

A note on completeness of conditional context-sensitive rewriting*

Salvador Lucas

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

Abstract

Restricting the ability of the rewrite relation to perform reductions can lead to loose computational power in rewriting-based programming languages and systems. Despite this fact, advanced specification and programming languages like *CafeOBJ*, *Maude*, and *OBJ* permit the introduction of such computational restrictions by means of programmable strategies and suitable syntactic annotations. The main reason is to achieve a better termination behavior. Canonical context-sensitive rewriting has been proposed as a suitable restriction of rewriting which can improve the termination behavior of TRSs whereas is still able to achieve useful (normalizing and infinitary normalizing) computations. In this paper we discuss a number of drawbacks arising when using canonical context-sensitive rewriting with Conditional Term Rewriting Systems (which provide an abstract computational model for the aforementioned languages) and propose some appropriate solutions.

Keywords: Programming languages, Conditional term rewriting.

1 Introduction

Context-sensitive rewriting (*CSR* [Luc98]) is a restriction of rewriting which forbids reductions on selected arguments of functions. In *CSR*, a *replacement map* μ indicates the set of argument positions $\mu(f)$ which can be rewritten for each symbol f . With *CSR* we can *achieve* a terminating behavior with non-terminating Term Rewriting Systems (TRSs), by pruning (all) infinite rewrite sequences. We refer to [GM04, Luc98, Luc02, Luc04b] for further details, applications, methods, and references about *CSR* and termination of *CSR*. On the other hand, although the use of *CSR* usually dismisses many rewriting steps,

*Work partially supported by Spanish MEC grant SELF TIN 2004-07943-C04-02, Acción Integrada HU 2003-0003, and EU-India Cross-Cultural Dissemination project ALA/95/23/2003/077-054.

with *canonical CSR* we are still able to compute head-normal forms which can then be used to reach normal forms and infinite normal forms of terms [Luc02]. Canonical *CSR* is an specialization of *CSR* that only uses replacement maps which are less restrictive than the *canonical* replacement map $\mu_{\mathcal{R}}^{can}$ of the TRS \mathcal{R} . The canonical replacement map $\mu_{\mathcal{R}}^{can}$ is *the most restrictive replacement map ensuring that the non-variable subterms of the left-hand sides of the rules of \mathcal{R} are replacing* [Luc98, Luc02]. This replacement map can be automatically computed for each TRS (for instance, the tool MU-TERM can do that [Luc04a]) thus giving the programmer the possibility of using *CSR* without explicitly introducing hand-crafted replacement restrictions.

As shown in [BM03, Luc01a, Luc01b], *CSR* provides a suitable framework to model computations with *strategy annotations* and *frozen arguments* as used in (unconditional) **Maude** programs [CDEL⁺02, CDEL⁺03]. Since advanced programming languages like **Maude** allow for the use of conditional rules and equations, context-sensitive rewriting for TRSs has been recently generalized to Conditional TRSs as part of a research effort to develop both theory and tools to reason about the termination of specifications in advanced equational languages [DLM⁺04]. This generalization is quite straightforward: just use *CSR* instead of rewriting when evaluating the (instances of the) *conditions* as part of the application of a conditional (oriented) rule.

When considering canonical replacement maps μ with conditional CTRSs, one is tempted to think that the good properties of canonical *CSR* (for TRSs) are also available in this new setting. In this paper we will see that this is *not* the case and discuss possible solutions to this problem. In Section 3, we focus on *normal* CTRSs, where each condition $s_i \rightarrow t_i$ satisfies that t_i is a *ground* normal form [Ohl02, Definition 7.1.3(2)].

Inspired by the results in [Luc98, Luc02], where it is proved that canonical *CSR* (for left-linear TRSs) is able to obtain *ground* normal forms t of terms s provided that either the replacement map μ makes all positions of t replacing ([Luc98, Theorem 12]), or we use the normalization via μ -normalization procedure, which performs a layered normalization of terms by computing intermediate μ -normal forms ([Luc02, Theorems 22 and 23]), we explore two main refinements of the basic computational model for conditional *CSR*:

1. *Change the notion of canonical replacement map for CTRSs*, to eventually relax the replacement restrictions as much as demanded by the terms t_i in the conditions $s_i \rightarrow t_i$ of all conditional rules in the CTRS, or
2. *Change the evaluation model for the conditional part of the rules* to evaluate the conditions by using normalization via μ -normalization. In this setting we also explore the use of *transformations* from CTRSs into CTRSs which simulate this process by using *CSR* thus keeping the simple evaluation of conditions by *CSR* without requiring any weakening of the canonical replacement map.

In Section 4, we further discuss the application of our results to more general CTRSs like join CTRSs. Section 5 concludes.

