Recursive Path Orderings can be Context-Sensitive

Cristina Borralleras[1], Salvador Lucas[2] and Albert Rubio[3⋆]

[1] Dept. Informàtica i Matemàtica, E.P.S. Universitat de Vic,
C/ de la Laura, 13. 08500 Vic Spain
Email: cristina.borralleras@uvic.es

[2] D.S.I.C. Universidad Politécnica de Valencia
C/ Camino de Vera, s/n. 46022 Valencia Spain
Email: slucas@dsic.upv.es

[3] Dept. L.S.I. Universitat Politècnica de Catalunya,
C/Jordi Girona, 1-3. 08034 Barcelona, Spain
Email: rubio@lsi.upc.es

Abstract. Context-sensitive rewriting (CSR) is a simple restriction of rewriting which can be used e.g. for modelling non-eager evaluation in programming languages. Many times *termination* is a crucial property for program verification. Hence, developing tools for automatically proving termination of CSR is necessary.

All known methods for proving termination of (CSR) systems are based on transforming the CSR system $\mathcal{R}$ into a (standard) rewrite system $\mathcal{R}'$ whose termination implies the termination of the CSR system $\mathcal{R}$.

In this paper first several negative results on the applicability of existing transformation methods are provided. Second, as a general-purpose way to overcome these problems, we develop the first (up to our knowledge) method for proving directly termination of context-sensitive rewrite systems: the *context sensitive recursive path ordering* (CSRPO).

Many interesting (realistic) examples that cannot be proved or are hard to prove with the known transformation methods are easily handled using CSRPO. Moreover, CSRPO is very suitable for automation.

1 Introduction

One of the main applications of automated deduction is (e.g. program) verification. Many correctness proofs include termination as an essential property. Therefore automatic tools for termination analysis may be crucial in verification.

Obviously, automatically proving termination of arbitrary programs written in languages like C is extremely hard. Hence, one focuses on more abstract computation formalisms like term rewrite systems (TRSs). Termination proofs

⋆ C. Borralleras and A. Rubio are supported by the CICYT TIC2001-2476-C03-01, A. Rubio also by the DURSI group 2001SGR 00254 and S. Lucas by CICYT TIC2001-2705-C03-01, Acciones Integradas HI 2000-0161, HA 2001-0059, HU 2001-0019, and Generalitat ValencianaGV01-424.

for other (e.g. logic or functional) programs can be obtained by translating them into a TRS (see e.g. [GW93,AZ96]).

Context-sensitive (CS) rewriting (*CSR* [Luc98]) is a simple generalization of rewriting which is formalized by imposing fixed syntactic restrictions on replacements: a *replacement map* μ discriminates, for each symbol of the signature, the argument positions on which replacements are allowed. E.g., with $\mu(\texttt{if-then-else}) = \{1\}$, reductions are permitted *only* on the first argument of calls to `if-then-else`. Many *eager* programming languages (e.g., Lisp, OBJ2, OBJ3, CafeOBJ, or Maude) provide a special (non-eager) semantics for this operator which can be modeled with a replacement map [Luc01]. Note that in the particular case where μ includes all argument positions CSR becomes standard rewriting.

Any terminating TRS is also *μ-terminating*, i.e., no term initiates an infinite sequence of *CSR* under μ. However, *CSR* can *enforce* termination, by pruning (all) infinite rewrite sequences. Therefore, particular methods for proving μ-termination have been developed. All currently known such methods are transformation methods [Luc96,Zan97,SX98,FR99,GM99,GM01]: one proves the termination of a CS-TRS $\mathcal{R}$ by building a (standard) TRS $\mathcal{R}'$ whose termination implies the μ-termination of $\mathcal{R}$. Then one can use standard techniques for proving the termination of $\mathcal{R}'$.

Example 1. The following term rewrite system (given in [GM01]) defines some operations on lists and, in particular, allow us to generate an infinite list of zeros (from *zeros*) and the list of all natural numbers (from *nats*).

$$\begin{array}{rlrl}
incr(nil) & \to nil & nats & \to adx(zeros) \\
incr(cons(x,l)) & \to cons(s(x), incr(l)) & zeros & \to cons(0, zeros) \\
adx(nil) & \to nil & head(cons(x,l)) & \to x \\
adx(cons(x,l)) & \to incr(cons(x, adx(l))) & tail(cons(x,l)) & \to l
\end{array}$$

This non-terminating TRS becomes μ-terminating with the replacement map $\mu(cons) = \mu(incr) = \mu(adx) = \mu(s) = \mu(head) = \mu(tail) = \{1\}$. Note that, since *cons* has its second argument blocked (not active) we cannot produce an infinite rewriting sequence starting from *zeros*.

All these transformation methods (except the one in [Luc96]) use new symbols and rules, often introducing a loss of structure and intuition due to the encoding of the CS-control by such new elements.

In [GM99] a complete transformation (i.e., a transformation that never turns CS-terminating $\mathcal{R}$ into non-terminating $\mathcal{R}'$) was given. The completeness of their method motivated the claim in [GM99] (pg. 272) that "it appears that from a termination point of view there is no reason to study context-sensitive rewriting further". However, such claims are based on the (mis)understanding that whenever the transformed system $\mathcal{R}'$ terminates it will be easily provable, obviating the fact that termination of a TRS is an undecidable property[1].

---

[1] Moreover, the existence of such complete transformations is no surprise: it can be directly extracted from the Turing completeness proof of TRS: simulate the CS-TRS

In this paper we give good reasons for believing that in many cases it may be more feasible to directly prove the CS-termination of the original system. For instance, with the complete method proposed in [GM99] (and also the incomplete one given there), even when the termination of the initial context-sensitive system is quite trivial, the termination of the transformed system can be hard (this is the case for the CS-TRS of Example 1 above): in Section 3 we prove that with the methods given in [GM99] the transformed TRS can be proved terminating with the *recursive path ordering* (RPO) [Der82] only if the original CS-TRS was already RPO-terminating *without the CS restrictions*. It can even be the case that the original system is RPO-terminating and the transformed one is not. Similar negative results are obtained for the *Knuth-Bendix ordering* (KBO) [KB70], the other main general-purpose termination proof method.

For all these reasons, here we focus on specific methods for directly proving the termination of *CSR*, and give evidence for the fact that simple but powerful automated termination methods for *CSR* can be obtained in this way.

Due to its simplicity and its success in the (standard) term rewriting setting, our aim has been to obtain a recursive path ordering which is context-sensitive, i.e., terms are ordered differently depending on the context they are in. In Section 4 we define such a context-sensitive RPO (CSRPO) and state its main properties (all proofs can be found in [BLR02]).

In sections 5 and 6 evidence for the power of CSRPO is given, by means of several realistic examples where CSRPO easily proves the termination of context-sensitive term rewrite systems which are not terminating without the context-sensitivity restrictions. For most of these examples the known transformation methods cannot work or do not work in practice.

Moreover, the ideas of the CSRPO, especially the ones concerning the treatment of variables, carry over to other more powerful ordering-based methods, obtaining its corresponding context-sensitive version. This may produce proper extensions of the most powerful existing methods for direct termination proofs of context-sensitive rewriting in many cases.

