LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

On Deterministic Conditional Rewriting

Computation Structures Group Memo 405
December 9, 1997

Massimo Marchiori

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

On Deterministic Conditional Rewriting

Massimo Marchiori
MIT Laboratory for Computer Science
545 Technology Square, Rm. NE43-257
Cambridge, MA 02139
max@lcs.mit.edu

Abstract

The class of Deterministic Conditional Term Rewriting Systems (DCTRSs) is of utmost importance for the tight relationships exhibited with functional programming, logic programming and inductive reasoning. However, its analysis is extremely difficult, and to date there are only very few works on the subject, each analyzing a particular aspect of DCTRSs. In this paper, we perform a thorough analysis of DCTRSs, ranging from the study of termination criteria, to new verification methods for the major properties of DCTRSs like termination and confluence, to the identification of subclasses of DTCRSs that exhibit a particularly nice behaviour. Moreover, we also address the study of *modularity* of DCTRSs, providing a number of new powerful results. This is particularly important, since to the best of our knowledge there is so far *not a single result* on the modularity of DCTRSs, and of 3-CTRSs in general. Finally, most of the analysis of the paper is performed relying on the recent tool of unravelings, that allows to automatically lift results from the much simpler unconditional rewriting systems to DCTRSs. This way, we clarify what are the links between TRSs and DCTRSs, providing better intuitions on how the gained experience on TRSs can be profitably reused to understand the much more complicated world of DCTRSs.

Keywords: oriented and deterministic CTRSs, modularity, verification, unraveling.

1 Introduction

Conditional rewriting, besides being a primary field of interest for the rewriting community, is nowadays assuming an increasingly greater importance in view of the tight connections with the functional and logic programming paradigms. To this extent, the basic forms of conditional rewriting (so called "1-CTRSs"), where the conditions to be evaluated are merely tests, and do not allow parameter passing and creation of new computed values, are no longer sufficient, since to reach the power of the functional and logic programming paradigms one has to go beyond. One of the most powerful classes of CTRSs so far object of research is that of so-called "3-CTRSs", where this creation of new computed values by the conditions is allowed. In particular, the class of deterministic CTRSs (cf. [4, 9, 13]) has shown to be of utmost importance, since it is sufficiently flexible to capture the power of local definitions proper of functional languages. However, flexibility has a cost: the analysis of deterministic CTRSs (briefly, DCTRSs) is extremely difficult when compared to simpler forms of rewriting like unconditional TRSs or even conditional rewriting systems like 1-CTRSs or 2-CTRSs. As a result, there are to date only very few works that deal with the analysis 3-CTRSs (cf. [4, 9, 10, 3, 34]). In this paper, we perform a thorough analysis of DCTRSs (and, en passant, of oriented CTRSs), studying their major properties like termination and confluence. First, we perform an abstract study of termination for oriented and deterministic CTRSs, presenting new criteria for their (effective) termination, and carefully analyzing their relative strength. Next, we proceed in the analysis of DCTRSs using the recent tool of unravelings (cf. [21]. Unravelings allow to automatically infer properties of conditional rewriting systems by lifting existing results for unconditional rewriting, and to provide better intuitions on the new phenomena occurring in conditional rewriting resorting on the familiar concepts proper of TRSs. Here we develop an unraveling tailored for deterministic CTRSs, that allows to analyze the major properties of DCTRSs like (effective) termination and confluence. We also identify a distinguished subclass of DCTRS, called semi-linear DCTRSs (SCTRSs), which are in a sense what left-linear TRSs are for TRSs. For this class, the analysis becomes particularly precise, allowing to develop new powerful results, and also providing intuitions on the intimate reasons motivating the results of previous works on DCTRSs. Moreover, using the developed unraveling, we are able to provide a bunch of new modularity results for DCTRSs. Besided the practical relevance of these results, they are particularly important since, to the best of our knowledge, there are so far no results at all on the modularity of 3-CTRSs. Finally, we hint at other relevant results on DCTRSs that can be inferred from our analysis.

2 Preliminaries

We assume knowledge of the basic notions regarding conditional term rewriting systems and term rewriting systems (cf. [6, 17]).

Given a sequence of terms S, $Var(S)$ denotes the set of its variables. Two sequences S_1 and S_2 are said to be *disjoint* if they have no variables in common, i.e. $Var(S_1) \cap Var(S_2) = \emptyset$. A sequence is said to be *linear* if every variable occurs in it at most once. Also, given a sequence S and a variable X, we denote with $|S|_X$ the number of occurrences of X in S. Sequences in formulae should be seen just as abbreviations: for example, if S is the sequence t_1, t_2, then $f(S)$ denotes the term $f(t_1, t_2)$.

A term s *overlaps* a term t if there is a renaming s' of s that is disjoint with t and unifiable with a non-variable subterm of t. If s' is unifiable with t, then s *overlays* t.

As usual, a partial ordering $\succ$ on terms is said: to be *closed under substitutions* if for every substitution σ, $s \succ t \Rightarrow s\sigma \succ t\sigma$; to be *closed under contexts* if $s \succ t \Rightarrow C[s] \succ C[t]$ for every context C; to have the *subterm property* if $C[t] \succ t$ for every context $C \neq \Box$. Also, $\succ_{st}$ denotes the smallest ordering that contains $\succ$ and has the subterm property.

Finally, we will use the acronyms CON$^{\rightarrow}$ and CON to denote respectively the properties of *consistency w.r.t. reduction* (a term cannot rewrite to two different variables) and *consistency* (two variables cannot be convertible).

2.1 Oriented and Deterministic CTRSs

In this paper, we will mainly deal with so-called *oriented CTRSs* (briefly *OCTRSs*), that is to say CTRSs where the rules have the form

$$t_0 \rightarrow s_{k+1} \Leftarrow s_1 \rightarrow^* t_1, \ldots, s_k \rightarrow^* t_k$$

(that is to say, the conditions are not symmetric, like in join CTRSs where they are of the form $s \downarrow t$, but they are oriented since one of the terms is required to rewrite to the other). The term t_0, as usual, is said the *left-hand side* of the rule, s_{k+1} the *right-hand side*, and the other terms make up the *conditions*.

In general, CTRSs are divided into four categories, according to the way variables are distributed in each rewrite rule (cf. [25]). In the class of *1-CTRSs*, the variables of the right-hand side and of the conditions are contained in the variables of the left-hand side (i.e., there are no "extra variables" w.r.t. the left-hand side). In the class of *2-CTRSs*, the variables of the right-hand side are contained in those of the right-hand side (i.e., extra variables can be present in the conditions). In the class of *3-CTRSs*, the variables of the right-hand side must be contained in union of the variables of the left-hand side with those of the conditions (i.e., extra variables are also admitted in the right-hand side, provided they appear in the conditions). Finally, the class of *4-CTRSs* has no restrictions.