2 Preliminaries

We refer the reader to [Ohl02] to recall the usual notions and notations regarding term rewriting and CTRSs. A conditional rule is as follows: $l \rightarrow r$ if $s_1 = t_1, \dots, s_n = t_n$, where $l, r, s_1, t_1, \dots, s_n, t_n$ are terms. As usual, l and r are called the left- and right-hand sides of the rule, and the sequence $s_1 = t_1, \dots, s_n = t_n$ (often denoted c) is the *conditional part* of the rule. Given a CTRS, $L(\mathcal{R})$ is the set of left-hand sides of rules and $RC(\mathcal{R})$ is the set of terms in the right hand side of a condition $s_i \rightarrow t_i$ in the conditional part of a conditional rule. Rewrite rules $l \rightarrow r$ if c are classified according to the distribution of variables among l , r , and c , as follows: type 1, if $\text{Var}(r) \cup \text{Var}(c) \subseteq \text{Var}(l)$; type 2, if $\text{Var}(r) \subseteq \text{Var}(l)$; type 3, if $\text{Var}(r) \subseteq \text{Var}(l) \cup \text{Var}(c)$; and type 4, if no restriction is given. An n -CTRS contains only rewrite rules of type $m \leq n$.

It is well-known that the conditions $s_i = t_i$ for $1 \leq i \leq n$ can be interpreted in a number of different ways [DO90]. Join CTRSs (often called *standard* CTRSs) interpret the equality symbol $=$ as joinability ($\downarrow_{\mathcal{R}}$). As in [Ohl02], however, we are mainly concerned with *oriented* CTRSs, whose (conditional) rules are written $l \rightarrow r$ if $s_1 \rightarrow t_1, \dots, s_n \rightarrow t_n$ indicating that the conditions $s_i \rightarrow t_i$ for $1 \leq i \leq n$ are intended to express the reachability of (instances of) t_i from (instances of) s_i . A *normal* CTRS \mathcal{R} is an oriented CTRS such that every t_i is a ground normal form (w.r.t. the unconditional TRS \mathcal{R}_u obtained by removing the conditional part from each conditional rule of \mathcal{R}) for $1 \leq i \leq n$. An oriented 3-CTRS \mathcal{R} is called *deterministic* if for each rule $l \rightarrow r$ if $s_1 \rightarrow t_1, \dots, s_n \rightarrow t_n$ in \mathcal{R} and each $1 \leq i \leq n$, we have $\text{Var}(s_i) \subseteq \text{Var}(l) \cup \bigcup_{j=1}^{i-1} \text{Var}(t_j)$.

Let \mathcal{R} be a CTRS. We inductively define unconditional TRSs \mathcal{R}_n for $n \in \mathbb{N}$ by $\mathcal{R}_0 = \emptyset$ and $\mathcal{R}_{n+1} = \{\sigma(l) \rightarrow \sigma(r) \mid l \rightarrow r \text{ if } s_1 \rightarrow t_1, \dots, s_n \rightarrow t_n \in \mathcal{R} \text{ and } \sigma(s_i) \rightarrow_{\mathcal{R}_n}^* \sigma(t_i) \text{ for all } i \in \{1, \dots, n\}\}$. The rewrite relation $\rightarrow_{\mathcal{R}}$ associated with a CTRS \mathcal{R} is $\bigcup_{n \in \mathbb{N}} \rightarrow_{\mathcal{R}_n}$.

A term t is a head-normal form if it does not reduce to a term s issuing a reduction step at the root.

2.1 Context-sensitive rewriting

A mapping $\mu : \mathcal{F} \rightarrow \mathcal{P}(\mathbb{N})$ is a *replacement map* (or \mathcal{F} -map) if $\forall f \in \mathcal{F}, \mu(f) \subseteq \{1, \dots, ar(f)\}$ [Luc98]. Let $M_{\mathcal{F}}$ be the set of all \mathcal{F} -maps. We write $\mu \sqsubseteq \mu'$ if for all $f \in \mathcal{F}, \mu(f) \subseteq \mu'(f)$. The set of μ -*replacing positions* $\mathcal{P}os^{\mu}(t)$ of $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is: $\mathcal{P}os^{\mu}(t) = \{\Delta\}$, if $t \in \mathcal{X}$ and $\mathcal{P}os^{\mu}(t) = \{\Delta\} \cup \bigcup_{i \in \mu(\text{root}(t))} i.\mathcal{P}os^{\mu}(t|_i)$, 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 $p \in \mathcal{P}os^{\mu}(t)$.

The *canonical replacement map* $\mu_{\mathcal{R}}^{can}$ for a TRS \mathcal{R} is the most restrictive replacement map which ensures that the (positions of) non-variable subterms of the left-hand sides of the rules of \mathcal{R} are replacing [Luc98, Luc02]. Note that $\mu_{\mathcal{R}}^{can}$ can be automatically associated to \mathcal{R} by means of a very simple calculus: for each symbol $f \in \mathcal{F}$ and $i \in \{1, \dots, ar(f)\}$,

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

Given a TRS \mathcal{R} , $CM_{\mathcal{R}} = \{\mu \in M_{\mathcal{R}} \mid \mu_{\mathcal{R}}^{can} \sqsubseteq \mu\}$ is the set of replacement maps which are less restrictive than or equally restrictive to $\mu_{\mathcal{R}}^{can}$. One of the most important properties of the canonical replacement map is the following.