2 Preliminaries

Given a binary relation R on A, we say that R is *terminating* (*strongly normalizing*) iff there is no infinite sequence $a_1 \; R \; a_2 \; R \; a_3 \cdots$. Throughout the paper, $\mathcal{X}$ denotes a countable set of variables and $\mathcal{F}$ denotes the set of function symbols $\{f, g, \ldots\}$, each having a fixed arity given by a mapping $ar : \mathcal{F} \to \mathbb{N}$. The set of terms built from $\mathcal{F}$ and $\mathcal{X}$ is $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and the set of ground terms built from $\mathcal{F}$ is $\mathcal{T}(\mathcal{F})$. Terms are viewed as labeled trees in the usual way. The set of positions of a term t is $\mathcal{P}os(t)$. The subterm at position p of t is denoted as $t|_p$ and $t[s]_p$ is the term t with the subterm at position p replaced by s. The symbol labeling the root of t is denoted as $root(t)$.

by a C-program (easy), which is simulated by a Turing machine (well-known as well), which is finally encoded by a (even single-rule) TRS as in [Dau92].

A rewrite rule is an ordered pair of terms, written $l \to r$, with $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. A TRS is a pair $\mathcal{R} = (\mathcal{F}, R)$ where R is a set of rewrite rules. An instance $\sigma(l)$ of a *lhs* l of a rule is a redex. A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ rewrites to s (at position p), written $t \xrightarrow{p}_{\mathcal{R}} s$ (or just $t \to s$), if $t|_p = \sigma(l)$ and $s = t[\sigma(r)]_p$, for some rule $l \to r \in R$, $p \in \mathcal{P}os(t)$ and substitution σ. A TRS is terminating if $\to$ is terminating.

2.1 Orderings

A (strict partial) ordering is a transitive and irreflexive relation. An ordering $\succ$ is compatible with an equality (or a congruence) relation $\sim$ if for all terms s, s', t, t' we have that $s' \sim s \succ t \sim t'$ implies $s' \succ t'$. An ordering $\succ$ is stable under substitutions if $s \succ t$ implies $s\sigma \succ t\sigma$ for all terms s and t and substitution σ, and it is said to be stable under ground substitutions if $s \succ t$ implies $s\sigma \succ t\sigma$ for all ground substitution σ. An ordering $\succ$ is well-founded if there exists no infinite sequence $t_1 \succ t_2 \succ \ldots$.

Given an ordering $\succ$ and a compatible equality relation $\sim$, the multiset extension of $\succ$ with respect to $\sim$, denoted by $\succ\!\succ$, is defined as the smallest transitive relation containing

$$X \cup \{s\} \succ\!\succ Y \cup \{t_1, \ldots, t_n\} \qquad \text{if } X \sim Y \text{ and } s \succ t_i \text{ for all } i \in \{1 \ldots n\}$$

If $\succ$ is a well-founded ordering on terms then $\succ\!\succ$ is a well-founded ordering on finite multisets of terms.

Let $\sim$ be an equality relation compatible with a given ordering $\succ$. Then the lexicographic extension of $\succ$ with respect to $\sim$, denoted by $\succ^{lex}$, is defined as $\langle s_1, \ldots, s_m\rangle \succ^{lex} \langle t_1, \ldots, t_n\rangle$ if and only if $m \geq n$ and there is some $i \in \{1 \ldots n\}$ such that $s_i \succ t_i$ and $s_k \sim t_k$ for all $k \in \{1 \ldots i-1\}$. If $\succ$ is a well-founded ordering on terms then $\succ^{lex}$ is a well-founded ordering on finite sequences of terms.

For standard rewriting one of the most successful general methods applied to prove termination is the *recursive path ordering* (RPO) [Der82]. Given a precedence $\succeq_{\mathcal{F}}$ on the set of function symbols, which is the union of a well-founded ordering $\succ_{\mathcal{F}}$ and a compatible equality $=_{\mathcal{F}}$, and a status function $stat(f) \in \{lex, mul\}$ s.t. if $f =_{\mathcal{F}} g$ then $stat(f) = stat(g)$, RPO is defined recursively as follows:

$s = f(s_1, \ldots, s_n) \succ_{rpo} t$ iff

1. $s_i \succeq_{rpo} t$, for some $i = 1, \ldots, n$ or
2. $t = g(t_1, \ldots, t_m)$ with $f \succ_{\mathcal{F}} g$ and $s \succ_{rpo} t_i$ for all $i = 1 \ldots n$ or
3. $t = g(t_1, \ldots, t_m)$ with $f =_{\mathcal{F}} g$, $stat(f) = mul$ and $\{s_1, \ldots, s_n\} \succ\!\succ_{rpo} \{t_1, \ldots, t_m\}$ or
4. $t = g(t_1, \ldots, t_m)$ with $f =_{\mathcal{F}} g$, $stat(f) = lex$, $\langle s_1, \ldots, s_n\rangle \succ_{rpo}^{lex} \langle t_1, \ldots, t_m\rangle$ and $s \succ_{rpo} t_i$ for all $i \in \{1 \ldots m\}$.

where $\succeq_{rpo}$ is the union of $\succ_{rpo}$ and syntactic equality.

2.2 Context-sensitive rewriting

A mapping $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ is a *replacement map* (or $\mathcal{F}$-map) if $\forall f \in \mathcal{F},\ \mu(f) \subseteq \{1, \ldots, ar(f)\}$ [Luc98]. A replacement map μ specifies the *argument* positions which can be reduced for each *symbol* in $\mathcal{F}$. Accordingly, the set of μ-*replacing* or *active* positions $\mathcal{P}os^{\mu}(t)$ of $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is: $\mathcal{P}os^{\mu}(t) = \{\Lambda\}$, if $t \in \mathcal{X}$ and $\mathcal{P}os^{\mu}(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(root(t))} i.\mathcal{P}os^{\mu}(t|_i)$, if $t \notin \mathcal{X}$.

In *context-sensitive rewriting* (*CSR* [Luc98]), we (only) contract *replacing* redexes: t μ-rewrites to s, written $t \hookrightarrow_{\mu} s$, if $t \xrightarrow{p}_{\mathcal{R}} s$ and $p \in \mathcal{P}os^{\mu}(t)$.

Example 2. Consider $\mathcal{R}$ and μ as in Example 1. Then, we have: $adx(zeros) \hookrightarrow_{\mu} adx(cons(0, zeros)) \hookrightarrow_{\mu} incr(cons(0, adx(zeros))) \hookrightarrow_{\mu} cons(s(0), incr(adx(zeros)))$. Since $2.1.1 \notin \mathcal{P}os^{\mu}(cons(s(0), incr(adx(zeros))))$, redex $zeros$ cannot be μ-rewritten.

A TRS $\mathcal{R}$ is μ-terminating if $\hookrightarrow_{\mu}$ is terminating.

2.3 μ-reduction orderings

Here we adapt the reduction orderings framework to the context sensitive case. In order to obtain an appropriate notion of reduction orderings we only have to modify the definition of monotonicity. In context-sensitive rewriting the redexes are always in active positions, therefore, we only need to ensure monotonicity for these positions (this notion was already considered in [Zan97]).

Definition 1. *An ordering $\succ$ is μ-monotonic iff $s \succ t$ implies $C[s]_p \succ C[t]_p$ for all context C and for all active position p.*

Definition 2. *$\succ$ is a μ-reduction ordering if it is a μ-monotonic, stable under substitutions, well-founded ordering.*

Then we have that μ-reduction orderings characterize μ-termination.

