Computationally Equivalent Elimination of Conditions

Traian Florin Şerbănuţă and Grigore Roşu

Department of Computer Science,
University of Illinois at Urbana-Champaign.
{tserban2,grosu}@cs.uiuc.edu

Abstract

An automatic and easy to implement transformation of conditional term rewrite systems into computationally equivalent unconditional term rewrite systems is presented. No special support is needed from the underlying unconditional rewrite engine. Since unconditional rewriting is more amenable to parallelization, our transformation is expected to lead to efficient concurrent implementations of rewriting.

1 Introduction

Conditional rewriting is a crucial paradigm in algebraic specification, since it provides a natural means for executing equational specifications. Many specification languages, including CafeOBJ [DF98], ELAN [BCD+00], Maude [CDE+03], OBJ [GWM+00], provide conditional rewrite engines to execute and reason about specifications. It also plays a foundational role in functional logic programming [Han94]. Conditional rewriting is, however, rather inconvenient to implement directly. To reduce a term, a rewrite engine needs to maintain a *control context* for each conditional rule that is tried. Due to the potential nesting of rule applications, such a control context may grow arbitrarily. The technique presented in this paper automatically translates conditional rewrite rules into unconditional rules, by *encoding the necessary control context into data context*. The obtained rules can be then executed on any unconditional rewrite engine, whose single task is to *match-and-apply* unconditional rules. Such a simplified engine can be seen as a *rewrite virtual machine*, which can be even implemented in hardware for increased efficiency, and our transformation technique can be seen as a compiler. One can also simulate the proposed transformation as part of the implementation of a conditional engine.

Experiments performed on two fast rewrite engines, Elan[BCD+00] and Maude[CDE+03], show that performance increases can be obtained on current engines if one uses the proposed transformation as a front-end. However, since

these rewrite engines are optimized for conditional rewriting, we expect significant further increases in performance if one just focuses on the much simpler problem of developing optimized *unconditional* rewrite engines and use our technique. Moreover, one can focus on developing *parallel rewrite machines* without worrying about conditions, which obstruct the potential for high parallelism.

On computational equivalence. Let us formalize the informal notion of "computationally equivalent elimination of conditions". Consider a conditional term rewriting system (CTRS) $\mathcal{R}$ over signature Σ and an (unconditional) term rewriting system (TRS) R' over signature Σ'. Also, assume some mapping φ from Σ-terms to Σ'-terms and some partial mapping ψ from Σ'-terms to Σ-terms that is an inverse to φ (i.e., $\psi(\varphi(s)) = s$ for any Σ-term s). φ can be thought of as translating input terms for $\mathcal{R}$ into input terms for $\mathcal{R}'$, while ψ as taking results of rewritings in $\mathcal{R}'$ into corresponding results for $\mathcal{R}$. In other words, φ and ψ can wrap a Σ' rewrite engine into a Σ rewrite engine. Σ'-terms $\varphi(s)$ are called *initial*, while terms t' with $\varphi(s) \to^*_{\mathcal{R}'} t'$ are called *reachable* in $\mathcal{R}'$. The (partial) mapping ψ only needs to translate reachable Σ'-terms into Σ-terms. Typically, φ and ψ are straightforward linear translators of syntax.