Deterministic CTRSs, introduced in [4] (see also [9]), are a particularly important class of 3-CTRSs:

Definition 2.1 A rule

$$t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$$

is said to be *deterministic* if $\forall i \in [1, k+1]$. $Var(s_i) \subseteq \cup_{j<i} Var(t_j)$. A CTRS is *deterministic* (briefly, a *DCTRS*), if each its rule is deterministic. $\square$

As well known (and, indeed, this was one of the main stimuli to the study of 2- and 3-CTRSs), DCTRSs are tightly linked with functional programming, since every rule $t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ can be seen in a functional language as

$$\begin{array}{l} t_0 = \texttt{let}\ t_1 = s_1\ \texttt{in} \\ \qquad\quad \ddots \\ \qquad\qquad \texttt{let}\ t_k = s_k\ \texttt{in}\ s_{k+1} \end{array}$$

That is to say, DCTRSs allows to capture the power of the functional *local definitions* provided by the `let` constructs (or, equivalently, by the `where` declarations).

2.2 Unravelings

Unravelings have been introduced in [21]. Here we will only sketch the basic notions that will be needed in the sequel.

An unraveling is a map $\mathbf{U}$ associating to every CTRS an approximating TRS, in the sense that for every CTRS R, $\downarrow_R \subseteq \downarrow_{\mathbf{U}(R)}$, and $\mathbf{U}(T \cup R) = T \cup \mathbf{U}(R)$ (T a TRS). The first condition requires that the join relation of the unraveled CTRS is an extension of the original CTRS: informally, it means that the TRS does not compute 'less' than its original CTRS; more formally, it states that the unraveled CTRS does not have less logical strength (cf. [8, 7]) than the original CTRS. The second condition says that if we are unraveling a CTRS, we can extract from it the part that is already a TRS, and then go on computing the unraveling. An unraveling is said to be *tidy* if it satisfies the following properties: 1) Compositionality: $\mathbf{U}(T_1 \cup T_2) = \mathbf{U}(T_1) \cup \mathbf{U}(T_2)$ 2) Finiteness: R finite $\Rightarrow$ $\mathbf{U}(R)$ finite 3) The unraveling of the empty TRS is the empty TRS. The first condition expresses the fact that the unraveling is compositional, i.e. that we can incrementally build up the unraveling of a CTRS by computing the unravelings of its parts. The second says that finite objects are mapped into finite objects, and the third condition implies that the unraveling of a TRS T is just T. By compositionality, tidy unravelings only need to be defined on single rules. Moreover, they are the identity function when restricted to TRSs. So, when defining a tidy unraveling we can only define it for rules with a non-void conditional part.

Unravelings allow to study properties of a CTRS by studying a corresponding property on the unraveled TRSs, the so-called ultra-property:

Definition 2.2 Let $\mathcal{P}$ be a property and $\mathbf{U}$ be an unraveling. The property *ultra-*$\mathcal{P}$ (w.r.t. $\mathbf{U}$), briefly $\mathbf{U}(\mathcal{P})$, is defined as follows: $T \in$ *ultra-*$\mathcal{P} \Leftrightarrow \mathbf{U}(T) \in \mathcal{P}$. An unraveling is said to be *sound* (resp. *complete*) for a property $\mathcal{P}$ if *ultra-*$\mathcal{P} \Rightarrow \mathcal{P}$ (resp. *ultra-*$\mathcal{P} \Leftrightarrow \mathcal{P}$). Moreover, we say that an unraveling *preserves* the property $\mathcal{P}$ if $\mathcal{P} \Rightarrow$ *ultra-*$\mathcal{P}$. $\square$

Soundness allows to infer the property $\mathcal{P}$ of a CTRSs just by studying $\mathcal{P}$ for the corresponding unraveled TRS, this way enabling to automatically lift every result on $\mathcal{P}$ for TRSs to CTRSs. In addition, preservation allows to extend a property of TRSs ($\mathcal{P}$) to CTRSs (*ultra-*$\mathcal{P}$). For instance, consider the duplication property: while this syntactic property makes sense for TRSs, it is rather awkward to say what is the proper extension of this concept to 3-CTRSs. Soundness and preservation allows to use ultra-duplication as a suitable extension. So far, the standard approach was to pass from $\mathcal{P}$ to "$\mathcal{P}$ for CTRSs" simply by forgetting about the conditional part of each rule, and applying $\mathcal{P}$ to the obtained "unconditional" TRS. This approach stays within the unraveling methodology,, since it is nothing but *ultra-*$\mathcal{P}$ w.r.t. the so-called trivial unraveling u, that translates each conditional rule $l \to r \Leftarrow \ldots$ into $l \to r$ (it is easy to check u is indeed an unraveling). However, the problem is that u may not be sound for $\mathcal{P}$, or if it is so *ultra-*$\mathcal{P}$ may be too weak to be of great interest (u gives very lossy approximant TRSs, since it just discards all the conditional part). Using better approximants (better unravelings), we can thus get better approximants of the right extension property.

Unravelings also provide a tool to automatically transfer modularity results obtained in the TRSs field to CTRSs. Recall that a map μ is said to be *compositional* w.r.t. an operator $\odot$ ($\odot$-compositional for short) if $\forall R, S.\, \mu(R \odot S) = \mu(R) \odot \mu(S)$. The following fundamental result explains why unravelings are so useful for the study of modularity:

Theorem 2.3 *Suppose an unraveling is $\odot$-compositional. Then $\mathcal{P}$ is $\odot$-modular for TRSs $\Rightarrow$ ultra-$\mathcal{P}$ is $\odot$-modular for CTRSs. Moreover, if the unraveling is complete for $\mathcal{P}$, then $\mathcal{P}$ is $\odot$-modular for TRSs $\Leftrightarrow$ $\mathcal{P}$ is $\odot$-modular for CTRSs.*

3 Termination

As well known, for finite CTRSs the termination of the rewrite relation does not imply its decidability (cf. [15]). What we need in order to ensure that a finite CTRS is computationally feasible is *effective termination* ([21, 16, 14, 8, 7]): a CTRS R is effectively terminating if $\underset{R}{\rightarrow}$ is terminating and it is decidable whether $s \underset{R}{\rightarrow} t$, $s \underset{R}{\rightarrow}^* t$, $s {\downarrow_R} t$, and if a term is in normal form. From now on, within this section we assume every CTRS to be finite.

As far as 1-CTRSs are concerned, three criteria that guarantee effective termination have been developed: *simplifyingness* ([16]), *reductivity* ([14]) and *decreasingness* ([8]). We recall here these basic notions:

Definition 3.1 A CTRS is *simplifying* if there is a simplification ordering $\succ$ such that for each its rewrite rule $t_0 \to s_{k+1} \Leftarrow s_1{\downarrow}t_1, \ldots, s_k{\downarrow}t_k$, we have $s_i \prec t_0 \succ t_i$ ($i \in [1, k+1]$). □

Definition 3.2 A CTRS R is *decreasing* if there is a partial order $\succ$ that is well-founded, has the subterm property, extends the rewrite relation of R (i.e. $\underset{R}{\rightarrow} \subseteq \succ$), and such that for each rule $t_0 \to s_{k+1} \Leftarrow s_1{\downarrow}t_1, \ldots, s_k{\downarrow}t_k$ of R, and substitution σ, we have $s_i\sigma \prec t_0\sigma \succ t_i\sigma$ for every $i \in [1, k]$. □

Reductivity is like decreasingness, but for the fact that $\succ$ is required to be closed under contexts instead of having the subterm property.

