a μ -normal form. Since $\text{erase}(s') = \text{erase}(s)$, $\text{erase}(s|_{q.i})$ is a μ -normal form. If $i = j$, then $q.i = p'$. Since $\text{root}(s'|_{p'}) = f_{\text{nil}}$, $\mathfrak{s}(s', p') = \varphi(f)$. By the induction hypothesis, and since for all $f \in \mathcal{D}$, $\varphi(f)$ ends in 0, it follows that $\text{erase}(s'|_{p'})$ is not a redex and that for all $i \in \mu(f)$, $\text{erase}(s'|_{p'.i})$ is a μ -normal form. Therefore, $\text{erase}(s'|_{p'})$ is a μ -normal form. Since $s' = s$, $\text{erase}(s|_{p'})$ is a μ -normal form.

2. Let $s'|_{p'} = f_{i:L}(t_1, \dots, t_k)$ with $i > 0$, $s = s'[f_L(t_1, \dots, t_k)]_{p'}$ and $p = p'.i$. Then, by the induction hypothesis, the conclusion follows.
3. $s'|_{p'} = f_{0:L}(t_1, \dots, t_k)$, $erase(s'|_{p'})$ is not a redex, $s = s'[f_L(t_1, \dots, t_k)]_{p'}$, $p = p'$. Then, the last element of $\mathfrak{s}(s, q)$ is zero and $erase(s'|_p)$ is not a redex. Since $erase(s) = erase(s')$, by I.H., the conclusion follows.
4. Let $s'|_{q'} = f_{0:L}(t_1, \dots, t_k) = \sigma(l')$, $erase(l') = l$, $s = t[\sigma(\varphi(r))]_{q'}$ for some $l \rightarrow r \in R$ and substitution σ , $q = q'$. If $q < p$, then $\mathfrak{s}(s, q) = \mathfrak{s}(s', q)$. Let $i \in \mathfrak{s}(s, q)$ be a natural number which is neither zero nor the last element of $\mathfrak{s}(s, q)$. Using Lemma 2, we obtain $q.i \parallel p$. Thus, $s|_{q.i} = s'|_{q.i}$. Since, by induction hypothesis, $erase(s'|_{q.i})$ is a μ -normal form, we have that $erase(s|_{q.i})$ also is. If $q \parallel p$, then the conclusion follows by I.H. Finally, if $q \geq p$, we distinguish two cases: if $q \in \mathcal{Pos}_{\mathcal{F}}(r)$, then $\mathfrak{s}(s, q) = nil$ and thus the conclusion follows. Otherwise, there exists $q' \in \mathcal{Pos}(s')$ such that $p' < q'$ and $s'|_{q'} = s|_q$. By I.H., the conclusion follows again.

□

Theorem 3 Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and φ be an elementary E -strategy map for \mathcal{R} . Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If

$$\langle \varphi(t), \Lambda \rangle = \langle t_1, p_1 \rangle \xrightarrow{\mathbb{T}_{\varphi}^* \circ \mathbb{R}_{\varphi}} \langle t_2, p_2 \rangle \xrightarrow{\mathbb{T}_{\varphi}^* \circ \mathbb{R}_{\varphi}} \dots \xrightarrow{\mathbb{T}_{\varphi}^* \circ \mathbb{R}_{\varphi}} \langle t_n, p_n \rangle,$$

$$\text{then } t = erase(t_1) \xrightarrow{i}_{\mu^\varphi} erase(t_2) \xrightarrow{i}_{\mu^\varphi} \dots \xrightarrow{i}_{\mu^\varphi} erase(t_n).$$

PROOF. By induction on n . The base case, $n = 0$ is obvious. If $n > 0$, by the induction hypothesis,

$$t = erase(t_1) \xrightarrow{i}_{\mu^\varphi} erase(t_2) \xrightarrow{i}_{\mu^\varphi} \dots \xrightarrow{i}_{\mu^\varphi} erase(t_{n-1}).$$

We expand the last $\xrightarrow{\mathbb{T}_{\varphi}^* \circ \mathbb{R}_{\varphi}}$ -step $\langle t_{n-1}, p_{n-1} \rangle \xrightarrow{\mathbb{T}_{\varphi}^* \circ \mathbb{R}_{\varphi}} \langle t_n, p_n \rangle$ as follows:

$$\langle t_{n-1}, p_{n-1} \rangle = \langle s_1, q_1 \rangle \xrightarrow{\mathbb{T}_{\varphi}} \langle s_2, q_2 \rangle \xrightarrow{\mathbb{T}_{\varphi}} \dots \xrightarrow{\mathbb{T}_{\varphi}} \langle s_m, q_m \rangle \xrightarrow{\mathbb{R}_{\varphi}} \langle s_{m+1}, q_{m+1} \rangle = \langle t_n, p_n \rangle$$