$\mathcal{R}'$ is *complete* for $\mathcal{R}$ iff any reduction in $\mathcal{R}$ has some corresponding reduction in $\mathcal{R}'$: $s \to^*_{\mathcal{R}} t$ implies $\varphi(s) \to^*_{\mathcal{R}'} \varphi(t)$. Completeness is typically easy to prove but, unfortunately, has a very limited practical use; it only allows to disprove reachability tasks in $\mathcal{R}$ by disproving corresponding tasks in $\mathcal{R}'$. $\mathcal{R}'$ is *sound* for $\mathcal{R}$ iff any reduction in $\mathcal{R}'$ of an initial term corresponds to some reduction in $\mathcal{R}$: $\varphi(s) \to^*_{\mathcal{R}'} t'$ implies $s \to^*_{\mathcal{R}} \psi(t')$. The soundness of $\mathcal{R}'$ allows to compute partial reachability sets in $\mathcal{R}$: applying ψ to all t' reached from $\varphi(s)$ in $\mathcal{R}'$, we get Σ-terms (not necessarily all) reachable from s in $\mathcal{R}$. The soundness and completeness of $\mathcal{R}'$ gives a procedure to *test* reachability in the CTRS $\mathcal{R}$ using any reachability analysis procedure for the TRS $\mathcal{R}'$: $s \to^*_{\mathcal{R}} t$ iff $\varphi(s) \to^*_{\mathcal{R}'} \varphi(t)$.

Soundness and completeness may seem the ideal properties of a transformation. Unfortunately, they do not yield the computational equivalence of the original CTRS to (the wrapping of) the resulting TRS. By *computational equivalence of $\mathcal{R}'$ to $\mathcal{R}$* we mean the following: if $\mathcal{R}$ is intended as a computational engine, that is, if it terminates on a given term s admitting a unique normal form t, then $\mathcal{R}'$ also terminates on $\varphi(s)$ and for any of its normal forms t', we have that $\psi(t') = t$. In other words, the unconditional $\mathcal{R}'$ *can be used transparently* to perform computations for $\mathcal{R}$. Example 3 shows that the soundness and completeness of a transformation do *not* imply computational equivalence, even if the original CTRS is confluent and terminates! Note that termination of $\mathcal{R}$ is *not* required. Indeed, termination of the CTRS may be too restrictive in certain applycations, e.g., in functional logic programming [ABH03].

On Termination. Rewriting of a given term in a CTRS may not terminate for two reasons [Ros04]: the reduction of the condition of a rule does not terminate, or there are some rules that can be applied infinitely often on the given term. In rewrite engines, the effect in both situations is the same: the system

loops forever or crashes running out of memory. For this reason, we do not make any distinction between the two cases, and simply call a Σ-rewriting system *operationally terminating* [LMM05] on Σ-term s iff it always reduces s to a normal form regardless of the order rules apply. Note that this notion is different from *effective termination* [Mar96]; Example 6 shows a system that is confluent and effectively terminating but *not* operationally terminating. Operational termination is based on the assumption that, in general, one cannot expect a rewrite engine to be "smart" enough to pick the right rewrite sequence to satisfy a condition. Formally, a CTRS $\mathcal{R}$ is operationally terminating on s if for any t, any proof tree attempting to prove that $s \to^*_{\mathcal{R}} t$ is finite. Operational termination is equivalent to decreasingness for normal CTRSs (and with quasi-decreasingness for deterministic CTRSs) [LMM05].

We give an automatic transformation technique of CTRSs into TRSs, taking ground confluent normal CTRSs $\mathcal{R}$ into computationally equivalent TRSs $\mathcal{R}'$. This technique can be extended to more general CTRSs including ones with extra variables in conditions (Section 6). Experiments show that the resulting TRSs yield performant computational engines for the original CTRSs. On the theoretical side, we show several results: the completeness of our transformation (Theorem 1); ground confluence (Theorem 2) or left linearity (Theorem 3) of $\mathcal{R}$ implies the soundness of $\mathcal{R}'$; if $\mathcal{R}$ is left-linear, then ground confluence of $\mathcal{R}$ implies ground confluence of $\mathcal{R}'$ on reachable terms (Theorem 4) and operational termination of $\mathcal{R}$ on s implies termination of $\mathcal{R}'$ on $\varphi(s)$ (Theorem 5); if $\mathcal{R}$ is finite, ground confluent and operationally terminating on s then $\mathcal{R}'$ is ground confluent on reachable terms and terminating on $\varphi(s)$ (Theorem 7). The latter implies the computational equivalence of our transformation. Additionally, we show that left linearity and ground confluence of $\mathcal{R}'$ on reachable terms implies ground confluence of $\mathcal{R}$ (Proposition 6), and termination of $\mathcal{R}'$ on reachable terms implies operational termination of $\mathcal{R}$ (Proposition 8); these results potentially enable one to use confluence and/or termination techniques on unconditional TRSs to show confluence and/or operational termination of the original CTRS, but this was not our purpose and consequently have not experimented with this approach.