Theorem 1. *Let $\mathcal{R}$ be a TRS and μ be a replacement map. Then $\mathcal{R}$ is μ-terminating if and only if there exists a μ-reduction ordering $\succ$ such that $l \succ r$ for every rule $l \to r$ in $\mathcal{R}$.*

In fact, stability under substitutions can be relaxed by considering only ground substitutions. Then we speak about *ground stable μ-reduction orderings.*

Definition 3. *$\succ$ is a ground stable μ-reduction ordering if it is a μ-monotonic, stable under ground substitutions, well-founded ordering.*

Ground stable μ-reduction orderings characterize μ-termination as well, provided there is at least one constant symbol in the signature, since any infinite sequence of rewriting can be instantiated to obtain a ground infinite sequence.

Theorem 2. *Let $\mathcal{R}$ be a TRS and μ be a replacement map. Then $\mathcal{R}$ is μ-terminating if and only if there exists a ground stable μ-reduction ordering $\succ$ such that $l \succ r$ for every rule $l \to r$ in $\mathcal{R}$.*

To simplify the proof of μ-monotonicity we can equivalently state the property in a different way.

Property 1. An ordering $\succ$ is μ-monotonic iff $s \succ t$ implies $f(\dots[s]_i \dots) \succ f(\dots[t]_i \dots)$ for all function symbol $f \in \mathcal{F}$ and for all $i \in \mu(f)$.

3 Transformation methods.

In this section we analyze the transformation methods presented in [GM99] from a practical point of view. In particular, we provide some results showing that general purpose methods like RPO or KBO (see the appendix) cannot be used for proving termination of the transformed system. This result shows that only very powerful termination techniques can be used to prove the termination of the transformed TRS even when the μ-termination of the original CS-TRS looks quite trivial. All proofs can be found in [BLR02].

In the first presented transformation method, denoted by $\mathcal{R}^{\mu}_{GM}$, the basic idea is to explicitly *mark* the replacing positions of a term (by using a new symbol `active`), since these positions are the only ones where *CSR* may take place. Given a TRS $\mathcal{R} = (\mathcal{F}, R)$ and μ a replacement map, the TRS $\mathcal{R}^{\mu}_{GM} = (\mathcal{F} \cup \{\texttt{active}, \texttt{mark}\}, R^{\mu}_{GM})$ consists of the following rules (for all $l \to r \in R$ and $f \in \mathcal{F}$):

$$\begin{aligned} \texttt{active}(l) &\to \texttt{mark}(r) \\ \texttt{mark}(f(x_1, \dots, x_k)) &\to \texttt{active}(f([x_1]_f, \dots, [x_k]_f)) \\ \texttt{active}(x) &\to x \end{aligned}$$

where $[x_i]_f = \texttt{mark}(x_i)$ if $i \in \mu(f)$ and $[x_i]_f = x_i$ otherwise. The second (confluent and terminating) collection of rules is labeled $\mathcal{M}$ in [GM99].

Example 3. Consider $\mathcal{R}$ and μ as in Example 1. Then, $\mathcal{R}^{\mu}_{GM}$ is

$$\begin{aligned} &\texttt{active}(incr(nil)) \to \texttt{mark}(nil) \\ &\texttt{active}(incr(cons(x, l))) \to \texttt{mark}(cons(s(x), incr(l))) \\ &\texttt{active}(adx(nil)) \to \texttt{mark}(nil) \\ &\texttt{active}(adx(cons(x, l))) \to \texttt{mark}(incr(cons(x, adx(l)))) \\ &\texttt{active}(nats) \to \texttt{mark}(adx(zeros)) \\ &\texttt{active}(zeros) \to \texttt{mark}(cons(0, zeros)) \\ &\texttt{active}(head(cons(x, l))) \to \texttt{mark}(x) \\ &\texttt{active}(tail(cons(x, l))) \to \texttt{mark}(l) \end{aligned}$$

$$\begin{aligned} &\texttt{mark}(incr(x)) \to \texttt{active}(incr(\texttt{mark}(x))) \\ &\texttt{mark}(nil) \to \texttt{active}(nil) \\ &\texttt{mark}(cons(x, y)) \to \texttt{active}(cons(\texttt{mark}(x), y)) \\ &\texttt{mark}(s(x)) \to \texttt{active}(s(\texttt{mark}(x))) \\ &\texttt{mark}(adx(x)) \to \texttt{active}(adx(\texttt{mark}(x))) \\ &\texttt{mark}(nats) \to \texttt{active}(nats) \\ &\texttt{mark}(zeros) \to \texttt{active}(zeros) \\ &\texttt{mark}(0) \to \texttt{active}(0) \\ &\texttt{mark}(head(x)) \to \texttt{active}(head(\texttt{mark}(x))) \\ &\texttt{mark}(tail(x)) \to \texttt{active}(tail(\texttt{mark}(x))) \\ &\texttt{active}(x) \to x \end{aligned}$$

Although it is not complete (see Example 1 of [GM99]), this transformation preserves termination for strictly more CS-TRSs than Lucas' contractive transformation [Luc96], denoted by $\mathcal{R}^{\mu}_{L}$, and Zantema's transformation [Zan97], denoted by $\mathcal{R}^{\mu}_{Z}$. For instance, μ-termination of $\mathcal{R}$ in the introduction cannot be proved by using either $\mathcal{R}^{\mu}_{L}$ or $\mathcal{R}^{\mu}_{Z}$, while $\mathcal{R}^{\mu}_{GM}$ of example 3 can be proved terminating using the dependency pairs technique [AG00] with a lexicographic combination of polynomial interpretations.

However in practice, many times $\mathcal{R}^{\mu}_{L}$ and $\mathcal{R}^{\mu}_{Z}$ can more easily be proved terminating automatically. In fact, although it is terminating, we do not know about any available automatic termination proof system able to prove the termination of $\mathcal{R}^{\mu}_{GM}$ in Example 3. In the Section 6 we will provide a very simple proof of termination using our method.

Here we show that RPO can only be used to prove termination of the transformed system when the original system is already provable terminating by RPO (i.e. the system is terminating without any restriction). Note that in such a case there is no reason for proving μ-termination.

Theorem 3. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and μ a replacement map. If $\mathcal{R}$ is not* rpo-*terminating, then $\mathcal{R}^{\mu}_{GM}$ is not* rpo-*terminating.*

Proof. For the second group of rules, any proof of *rpo*-termination of $\mathcal{R}^{\mu}_{GM}$ would need $\mathtt{mark} \succ_{\mathcal{F}} \mathtt{active}$ in the precedence. Therefore, for the first group of rules we also need $l \succeq_{rpo} \mathtt{mark}(r)$ for all rule $l \to r \in R$, which implies $l \succ_{rpo} r$.

Moreover note that even when $\mathcal{R}$ is *rpo*-terminating, it can still happen that $\mathcal{R}^{\mu}_{GM}$ is not *rpo*-terminating. This is showed in the following example.

Example 4. Consider the following *rpo*-terminating TRS $\mathcal{R}$ (consider $f \succ_{\mathcal{F}} b$):

$$f(b(x)) \to b(f(x))$$

Let $\mu(b) = \mu(f) = \{1\}$. Then, $\mathcal{R}^{\mu}_{GM}$:

$$\begin{array}{llll} \mathtt{active}(f(b(x))) & \to \mathtt{mark}(b(f(x))) & \mathtt{mark}(b(x)) & \to \mathtt{active}(b(\mathtt{mark}(x))) \\ \mathtt{mark}(f(x)) & \to \mathtt{active}(f(\mathtt{mark}(x))) & \mathtt{active}(x) & \to x \end{array}$$

is not *rpo*-terminating: the rule $\mathtt{mark}(f(x)) \to \mathtt{active}(f(\mathtt{mark}(x)))$ requires $\mathtt{mark} \succ_{\mathcal{F}} \mathtt{active}, f$; in that case, the rule $\mathtt{active}(f(b(x))) \to \mathtt{mark}(b(f(x)))$ cannot be satisfied by any RPO.