We have that simplifyingness $\Rightarrow$ reductivity $\Rightarrow$ decreasingness, and all the implications are strict.

The interest in simplifyingness lies in the fact that it is easily automatizable, for instance via RPO's.

These criteria have been developed for join CTRSs; however, it is trivial to see that they also also hold for OCTRSs.

Nevertheless, when dealing with OCTRSs, we can further refine these notions by taking advantage of the orientation of the conditions. So, we introduce these two new concepts:

Definition 3.3 An oriented CTRS is *o-simplifying* if there is a simplification ordering $\succ$ such that for each its rewrite rule $t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$, we have $t_0 \succ s_i$ $(i \in [1, k+1])$. □

Definition 3.4 An oriented CTRS R is *o-decreasing* if there is a partial ordering $\succ$ that is well-founded, has the subterm property, $\underset{R}{\to} \subseteq \succ$, and such that for each rule of R, $t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$, and substitution σ, we have $t_0\sigma \succ s_i\sigma$ $(i \in [1, k])$. □

The following results justify the introduction of these concepts, showing they behave analogously to the simplifyingness and reductivity criteria:

Theorem 3.5 *Every o-decreasing OCTRS is effectively terminating.*

Theorem 3.6 *For OCTRSs, o-simplifying $\Rightarrow$ o-decreasing, and o-simplifying $\not\Leftarrow$ o-decreasing.*

Corollary 3.7 *Every o-simplifying OCTRS is effectively terminating.*

As far as the power of these criteria is concerned, we have:

Theorem 3.8 *For DCTRSs, decreasing $\Rightarrow$ o-decreasing and simplifying $\Rightarrow$ o-simplifying. For normal CTRSs, decreasing $\not\Leftarrow$ o-decreasing, and simplifying $\not\Leftarrow$ o-simplifying.*

That is to say, o-decreasingness is strictly more powerful than decreasingness, even when restricting to normal CTRSs, and the same holds for o-simplifyingness w.r.t. simplifyingness.

We now focus on DCTRSs. For these class of 3-CTRSs, the criterion of *quasi-reductivity* has been developed (cf. [4, 9, 10]):

Definition 3.9 A DCTRS R is *quasi-reductive* if there is a partial ordering $\succ$ that is well-founded, closed under contexts and substitutions, and such that for every rule of R, $t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$, and substitution σ, we have that

$$\left(\bigwedge_{1 \le j \le i} \sigma(s_i) \succeq \sigma(t_j)\right) \Rightarrow \sigma(t_0) \succ_{st} \sigma(s_{i+1}) \qquad (i \in [0,k])$$

and that

$$\left(\bigwedge_{1 \le j \le k} \sigma(s_i) \succeq \sigma(t_j)\right) \Rightarrow \sigma(s_0) \succ \sigma(s_{k+1})$$

□

It has been proved in [9] that quasi-reductivity implies effective termination.

Compared to reductivity and decreasingness, the following result holds:

Theorem 3.10 *For DCTRSs, quasi-reductive $\not\Rightarrow$ decreasing, and quasi-reductive $\not\Leftarrow$ reductive.*

This means that quasi-reductivity is not more powerful than reductivity or decreasingness, and vice versa. However, the interest in quasi-reductivity lies in the fact that while the latter criteria only cope with 1-CTRSs, quasi-reductivity allows to deal with DCTRSs that are not 1-CTRSs.

We will now introduce a new concept, which can be seen as the extension of (o-)decreasingness to DCTRSs:

Definition 3.11 A DCTRS R is *d-decreasing* if there is a partial ordering $\succ$ that is well-founded, has the subterm property, $\underset{R}{\to} \subseteq \succ$, and such that for each rule of R, $t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$, and substitution σ, we have

$$\left(\bigwedge_{1 \le j \le i} \sigma(s_j) \succeq \sigma(t_j)\right) \Rightarrow \sigma(t_0) \succ \sigma(s_{i+1}) \qquad (i \in [0,k])$$

□

As expected, we have the following result:

Theorem 3.12 *Every d-decreasing DCTRS is effectively terminating.*

As far as the power of this criterion w.r.t. the others is concerned, we have:

Theorem 3.13 *For DCTRSs, o-decreasing $\Rightarrow$ d-decreasing. For normal CTRSs, o-decreasing $\not\Leftarrow$ d-decreasing.*

That is, d-decreasingness is strictly more powerful than o-decreasingness (and thus than decreasingness), even when restricting to normal CTRSs.

Theorem 3.14 *For DCTRSs, quasi-reductive $\Rightarrow$ d-decreasing, and quasi-reductive $\not\Leftarrow$ d-decreasing.*

So, d-decreasingness is more powerful than quasi-reductivity as well.

4 The Unraveling

In this section we will develop an unraveling for DCTRSs.

We assume that CTRSs are composed of terms from a certain set TERMS, built up from variables $\mathcal{V}$ and function symbols $\mathcal{F}$. Also, when unraveling a CTRS into a TRS we will need some extra symbols: for every conditional rule ρ we take new fresh symbol $\mathcal{U}_\rho^i$ ($i \in \mathbb{N}$): thus, we consider a set TERMS$^+$ of 'extended terms' to be the terms obtained from the variables $\mathcal{V}$ and the terms $\mathcal{F}$ plus these new symbols $\mathcal{U}_\rho^i$.

Finally, we employ an operator VAR that, once applied to a term, gives the sequence of its variables in some fixed order (e.g. left-to-right writing order): for example, $\mathrm{VAR}(f(X, g(Y, Z), Y))$ gives the sequence X, Y, Z, Y.

Definition 4.1 (Unraveling $\mathbb{U}_D$)
Take a rule $\rho:\ t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ ($k \geq 1$). Its (tidy) unraveling $\mathbb{U}_D(\rho)$ is equal to

$$\begin{array}{ll}
t_0 \to \mathcal{U}_\rho^1(s_1, \mathrm{VAR}(t_0)) & \\
\mathcal{U}_\rho^j(t_j, \mathrm{VAR}(t_0, \ldots, t_{j-1})) \to \mathcal{U}_\rho^{j+1}(s_{j+1}, \mathrm{VAR}(t_0, \ldots, t_j)) & (1 \leq j \leq k) \\
\mathcal{U}_\rho^k(t_k, \mathrm{VAR}(t_0, \ldots, t_k - 1)) \to s_{k+1} &
\end{array}$$

□

This mapping can be explained as follows. The first rule starts the verification of the conditions, beginning from the first one ($s_1 \to^* t_1$). If this condition is verified, the subsequent rule $\mathcal{U}_\rho^0(t_0)) \to \mathcal{U}_\rho^1(s_1, \mathrm{VAR}(t_0, t_1))$ can be applied, and the second condition ($s_2 \to^* t_2$) is tested. This process goes on until, if all the conditions are satisfied, the last rule $\mathcal{U}_\rho^k(t_k, \mathrm{VAR}(t_0, \ldots, t_k -$

$1)) \to s_{k+1}$ is applied, thus completing the simulation of the original conditional rewrite rule. We also need some form of parameter passing in order to transfer the computed results from one rule to another, and this is accomplished by the occurrences of VAR that store the (content of) the variables of the t_i's.