Theorem 1 [Luc98] *Let \mathcal{R} be a left-linear TRS and $\mu \in CM_{\mathcal{R}}$. Every μ -normal form is a head-normal form.*

Context-sensitive rewriting has been extended to CTRSs in [DLM⁺04]. Let $\mathcal{R} = (\mathcal{F}, R)$ be a CTRS and $\mu \in M_{\mathcal{F}}$ be a replacement map. We inductively define unconditional TRSs \mathcal{R}_n for $n \in \mathbb{N}$ by $\mathcal{R}_0 = \emptyset$ and $\mathcal{R}_{n+1} = \{\sigma(l) \rightarrow \sigma(r) \mid l \rightarrow r \text{ if } s_1 \rightarrow t_1, \dots, s_n \rightarrow t_n \in \mathcal{R} \text{ and } \sigma(s_i) \hookrightarrow_{\mathcal{R}_n}^* \sigma(t_i) \text{ for all } i \in \{1, \dots, n\}\}$. The rewrite relation $\hookrightarrow_{\mathcal{R}}$ associated with a CTRS \mathcal{R} is $\bigcup_{n \in \mathbb{N}} \hookrightarrow_{\mathcal{R}_n}$. The definition of canonical replacement map for a CTRS is the same given above (when considering conditional rules instead of unconditional ones).

3 Refined computation models for conditional CSR with normal CTRSs

As remarked in the introduction, the good properties of canonical CSR (for TRSs) are not immediately available when CTRSs and conditional CSR are considered. In this section we focus on *normal* CTRSs; the following example illustrates the problem:

Example 1 *Consider the following normal CTRS \mathcal{R} :*

$$\begin{array}{ll} b \rightarrow c(d) & a \rightarrow h(e) \text{ if } b \rightarrow c(e) \\ d \rightarrow e & a \rightarrow c(a) \end{array}$$

together with $\mu(h) = \{1\}$ and $\mu(c) = \emptyset$. Note that $\mu = \mu_{\mathcal{R}}^{can} \in CM_{\mathcal{R}}$. However, although the reduction step

$$a \rightarrow h(e)$$

can be obviously performed by using the conditional rule, it is not possible with CSR. The reason is that the auxiliary sequence

$$b \rightarrow c(d) \rightarrow c(e)$$

which is necessary to trigger the conditional rule is not possible with CSR; we have

$$b \hookrightarrow c(d)$$

but $c(d)$ is a μ -normal form: no further μ -rewriting step is possible.

Thus, the problem is that, since conditional rewriting in normal CTRSs embeds normalizing reduction sequences issued to decide about the satisfaction of the conditions, the satisfaction of the conditions can be impossible with CSR. In [Luc98, Luc02], we have investigated how to directly or indirectly obtain normal forms by using *canonical* context-sensitive rewriting computations (with TRSs). In the following sections we investigate possible solutions to the incompleteness of conditional CSR with normal CTRSs.

3.1 Weakening the canonical replacement map

In [Luc98], we have investigated how to obtain normal forms by using *canonical CSR* (with TRSs). Given a term $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, μ_s is the most restrictive replacement map ensuring that all non-variable symbols in s are replacing, i.e., $\mathcal{P}os_{\mathcal{F}}(s) \subseteq \mathcal{P}os^{\mu_s}(s)$ [Luc98]. We have the following generalization of [Luc98, Theorem 12] whose proof is, actually, the same.

Theorem 2 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear TRS, $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, and $s \in \mathcal{T}(\mathcal{F})$. Let $\mu \in CM_{\mathcal{R}}$ be such that $\mu_s \sqsubseteq \mu$. Then, $t \rightarrow^* s$ iff $t \hookrightarrow^* s$.*

As shown in Example 1, this result does not hold for (normal) CTRSs. Thus, regarding its extension to CTRSs, we define the *cond-canonical* replacement map $\mu_{\mathcal{R}}^{ccan}$ for a CTRS \mathcal{R} as the most restrictive replacement map satisfying that for all left-hand sides l of the (conditional or unconditional) rules and *right-hand sides* t_i of all conditions $s_i \rightarrow t_i$, the non-variable subterms are replacing: For each symbol $f \in \mathcal{F}$ and $i \in \{1, \dots, ar(f)\}$,

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

Obviously, $\mu_{\mathcal{R}}^{can} \sqsubseteq \mu_{\mathcal{R}}^{ccan}$. Let $CCM_{\mathcal{R}}$ be the set of replacement maps which are less restrictive than $\mu_{\mathcal{R}}^{ccan}$; we have:

Theorem 3 *Let \mathcal{R} be a left-linear normal CTRS and $\mu \in CCM_{\mathcal{R}}$. Every μ -normal form is a head-normal form.*

Consider the Maude modules **LazyLNat**, **DIVIDES** and **PRIMES** in Appendix A which implement the generation of the list of prime numbers. The strategy annotations (0) and (10) given to the constructors **s** and ‘.’ of natural numbers and lists of natural numbers, respectively (see module **LazyLNat** in Appendix A) are treated as the replacement restrictions $\mu(\mathbf{s}) = \emptyset$ and $\mu(\cdot) = \{1\}$ [Luc01a, Luc01b]. On the other hand, no strategy annotation is given to any other symbol; thus, we have $\mu(f) = \{1, \dots, ar(f)\}$ for any other symbol f in the program. It is not difficult to see that $\mu \in CCM_{\mathcal{R}}$. This three modules conform a *normal* CTRS: although the conditions have equational syntax, since the right-hand sides of the equations are (constant) normal forms, they can equivalently be considered as oriented conditions (from left to right). Therefore, since $\mu_{\mathcal{R}}^{can} = \mu_{\mathcal{R}}^{ccan}$, Theorem 3 applies to guarantee that the evaluation of any expression yields (at least) a head-normal form.

```
Maude> red primes .
reduce in PRIMES : primes .
rewrites: 4 in 0ms cpu (0ms real) (~ rewrites/second)
result LNat: s(0 + s(0)) . filter(remove(s(0 + s(0)),
natsFrom(s(0) + s(0 + s(0))))))
```