Note that, by Theorem 1 and Lemma 1, and by definition of $\xrightarrow{\mathbb{R}_{\varphi}}$, $p_n = q_{m+1} = q_m \in \mathcal{Pos}^{\mu^\varphi}(erase(t_{n-1}))$ is a position of a μ^φ -replacing redex of $erase(t_{n-1})$; also $erase(t_{n-1}) \xrightarrow{i}_{\mu^\varphi} erase(t_n)$. Now, in order to prove that the last step $erase(t_{n-1}) \xrightarrow{i}_{\mu^\varphi} erase(t_n)$ is an innermost μ^φ -rewriting step, we have to prove that $erase(t_{n-1})|_{p_n}$ contains no μ^φ -rewriting redex.

By Proposition 5(2.1) $erase(t_{n-1})|_{p_n.i} = erase(s_m)|_{q_m.i}$ is a μ -normal form for each $i \in \mathfrak{s}(s_m, q_m)$ with $i \neq 0$. By elementarity of ς , $root(s_m)|_{q_m} = f(0)$. Therefore, $s(s_m, q_m)$ contains all indices in $\mu^\varphi(f)$. Hence, $erase(s_m)|_{q_m.i}$ is a μ -normal form for each $i \in \mu(f)$. Thus, $erase(s_m)|_{q_m} = erase(t_{n-1})|_{p_n}$ is an innermost μ^φ -replacing redex. □

Theorem 4 Let $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and φ be an elementary E-strategy map for \mathcal{R} . If \mathcal{R} is innermost μ -terminating, then \mathcal{R} is φ -terminating.

PROOF. Assume that \mathcal{R} is not φ -terminating. By Proposition 4, there is a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and an infinite sequence

$$\langle s_1, q_1 \rangle \xrightarrow{\text{T}}_{\varphi}^* \circ \xrightarrow{\text{R}}_{\varphi} \langle s_2, q_2 \rangle \xrightarrow{\text{T}}_{\varphi}^* \circ \xrightarrow{\text{R}}_{\varphi} \cdots \xrightarrow{\text{T}}_{\varphi}^* \circ \xrightarrow{\text{R}}_{\varphi} \langle s_n, q_n \rangle \xrightarrow{\text{T}}_{\varphi}^* \circ \xrightarrow{\text{R}}_{\varphi} \cdots$$

where $\langle \varphi(t), \Lambda \rangle = \langle s_1, q_1 \rangle$. By Theorem 3 there exists an infinite sequence $\text{erase}(t_1) \xrightarrow{i}_{\mu} \cdots \xrightarrow{i}_{\mu} \text{erase}(t_n) \xrightarrow{i}_{\mu} \cdots$ which contradicts innermost μ -termination of \mathcal{R} . \square

Theorem 5 Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and φ be an elementary E-strategy map for \mathcal{R} . If \mathcal{R} is φ -terminating, then \mathcal{R} is innermost μ^φ -terminating.

PROOF. Reasoning by contradiction, if \mathcal{R} is not innermost μ -terminating, then there exists an infinite innermost derivation

$$A : t_1 \xrightarrow{i}_{\mu^\varphi} t_2 \xrightarrow{i}_{\mu^\varphi} \cdots$$

The assumption that there is a maximum depth of reductions in A would lead to a contradiction, since this would be equivalent to consider that A splits as follows:

$$t_1 = t_{n_1} \xrightarrow{i}_{\mu^\varphi}^* t_{n_2} \xrightarrow{i}_{\mu}^* t_{n_3} \xrightarrow{i}_{\mu^\varphi}^* \cdots$$

where each of the finite segments $t_{n_i} \xrightarrow{i}_{\mu^\varphi}^* t_{n_{i+1}}$ of innermost μ^φ -reductions first μ^φ -normalizes the μ^φ -replacing arguments of $\text{root}(t_{n_i})$ and next apply a μ^φ -reduction step on t_{n_i} to produce $t_{n_{i+1}}$. However, it is not difficult to see that this decomposition implies the existence of an infinite \rightarrow_φ -sequence (here we use elementarity of φ)

$$\langle s_1, p_1 \rangle \rightarrow_{\varphi}^* \langle s_2, p_2 \rangle \rightarrow_{\varphi}^* \langle s_3, p_3 \rangle \rightarrow_{\varphi}^* \cdots$$

where $s_1 = \varphi(t_1)$, $p_1 = \Lambda$, and $s_i = \text{erase}(t_{n_i})$ for $i \geq 1$. This contradicts φ -termination of \mathcal{R} .