Example 4.2 Consider the following classic DCTRS computing Fibonacci numbers:

$$\begin{array}{l}
\rho 1: \ 0 + X \to X \\
\rho 2: \ s(X) + Y \to X + s(Y) \\
\rho 3: \ \mathit{fib}(0) \to \mathit{pair}(s(0), 0) \\
\rho 4: \ \mathit{fib}(s(X)) \to \mathit{pair}(W, Y) \Leftarrow \mathit{fib}(X) \to^* \mathit{pair}(Y, Z), Y + Z \to^* W
\end{array}$$

Its unraveled TRS using $\mathbb{U}_D$ consists of the unconditional rules $\rho 1$, $\rho 2$ and $\rho 3$, plus the rules

$$\begin{array}{l}
\mathit{fib}(s(X)) \to \mathcal{U}^1_{\rho 4}(fib(X), X) \\
\mathcal{U}^1_{\rho 4}(\mathit{pair}(Y, Z), X) \to \mathcal{U}^2_{\rho 4}(Y + Z, X, Y, Z) \\
\mathcal{U}^2_{\rho 4}(W, X, Y, Z) \to \mathit{pair}(W, Y)
\end{array}$$

$\square$

The mapping $\mathbb{U}_D$ safely approximates the structure of a DCTRS, since:

Theorem 4.3 *$\mathbb{U}_D$ is a (tidy) unraveling for DCTRSs.*

The following soundness results allow to use $\mathbb{U}_D$ for the study of the corresponding properties:

Theorem 4.4 *For DCTRSs, $\mathbb{U}_D$ is sound for termination and innermost termination.*

Theorem 4.5 *For DCTRSs, $\mathbb{U}_D$ is sound for* $\mathrm{CON}^{\to}$ *and* CON.

In the next two subsections, we will first focus on the effective termination analysis, and then on the extension of syntactical properties of TRSs to DCTRS.

4.1 Effective Termination

In order to face the problem of effective termination, the obvious approach is to use ultra-termination. Comparing ultra-termination and d-decreasingness, we have that

Lemma 4.6 *For DCTRSs, Ultra-termination $\Rightarrow$ d-decreasingness*

Hence, ultra-termination guarantees effective termination:

Corollary 4.7 *Ultra-terminating finite DCTRSs are effectively terminating.*

As far as the power of ultra-termination is concerned, we have the following result:

Theorem 4.8 *For DCTRSs,*

$$\textit{quasi-reductivity} \not\Rightarrow \textit{ultra-termination} \not\Rightarrow \textit{o-decreasingness}$$

and

$$\textit{quasi-reductivity} \not\Leftarrow \textit{ultra-termination} \not\Leftarrow \textit{o-decreasingness.}$$

That is to say, quasi-reductivity and o-decreasingness are not more powerful than ultra-termination, and vice versa.

On the other hand, note that ultra-termination is of great practical importance since it can be checked by utilizing all the existing (and future) techniques to prove termination of TRSs.

For instance, let us consider one of the most successful techniques developed for TRSs: simplification orderings. We have already provided in Section 3 the criterion of o-simplifyingness, which is the natural extension of Kaplan's simplifyingness criterion developed for 1-CTRSs. What about, instead, verifying ultra-termination using simplification orderings on the unraveled TRS, i.e. using *ultra-simplifyingness*? On the practical side, ultra-simplifyingness is particularly appealing, since one does not have to develop from the scratch a checker (like in the Kaplan's simplifyingness and o-simplifyingness cases), but can directly employ all the numerous existing implementations for TRSs. On the theoretical side, the power of the approach is clarified by the following result:

Lemma 4.9 *For DCTRSs, ultra-simplifyingness is strictly more powerful than o-simplifyingness.*

4.2 Syntactical Properties

We now turn to the problem of extending the major syntactical properties from TRSs to DCTRSs.

Lemma 4.10 *For DCTRSs, $\mathbb{U}_D$ is complete for the following properties: being non-collapsing, non-erasing, an overlay system.*

It turns out that all the ultra-properties of the above three properties are just the standard properties "for CTRSs" (cf. Subsection 2.2), thus giving an indication that the standard extensions are the correct ones.

Two other major syntactical properties remain: non-duplication and left-linearity. This time, $\mathbb{U}_D$ is not complete (more precisely, the above two properties are not preserved), thus indicating that the standard extensions of these properties are not the right ones. We will so investigate the corresponding ultra-properties. The case of left-linearity is so important that we will face it in the next section. As far as non-duplication is concerned, $\mathbb{U}_D$ is complete for ultra-nonduplication, as it is easy to see (note that in general this is not automatically true, since only the preservation of *ultra-$\mathcal{P}$* is guaranteed, but not the soundness). Therefore, we can safely employ ultra-nonduplication. The definition of this ultra-property can be also expressed in a more expressive, syntactic way, that is independent from the unraveling $\mathbb{U}_D$:

Lemma 4.11 *A DCTRS is ultra-non-duplicating if and only if for each of its rules $t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ the terms $s_1, \ldots, s_k$ are ground, and for every variable X, $|s_{k+1}|_X \leq |t_0, \ldots, t_k|_X$.*

We will see the applications of ultra-nonduplication in Section 6.

5 Semilinear DCTRSs

For TRSs, left-linearity is such an important property that it is of utmost relevance to try to see what is its correct extension for (3-)CTRSs. Like in the non-duplication case, we are lucky since $\mathbb{U}_D$ is complete for ultra-left-linearity, and so we can safely employ this notion. Analogously, we can reformulate it in a completely syntactic way without mentioning the unraveling $\mathbb{U}_D$. Since also the adjective "ultra-left-linear" is a bit verbose (and refers implicitly to $\mathbb{U}_D$), we use an alternative acronym: semilinear.

Definition 5.1 A DCTRS is said to be *semilinear* (briefly, an *SCTRS*), if for each its rewrite rule $t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ the sequence $t_0, \ldots, t_k$ is linear. $\Box$

It is easy to verify that semilinearity coincides with ultra-left-linearity.

Semilinearity has also a significance when recalling the importance of DCTRSs for functional programming. Indeed, most of functional languages (eg ML, CLEAN, Haskell etc.) require the defining patterns to be left-linear: this, recalling the parallelism between DCTRSs and functional programs seen in Subsection 2.1, is tantamount to requiring semiliinearity of the DCTRS.

The key result that distinguishes semilinear DCTRSs is that their structure is completely preserved by the unraveling $\mathbb{U}_D$:

Theorem 5.2 *For every SCTRS R,* $\forall s, t \in \textsc{Terms}.\ s \underset{R}{\to}^* t \Leftrightarrow s \underset{\mathbb{U}_D(R)}{\longrightarrow}^* t$

That is to say, when restricted to $\textsc{Terms}$, an SCTRS R and its unraveled TRS $\mathbb{U}_D(R)$ have the same rewrite relation $\underset{R}{\to}^*$.