Using Knuth-Bendix orderings is even worse, since only when the original system is already kbo-terminating and we are considering the minimum replacement map $\mu_{\perp}$ (defined as $\mu_{\perp}(f) = \emptyset$ for all symbols f) the termination of $\mathcal{R}^{\mu}_{GM}$ may be provable by KBO.

Theorem 4. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and μ a replacement map. If $\mu \neq \mu_{\perp}$ or $\mathcal{R}$ is not* kbo-*terminating, then $\mathcal{R}^{\mu}_{GM}$ is not* kbo-*terminating.*

Proof. (sketch) First it is shown that if $\mu \neq \mu_\perp$ then for any admissible weight function there is at least one rule in the second group rules that cannot be proved. Second, if $\mu = \mu_\perp$ then using the fact that $\mathcal{R}$ is not *kbo*-terminating then for any admissible weight function either we cannot prove the rules of the first group or we cannot prove the rules of the second one.

Giesl and Middeldorp proposed a second transformation method, denoted by $\mathcal{R}^\mu_C$, which is correct and complete. The idea is to permit *a single* (context-sensitive) reduction step each time. They achieve this by using new symbols f' for each (non-constant) symbol $f \in \mathcal{F}$ and shifting a single symbol `active` to (non-deterministically) reach the replacing position where the redex is placed. The application of a rewrite rule changes `active` into `mark` which is propagated upwards through the term, in order to be replaced by a new symbol `active` that enables new reduction steps. After checking that no 'strange' symbols remain uncontrolled (using a symbol `proper` such that $\mathtt{proper}(t)$ reduces to $\mathtt{ok}(t)$ if and only if t is a ground term of the original signature), a rule $\mathtt{top}(\mathtt{ok}(x)) \to \mathtt{top}(\mathtt{active}(x))$ enables a new reduction step (see [GM99] for a detailed explanation). Given a TRS $\mathcal{R} = (\mathcal{F}, R)$ and a replacement map μ, the TRS $\mathcal{R}^\mu_C = (\mathcal{F} \cup \{f' \mid f \in \mathcal{F} \wedge ar(f) > 0\} \cup \{\mathtt{active}, \mathtt{mark}, \mathtt{ok}, \mathtt{proper}, \mathtt{top}\}, R^\mu_C)$ consists of the following rules: for all $l \to r \in R$, $f \in \mathcal{F}$ such that $k = ar(f) > 0$, $i \in \mu(f)$, and constants $c \in \mathcal{F}$,

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

Regarding *rpo*-termination this new transformation is even less powerful than the previous one since *rpo*-termination of $\mathcal{R}^\mu_C$ implies *rpo*-termination of $\mathcal{R}^\mu_{GM}$. Therefore only in a few (useless) cases RPO can be applied.

Theorem 5. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and μ a replacement map. If $\mu \neq \mu_\perp$ or $\mathcal{R}$ is not* rpo-*terminating, then $\mathcal{R}^\mu_C$ is not* rpo-*terminating.*

Finally regarding *kbo*-termination we have the following result, which states that if there is a constant symbol (which is always the case) and a non-unary function symbol then KBO cannot be applied.

Theorem 6. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and μ a replacement map. If there exist $c, f \in \mathcal{F}$ such that $ar(c) = 0$ and $ar(f) \geq 2$, then $\mathcal{R}^\mu_C$ is not* kbo-*terminating.*

4 The context-sensitive recursive path ordering

First we provide some intuition behind the definition of the ordering we are going to present. Following the example of the introduction we consider a replacement map satisfying $\mu(cons) = 1$ (i.e. blocking the second argument).

A first attempt to define an RPO for context-sensitive rewriting (which closely relates to the ideas behind the transformation method given in [Luc96]) is to simply consider that the second argument of *cons* does not exist, i.e. it should not be considered when *cons* is heading the right hand side term and it cannot be used when *cons* is heading the left hand side term. This ordering is a very simple μ-reduction ordering, but unfortunately, it will fail in many cases due to the variables occurring in an active position in the right hand side which are in a blocked position in the left hand side.

Therefore, since removing blocked subterms does not work, the reasonable alternative would be to mark the symbols which are in blocked positions and consider them smaller than the active ones. Therefore terms in blocked positions become smaller.

However, if we simply apply RPO to the terms after marking the symbols in blocked positions the resulting ordering is not stable under substitutions. The problem comes from the treatment of variables, since variables in blocked positions are somehow smaller than variables in active positions, which is not taken into account in RPO. For instance, following the example above, when comparing the terms $cons(x, l)$ and l, the variable l in the first term is in a blocked position while in the second one it is not.

If we solve this problem by simply marking the variables in blocked positions then the resulting ordering would be too weak. For instance, a term t like $tail(cons(x, l))$ could not be greater than l, since the l in t would be marked.

The key idea for obtaining a powerful ordering is using not only the variable l in $tail(cons(x, l))$ but the whole term to take care of the l, since if *tail* is the greatest symbol in the precedence for any substitution σ we have that $tail(cons(x, l))\sigma$ is greater than $l\sigma$, since *tail* is "taking care" of the head symbol of $l\sigma$ which is the one that is not marked.

However, this idea is not enough if we mark all symbols below blocked positions. The alternative is to mark only the necessary symbols. Therefore, an additional ingredient to our ordering is a *marking map*, denoted by $\mathfrak{m}$, which defines for every symbol and every blocked position the set of symbols that should be marked. For instance, in our example, we may have $\mathfrak{m}(cons, 2) = \{zeros, incr\}$, which means that if either the constant symbol *zeros* or the function symbol *incr* are heading the second argument of *cons* they will become the marked symbols $\underline{zeros}$ and $\underline{incr}$ respectively. For any other symbol no marking is applied. With such a marking, taking $tail \succ_{\mathcal{F}} zeros$ and $tail \succ_{\mathcal{F}} incr$ is enough to ensure that $tail(cons(x, l))$ is greater than l for all substitutions.

Finally, many times it is necessary to propagate the marking to the arguments of marked symbols. This can easily be expressed using the same marking map $\mathfrak{m}$ but defined on the arguments of the marked symbols. For instance, if we want to propagate the marking of $\underline{incr}$ to the symbol *cons* we take $\mathfrak{m}(\underline{incr}, 1) = \{cons\}$ and again we can decide whether we want to propagate through $\underline{cons}$ or not. Note that we can mark any symbol under a marked symbol since they are all blocked positions.

This notion of marking map generalizes the replacement map if we consider that positions headed by a marked symbol are blocked. Hence marking maps may be seen as a more flexible way to define blocked positions. As a consequence of our work, it turns out that many times considering a smaller set of blocked positions can simplify the termination proof.

To conclude this first look at our method, let us mention that we will not apply the marking map to the whole term but only to the top symbol of the arguments in the recursive calls of the definition of our recursive path ordering. Therefore the only one symbol that can be marked in the terms is the head symbol.