Therefore, any infinite innermost μ -derivation

$$A : t_1 \xrightarrow{i}_{\mu^\varphi} t_2 \xrightarrow{i}_{\mu^\varphi} \cdots$$

contracts deeper and deeper replacing redexes. Then, for each depth d , there exists $n > 0$ and a replacing subterm s at depth d within t_n such that s starts an infinite innermost μ^φ -derivation. Since such infinite derivation is preserved under modifications of the surrounding context, the \rightarrow_φ -sequence starting from $\langle \varphi(t_n), \Lambda \rangle$ (that μ^φ -normalizes the context around s) will eventually get into an infinite sequence inside s . This contradicts φ -termination of \mathcal{R} . \square

Proofs of Section 5

Proposition 6 [Luc98] *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, $\mu \in M_{\mathcal{F}}$, and $p \in \mathcal{P}os^{\mu}(C[t]_p)$. If $t \hookrightarrow s$, then $C[t]_p \hookrightarrow C[s]_p$.*

Theorem 7 *Let \mathcal{R} be a TRS, ς be a strategy annotation, and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If $t \rightarrow_{\varsigma} s$, then $t \hookrightarrow_{\mu^{\varsigma}} s$.*

PROOF. We consider pairs (t, L) (where t is a term and L is a suffix of $\varsigma(\text{root}(t))$) ordered by the lexicographic product $(\succ, >)$ of (strict) orderings \succ (the subterm ordering, $t \succ s$ if $\exists p \in \mathcal{P}os(t) - \{\Lambda\}, s = t|_p$), and $>$ (the *suffix* ordering on lists, $L > L'$ if L' is a strict suffix of L).

We prove, by induction on pairs (t, L) as above, and using $(\succ, >)$ that whenever $\text{rewr}_{\varsigma}(t, L) = (p, s)$, we have $t \xrightarrow{p}_{\mu^{\varsigma}} s$. The base case involves pairs (t, nil) , where t is a constant symbol or a variable. Then, $\text{rewr}_{\varsigma}(t, \text{nil}) = \perp$ and the conclusion follows.

Now, let (t, L) be an arbitrary pair as above:

1. If $L = (l \rightarrow r : L')$, we consider two cases:
 - (a) If $t = \sigma(l)$, then $\text{rewr}_{\varsigma}(t, L) = (\Lambda, s)$ where $s = \sigma(r)$. Since $\Lambda \in \mathcal{P}os^{\mu^{\varsigma}}(t)$, we have that $t \hookrightarrow_{\mu^{\varsigma}} s$.
 - (b) If t is not a redex of the rule, then $\text{rewr}_{\varsigma}(t, L) = \text{rewr}_{\varsigma}(t, L')$ and the conclusion follows by the I.H.
2. If $L = (i : L')$ for some $i \in \{1, \dots, \text{ar}(f)\}$, then, we consider two cases
 - (a) If $\text{rewr}_{\varsigma}(t|_i) = \text{rewr}_{\varsigma}(t|_i, \varsigma(\text{root}(t|_i))) = (q, u)$, then $\text{rewr}_{\varsigma}(t, L) = (i.q, t[u]_i)$. According to $(\succ, >)$, (t, L) is greater than $(t|_i, \varsigma(\text{root}(t|_i)))$. Thus, by the I.H., $t|_i \xrightarrow{q}_{\mu^{\varsigma}} u$. Since L is a suffix of $\varsigma(\text{root}(t))$, we have that $i \in \mu^{\varsigma}(\text{root}(t))$. Therefore, by Proposition 6, we have $t \xrightarrow{i.q}_{\mu^{\varsigma}} t[u]_i$.
 - (b) If $\text{rewr}_{\varsigma}(t|_i) = \perp$, then $\text{rewr}_{\varsigma}(t, L) = \text{rewr}_{\varsigma}(t, L')$. By the induction hypothesis, the conclusion follows.