This, in turn, implies that we can obtain a more precise analysis of an SCTRS using ultra-properties, as we will see in the next subsections.

5.1 Effective Termination for SCTRSs

We face again the problem of effective termination, this time for SCTRSs. We have seen that ultra-termination for DCTRSs provides a powerful criterion for effective termination. Yet, it is not as powerful as d-decreasingness. Quite surprisingly, in the SCTRSs case we manage to reach d-decreasingness, since the following result holds:

Theorem 5.3 *For SCTRSs, ultra-termination = d-decreasingness.*

Hence, the notion of d-decreasingness is provided with a much more meaningful justification, being just the termination of the unraveled CTRS.

Note that all the difficulties and possible objections in expressing what the 'right notion' of effective termination is, are nicely coped with using ultra-termination: the leading intuition was that things for effectively terminating DCTRSs should be like for terminating TRSs; and ultratermination is just the concept that the TRS corresponding to a DCTRS terminates.

5.2 Confluence for SCTRSs

Theorem 5.4 *For SCTRSs, $\mathbb{U}_D$ is sound for confluence.*

An immediate corollary of the above result and of Theorem 4.4 is that also completeness can be coped with:

Corollary 5.5 *For SCTRSs, $\mathbb{U}_D$ is sound for completeness.*

Theorem 5.4 enables us to lift every result on the confluence of left-linear TRSs, giving new insights on the reasons and failures of existing criteria for confluence.

So far, there are two recent powerful results on the confluence of DCTRSs. The first is that of [3], where Avenhaus and Loría-Sáenz proved that a DCTRS is confluent if it is "strongly deterministic", quasi-reductive and with convergent conditional critical pairs. The other one is that of [34], where Suzuki, Middeldorp and Ida proved that every orthogonal "properly oriented" and "right stable" 3-CTRS is (level-)confluent.

These two results resemble the two well-known major criteria for confluence of TRSs, namely respectively the fact that a terminating TRSs with convergent critical pair is convergent, and the result that orthogonal TRSs are confluent. However, the authors had to face a number of new problems and phenomena occurring in the 3-CTRSs realm, as is well-illustrated in [3, 34]. For instance, consider these two SCTRS after [3]:

$$\mathcal{R}_1 = \begin{cases} 0 + X \to X \\ s(X) + Y \to X + s(Y) \\ f(X,Y) \to Z \Leftarrow X + Y \to^* Z + Z' \end{cases}$$

$$\mathcal{R}_2 = \begin{cases} a \to c \\ g(a) \to h(b) \\ h(b) \to g(c) \\ f(X) \to Z \Leftarrow g(X) \to^* h(Z) \end{cases}$$

These two SCTRSs are both orthogonal and terminating (even quasi-reductive). Nevertheless, they are not confluent. $\mathcal{R}_1$ is not confluent, since $f(s(0),0)$ reduces both to $s(0)$ and to 0. Here the problem is with the overlapping of a rule with itself (at root level): this new phenomenon is called *improper critical pair* in [3]. Also, $\mathcal{R}_2$ is not confluent, since $f(a)$ reduces both to $f(c)$ and to b. The problem here is called *variable overlappingness* in [3].

All of these apparent different and weird new phenomena, clashing with the intuitions developed for TRSs, can be better comprehended and unified using ultra-confluence. Indeed, when unraveling the two above SCTRSs we have that: in $\mathcal{R}_1$, $Z+Z'$ overlaps with the lhs's of the other non conditional rules; in $\mathcal{R}_2$, $h(Z)$ overlaps with the lhs of the second rule, $h(b)$.

Thus, orthogonality is only apparent: indeed, what we are lead to consider is just orthogonality "for CTRSs" (cf. Subsection 2.2), which is not the correct notion, since $\mathbb{U}_D$ is not complete for this property. The correct notion is obtained by using ultra-orthogonality, which gives the following major result:

Theorem 5.6 *Ultra-orthogonal SCTRSs are confluent.*

Proof By lifting the well-known fact that orthogonal TRSs are confluent (being $\mathbb{U}_D$ sound for confluence and complete for ultra-orthogonality). $\square$

Ultra-orthogonality can be syntactically expressed without mentioning $\mathbb{U}_D$:

Lemma 5.7 *An SCTRS R is ultra-orthogonal if and only if for each its rule $\rho:\ t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ there is not a left-hand side of a rule in R that overlaps a t_i ($i \in [0,k]$), but for the trivial overlay of the left-hand side of ρ.*

Note that ultra-orthogonality (like orthogonality) is a decidable criterion. The problem with the criterion given in [3], even assuming quasi-decreasingness of the DCTRS, being "strongly deterministic" is undecidable. Thus, in the same paper the authors also give a powerful decidable criterion for being strongly deterministic, namely being *absolutely deterministic*. Interestingly, it is easy to see that for orthogonal SCTRSs, *being absolutely deterministic is the same as being ultra-orthogonal*: thus, we are able to automatically reconstruct the right notions ensuring confluence of a DCTRS (also, the more general, and undecidable, notion of being "strongly deterministic" is, rather reasonably, obtained by loosening the non-overlapping condition of the t_i's requiring that all their instances are irreducible). However, note the fundamental difference, that in our case we *do not require* not only quasi-reductivity, but termination at all (on the other hand, we cope only with SCTRSs). For instance, the Fibonacci SCTRS of Example 4.2 can be immediately proved confluent, since it is readily ultra-orthogonal, without having to prove its quasi-reductivity.

To this extent, a similar result is that obtained in the other aforementioned paper, [34], which can be seen as an attempt to lift the orthogonality criterion of TRSs to 3-CTRSs. Again, the obtained conditions, when restricted to SCTRSs, are also very similar (while "properly orientedness" is a condition alike to determinism, their other condition, "right stability", is extremely similar to ultra-orthogonality). As far as the relative strength of the approaches is concerned, their criterion is not more powerful than ultra-orthogonality, and vice versa (although it should be noticed that their criterion is able to prove not only confluence, but level-confluence).

Finally, another criterion that we can automatically obtain is the lifting of the other major criterion for TRSs previously mentioned:

Theorem 5.8 *A d-decreasing SCTRS R is confluent if every critical pair of $\mathbb{U}_D(R)$ is convergent.*

Proof By the fact that for terminating TRSs convergence of critical pairs implies confluence (being d-decreasingness equivalent to ultra-termination, and $\mathbb{U}_D$ sound for termination and confluence). □

Again, note the similarity with the criterion of [3]. Regarding the power, again, the two criteria are uncomparable each other.

In a nutshell, although in this case we showed that we can automatically obtain results not subsumed by the other works, we do not claim that unravelings necessarily provide each time such powerful solutions. Rather, we claim that they should be seen as a tool to provide a better understanding of the new phenomena occurring with 3-CTRSs, resorting on the familiar experience gained for TRSs: unravelings can provide right away new criteria for free, lifting old ones of TRSs, and thus providing the correct intuitions on how to get better criteria.

5.3 Consistency w.r.t. reduction for SCTRSs

Finally, $\mathbb{U}_D$ allows to fully grasp the property of consistency w.r.t. reduction:

Theorem 5.9 *For SCTRSs, $\mathbb{U}_D$ is complete for* $\mathrm{CON}^{\rightarrow}$.