4.1 Marked symbols and marking maps

Let μ be a replacement map and $\mathcal{F}$ be a signature. By $\underline{\mathcal{F}}$ we denote the set of marked symbols corresponding to $\mathcal{F}$ and by $\underline{\mathcal{X}}$ we denote the set of labeled variables corresponding to $\mathcal{X}$. The variables in $\underline{\mathcal{X}}$ are labeled by subsets of $\mathcal{F}$, for instance $x_{\{f,g,h\}}$, and we will ambiguously use the variables of $\mathcal{X}$ to denote variables labeled by the empty set.

Given a symbol f in $\mathcal{F} \cup \underline{\mathcal{F}}$ and a positive integer i in $\{1...ar(f)\}$, a marking map $\mathfrak{m}$ provides the subset of symbols in $\mathcal{F}$ that should be marked, i.e. $\mathfrak{m}(f,i) \subseteq \mathcal{F}$.

For instance, given the symbols $f, g, h \in \mathcal{F}$, $\mathfrak{m}(f,2) = \{g,h\}$ means that if the top of the second component of a term headed by f is g or h, when we obtain this subterm, its top symbol (g or h) will be marked, and $\mathfrak{m}(\underline{g},1) = \{h\}$ means that if the top of the first component of a term headed by $\underline{g}$ is h, when we reach the subterm this symbol h will be marked.

The precedence $\succeq_{\mathcal{F}}$ and the marking map $\mathfrak{m}$ has to satisfy some conditions. On the one hand, to preserve μ-monotonicity we need to ensure that $\mathfrak{m}$ never marks an active position, i.e. it must mark only blocked positions, and on the other hand, to be able to have a powerful treatment of variables, any symbol in $\mathcal{F}$ has to be greater than or equal to in the precedence than its marked version and it cannot mark more than its marked version. If these conditions are fulfilled, we say that $(\succeq_{\mathcal{F}}, \mathfrak{m})$ is a *valid* marking pair.

Definition 4. *Let $\succeq_{\mathcal{F}}$ be a precedence, μ a replacement map and $\mathfrak{m}$ a marking map. Then $(\succeq_{\mathcal{F}}, \mathfrak{m})$ is a valid marking pair if*

1. $\mathfrak{m}(f,i) = \emptyset \quad \forall f \in \mathcal{F}, \quad \forall i \in \mu(f)$
2. $f \succeq_{\mathcal{F}} \underline{f} \quad \forall f \in \mathcal{F}$
3. $\mathfrak{m}(f,i) \subseteq \mathfrak{m}(\underline{f},i) \quad \forall f \in \mathcal{F}, \ \forall i \in \{1...ar(f)\}$

When using the ordering, the marking map tell us whether we have to mark the top symbol every time we go to an argument. Therefore, if we have a term $f(s_1...s_k)$, we will access to the arguments using $mt(s_i, \mathfrak{m}(f,i))$, which represents the result of marking the top symbol when required, and it is defined as:

$$mt(f(s_1...s_n), W) = \begin{array}{ll} \underline{f}(s_1...s_n) & if \quad f \in W \\ f(s_1...s_n) & otherwise \end{array}$$

$$mt(x, W) = x_W \quad \text{where } x \text{ is a variable}$$

We say that a term $s \in \underline{\mathcal{T}}(\mathcal{F}, \mathcal{X})$ if $s \in \underline{\mathcal{X}}$, or $s = f(s_1...s_n)$, with $f \in \mathcal{F} \cup \underline{\mathcal{F}}$ and $s_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ for all $i \in \{1...n\}$. Note that, as said, marked symbols can only appear at the top of a term. A ground term s is in $\underline{\mathcal{T}}(\mathcal{F})$ if it is in $\underline{\mathcal{T}}(\mathcal{F}, \mathcal{X})$ and contains no variable.

4.2 Dealing with labeled variables

In this section we will present the way we are going to deal with the labeled variables. Roughly speaking, given a term s, we will provide the set of labeled variables that can be considered smaller than (or equal to) s without risk of losing stability under substitutions.

The basic idea is the following. As seen, we label the variables with the symbols that should be marked in case of applying a substitution. To ensure that some labeled variable x_W is in the set of safe (wrt. stability) labeled variables of a term s, we need x to occur in s and to be sure that for any substitution σ we have that $mt(x\sigma, W)$ is smaller than $s\sigma$. Therefore, assuming that x occurs in s, the important point is what happens with the function symbols heading $x\sigma$. Due to this we analyze which function symbols are harmless as head symbols. In all cases the symbols which are included in the label W of x. Additionally, all function symbols which do not appear in the label when we reach some occurrence of x in s are safe. Finally, and more importantly, the symbols g that can be proved to be safe because the head symbol of s (or recursively using some subterm of s containing x) is greater than or equal to g (and in the latter case they have multiset status), and $\underline{g}$ and g have the same marking.

Definition 5. *Let s be a non-variable term in $\underline{\mathcal{T}}(\mathcal{F}, \mathcal{X})$ and x_W a labeled variable. Then $x_W \in Stable(s)$ if and only if $x \in \mathcal{V}ar(s)$ and $f \in Safe(s, x)$ for all $f \in \mathcal{F} \setminus W$.*

The set $Safe(s, x)$ for some variable x s.t. $x \in \mathcal{V}ar(s)$ or $s = x_V$ (for some label V) is defined as the smallest subset of $\mathcal{F}$ containing

1. *if $s = x_V$ then all symbols in $\mathcal{F} \setminus V$.*
2. *if $s = f(s_1, \ldots, s_n)$ then*
   (a) *the union of all $Safe(mt(s_i, \mathfrak{m}(f, i)), x)$ with $i \in \{1 \ldots n\}$ and $x \in \mathcal{V}ar(s_i)$, and*
   (b) *all $g \in \mathcal{F}$ such that $f =_{\mathcal{F}} g$ and $stat(f) = stat(g) = mul$, and $(\mathfrak{m}(\underline{g}, i) = \mathfrak{m}(g, i))$ for all $i \in \{1...ar(g)\}$.*
   (c) *all $g \in \mathcal{F}$ such that $f \succ_{\mathcal{F}} g$ and $(\mathfrak{m}(\underline{g}, i) = \mathfrak{m}(g, i))$ for all $i \in \{1...ar(g)\}$.*

For instance, given the term $tail(cons(x, l))$, the marking map $\mathfrak{m}(cons, 2) = \{zeros, incr\}$ and the precedence $tail \succ_{\mathcal{F}} zeros$ and $tail \succ_{\mathcal{F}} incr$, we have that $l_\emptyset \in Stable(tail(cons(x, l)))$ as $l \in \mathcal{V}ar(tail(cons(x, l)))$ and for all $f \in \mathcal{F} \setminus \emptyset$, $f \in Safe(tail(cons(x, l)), x)$. Since $\mathfrak{m}(cons, 2) = \{zeros, incr\}$ then for all $f \in \mathcal{F} \setminus \{zeros, incr\}$ we have that $f \in Safe(tail(cons(x, l)), x)$ by applying first case 2a twice and then case 1. Finally, we have that $zeros, incr \in Safe(tail(cons(x, l)), x)$ by case 2c twice since $tail \succ_{\mathcal{F}} zeros$ and $tail \succ_{\mathcal{F}} incr$ and $\mathfrak{m}(incr, 1) = \mathfrak{m}(\underline{incr}, 1) = \emptyset$.