We obtain a list in head-normal form whose first component is the first prime number (2, or, more properly, $\mathbf{s}(\mathbf{s}(0))$), although represented in the non-completely evaluated form $\mathbf{s}(0 + \mathbf{s}(0))$. This is due to the configuration of

the replacement map μ . Although the design of the program could be improved to avoid this (e.g., to obtain $\mathbf{s}(\mathbf{s}(\mathbf{0}))$ instead), the current version of the program is intended to easily show the problems we are interested in (also in Section 4 below). The following result extends Theorem 2 to conditional *CSR* with normal CTRSs and cond-canonical replacement maps.

Theorem 4 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear normal CTRS, $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, and $s \in \mathcal{T}(\mathcal{F})$. Let $\mu \in CCM_{\mathcal{R}}$ be such that $\mu_s \sqsubseteq \mu$. Then, $t \rightarrow^* s$ iff $t \hookrightarrow^* s$.*

The following example shows that the requirement of being a *normal* CTRS cannot be dropped in Theorem 4.

Example 2 *Consider the following oriented CTRS \mathcal{R} :*

$$\begin{array}{ll} \mathbf{b} \rightarrow \mathbf{c}(\mathbf{d}) & \mathbf{f}(\mathbf{x}) \rightarrow \mathbf{h}(\mathbf{x}) \text{ if } \mathbf{b} \rightarrow \mathbf{c}(\mathbf{x}) \\ \mathbf{d} \rightarrow \mathbf{e} & \mathbf{a} \rightarrow \mathbf{c}(\mathbf{a}) \end{array}$$

together with $\mu(\mathbf{f}) = \mu(\mathbf{h}) = \mu(\mathbf{c}) = \emptyset$. Note that $\mu = \mu_{\mathcal{R}}^{ccan} \in CCM_{\mathcal{R}}$. However, although the reduction step

$$\mathbf{f}(\mathbf{e}) \rightarrow \mathbf{h}(\mathbf{e})$$

can be performed by using the conditional rule, it is not possible with CSR. Again, we need the auxiliary sequence in Example 1 which is still not possible with CSR.

According to Theorem 4, the modification of the replacement map μ given in Example 1 to let $\mu(\mathbf{c}) = \{1\}$ makes *CSR* able to obtain $\mathbf{h}(\mathbf{e})$ from \mathbf{a} with the CTRS \mathcal{R} of Example 1. The problem, now, is that the system becomes non-terminating (due to the rule $\mathbf{a} \rightarrow \mathbf{c}(\mathbf{a})$). In the following sections, we show how to avoid this by using a new computation model to evaluate the conditional part of the rules of the CTRS.

3.2 Layered evaluation of the conditions

As shown in [Luc02], when considering left-linear TRSs \mathcal{R} and replacement maps $\mu \in CM_{\mathcal{R}}$, we can use the normalization via μ -normalization procedure to obtain normal forms. It performs a layered normalization of terms by computing intermediate μ -normal forms. In fact, we can formalize this as a reduction relation as follows. Given a replacement map $\mu \in M_{\mathcal{F}}$ and a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, the *maximal replacing context* $MRC^{\mu}(t)$ of t consists of the maximal part of t whose positions are μ -replacing in t [Luc02, Definition 1]. Then, we introduce the following.

Definition 1 (Layered Context-Sensitive Rewriting) *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, $\mu \in M_{\mathcal{R}}$ and $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. We write $t \hookrightarrow s$ if either*

1. $t \hookrightarrow s$, or
2. t is a μ -normal form, $t = C[t_1, \dots, t_i, \dots, t_n]$, where $C[\] = MRC^{\mu}(t)$, $s = C[t_1, \dots, s_i, \dots, t_n]$, and $t_i \hookrightarrow s_i$ for some $i \in \{1, \dots, n\}$.

It is not difficult to see that, in contrast with *CSR*, the normal forms of layered *CSR* (abbreviated *LCSR*) are always normal forms, i.e., $\text{NF}_{\hookrightarrow_{\mathcal{R}}} = \text{NF}_{\rightarrow_{\mathcal{R}}}$. Now, we can rephrase [Luc02, Theorem 22] as follows:

Theorem 5 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear TRS, $\mu \in \text{CM}_{\mathcal{R}}$, and $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Then, $t \rightarrow^! s$ if and only if $t \hookrightarrow^! s$.*

This result does not hold if the requirement of left-linearity is dropped.