Section 2 discusses previous transformations of CTRSs into TRSs. We only focus on ones intended to be computationally equivalent and discuss their limitations. Section 3 presents our transformation. Section 4 shows it at work on several examples; some of these examples have been expiremented with on two rewrite engines, Elan [BCD$^+$00] and Maude [CDE$^+$03], with promising performance results. Section 5 is concern with the theoretical results. Section 6 discusses extensions of our transformation technique to more general CTRSs (with extra variables in conditions and matching modulo equations). Finally, Section 7 concludes the paper.

2 Previous transformations

Stimulated by the benefits of transforming CTRSs into equivalent TRSs, there has been much research on this topic. Despite the apparent simplicity of most transformations, they typically work for restricted CTRSs and their correctness, when true, is quite involved. We focus on transformations that generate TRSs intended to be *transparently* used to reduce terms or test reachability in the original CTRSs. Significant efforts have been dedicated to transformations preserving only certain properties, e.g., termination or confluence [Ohl02]; we do not discuss these here. We use the following two examples to illustrate the different transformations and to analyze their limitations.

Example 1 [Mar96, Ohl02]. The CTRS $\mathcal{R}_s$ will be used to test if a transformation is sound and $\mathcal{R}_t$ to test if it preserves termination. Let $\mathcal{R}_s$ be the CTRS

$$\begin{array}{lll} A \to h(f(a), f(b)) & g(d,x,x) \to B & a \to c \quad b \to c \quad c \to e \quad d \to m \\ h(x,x) \to g(x,x,f(k)) & f(x) \to x \ \textit{if}\ x \to e & a \to d \quad b \to d \quad c \to l \quad k \to l \\ & & \qquad\qquad\qquad\quad k \to m \end{array}$$

Let $\mathcal{R}_t$ be $\mathcal{R}_s \cup \{B \to A\}$; then $A \not\to^*_{\mathcal{R}_t} B$ and $\mathcal{R}_t$ operationally terminates. □

Example 2 [ABH03]. The two-rule canonical CTRS $\{f(g(x)) \to x \ \textit{if}\ x \to 0, g(g(x)) \to g(x)\}$ will be used to test whether a transformation preserves confluence. □

Bergstra&Klop. The first CTRS-to-TRS transformation appeared in [BK86]: start with a rule $Ix \to x$ and to each rule $\rho_i : l \to r \ \textit{if}\ cl \to cr$ associate rules $\rho'_i : l \to \sigma_i(cl)r$ and $\rho''_i : \sigma_i(cr) \to I$. The transformation is proved to be complete in [BK86] and claimed to also be sound Let us apply this transformation on $\mathcal{R}_s$ in Example 1. Rule $f(x) \to x \ \textit{if}\ x \to e$ is replaced by $f(x) \to \sigma_1(x)x$ and $\sigma_1(e) \to I$, and rule $Ix \to x$ is added. Then:

$$\begin{array}{l} A \to h(f(a), f(b)) \to h(\sigma_1(a)a, f(b)) \to h(\sigma_1(a)d, f(b)) \to h(\sigma_1(c)d, f(b)) \\ \to h(\sigma_1(c)d, \sigma_1(b)b) \to h(\sigma_1(c)d, \sigma_1(b)d) \to h(\sigma_1(c)d, \sigma_1(c)d) \\ \to g(\sigma_1(c)d, \sigma_1(c)d, f(k)) \to g(\sigma_1(e)d, \sigma_1(c)d, f(k)) \to g(Id, \sigma_1(c)d, f(k)) \\ \to g(d, \sigma_1(c)d, f(k)) \to g(d, \sigma_1(l)d, f(k)) \to g(d, \sigma_1(l)m, f(k)) \\ \to g(d, \sigma_1(l)m, \sigma_1(k)k) \to g(d, \sigma_1(l)m, \sigma_1(l)k) \to g(d, \sigma_1(l)m, \sigma_1(l)m) \to B \end{array}$$

So this transformation is *not sound*. Transforming $\mathcal{R}_t$, we can see that this transformation does *not preserve termination*, because $A \to^+ A$. For the system in Example 2, $f(g(x)) \to x \ \textit{if}\ x \to 0$ is replaced by $f(g(x)) \to \sigma_1(x)x$ and $\sigma_1(0) \to I$, so $f(g(g(0))) \to f(g(0)) \to \sigma_1(0)0 \to I0 \to 0$ and $f(g(g(0))) \to \sigma_1(g(0))g(0)$, both of them normal forms. Thus the resulting TRS is *not confluent*. Consequently, this transformation does not produce computationally equivalent TRSs.

Giovanetti&Moiso. The transformation in [GM87] (suggested in [DP88]) replaces each rule $\rho_i : l \to r \ \mathit{if}\ cl \to cr$ by $l \to \mathit{if}_i(\mathit{Var}(l), cl)$ and $\mathit{if}_i(\mathit{Var}(l), cr) \to r$. However, this transformation is complete and computationally equivalent only when the original CTRS is *safely transformable* [GM87], that is, has no superposition, is simply terminating, and is non-overlapping on conditions. The "safely transformable" CTRSs are too restrictive; our transformation yields computationally equivalent TRSs imposing only ground confluence (safely transformable CTRSs are ground confluent) on the original CTRS.

Hintermeier [Hin94] proposes a technique where an "interpreter" for a CTRS is specified using unconditional rewrite rules, defining the detailed steps of the application of a conditional rewrite rule including rewrite-based implementations of matching and substitution application. Although this is an interesting result, it has little practical relevance (the "meta" stepwise simulation of a rewrite system using another rewrite system leads to dramatic performance loss).

Marchiori's Unravellings. An abstract notion of transformation, called *unravelling*, and several concrete instances of it, were introduced by Marchiori in [Mar96]; these were further studied [Mar97, Ohl99, Ohl02, NSS04]. An *unraveling* is a computable map U from CTRSs to TRSs over the same signature, except a special operation U_ρ for each rule ρ, such that $\downarrow_R \subseteq \downarrow_{U(R)}$ ($\downarrow$ stands for "join", i.e., $\to;\leftarrow$) and $U(T \cup R) = T \cup U(R)$ if T is a TRS. The concrete instance transformations are similar to that in [GM87]: each conditional rule $\rho : l \to r \ \mathit{if}\ cl \to cr$ is replaced by its unravelling, rules $l \to U_\rho(cl, \mathit{Var}(r))$ and $U_\rho(cr, \mathit{Var}(r)) \to r$. Completeness holds, but soundness does not hold without auxiliary hypotheses [Mar96] (see Example 1) such as left linearity [Mar96, Ohl02]. Also, (quasi) decreasingness and left linearity of the CTRS imply termination of the corresponding TRS.

Example 3 A sound and complete transformation does not necessarily yield computational equivalence even if the original CTRS is canonical. The unravelling of the system in Example 2 is $\{f(g(x)) \to U_1(x, x), U_1(0, x) \to x, g(g(x)) \to g(x)\}$. The original CTRS is left-linear, so the unravelling is sound and complete, but is *not* computationally equivalent: $f(g(g(0)))$ reduces to $U_1(g(0), g(0))$, a normal form with no correspondent normal form in the original CTRS. □

Unfortunately, unravellings do not preserve confluence as seen above, and, indeed, they do not yield computationally equivalent TRSs. Therefore, it is not surprising that the more recent transformations discussed next, including ours, are *not* unravellings (they significantly modify the original signature).