By definition, $t \rightarrow_{\varsigma} s$ if $\text{rewr}_{\varsigma}(t) = \text{rewr}_{\varsigma}(t, \varsigma(\text{root}(t))) = (p, s)$ for some $p \in \mathcal{P}os_{\mathcal{R}}(t)$. Hence, $t \xrightarrow{p}_{\mu^{\varsigma}} s$ whenever $t \rightarrow_{\varsigma} s$. \square

Theorem 9 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, ς be a strategy annotation and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If $\text{norm}_{\varsigma}(t)$ is defined for t , then $t \hookrightarrow_{\mu^{\varsigma}}^* \text{norm}_{\varsigma}(t)$.*

PROOF. Note that norm_{ς} is deterministic, i.e., there is only one value -if any- returned by norm_{ς} for a given term s or pair (s, L) . Now, since $\text{norm}_{\varsigma}(t)$ is defined for t , each call to either $\text{norm}_{\varsigma}(s, L)$ or $\text{norm}_{\varsigma}(u)$ which is invoked to obtain $\text{norm}_{\varsigma}(t)$ is also defined.

We prove that $t \hookrightarrow_{\mu^{\varsigma}}^* \text{norm}_{\varsigma}(t, L)$, by induction on the number $n_{(s, L)}$ of recursive calls to either $\text{norm}_{\varsigma}(s)$ or $\text{norm}_{\varsigma}(s, L)$ which are necessary to compute

$norm_\varsigma(t)$. If $n_{t,L} = 0$, then $L = nil$, and $norm_\varsigma(t, nil) = t$ and the conclusion obviously follows. If $L = (l \rightarrow r : L')$ then we have two cases:

1. $t = \sigma(l)$ for some substitution σ . Then, $norm_\varsigma(t, L) = norm_\varsigma(\sigma(r)) = norm_\varsigma(\sigma(r), \varsigma(root(\sigma(r))))$. Obviously, $n_{(\sigma(r), \varsigma(root(\sigma(r))))} < n_{(t,L)}$. Then, by I.H., $t \hookrightarrow_{\mu^\varsigma}^* norm_\varsigma(\sigma(r), \varsigma(root(\sigma(r))))$ and also $t \hookrightarrow_{\mu^\varsigma}^* norm_\varsigma(t, L)$.
2. $t \neq \sigma(l)$ for any substitution σ . Then, $norm_\varsigma(t, l \rightarrow r : L) = norm_\varsigma(t, L')$. Again, $n_{(t,L)} > n_{(t,L')}$ and the conclusion follows by I.H.

If $L = (i : L')$, then $i \in \mu^\varsigma(root(t))$. Since $norm_\varsigma(t, L) = norm_\varsigma(t[norm_\varsigma(t|_i)]_i, L')$ and $n_{(t,L)} > n_{(t|_i, \varsigma(root(t|_i)))}$, we have $t|_i \hookrightarrow_{\mu^\varsigma}^* norm_\varsigma(t|_i)$. Let $s = norm_\varsigma(t|_i)$. Then, $norm_\varsigma(t, L) = norm_\varsigma(t[s]_i, L')$. By I.H., $t[s]_i \hookrightarrow_{\mu^\varsigma}^* norm_\varsigma(t, L)$. Since $i \in \mu^\varsigma(f)$, by a repeated application of Proposition 6, we also have $t[t|_i]_i \hookrightarrow_{\mu^\varsigma}^* t[s]_i \hookrightarrow_{\mu^\varsigma}^* norm_\varsigma(t, L)$. \square

In the proof of the following theorem, we use the function $norm'_\varsigma$ given by van de Pol as follows [Pol01, Section 3.1]:

$$norm'_\varsigma(t, p, L) = t[norm_\varsigma(t|_p, L)]_p$$

Then, van de Pol gives the following recursive definition for $norm'_\varsigma$:

$$\begin{aligned} norm'_\varsigma(t, p, nil) &= t \\ norm'_\varsigma(t, p, (l \rightarrow r : L)) &= \begin{cases} norm'_\varsigma(t[\sigma(r)]_p, p, \varsigma(root(\sigma(r)))) \\ \text{if } t = \sigma(l) \text{ for some } \sigma \\ norm'_\varsigma(t, p, L), \text{ otherwise} \end{cases} \\ norm'_\varsigma(t, p, (i : L)) &= norm'_\varsigma(norm'_\varsigma(t, p, i, \varsigma(root(t|_p, i))), p, L) \end{aligned}$$

Note that $norm_\varsigma(t, L) = norm'_\varsigma(t, \Lambda, L)$.