4.3 The ordering

First we give the definition of the equality relation, induced by the equality on function symbols, that we will use.

Definition 6. *Given two terms in* $\underline{\mathcal{T}}(\mathcal{F}, \mathcal{X})$*, we define* $=_{\mathcal{S}}$ *as follows:*

- $f(s_1...s_k) =_{\mathcal{S}} g(t_1...t_k)$ *iff* $f =_{\mathcal{F}} g$ *and* $mt(s_i, \mathfrak{m}(f,i)) =_{\mathcal{S}} mt(t_i, \mathfrak{m}(g,i))$ *for every* $i \in \{1, \dots, k\}$.
- $x_W =_{\mathcal{S}} x_{W'}$ *iff* $W = W'$

We can enlarge the equality relation by considering permutations of arguments of symbols with multiset status.

Now we can give the definition of our context-sensitive recursive path ordering.

Definition 7. *Let* $s, t \in \underline{\mathcal{T}}(\mathcal{F}, \mathcal{X})$
$s = f(s_1...s_n) \succ_{\mathcal{S}} t$ *with* $t \in \underline{\mathcal{X}}$ *or* $t = g(t_1...t_m)$ *iff*

1. $t = x_W \in Stable(s)$
2. *or* $mt(s_i, \mathfrak{m}(f,i)) \succeq_{\mathcal{S}} t$*, for some* $i \in \{1...n\}$
3. *or* $t = g(t_1...t_m)$ *with* $f \succ_{\mathcal{F}} g$ *and* $s \succ_{\mathcal{S}} mt(t_i, \mathfrak{m}(g,i))$ *for all* $i \in \{1...m\}$
4. *or* $t = g(t_1...t_m)$ *with* $f =_{\mathcal{F}} g$*,* $stat(f) = mul$ *and* $\{mt(s_1, \mathfrak{m}(f,1)), ..., mt(s_n, \mathfrak{m}(f,n))\} \succ\succ_{\mathcal{S}} \{mt(t_1, \mathfrak{m}(g,1)), ..., mt(t_m, \mathfrak{m}(g,m))\}$
5. *or* $t = g(t_1...t_m)$*,* $f =_{\mathcal{F}} g$*,* $stat(f) = lex$*,* $\langle mt(s_1, \mathfrak{m}(f,1)), ..., mt(s_n, \mathfrak{m}(f,n))\rangle \succ_{\mathcal{S}}^{lex} \langle mt(t_1, \mathfrak{m}(g,1)), ..., mt(t_m, \mathfrak{m}(g,m))\rangle$ *and* $s \succ_{\mathcal{S}} mt(t_i, \mathfrak{m}(g,i))$ *for all* $i \in \{1...m\}$.

where $s \succeq_{\mathcal{S}} t$ *denotes* $s \succ_{\mathcal{S}} t$ *or* $s =_{\mathcal{S}} t$*, and* $\succ\succ_{\mathcal{S}}$ *and* $\succ_{\mathcal{S}}^{lex}$ *are respectively the multiset and lexicographic extension of* $\succ_{\mathcal{S}}$ *wrt.* $=_{\mathcal{S}}$.

The ordering satisfies the following properties:

Lemma 1.

1. $=_{\mathcal{S}}$ *and* $\succ_{\mathcal{S}}$ *are transitive.*
2. $\succ_{\mathcal{S}}$ *is compatible with* $=_{\mathcal{S}}$.
3. $f(t_1...t_k) \succeq_{\mathcal{S}} \underline{f}(t_1...t_k)$ *for all* $f \in \mathcal{F}$ *and* $\forall t_i \in \mathcal{T}(\mathcal{F})$.

Theorem 7. $\succ_{\mathcal{S}}$ *over terms in* $\mathcal{T}(\mathcal{F}, \mathcal{X})$ *is a ground stable* μ*-reduction ordering.*

μ-monotonicity easily follows, and well-foundedness is shown directly by contradiction on the existence of a minimal (in the sense of e.g. Nash-Williams' proof of Kruskal's theorem [NW63]) infinite decreasing sequence with $\succ_{\mathcal{S}}$. It can also be proved using the general method described in [Gou01].

For stability we first proof that $\succ_{\mathcal{S}}$ is stable under ground μ-substitutions, which are substitution satisfying that $x_W\sigma = mt(x\sigma, W)$ for all variables in the domain of σ. This is the most difficult part, an requires to prove that if $x_W \in Stable(s)$ then $s\sigma \succ_{\mathcal{S}} x_W\sigma$ for all ground μ-substitutions σ. Finally, for terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$, i.e. without marked symbols, we can show stability under ground substitutions.

Note that the ordering can be adapted to be stable under substitutions (not only ground). We have not included this definition since it is a bit more complex and there is no gain in practice.

5 Examples

First, two simple "training" examples are presented. Although none of them can be proved by $\mathcal{R}^\mu_Z$, only the second one holds with $\mathcal{R}^\mu_L$ and e.g. dependency pairs technique is needed for $\mathcal{R}^\mu_{GM}$, they both have a very easy proof using CSRPO.

Example 5. (taken from [GM99]).

$$\begin{array}{rcl} g(x) & \to & h(x) \\ c & \to & d \\ h(d) & \to & g(c) \end{array}$$

with $\mu(g) = \mu(h) = \emptyset$.

Termination is proved using CSRPO with the marking $\mathfrak{m}(g,1) = \mathfrak{m}(\underline{g},1) = \mathfrak{m}(h,1) = \mathfrak{m}(\underline{h},1) = \{c\}$ and the precedence $c \succ_{\mathcal{F}} d \succ_{\mathcal{F}} g \succ_{\mathcal{F}} h$ and $d \succ_{\mathcal{F}} \underline{c}$.

Example 6. (taken from [Zan97]).

$$f(x) \to g(h(f(x)))$$

with $\mu(f) = \mu(h) = \{1\}$ and $\mu(g) = \emptyset$.

We can prove that this rule is included in $\succ_{\mathcal{S}}$ with the marking $\mathfrak{m}(g,1) = \mathfrak{m}(\underline{g},1) = \{h\}$, $\mathfrak{m}(\underline{h},1) = \{f\}$ and the precedence $f \succ_{\mathcal{F}} g, \underline{h}, \underline{f}$.

The following example present the definition of some functions that can handle infinite lists.

Example 7. Lists.

$$\begin{array}{ccc} from(x) & \to & cons(x, from(s(x))) \\ head(cons(x, xs)) & \to & x \\ 2nd(cons(x, xs)) & \to & head(xs) \\ take(0, xs) & \to & nil \\ take(s(n), cons(x, xs)) & \to & cons(x, take(n, xs)) \\ sel(0, cons(x, xs)) & \to & x \\ sel(s(n), cons(x, xs)) & \to & sel(n, xs) \end{array}$$

with $\mu(cons) = \mu(2nd) = \mu(from) = \mu(head) = \mu(s) = \{1\}$ and $\mu(take) = \mu(sel) = \{1, 2\}$.

This CS-TRS is included in $\succ_{\mathcal{S}}$ with the marking $\mathfrak{m}(cons, 2) = \mathfrak{m}(\underline{cons}, 2) = \{from\}$; the precedence $2nd \succ_{\mathcal{F}} \{head, from\}$, $take \succ_{\mathcal{F}} \{from, nil, cons\}$ and $sel \succ_{\mathcal{F}} from \succ_{\mathcal{F}} \{cons, s, \underline{from}\}$; and the status $stat(take) = stat(sel) = lex$ and mul for all others. Note that $take$ and sel must be greater than $from$, because it is marked by $cons$.