Example 3 *Consider the following TRS \mathcal{R} [Luc02, Example 21]:*

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

together with $\mu(\mathbf{f}) = \{1\}$. Note that $\mu \in \text{CM}_{\mathcal{R}}$. Although

$$\mathbf{f}(\mathbf{b}, \underline{\mathbf{c}}) \rightarrow \underline{\mathbf{f}(\mathbf{b}, \mathbf{b})} \rightarrow \mathbf{a}$$

we cannot reach \mathbf{a} from $t = \mathbf{f}(\mathbf{b}, \mathbf{c})$ by using \hookrightarrow because the only \hookrightarrow -step which is possible from t yields t again:

$$\mathbf{f}(\underline{\mathbf{b}}, \mathbf{c}) \hookrightarrow \mathbf{f}(\mathbf{b}, \mathbf{c})$$

Theorem 5 does not hold if $\mu \notin \text{CM}_{\mathcal{R}}$ (see [Luc02, Example 37]). Unfortunately (although not surprising, since $\hookrightarrow \subseteq \rightarrow$), Theorem 5 does not hold for arbitrary reducts (i.e., we cannot change $\rightarrow^!$ by \rightarrow^* and $\hookrightarrow^!$ by \hookrightarrow^*):

Example 4 *Consider the following left-linear TRS \mathcal{R} :*

$$\begin{array}{l} \mathbf{g}(x) \rightarrow \mathbf{c}(x, x) \\ \mathbf{a} \rightarrow \mathbf{b} \end{array}$$

together with $\mu(\mathbf{g}) = \mu(\mathbf{c}) = \{1\}$. Again, $\mu \in \text{CM}_{\mathcal{R}}$. Then,

$$\underline{\mathbf{g}(\mathbf{a})} \rightarrow \mathbf{c}(\underline{\mathbf{a}}, \underline{\mathbf{a}}) \rightarrow \mathbf{c}(\mathbf{a}, \mathbf{b})$$

However, although $\mathbf{g}(\mathbf{a}) \hookrightarrow \mathbf{c}(\mathbf{a}, \mathbf{a})$, since $\mathbf{c}(\mathbf{a}, \mathbf{a})$ is not a μ -normal form and the second reduction step is issued on a non-replacing argument of \mathbf{c} , we have that $\mathbf{c}(\mathbf{a}, \mathbf{a}) \not\hookrightarrow \mathbf{c}(\mathbf{a}, \mathbf{b})$. On the other hand, if we apply more \hookrightarrow -reduction steps, then we definitely lose the opportunity of reaching $\mathbf{c}(\mathbf{a}, \mathbf{b})$:

$$\mathbf{c}(\underline{\mathbf{a}}, \mathbf{a}) \hookrightarrow \mathbf{c}(\mathbf{b}, \underline{\mathbf{a}}) \hookrightarrow \mathbf{c}(\mathbf{b}, \mathbf{b})$$

i.e., in fact, $\mathbf{c}(\mathbf{a}, \mathbf{a}) \not\hookrightarrow^* \mathbf{c}(\mathbf{a}, \mathbf{b})$. This problem also happens if we replace variable occurrences in the first rule by symbol \mathbf{a} to obtain a modified TRS.

Theorem 5 suggests the following refinement of the definition of conditional *CSR*: Define now the unconditional TRSs \mathcal{R}_n for $n \in \mathbb{N}$ by $\mathcal{R}_0 = \emptyset$ and $\mathcal{R}_{n+1} = \{\sigma(l) \rightarrow \sigma(r) \mid l \rightarrow r \text{ if } s_1 \rightarrow t_1, \dots, s_n \rightarrow t_n \in \mathcal{R} \text{ and } \sigma(s_i) \hookrightarrow_{\mathcal{R}_n}^* \sigma(t_i) \text{ for all } i \in \{1, \dots, n\}\}$. We will speak, then, of oriented (or normal) *layered* CTRSs. The μ -rewriting relation $\hookrightarrow_{\mathcal{R}}$ associated with a CTRS \mathcal{R} is, again, $\bigcup_{n \in \mathbb{N}} \hookrightarrow_{\mathcal{R}_n}$ (so, what is actually changing is the ‘number’ of unconditional TRSs \mathcal{R}_n which contribute to the context-sensitive rewriting relation).

Theorem 6 *Let \mathcal{R} be a left-linear normal layered CTRS and $\mu \in \text{CM}_{\mathcal{R}}$. Every μ -normal form is a head-normal form.*

We can go further beyond and associate a layered context-sensitive rewrite relation \hookrightarrow to the layered CTRS. In this case, we are able to obtain normal forms.

Theorem 7 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear normal layered CTRS, $\mu \in CM_{\mathcal{R}}$, and $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Then, $t \rightarrow^! s$ if and only if $t \hookrightarrow^! s$.*

Unfortunately, Theorem 7 does not apply to oriented CTRSs which are not normal.