This means there is no loss of precision when studying $\mathrm{CON}^{\rightarrow}$ of a SCTRS R using $\mathbb{U}_D$: we can simply use the corresponding TRS $\mathbb{U}_D(R)$.

6 Modularity

As said in the introduction, so far, to the best of our knowledge, *there are no results on the modularity of 3-CTRSs*. Using unravelings, we can get for free a bunch of new powerful results, simply lifting to DCTRSs the existing results on the modularity of TRSs.

In order to do this, the unraveling must be compositional w.r.t. the composition operator (cf. Subsection 2.2). The main modularity operators so far introduced (cf. e.g. [27, 17, 21]), are: *disjoint union* $\oplus$ (disjoint signatures); *constructor-sharing union* $\underset{cs}{\oplus}$ (sharing only of constructor symbols); *composable union* $\underset{comp}{\oplus}$ (alike $\underset{cs}{\oplus}$ but, in addition, rules defining shared defined symbols are shared); and *hierarchical union* $\underset{hier}{\oplus}$: two OCTRSs R_1 and R_2 can be composed via $\underset{hier}{\oplus}$ if R_1 does not have defined symbols of R_2, and for every rule $t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ of R_2 no defined symbol of R_1 appears in $t_0, \ldots, t_k$.

The next result shows that all of these operators can be coped with:

Theorem 6.1 *$\mathbb{U}_D$ is compositional w.r.t. the operators $\oplus$, $\underset{cs}{\oplus}$, $\underset{comp}{\oplus}$ and $\underset{hier}{\oplus}$.*

Hence, by this result we can use Theorem 2.3 to lift every modularity result for TRSs to DCTRSs. In order to save space and enhance readability, in the subsequent proofs we will usually consider usage of the aforementioned method to be understood, and just name the original modularity property of TRSs that is lifted.

6.1 Modularity of DCTRSs

Termination

Theorem 6.2 *Ultra-termination is modular for non-collapsing composable DCTRSs.*

Proof By the modularity of termination for non-collapsing composable TRSs ([27]). □

Theorem 6.3 *Ultra-termination is modular for ultra-non-overlapping composable DCTRSs.*

Proof By the modularity of termination for non-overlapping composable TRSs ([5, 27]). □

Theorem 6.4 *Ultra-termination is modular for non-ultraduplicating composable DCTRSs.*

Proof By the modularity of termination for non-duplicating composable TRSs ([12, 27]). $\Box$

The modularity of $\mathcal{C}_{\mathcal{E}}$-termination (cf. [26, 12]) is one of the most powerful results ever obtained for TRSs. Recall that a TRS T is said to be $\mathcal{C}_{\mathcal{E}}$-terminating if $T \oplus \{or(X,Y) \to X, or(X,Y) \to Y\}$ is terminating (cf. [26]). The lifted ultraproperty can be expressed in a similar way: it can be proved that T *is ultra-*$\mathcal{C}_{\mathcal{E}}$*-terminating iff* $T \oplus \{or(X,Y) \to X, or(X,Y) \to Y\}$ *is ultra-terminating*. Thus we get:

Theorem 6.5 *Ultra-*$\mathcal{C}_{\mathcal{E}}$*-termination is modular for DCTRSs.*

Proof By the modularity of $\mathcal{C}_{\mathcal{E}}$-termination for TRSs ([12, 26]). $\Box$

This result is particularly relevant in view of the practical importance of ultra-simplifyingness (cf. Subsection 4.1 and Lemma 4.9):

Theorem 6.6 *Ultra-simplifyingness is modular for composable DCTRSs.*

Proof By the modularity of simplifyingness for composable TRSs ([18]). $\Box$

Theorem 6.7 *Ultra simple termination is modular for DCTRSs.*

Proof By the modularity of simple termination for composable TRSs ([18]). $\Box$

As far as hierarchical union is concerned, Krishna Rao in [29] proved that simplifyingness is modular for hierarchical TRSs forming a 'proper extension'. From his result we obtain:

Theorem 6.8 *Ultra-simplifyingness is modular for hierarchical DCTRSs forming a 'ultra proper extension'.*

Unravelings allow to lift not only "classic" modularity results, but also "hybrid" results like the ones obtained by Middeldorp and Ohlebusch: Middeldorp in [24] proved that if one of two terminating TRSs is both non-collapsing and non-duplicating, then their disjoint sum is terminating as well. Later, Ohlebusch ([27]) managed to extend this result to composable union of TRSs.

Theorem 6.9 *If one of two ultra-terminating DCTRSs is both non-collapsing and non-ultraduplicating, then their composable union is ultra-terminating.*

Proof Take two ultra-terminating DCTRSs R and R', with R non-collapsing and non-ultraduplicating. We have that $\mathbb{U}_D(R)$ is terminating, non-collapsing (by Lemma 4.10) and non-duplicating. Hence, by the aforementioned result of [27], the composable union of $\mathbb{U}_D(R')$ and $\mathbb{U}_D(R)$ is terminating, which implies by Theorem 6.1 that the composable union of R and R' is ultra-terminating. $\square$

Other Properties

Last but not least, we consider innermost termination and the consistency property:

Theorem 6.10 *Ultra innermost termination is modular for composable DCTRSs.*

Proof By the modularity of innermost termination for composable TRSs ([11, 27]). $\square$

Theorem 6.11 *Ultra-*CON *is modular for DCTRSs.*

Proof By the modularity of CON for TRSs ([32]). $\square$

6.2 Modularity of SCTRSs

In the case of SCTRSs, several new powerful modularity results can be obtained. Here, in some proofs we will also implicitly use the fact that, for SCTRSs, $\mathbb{U}_D$ is (trivially) complete for left-linearity.

Termination

Theorem 6.12 *d-decreasingness is modular for non-collapsing composable SCTRSs.*

Proof By the modularity of termination for non-collapsing composable TRSs ([27]). $\square$

Theorem 6.13 *d-decreasingness is modular for non-overlapping composable SCTRSs.*

Proof By the modularity of termination for non-overlapping composable TRSs ([5, 27]). $\square$

Theorem 6.14 *d-decreasingness is modular for non-ultraduplicating composable SCTRSs.*

Proof By the modularity of termination for non-duplicating composable TRSs ([12, 27]). □

Corollary 6.15 *d-decreasingness is modular for ultra-$\mathcal{C}_\mathcal{E}$-terminating SC-TRSs.*

Proof By Theorem 6.5, since ultra-$\mathcal{C}_\mathcal{E}$-termination implies d-decreasingness. □

Regarding ultra-$\mathcal{C}_\mathcal{E}$-termination, note that it can be nicely expressed without using $\mathbb{U}_D$, since it can be proved that *T is ultra-$\mathcal{C}_\mathcal{E}$-terminating (for SCTRSs) iff $T \oplus \{or(X,Y) \to X, or(X,Y) \to Y\}$ is d-decreasing.*

Theorem 6.16 *d-decreasingness is modular for* $\mathrm{CON}^{\rightarrow}$ *SCTRSs.*