In the next example, the function fib produces (in an efficient way) the infinite sequence of Fibonacci numbers. This example can also be proved using $\mathcal{R}^\mu_Z$ and RPO.

Example 8. Fibonacci.

$$\begin{array}{rcl} fib(n) & \to & sel(n, fib1(s(0), s(0))) \\ fib1(x, y) & \to & cons(x, fib1(y, add(x, y))) \\ add(0, x) & \to & x \\ add(s(x), y) & \to & s(add(x, y)) \\ sel(0, cons(x, xs)) & \to & x \\ sel(s(n), cons(x, xs)) & \to & sel(n, xs) \end{array}$$

with $\mu(cons) = \mu(fib) = \mu(s) = \{1\}$ and $\mu(fib1) = \mu(add) = \mu(sel) = \{1, 2\}$.

We can prove it with $\succ_{\mathcal{S}}$ using the marking $\mathfrak{m}(cons, 2) = \mathfrak{m}(\underline{cons}, 2) = \{fib1\}$; the precedence $fib \succ_{\mathcal{F}} \{sel, fib1, s, 0\}$, $sel \succ_{\mathcal{F}} fib1 \succ_{\mathcal{F}} \{cons, add, \underline{fib1}\}$ and $add \succ_{\mathcal{F}} s$; and the status $stat(sel) = lex$ and mul for all others.

The last example of this section defines a function which produces the infinite list of prime numbers, with some zeros inserted. In fact, the list coincides with the list of natural numbers (from 2 on), where every non-prime number has been replaced by zero, e.g. $2, 3, 0, 5, 0, 7, 0, 0, 0, 11, \ldots$ This example is a slight modification of a definition given in [KdV02]. This example can also be proved with $\mathcal{R}^{\mu}_{L}$ and RPO, but not with $\mathcal{R}^{\mu}_{Z}$ and RPO.

Example 9. Primes.

$$\begin{array}{rcl} filter(cons(x, y), 0, m) & \to & cons(0, filter(y, m, m)) \\ filter(cons(x, y), s(n), m) & \to & cons(x, filter(y, n, m)) \\ sieve(cons(0, y)) & \to & cons(0, sieve(y)) \\ sieve(cons(s(n), y)) & \to & cons(s(n), sieve(filter(y, n, n))) \\ nats(n) & \to & cons(n, nats(s(n))) \\ zprimes & \to & sieve(nats(s(s(0)))) \end{array}$$

with $\mu(cons) = \mu(nats) = \mu(sieve) = \mu(s) = \{1\}$ and $\mu(filter) = \{1, 2, 3\}$.

We can prove it with $\succ_{\mathcal{S}}$ using the marking $\mathfrak{m}(cons, 2) = \mathfrak{m}(\underline{cons}, 2) = \mathfrak{m}(\underline{filter}, 1) = \mathfrak{m}(\underline{sieve}, 1) = \{filter, sieve, nats\}$; the precedence $zprimes \succ_{\mathcal{F}} \{sieve, nats, s, 0\}$, $nats \succ_{\mathcal{F}} \{cons, \underline{nats}, s\}$ and $sieve \succ_{\mathcal{F}} \{cons, \underline{sieve}, \underline{filter}\}$ $filter \succ_{\mathcal{F}} \{cons, \underline{filter}\}$; and the status mul for all symbols.

Note that the list of prime numbers can be obtained by removing the zeroes from the list, but then proving termination of this example would be much harder (maybe impossible automatically) since termination follows from the fact that there are no infinitely many consecutive zeros in the list (which holds since there are infinitely many prime numbers).

6 Improvements

The presented ordering, although it is quite powerful, cannot prove examples like the one in the introduction. This is due to the fact that conditions we have imposed in the definition of $Safe$ in order to ensure stability under substitution are too strong. In particular, in case 2c, when we say that g is a safe symbol for a term s headed by f with $f \succ_{\mathcal{F}} g$, the condition $\mathfrak{m}(g,i) = \mathfrak{m}(\underline{g},i)$ is added to avoid problems with the propagated marks by $\underline{g}$. By a deeper analysis, we can see that if f is greater than all symbols that are marked by propagation from $\underline{g}$ then g is also safe. In fact we only have to be careful with the symbols that are marked by propagation from $\underline{g}$ and not from g.

The following definition describes this set of "dangerous" symbols with respect to propagation of marks for a given symbol g.

Definition 8. *Given a mapping $\mathfrak{m}$ and a function symbol g in $\mathcal{F}$, the set $prop(g)$ is defined as the smallest set satisfying that $h \in prop(g)$ if*

- *$h \in \mathfrak{m}(\underline{g},i) \setminus \mathfrak{m}(g,i)$ for some $i \in \{1...ar(f)\}$*
- *$h \in prop(f)$ and $f \in prop(g)$*

Note that we can actually compute $prop(g)$ as a closure, starting from the symbols satisfying the first condition and computing their propagation until no new symbol is added to the set.

Let us give the new definition of $Safe$. Note that the definition of *Stable* and the definition of the ordering remain unchanged.

Definition 9. *Let s be a non-variable term in $\underline{\mathcal{T}}(\mathcal{F},\mathcal{X})$. The set $Safe(s,x)$ for some variable x s.t. $x \in \mathcal{V}ar(s)$ or $s = x_V$ (for some label V) is defined as the smallest subset of $\mathcal{F}$ containing*

1. *if $s = x_V$ then all symbols in $\mathcal{F} \setminus V$.*
2. *if $s = f(s_1, \ldots, s_n)$ then*
   (a) *the union of all $Safe(mt(s_i, \mathfrak{m}(f,i)), x)$ with $i \in \{1 \ldots n\}$ and $x \in \mathcal{V}ar(s_i)$, and*
   (b) *all $g \in \mathcal{F}$ such that $f =_{\mathcal{F}} g$ and $stat(f) = stat(g) = mul$, and $(\mathfrak{m}(\underline{g},i) = \mathfrak{m}(g,i))$ for all $i \in \{1...ar(g)\}$.*
   (c) *all $g \in \mathcal{F}$ such that $f \succ_{\mathcal{F}} g$ and $f \succ_{\mathcal{F}} h$ for all $h \in prop(g)$.*

With this new definition of $Safe$ the ordering keeps the same properties (and the modification only affects to the proof of stability under ground substitutions).

Theorem 8. *$\succ_{\mathcal{S}}$ over terms in $\mathcal{T}(\mathcal{F},\mathcal{X})$ is a ground stable μ-reduction ordering.*

Now we can easily prove the termination of the example in the introduction.

Example 10. Proof of example 1.

For this example we just need the marking map $\mathfrak{m}(cons,2) = \mathfrak{m}(\underline{cons},2) = \mathfrak{m}(\underline{adx},1) = \mathfrak{m}(\underline{incr},1) = \{zeros, adx, incr\}$, (for all other symbol and argument positions it is the empty set); the precedence $tail \succ_{\mathcal{F}} \{zeros, adx, incr\}$, $zeros \succ_{\mathcal{F}} \{cons, \underline{zeros}, 0\}$, $nats \succ_{\mathcal{F}} \{adx, zeros\}$, $adx \succ_{\mathcal{F}} incr \succ_{\mathcal{F}} \{cons, s\}$, $adx \succeq_{\mathcal{F}} \underline{adx}$, $incr \succeq_{\mathcal{F}} \underline{incr}$; and multiset status for all symbols.
Note that to prove the last rule we need to use case 2c of the definition of $Safe$.