Example 5 *Consider the following oriented CTRS \mathcal{R} :*

$$\begin{array}{l} g(\mathbf{x}) \rightarrow c(\mathbf{x}, \mathbf{x}) \quad \mathbf{f}(\mathbf{x}) \rightarrow \mathbf{h}(\mathbf{x}) \text{ if } g(\mathbf{x}) \rightarrow c(\mathbf{x}, \mathbf{b}) \\ \mathbf{a} \rightarrow \mathbf{b} \end{array}$$

together with $\mu(c) = \{1\}$ and $\mu(\mathbf{f}) = \mu(\mathbf{h}) = \emptyset$. Note that $\mu = \mu_{\mathcal{R}}^{can} \in CM_{\mathcal{R}}$. However, although the reduction step

$$\mathbf{f}(\mathbf{a}) \rightarrow \mathbf{h}(\mathbf{a})$$

is possible due to the existence of a reduction sequence

$$\underline{g}(\underline{\mathbf{a}}) \rightarrow c(\underline{\mathbf{a}}, \underline{\mathbf{a}}) \rightarrow c(\underline{\mathbf{a}}, \mathbf{b})$$

this sequence is not possible with LCSR.

3.3 Transforming normal CTRSs

Another possibility to repair the completeness problem that we are discussing *without* changing the simple computational model of *CSR* given in Section 2.1 is to simulate the ‘layered’ evaluation of the conditions with \hookrightarrow by performing a *layered flattening* of the right-hand sides t_i of the conditions $s_i \rightarrow t_i$ of a conditional rule. Let $lflat(s \rightarrow t)$ be a sequence $s_1 \rightarrow t_1, \dots, s_n \rightarrow t_n$ of pairs of terms $s_i \rightarrow t_i$ inductively defined as follows:

1. if $MRC^{\mu}(t) = t$, then $lflat(s \rightarrow t) = s \rightarrow t$;
2. otherwise, let $C[\] = MRC^{\mu}(t)$, $t = C[t_1, \dots, t_m]$, and x_1, \dots, x_m be fresh (renamed apart) variables. Then,

$$lflat(s \rightarrow t) = s \rightarrow C[x_1, \dots, x_m], lflat(x_1 \rightarrow t_1), \dots, lflat(x_m \rightarrow t_m)$$

Now, given a normal, layered CTRS \mathcal{R} , we obtain a *deterministic* (but not layered!) CTRS $F(\mathcal{R})$ as follows:

$$F(\mathcal{R}) = \{l \rightarrow r \text{ if } \bigwedge_{s_i \rightarrow t_i \in c} lflat(s_i \rightarrow t_i) \mid l \rightarrow r \text{ if } c \in \mathcal{R}\}$$

Note that this transformation does *not* introduce any new function symbol. Note, however, that, in general, $F(\mathcal{R})$ is *not* a normal CTRS. When considering replacement maps μ with \mathcal{R} , we assume that the same replacement map is used with $F(\mathcal{R})$.

Example 6 Consider the normal CTRS \mathcal{R} and μ as in Example 1. Then, $F(\mathcal{R})$ is:

$$\begin{array}{ll} b \rightarrow c(d) & a \rightarrow h(e) \text{ if } b \rightarrow c(x), x \rightarrow e \\ d \rightarrow e & a \rightarrow c(a) \end{array}$$

The following result establishes that this transformation permits a strong simulation of a left-linear and layered normal CTRSs \mathcal{R} by $F(\mathcal{R})$.

Theorem 8 Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear normal layered CTRS, $\mu \in CM_{\mathcal{R}}$, and $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Then, $t \hookrightarrow_{\mathcal{R}} s$ if and only if $t \hookrightarrow_{F(\mathcal{R})} s$.

As a consequence of this result, the μ -termination of \mathcal{R} for a layered normal CTRS \mathcal{R} can be proved as μ -termination of the CTRS $F(\mathcal{R})$. According to [DLM⁺04, Section 3.2], this can be done by using a variant of Ohlebusch's transformation [Ohl02] which takes into account the replacement maps.

Example 7 The μ -termination of $F(\mathcal{R})$ in Example 6 (where μ is as in Example 1) can be proved as μ' -termination of the following TRS $U(F(\mathcal{R}))$:

$$\begin{array}{ll} b \rightarrow c(d) & a \rightarrow u1(b) \\ d \rightarrow e & u1(c(x)) \rightarrow u2(x, x) \\ a \rightarrow c(a) & u2(e, x) \rightarrow h(e) \end{array}$$

where $\mu'(h) = \mu'(u1) = \mu'(u2) = \{1\}$ and $\mu'(c) = \emptyset$. The μ' -termination of $U(F(\mathcal{R}))$ can be proved by using Zantema's transformation, see [GM04, Luc04b] for further details.

An immediate consequence of Theorems 7 and 8 is the following:

Theorem 9 Let $\mathcal{R} = (\mathcal{F}, R)$ be a left-linear normal layered CTRS, $\mu \in CM_{\mathcal{R}}$, and $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Then, $t \rightarrow_{\mathcal{R}}^! s$ if and only if $t \hookrightarrow_{F(\mathcal{R})}^! s$.