Proof By the modularity of termination for left-linear $\mathrm{CON}^{\rightarrow}$ TRSs ([20, 33]). □

Theorem 6.17 *If one of two d-decreasing SCTRSs is both* $\mathrm{CON}^{\rightarrow}$ *and ultra-$\mathcal{C}_\mathcal{E}$-terminating, then their disjoint union is d-decreasing.*

Proof Take two d-decreasing left-linear normal CTRSs R and R', with R $\mathrm{CON}^{\rightarrow}$ and ultra-$\mathcal{C}_\mathcal{E}$-terminating. $\mathbb{U}_D(R)$ is both $\mathrm{CON}^{\rightarrow}$ (by Theorem 5.9) and $\mathcal{C}_\mathcal{E}$-terminating. Moreover, both $\mathbb{U}_D(R)$ and $\mathbb{U}_D(R')$ are readily left-linear. By the result proved in [23] for TRSs, the disjoint union of $\mathbb{U}_D(R)$ and $\mathbb{U}_D(R')$ is terminating, which implies by Theorems 6.1 and 5.3 that the disjoint union of R and R' is d-decreasing. □

Confluence

Theorem 6.18 *Ultra-confluence is modular for non-collapsing composable SCTRSs.*

Proof By the modularity of confluence for non-collapsing composable TRSs ([27]). □

Theorem 6.19 *Ultra-confluence is modular for constructor-sharing SC-TRSs.*

Proof By the modularity of confluence for left-linear constructor-sharing TRSs ([31]). □

Theorem 6.20 *Ultra-confluence is modular for hierarchical SCTRSs.*

Proof By the modularity of confluence for hierarchical left-linear TRSs (cf. [31]). $\square$

Finally, we mention that we can also lift the recent result of Verma ([35]) on the modularity of confluence for *lr-combinations* (a rather flexible and powerful kind of combination): without entering into details, we just mention that being $\mathbb{U}_D$ composable w.r.t. *lr*-combinations, we can lift this result obtaining a sufficient criterion for the confluence of *lr*-combinations of SCTRSs.

Other Properties

As far as consistency w.r.t. reduction is concerned, we can obtain the following important result:

Theorem 6.21 $\mathrm{CON}^{\rightarrow}$ *is modular for SCTRSs.*

Proof By the modularity of $\mathrm{CON}^{\rightarrow}$ for left-linear TRSs (see [20, 33, 23]). $\square$

Finally, we hint at the fact that, by Corollary 5.5, we can lift by unravelings also all the results obtained for the completeness of hierarchical combinations of TRSs, namely those of Dershowitz ([5]) and Rao ([28]).

7 Further Applications

In this final section we hint at other further powerful results that can be derived from the presented approach.

7.1 Power of the Analysis

We have seen that the better the approximant TRS (i.e., the better the unraveling), the more precise the analysis becomes. It is therefore natural to ask what is the intrinsic limit of the unraveling approach that we have successfully used in this paper. The best situation is when we manage to reach *completeness* of the unraveling for the given property: this means we are not losing information when passing from DCTRSs to TRSs. We have performed an abstract study of the unraveling approach for DCTRSs (along the same lines as done in [22] for TRSs and CTRSs), and shown that not only this is not possible for all the major properties of DCTRSs (termination, confluence, completeness etc.), but it is even impossible to faithfully translate a 3-CTRS into a 2-CTRS, and a 2-CTRS into a 1-CTRS (the gap from 1-CTRSs to TRss is already known from [21]). In other words, there are expressive gaps inbetween each couple of classes, so we have no hope to be able to fully analyze one of these classes when resorting on a more limited one: the analysis will always be approximate.

7.2 Logic Programming

DCTRSs have tight connections with logic programs, as shown in a nice paper ([10]) by Ganzinger and Waldmann. Without entering into too much details, the idea is that a relevant class of logic programs (so-called well moded logic programs) can be elegantly translated in a natural way into DCTRSs, in such a way that proving quasi-reductivity of the obtained CTRS implies termination of the original logic program. This approach is extremely neat, the only disadvantage being that one has to prove quasi-reductivity of a DCTRSs, which may not be easy. On the other hand, another approach is to develop such a transformation of logic programs into TRSs, like done in [30] and further investigated in [1, 19, 2]. This methodology, conversely, is rather powerful, since proving termination of a TRS is generally much simpler than proving termination of CTRS, but not very elegant, since the transformations have the form of a low-level compilation. Using the results presented in this paper, we can combine the advantages of both approaches: first, the elegant transformation of [10] is employed. Then, the obtained CTRSs is transformed into a TRSs using $\mathbb{U}_D$, and analyzed for termination (i.e., we employ ultra-termination). Thus, we gather together the simplicity of the approach of [10] with the effectiveness of the alternative approach of [30, 1, 19, 2]. Note that, in order for this approach to work, we had to extend the original result in [10], by proving that d-decreasingness of the corresponding CTRS implies termination of the original logic program.

7.3 Other Properties

In this paper we have analyzed DCTRSs for the properties of termination, innermost termination, confluence, and the consistency properties. Similarly, it is possible to use unravelings to study all the other main properties of DCTRSs, namely completeness, the normal form properties (uniqueness of normal forms w.r.t. reduction, uniqueness of normal forms, normal form property) and the normalization properties (weak normalization, innermost weak normalization, semicompleteness). In particular, $\mathbb{U}_D$ suffices to prove the normal forms properties, while for the normalization ones a slight variation is needed.

7.4 π-CTRSs

Definition 7.1 An OCTRS R is *of type* π (briefly, a π*-OCTRS*) if for each rule $t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ of R the conditions are disjoint, that is to say $\forall i, j \in [1, k], i \neq j.\ Var(s_i, t_i) \cap Var(s_j, t_j) = \emptyset$ □

The class of π-OCTRSs is of distinguished practical importance because of the following fact: when applying a rule, the conditions can be checked *in full parallel*, being completely independent one another. We consider now the class of π-SCTRSs. Being SCTRSs, π-CTRSs inherit all the good properties seen for this class. Also, it is not difficult to see that every o-simplifying SCTRS is a π-SCTRS. Moreover, we can develop some nice further results on their modularity, in view of the following fact. In [21] the unraveling $\mathbb{U}_n$ has been introduced for the study of normal CTRSs. Its definition is simply that each rule $\rho:\ t_0 \to s_{k+1} \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ is translated into the two rules $t_0 \to \mathcal{U}_\rho(s_1, \ldots, s_k, \mathrm{VAR}(t_0))$ and $\mathcal{U}_\rho(t_1, \ldots, t_k, \mathrm{VAR}(t_0)) \to s_{k+1}$. Now, we have proved that all the results developed in [21] for normal CTRSs do extend to π-SCTRSs, thus when analyzing π-SCTRSs the simpler TRSs produced by $\mathbb{U}_n$ can be used, in place of those produced by $\mathbb{U}_D$ (the drawback is a loss of power, since it can be proved that $\mathbb{U}_D$ is strictly more powerful than $\mathbb{U}_n$ for every major property). Moreover, an important new fact can be proved:

Theorem 7.2 *For π-SCTRSs,*

ultra-termination (w.r.t. $\mathbb{U}_n$) = o-decreasingness.