Theorem 11 *Let \mathcal{R} be a TRS and ς be a strategy annotation for \mathcal{R} . If \mathcal{R} is μ^ς -terminating, then \mathcal{R} is ς -terminating.*

PROOF. Since \mathcal{R} is μ^ς -terminating, $\hookrightarrow_{\mu^\varsigma}^+$ is a well-founded, stable and μ^ς -monotonic ordering on terms. We consider tuples (t, p, L) of arbitrary terms t , positions $p \in \mathcal{Pos}^{\mu^\varsigma}(t)$, and lists L which are a suffix of $\varsigma(root(t|_p))$. We consider the following ordering \sqsupset on these tuples: $(t, p, L) \sqsupset (s, q, L')$, if

1. $t \hookrightarrow_{\mu^\varsigma}^+ s$, or
2. $t = s$ and $p < q$, or
3. $p = q$ and L' is a suffix of L

It is not difficult to see that \sqsupset is a well-founded ordering (note that $<$ is well-founded because we always compare positions of the same term). We prove by induction on \sqsupset that $norm'_\varsigma(t, p, L)$ is defined for every tuple (t, p, L) as above.

For the base case, we have to consider tuples (t, p, nil) where t is a μ^ς -normal form and p is a maximal position in t . Then, $norm'_\varsigma(t, p, nil) = t$.

Let (t, p, L) be one of the considered pairs. We distinguish three cases:

1. If $L = \text{nil}$, then $\text{norm}'_{\zeta}(t, p, L) = t$.
2. If $L = l \rightarrow r : L'$, then
 - (a) If $t = \sigma(l)$ for some substitution σ , then

$$\text{norm}'_{\zeta}(t, p, L) = \text{norm}'_{\zeta}(t[\sigma(r)]_p, p, \zeta(\text{root}(\sigma(r)))).$$

Since $p \in \mathcal{Pos}^{\mu^{\zeta}}(t)$, it follows that $t \hookrightarrow_{\mu^{\zeta}} t[\sigma(r)]_p$. Thus, by the I.H., $\text{norm}'_{\zeta}(t[\sigma(r)]_p, p, \zeta(\text{root}(\sigma(r))))$ is defined and $\text{norm}'_{\zeta}(t, p, L)$ also is.

- (b) If t does not match l , then $\text{norm}_{\zeta}(t, p, l \rightarrow r : L') = \text{norm}'_{\zeta}(t, p, L)$. Since $(t, p, L) \sqsupset (t, p, L')$, by the I.H., the conclusion follows.

3. If $L = i : L'$, then

$$\text{norm}_{\zeta}(t, p, i : L') = \text{norm}'_{\zeta}(\text{norm}'_{\zeta}(t, p.i, \zeta(\text{root}(t|_{p.i}))), p, L).$$

Note that, since $p \in \mathcal{Pos}^{\mu^{\zeta}}(t)$ and L is a suffix of $\zeta(\text{root}(t|_p))$, $i \in \mu^{\zeta}(\text{root}(t|_p))$, and we have that $p.i \in \mathcal{Pos}^{\mu^{\zeta}}(t)$. Thus, $(t, p, i : L') \sqsupset (t, p.i, \zeta(\text{root}(t|_{p.i})))$. By I.H., $\text{norm}'_{\zeta}(t, p.i, \zeta(\text{root}(t|_{p.i})))$ is defined. Since $\text{norm}'_{\zeta}(t, p.i, \zeta(\text{root}(t|_{p.i}))) = t[\text{norm}_{\zeta}(t|_{p.i}, \zeta(\text{root}(t|_{p.i})))]_{p.i} = t[\text{norm}_{\zeta}(t|_{p.i})]_{p.i}$, $\text{norm}_{\zeta}(t|_{p.i})$ is also defined and, by Theorem 9, $t|_{p.i} \hookrightarrow_{\mu^{\zeta}}^* s$ where $s = \text{norm}_{\zeta}(t|_{p.i})$. Thus, $\text{norm}'_{\zeta}(t, p.i, \zeta(\text{root}(t|_{p.i}))) = t[s]_{p.i}$. Now, we distinguish two cases:

- (a) If $t|_{p.i} \hookrightarrow_{\mu^{\zeta}}^+ s$, then since $p.i \in \mathcal{Pos}^{\mu^{\zeta}}(t)$, by a repeated application of Proposition 6, $t = t[t|_{p.i}]_{p.i} \hookrightarrow_{\mu^{\zeta}}^+ t[s]_{p.i}$. Hence, by the I.H., $\text{norm}_{\zeta}(t[s]_{p.i}, p, L)$ is defined and, $\text{norm}_{\zeta}(t, p, L)$ also is.
- (b) If $t|_{p.i} = s$, then since $(t, p, L) \sqsupset (t, p, L')$, by the I.H., $\text{norm}_{\zeta}(t, p, L')$ is defined and $\text{norm}_{\zeta}(t, p, L)$ also is.

Now, since $\text{norm}_{\zeta}(t) = \text{norm}_{\zeta}(t, \zeta(\text{root}(t))) = \text{norm}'_{\zeta}(t, \Lambda, \zeta(\text{root}(t)))$, we conclude that $\text{norm}_{\zeta}(t)$ is defined for all terms t . \square

B Proof of termination of FIRST_FROM_LENGTH using MUTERM and CiME

Consider the program FIRST_FROM_LENGTH in Example 2:

```
obj FIRST_FROM_LENGTH is
  sorts Nat LNat .
  op 0      : -> Nat .
  op s      : Nat -> Nat [strat (0)] .
  op nil    : -> LNat .
  op cons   : Nat LNat -> LNat [strat (1)] .
  op fst    : Nat LNat -> LNat .
  op from   : Nat -> LNat .
```

```

op add  : Nat Nat -> Nat .
op len  : LNat -> Nat .
vars X Y : Nat .
var Z   : LNat .
eq fst(0,Z) = nil .
eq fst(s(X),cons(Y,Z)) = cons(Y,fst(X,Z)) .
eq from(X) = cons(X,from(s(X))) .
eq add(0,X) = X .
eq add(s(X),Y) = s(add(X,Y)) .
eq len(nil) = 0 .
eq len(cons(X,Z))= s(len(Z)) .
endo

```

In order to obtain a proof of termination of FIRST_FROM_LENGTH, with MUTERM and CiME, we proceed as follows:

1. Use option 2. Load TRS in simple OBJ format of menu FILE to load the program in MUTERM. The program is loaded as a CS-TRS, i.e., a TRS together with a replacement map which is obtained from the program. You can check it by using option 4. (CS-)TRS of menu VIEW:

Showing the CS-TRS FIRST_FROM_LENGTH:

Functions: 0 s nil cons fst from add len

Variables: X Y Z

Arities:

(0,0)

(s,1)

(nil,0)

(cons,2)

(fst,2)

(from,1)

(add,2)

(len,1)

Replacement map:

(0, [])

(s, [])

(nil, [])

(cons, [1])

(fst, [1,2])

```
(from,[1])
(add,[1,2])
(len,[1])
```

Rules:

```
fst(0,Z) -> nil
fst(s(X),cons(Y,Z)) -> cons(Y,fst(X,Z))
from(X) -> cons(X,from(s(X)))
add(0,X) -> X
add(s(X),Y) -> s(add(X,Y))
len(nil) -> 0
len(cons(X,Z)) -> s(len(Z))
```

2. Transform the CS-TRS by using the transformation in [Luc96] which is available as option 1. Lucas' transformation of menu TRANSFORMATIONS. Select the transformed system as the 'current TRS' by using option 9 of the same menu. In menu VIEW, you can take a look to the rules of the obtained TRS by using option 3. Rules:

```
fst(0,Z) -> nil
fst(s,cons(Y)) -> cons(Y)
from(X) -> cons(X)
add(0,X) -> X
add(s,Y) -> s
len(nil) -> 0
len(cons(X)) -> s
```

3. Generate a CiME file by using option 4. CiME TRS of menu SAVE AS. The TRS which is stored in this file can also be retrieved by using option 4. CiME TRS of menu ALL FORMATS:

```
let Sig = signature "
nil,s,0 : constant;
len,from,cons : unary;
add,fst : binary;
";
let X = vars "X Y Z";
let FIRST_FROM_LENGTH = TRS Sig X "
fst(0,Z) -> nil;
fst(s,cons(Y)) -> cons(Y);
```

```

from(X) -> cons(X);
add(0,X) -> X;
add(s,Y) -> s;
len(nil) -> 0;
len(cons(X)) -> s;
";