4 Beyond normal CTRSs

Although the results in Section 3 provide a suitable setting for using (layered) CSR with normal (layered) CTRSs, there are many applications where we need more flexibility, i.e., we need to use join or oriented CTRSs. Consider the following Maude module **TwinPRIMES** which (together with the modules in Appendix A) permits to obtain the list of the *twin* primes, which are pairs of primes which differ by two (e.g., $\langle 3, 5 \rangle$, $\langle 5, 7 \rangle$, $\langle 11, 13 \rangle$, etc.).

```
fmod TwinPRIMES is
pr PRIMES .
sorts PNat .
subsort PNat < Nat .
op <_,_> : Nat Nat -> PNat .
op takePairs : LNat -> LNat .
op twinPrimes : -> LNat .
vars P Q : Nat .
```

```

vars PS QS : LNat .
ceq takePairs(P . PS) = < P , Q > . takePairs(Q . PS)
  if Q . QS := PS /\ s(s(P)) = Q .
ceq takePairs(P . Q . PS) = takePairs(Q . PS)
  if Q . QS := PS /\ le(s(s(P)),Q) = true .
ceq twinPrimes = takePairs(PS) if P . PS := primes .
endfm

```

Here, **Maude** pattern matching expressions $Q . QS := PS$ are treated as oriented conditions $PS \rightarrow Q . QS$ and the equality conditions (e.g., $s(s(P)) = Q$) are given *joinability* semantics: **Maude** proceeds by first evaluating s and t and then comparing the corresponding evaluated expressions \bar{s} and \bar{t} for syntactic equality. The evaluation of `twinPrimes` in **Maude** yields:

```

Maude> red twinPrimes .
reduce in TwinPRIMES : twinPrimes .
rewrites: 185 in 0ms cpu (0ms real) (~ rewrites/second)
result LNat: takePairs(s(0 + s(0 + s(0))) . filter(remove(s(0 + s(0 + s(0))),
  remove(s(0 + s(0)), natsFrom(s(0) + s(0 + s(0 + s(0)))))))

```

Note that we do not obtain neither the first pair of twin primes nor a head-normal form! This is due to the failed evaluation of the condition $s(s(P)) = Q$ in the first conditional rule for `takePairs`: P is bound to $s(0 + s(0 + s(0)))$, i.e., the head of the list which is passed to `takePairs`; the tail of the list instantiates PS . Variable Q takes the value $s(0 + s(0 + s(0 + s(0 + s(0))))$ which is the head of the result of evaluating PS :

```

Maude> red filter(remove(s(0 + s(0 + s(0))),
>   remove(s(0 + s(0)), natsFrom(s(0) + s(0 + s(0 + s(0)))))) .
reduce in TwinPRIMES : filter(remove(s(0 + s(0 + s(0))), remove(s(0 + s(0)),
  natsFrom(s(0) + s(0 + s(0 + s(0)))))) .
rewrites: 142 in 0ms cpu (0ms real) (~ rewrites/second)
result LNat: s(0 + s(0 + s(0 + s(0 + s(0)))) . filter(remove(s(0 + s(0 + s(0 +
  s(0 + s(0))))), remove(s(0 + s(0 + s(0))), remove(s(0 + s(0)), natsFrom(s(
  0) + s(0 + s(0 + s(0 + s(0 + s(0))))))))))

```

Now, the test of the instance

$$s(s(s(0 + s(0 + s(0)))))) = s(0 + s(0 + s(0 + s(0 + s(0))))))$$

of $s(s(P)) = Q$ *fails* due to the strategy annotation (0) for s (i.e., $\mu(s) = \emptyset$): both $s(s(s(0 + s(0 + s(0))))))$ and $s(0 + s(0 + s(0 + s(0 + s(0))))$ are already evaluated because no reduction is possible below symbol s . Since $s(s(s(0 + s(0 + s(0)))))) \neq s(0 + s(0 + s(0 + s(0 + s(0))))$, the conditional rule is not triggered.

Equational conditions $s = t$ as given in module `TwinPrimes` can be easily simulated by a normal CTRS by introducing a new symbol `eq` defined by the rule $eq(x, x) \rightarrow true$, and encoding the condition as $eq(s, t) \rightarrow true$ (see, e.g., [DLM⁺04]). Unfortunately, however, the addition of such a rule destroys left-linearity and makes our current results unapplicable (and, in fact, it does not work with **Maude**, since this is what the interpreter does!). Nevertheless,

consider the previous Maude program. If we add a new symbol `!=` for *strict* equality of expressions of the sort `Nat` defined by:

```
eq 0 != 0 = true .
eq s(P) != s(Q) = P != Q .
```

together with $\mu(=!) = \{1, 2\}$, then the ‘problematic’ condition $s(s(P)) = Q$ becomes $(s(s(P)) != Q) = \text{true}$. Now, the evaluation of `twinPairs` yields:

```
Maude> red twinPrimes .
reduce in TwinPRIMESeq : twinPrimes .
rewrites: 198 in 0ms cpu (0ms real) (~ rewrites/second)
result LNat: < s(0 + s(0 + s(0))), s(0 + s(0 + s(0 + s(0 + s(0)))) > .
      takePairs(s(0 + s(0 + s(0 + s(0 + s(0)))) . filter(remove(s(0 + s(0 + s(
      0))), remove(s(0 + s(0)), natsFrom(s(0) + s(0 + s(0 + s(0))))))
```