Besides being important on its own, this theorem enables us to state a new bunch of results on the modularity of o-decreasingness: briefly, *all of the results* that we have proved for (the modularity of) d-decreasingness for SCTRSs (Subsection 6.2) hold true for o-decreasingness, when considering π-SCTRSs.

References

[1] G. Aguzzi and U. Modigliani. Proving termination of logic programs by transforming them into equivalent term rewriting systems. In *Founda-*

tions of Software Technology and Theoretical Computer Science, volume 761 of *LNCS*, pages 114–124. Springer–Verlag, 1993.

[2] T. Arts and H. Zantema. Termination of logic programs using semantic unification. In *Proceedings of the Fifth Workshop on Logic Program Synthesis and Transformation*, LNCS. Springer–Verlag, 1996. To appear.

[3] J. Avenhaus and C. Loría-Sáenz. On conditional rewrite systems with extra variables and deterministic logic programs. In *Proc. Int. Conference on Logic Programming and Automated Reasoning*, volume 822 of *LNAI*, pages 215–229. Springer–Verlag, 1994.

[4] H. Bertling and H. Ganzinger. Completion-time optimization of rewrite-time goal solving. In *Proceedings of the Third International Conference on Rewriting Techniques and Applications*, volume 355 of *LNCS*, pages 45–58. Springer–Verlag, 1989.

[5] N. Dershowitz. Hierarchical termination. In N. Dershowitz and N. Lindenstrauss, editors, *Proceedings 4th International Workshop on Conditional and Typed Rewriting Systems*, volume 968 of *LNCS*. Springer–Verlag, 1995.

[6] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 6, pages 243–320. Elsevier – MIT Press, 1990.

[7] N. Dershowitz and M. Okada. A rationale for conditional equational programming. *Theoretical Computer Science*, 75:111–138, 1990.

[8] N. Dershowitz, M. Okada, and G. Sivakumar. Canonical conditional rewrite systems. In *Proceedings of the 9th Conference on Automated Deduction*, volume 310 of *LNCS*, pages 538–549. Springer–Verlag, 1988.

[9] H. Ganzinger. Order-sorted completion: the many-sorted way. *TCS*, 89:3–32, 1991.

[10] H. Ganzinger and U. Waldmann. Termination proofs of well-moded logic programs via conditional rewrite systems. In M. Rusinowitch and J.L. Rémy, editors, *CTRS'92*, volume 656 of *LNCS*, pages 216–222. Springer–Verlag, 1993.

[11] B. Gramlich. Relating innermost, weak, uniform and modular termination of term rewriting systems. In *International Conference on Logic Programming and Automated Reasoning*, volume 624 of *LNAI*, pages 285–296. Springer-Verlag, 1992.

[12] B. Gramlich. Generalized sufficient conditions for modular termination of rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 5:131–158, 1994.

[13] M. Hanus. The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming*, 19&20:583–628, 1994.

[14] J.-P. Jouannaud and B. Waldmann. Reductive conditional term rewrite systems. In *3rd IFIP Working Conference on Formal Description of Programming Concepts*, pages 223–244, Ebberup, Denmark, 1986.

[15] S. Kaplan. Conditional rewrite rules. *Theoretical Computer Science*, 33(2):175–193, 1984.

[16] S. Kaplan. Simplifying conditional term rewriting systems. *Journal of Symbolic Computation*, 4(3):295–334, 1987.

[17] J.W. Klop. Term rewriting systems. In S. Abramsky, Dov M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 1, pages 1–116. Clarendon Press, Oxford, 1992.

[18] M. Kurihara and A. Ohuchi. Modularity of simple termination of term rewriting systems. *Journal of IPS Japan*, 31(5):633–642, 1990.

[19] M. Marchiori. Logic programs as term rewriting systems. In G. Levi and M. Rodríguez-Artalejo, editors, *Proceedings of the Fourth International Conference on Algebraic and Logic Programming*, volume 850 of *LNCS*, pages 223–241. Springer–Verlag, 1994.

[20] M. Marchiori. On the modularity of normal forms in rewriting. *Journal of Symbolic Computation*, 22(2):143–154, 1996.

[21] M. Marchiori. Unravelings and ultra-properties. In *Proceedings of the Fifth International Conference on Algebraic and Logic Programming (ALP'96)*, volume 1139 of *LNCS*, pages 107–121. Springer–Verlag, 1996.

[22] M. Marchiori. On the expressive power of rewriting. In *Proceedings of the Seventeenth International Conference on the Foundations of Software Technology and Theoretical Computer Science (FST&TCS')*, LNCS. Springer–Verlag, 1997. In press.

[23] M. Marchiori. Bubbles in modularity. *Theoretical Computer Science*, 192(1):31–54, 1998.

[24] A. Middeldorp. A sufficient condition for the termination of the direct sum of term rewriting systems. In *Proceedings of the Fourth IEEE Symposium on Logic in Computer Science*, pages 396–401, 1989.

[25] A. Middeldorp and E. Hamoen. Completeness results for basic narrowing. *Applicable Algebra in Engineering, Communication and Computing*, 5:213–253, 1994.

[26] E. Ohlebusch. On the modularity of termination of term rewriting systems. *Theoretical Computer Science*, 136(2):333–360, 1994.

[27] E. Ohlebusch. Modular properties of composable term rewriting systems. *Journal of Symbolic Computation*, 20(1):1–41, 1995.

[28] K. Rao. Completeness of hierarchical combinations of term rewriting systems. In *Proc. 13th FST&TCS*, volume 761 of *LNCS*, pages 125–139. Springer–Verlag, 1993.

[29] K. Rao. Simple termination of hierarchical combinations of term rewriting systems. In M. Hagiya and J.C. Mitchell, editors, *Proc. TACS*, volume 789 of *LNCS*, pages 203–223. Springer–Verlag, 1994.

[30] K. Rao, D. Kapur, and R.K. Shyamasundar. A transformational methodology for proving termination of logic programs. In *Proceedings of the Fifth Conference on Computer Science Logic*, volume 626 of *LNCS*, pages 213–226, Berlin, 1992. Springer–Verlag.

[31] K.-C. Raoult and J. Vuillemin. Operational and semantic equivalence between recursive programs. *Journal of the ACM*, 27(4):772–796, 1980.

[32] M. Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *Journal of Symbolic Computation*, 8(1,2):51–99, 1989.

[33] M. Schmidt-Schauß, M. Marchiori, and S.E. Panitz. Modular termination of r-consistent and left-linear term rewriting systems. *Theoretical Computer Science*, 149(2):361–374, 1995.

[34] T. Suzuki, A. Middeldorp, and T. Ida. Level-confluence of conditional rewrite systems with extra-variables in right-hand side. In *Proceedings of the Sixth International Conference on Rewriting Techniques and Applications*, volume 914 of *LNCS*, pages 179–193. Springer–Verlag, 1995.

[35] R.M. Verma. Unique normal forms and confluence of rewrite systems: Persistence. In *Proc. 14th IJCAI*, volume 1, pages 362–368, 1995.