Finally let us mention that using this notion of propagation but defined on marked symbols as well, we can relax one of the conditions for having a valid marking pair (see Definition 4), but by now we have not seen any interesting example in which this improvement is necessary.

7 Conclusions

In this paper we have presented the first method for directly proving termination of context-sensitive rewriting. Our method, called the context-sensitive recursive path ordering, is the first alternative to transformation methods. We have presented some negative results on the practical applicability of Giesl's and Middeldorp transformation methods. Our results show that these transformations are not useful combined with general-purpose termination proof methods like RPO or KBO.

In fact, we have also made some experiments using the C*i*ME 2.0 system (available at `http://cime.lri.fr`), which includes an implementation of the dependency pairs technique with polynomial interpretations, for proving the termination of the transformed system (with the different methods) for all examples given in this paper. The results have been quite poor: with $\mathcal{R}^{\mu}_{L}$ only examples 6 and 9 can be proved; with $\mathcal{R}^{\mu}_{Z}$ no example could be proved; with $\mathcal{R}^{\mu}_{GM}$ only examples 6 and 9 hold (and example 5 holds as well if we use an improved version of $\mathcal{R}^{\mu}_{GM}$ which consists of normalizing the right-hand sides of the transformed system); and with the complete transformation $\mathcal{R}^{\mu}_{C}$ only example 6 was proved.

As we have seen, using CSRPO we can easily prove termination of all these examples, including the ones where the transformation methods cannot work or do not work in practice. Moreover, CSRPO is very suitable for automation. The main difficulty for obtaining a fully automatic system based on CSRPO is to automatically generate in a clever way, not only the precedence (as in the implementations of standard RPO) but also the marking map. Note that a naive implementation, which exhaustively checks all possible precedences and markings, may be inefficient (for examples involving many function symbols with large arities), but, in practice, only a few well-chosen markings would have to be tried.

As future developments, apart from the implementation of the method, we want to extend the ideas behind CSRPO, especially the ones concerning the treatment of variables, to more powerful ordering-based methods in order to obtain for context-sensitive rewriting the same kind of automatic tools that can be used for standard rewriting.

Acknowledgements. We would like to thank Robert Nieuwenhuis for his helpful comments on this work.

References

[AG00] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.

[AZ96] Thomas Arts and Hans Zantema. Termination of logic programs using semantic unification. *Fifth International Workshop on Logic Program Synthesis and Transformation*, LNCS 1048:219–233. Springer, 1996.

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

[BLR02] C. Borralleras, S. Lucas and A. Rubio. Recursive Path Orderings can be Context-Sensitive. Available at `www.lsi.upc.es/~albert/papers.html`, 2002. Long version.

[Dau92] Max Dauchet. Simulation of turing machines by a regular rewrite rule. *Theoretical Computer Science*, 103(2):409–420, 1992.

[Der82] Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.

[FR99] M.C.F. Ferreira and A.L. Ribeiro. Context-Sensitive AC-Rewriting. *Proc. of 10th International Conference on Rewriting Techniques and Applications*, LNCS 1631:286-300, Springer, 1999.

[GM99] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. *Proc. of 10th International Conference on Rewriting Techniques and Applications*, LNCS 1631:271–285, Springer, 1999.

[GM01] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. *Proc. of 1th International Workshop on Rewriting Proof and Computation, RPC'01*, pages 14-33, RIEC, Tohoku University, 2001.

[Gou01] Jean Goubault-Larrecq. Well-Founded Recursive Relations. *Proc. 15th Int. Workshop Computer Science Logic*, LNCS 2142:484–497, Springer, 2001.

[GW93] Harald Ganzinger and Uwe Waldmann. Termination proofs of well-moded logic programs via conditional rewrite systems. *Proc. of 3rd International Workshop on Conditional Term Rewriting Systems*, LNCS 656:113–127, Springer, 1993.

[KB70] D.E. Knuth and P.B. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[KdV02] J. Kennaway and F.J. de Vries. Infinitary rewriting. In *Term Rewriting Systems*. Cambridge University Press, 2002. To appear.

[Luc96] S. Lucas. Termination of context-sensitive rewriting by rewriting. *Proc. of 23rd. International Colloquium on Automata, Languages and Programming, ICALP'96*, LNCS 1099:122-133, Springer, 1996.

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

[Luc01] S. Lucas. Termination of Rewriting With Strategy Annotations. *Proc. of 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, LNAI 2250:669-684, Springer, 2001.

[NW63] C. St. J. A. Nash-Williams. On well-quasi-ordering finite trees. *Proceedings of the Cambridge Philosophical Society*, 59(4):833–835, 1963.

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

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

Appendix

Given a finite signature $\mathcal{F}$, a Knuth-Bendix ordering on $\mathcal{T}(\mathcal{F},\mathcal{X})$ is determined by [BN98]:

1. A strict ordering $\succ_{\mathcal{F}}$ on $\mathcal{F}$,
2. A weight function

$$w : \mathcal{F} \cup \mathcal{X} \rightarrow \mathbb{R}_0^+$$

   where $\mathbb{R}_0^+$ are the non-negative real numbers.

The weight function w is admissible if

1. There exists $w_0 \in \mathbb{R}_0^+ - \{0\}$ such that $w(x) = w_0$ for all variables $x \in \mathcal{X}$ and $w(c) > w_0$ for all constants $c \in \mathcal{F}$.
2. If $f \in \mathcal{F}$ is a unary symbol of weight $w(f) = 0$, then f is the greatest element in $\mathcal{F}$, i.e., $f \geq g$ for all $g \in \mathcal{F}$.

The weight function extends to $\mathcal{T}(\mathcal{F},\mathcal{X})$ as follows: for $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$,

$$w(t) \;=\; \sum_{x \in \mathcal{V}ar(t)} w(x) \cdot |t|_x \quad + \quad \sum_{f \in \mathcal{F}} w(f) \cdot |t|_f$$

According to this, the Knuth-Bendix ordering $>_{kbo}$ on $\mathcal{T}(\mathcal{F},\mathcal{X})$ induced by $>$ and w is as follows: for $t, s \in \mathcal{T}(\mathcal{F},\mathcal{X})$,

$$t >_{kbo} s$$

if and only if $|t|_x \geq |s|_x$ for every $x \in \mathcal{X}$ and either

KBO1 $w(t) > w(s)$, or
KBO2 $w(t) = w(s)$, and one of the following properties holds
- **KBO2a** There are a unary function symbol f, a variable x and a positive integer n such that $t = f^n(x)$ and $s = x$.
- **KBO2b** There exist function symbols $f, g \in \mathcal{F}$ such that $f \succ_{\mathcal{F}} g$ and $t = f(t_1, \ldots, t_{ar(f)})$, $s = g(s_1, \ldots, s_{ar(g)})$.
- **KBO2c** There exist a function symbol $f \in \mathcal{F}$ and an index i, $1 \leq i \leq ar(f)$ such that $t = f(t_1, \ldots, t_{ar(f)})$, $s = f(s_1, \ldots, s_{ar(f)})$, $t_j = s_j$ for $1 \leq j < i$ and $t_i >_{kbo} s_i$.