5 Conclusions

We have discussed a number of drawbacks arising when using (canonical) *CSR* with CTRSs and proposed some appropriate solutions. We have introduced the notion of *cond-canonical* replacement map $\mu_{\mathcal{R}}^{can}$ which (in this setting) plays the role of the canonical replacement map $\mu_{\mathcal{R}}^{an}$ (in the unconditional one). We have also introduced the notions of *layered* context-sensitive rewriting (*LCSR*), which formalizes the normalization via μ -normalization procedure of [Luc02] as a reduction relation, and that of *layered* CTRS, where the evaluation model for the conditional part of the rules uses *LCSR* instead of *CSR*. In this way, we can avoid the use of $\mu_{\mathcal{R}}^{can}$ (which is less restrictive than $\mu_{\mathcal{R}}^{an}$ and can, then, induce a worse termination behavior) to ensure the desired properties. We have also introduced a *transformation* from normal, layered CTRSs into CTRSs which simulate conditional *CSR* with normal, layered CTRSs by using conditional *CSR* (instead of *LCSR*). This transformation permits to prove termination of *CSR* with a layered CTRS \mathcal{R} as termination of *CSR* in the transformed CTRS, for which we can use existing transformations to obtain a CS-TRS (i.e., a TRS together with a replacement map) [DLM⁺04] to which apply existing method and tools for proving termination of *CSR* [GM04, Luc04b, Luc04a]. As shown in our examples, the results and methods developed here are useful to understand and improve computations with specification and programming languages like *Maude*.

Future work

Our main results concern normal CTRSs. We think, however, that our techniques and results could also be extended to more general classes of oriented CTRSs. The concrete conditions for this, however, should be further investigated.

References

- [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.
- [CDEL⁺02] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. 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.
- [DLM⁺04] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Termination of Membership Equational Programs. In P. Sestoft and N. Heintze, editors, *Proc. of ACM SIGPLAN 2004 Symposium on Partial Evaluation and Program Manipulation, PEPM'04*, pages 147-158, ACM Press, New York, 2004.
- [DO90] N. Dershowitz and M. Okada. A rationale for conditional equational programming. *Theoretical Computer Science*, 75:111-138, 1990.
- [GM04] J. Giesl and A. Middeldorp. Transformation Techniques for Context-Sensitive Rewrite Systems. *Journal of Functional Programming*, 14(4):379-427, 2004.
- [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 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.
- [Luc01b] 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.
- [Luc02] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):293-343, 2002.
- [Luc04a] S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In V. van Oostrom, editor, *Proc. of 15th International Conference on Rewriting Techniques and Applications, RTA'04*, LNCS 3091:200-209, Springer-Verlag, Berlin, 2004. Available at <http://www.dsic.upv.es/~slucas/csr/termination/muterm>.
- [Luc04b] S. Lucas. Proving Termination of Context-Sensitive Rewriting by Transformation. Technical Report DSIC-II/18/04, DSIC, Universidad Politécnica de Valencia, 2004.
- [Ohl02] E. Ohlebusch. Advanced Topics in Term Rewriting. Springer-Verlag, Berlin, 2002.

A Prime numbers with lazy lists of lazy naturals

```
fmod LazyLNat is
  sort Nat LNat .
  op 0 : -> Nat .
  op s : Nat -> Nat [strat (0)] .
  op nil : -> LNat .
  op _.._ : Nat LNat -> LNat [strat (1 0)] .
  op _+_ : Nat Nat -> Nat .
  op length : LNat -> Nat .
  op natsFrom : Nat -> LNat .
  vars M N : Nat .
  var NS : LNat .
  eq 0 + N = N .
  eq s(M) + N = s(M + N) .
  eq length(nil) = 0 .
  eq length(N . NS) = s(length(NS)) .
  eq natsFrom(N) = N . natsFrom(s(0) + N) .
endfm

fmod DIVIDES is
  pr BOOL .
  pr LazyLNat .
  op le : Nat Nat -> Bool .
  op minus : Nat Nat -> Nat .
  op ifmod : Bool Nat Nat -> Nat .
  op mod : Nat Nat -> Nat .
  op _divides_ : Nat Nat -> Bool .
  vars M N : Nat .
  eq le(0,N) = true .
  eq le(s(M),0) = false .
  eq le(s(M),s(N)) = le(M,N) .
  eq minus(M,0) = M .
  eq minus(s(M),s(N)) = minus(M,N) .
  eq mod(0,N) = 0 .
  eq mod(s(M),0) = 0 .
  eq mod(s(M),s(N)) = ifmod(le(N,M),s(M),s(N)) .
  eq ifmod(true,s(M),s(N)) = mod(minus(M,N),s(N)) .
  eq ifmod(false,s(M),s(N)) = s(N) .
  ceq s(M) divides N = true if mod(N,s(M)) = 0 .
  eq M divides N = false .
endfm

fmod PRIMES is
  pr LazyLNat .
  pr DIVIDES .
  op filter : LNat -> LNat .
  op remove : Nat LNat -> LNat .
  op primes : -> LNat .
  vars P N : Nat .
  vars NS : LNat .
  eq filter(P . NS) = P . filter(remove(P,NS)) .
  ceq remove(P,N . NS) = remove(P,NS) if P divides N = true .
  ceq remove(P,N . NS) = N . remove(P,NS) if P divides N = false .
  eq primes = filter(natsFrom(s(0) + s(0))) .
endfm
```