```

4. Use CiME to prove termination of the previous TRS:

```

CiME> termination FIRST_FROM_LENGTH;
Entering the termination expert. Verbose level = 0
[nil] = 0;
[s] = 0;
[0] = 0;
[len](X0) = X0 + 1;
[from](X0) = X0 + 1;
[cons](X0) = X0;
[add](X0,X1) = X1 + X0 + 1;
[fst](X0,X1) = X1 + X0 + 1;
Execution time: 0.050000 sec
- : unit = ()

```

CiME shows the polynomial interpretation which can be used to prove termination of the system.

C Proof of termination of check-full-arg using MUTERM and AProVE

Consider the program check-full-arg in Example 21:

```

signature
  0(0)
  S(1)
  nth(2)
  cons(2)
  nil(0)
  from(1)
rules
  n1([x,y],nth(0,cons(x,y)),x)
  n2([m,x,y],nth(S(m),cons(x,y)),nth(m,y))
  f1([x],from(x),cons(x,from(S(x))))
default justintime
strategies
  cons([1])

```

```
end
stop
```

In order to obtain a proof of termination of `check-full-arg`, with `MUTERM` and `AProVE`, we proceed as follows:

1. Use option 1. Load `TRS in simple format` of menu `FILE` to load the program in `MUTERM`; since `MUTERM` does not accept `JITty` modules, we have to express the program in ‘simple format’ as follows:

```
nth(0,cons(X,Y)) -> X
nth(s(M),cons(X,Y)) -> nth(M,Y)
from(X) -> cons(X,from(s(X)))
```

Note the uppercase for variable symbols: this is required to distinguish between function symbols (always starting in lowercase) and variable symbols.

2. Assign the replacement map $\mu^s(\text{cons}) = \{1\}$ and $\mu^s(f) = \{1, \dots, ar(f)\}$ for all other symbols f to the TRS by using menu `REPLACEMENT MAP`: first use option (6. `Set the greatest replacement map`) to start with the replacement map which makes all argument of all function symbols to be replacing. Then, use option 2. `Modify replacement map` to let the replacement map of `cons` be as indicated above.
3. Now, the program can be retrieved as a CS-TRS; use option 4. `(CS-)TRS` of menu `VIEW`:

```
Showing the CS-TRS  check-full-arg:
```

```
Functions:  nth 0 cons s from
```

```
Variables:  X Y M
```

```
Arities:
```

```
(nth,2)
```

```
(0,0)
```

```
(cons,2)
```

```
(s,1)
```

```
(from,1)
```

```
Replacement map:
```

```
(nth,[1,2])
```

```
(0,[])
```

```
(cons,[1])
(s,[1])
(from,[1])
```

Rules:

```
nth(0,cons(X,Y)) -> X
nth(s(M),cons(X,Y)) -> nth(M,Y)
from(X) -> cons(X,from(s(X)))
```

4. Transform the CS-TRS by using the transformation in [Zan97]. It is available as option 2. Zantema's transformation of menu TRANSFORMATIONS. Select the transformed system as the 'current TRS' by using option 9 of the same menu. In menu VIEW, you can take a look to the rules of the obtained TRS by using option 3. Rules:

```
nth(0,cons(X,Y)) -> X
nth(s(M),cons(X,Y)) -> nth(M,activate(Y))
from(X) -> cons(X,n__from(s(X)))
from(X) -> n__from(X)
activate(n__from(X)) -> from(X)
activate(X) -> X
```

5. Generate an AProVE file by using option 3. AProVE TRS of menu SAVE AS. The TRS is stored in the file by using the appropriate format for AProVE. You can also see it by using option 3. AProVE TRS of menu ALL FORMATS:

```
[X, Y, M]
nth(0,cons(X,Y)) -> X
nth(s(M),cons(X,Y)) -> nth(M,activate(Y))
from(X) -> cons(X,n__from(s(X)))
from(X) -> n__from(X)
activate(n__from(X)) -> from(X)
activate(X) -> X
```

6. Use AProVE to prove termination of the previous TRS. Direct termination by using LPO (lexicographic path ordering) works fine. The precedence computed by the system is:

```
nth > activate > from > cons
nth > activate > from > s
nth > activate > from > n__from
```