Viry. The transformation in [Vir99] (inspired form [AGM90]) inspired all subsequent approaches. It modifies the signature by adding to each operation as many arguments as conditional rules having it at top of their lhs. Two unconditional rules replace each conditional rule, one for initializing the auxiliary

arguments and the other for the actual rewrite step. Formally: let $\rho_{\sigma,i}$ denote the ith rule whose lhs is topped in σ; add as many arguments to σ as the number of rules $\rho_{\sigma,i}$; let $c_{\sigma,i}$ be the number $\mathit{arity}(\sigma) + i$, corresponding to the i^{th} auxiliary argument added to σ; transform each rule $\rho_{\sigma,i} : l \to r \ \mathit{if}\ cl \to cr$ into

$$\rho'_{\sigma,i} : \widetilde{l}[c_{\sigma,i} \leftarrow \bot] \to \widetilde{l}[c_{\sigma,i} \leftarrow [\overline{cl}, \mathit{Var}(l)]] \text{ and } \rho''_{\sigma,i} : l^*[c_{\sigma,i} \leftarrow [cr, \mathit{Var}(l)]] \to \overline{r},$$

where "$\bot$" is a special constant stating that the corresponding conditional rule has not been tried yet on the current position, $\overline{s}$ lifts a term by setting all new arguments to $\bot$, $\widetilde{s}$ lifts a term with fresh variables on the new arguments, and $s^* = s$ replaces all variables in $\widetilde{s}$ with fresh variables. Structures $[u, \overrightarrow{s}]$ comprise the reduction status of conditions (u) together with corresponding substitutions ($\overrightarrow{s}$) when were started. The substitution is used to correctly initiate the reduction of the rhs of the original conditional rule. Viry proved in [Vir99] his transformation sound and complete and that it preserves termination. We believe the completeness indeed holds, but have counter-examples for the other properties. Let us transform the CTRS $\mathcal{R}_s$ from Example 1. First, rules $h(x,x) \to g(x,x,f(k))$ and $g(d,x,x) \to B$ are replaced by $h(x,y) \to g(x,x,f(k))\ \mathit{if}\ \mathit{eq}(x,y) \to \mathit{true}$ and $g(d,x,y) \to B\ \mathit{if}\ \mathit{eq}(x,y) \to \mathit{true}$ to resolve non-left-linearity, where $\mathit{eq}(x,x) \to \mathit{true}$ is the only non-left-linear rule allowed [Vir99]. Then the above conditional rules, together with $f(x) \to x\ \mathit{if}\ x \to e$ and $A \to h(f(a), f(b))$ are transformed into:

$$\begin{array}{ll}
f(x,\bot) \to f(x,[x,x]) & h(x,y,\bot) \to h(x,y,[\mathit{eq}(x,y),x,y] \\
f(y,[e,x]) \to x & h(z,w,[\mathit{true},x,y]) \to g(x,x,f(k,\bot),\bot) \\
h(d,z,w,[\mathit{true},x,y]) \to B & g(d,x,y,\bot) \to g(d,x,y,[\mathit{eq}(x,y),x,y] \\
\multicolumn{2}{c}{A \to h(f(a,\bot),f(b,\bot),\bot)}
\end{array}$$

The following is a valid sequence in the generated unconditional TRS:

$$\begin{array}{ll}
A & \to h(f(a,\bot),f(b,\bot),\bot) \to h(f(a,[a,a]),f(b,\bot),\bot) \\
& \to h(f(d,[a,a]),f(b,\bot),\bot) \to h(f(d,[c,a]),f(b,\bot),\bot) \\
& \to h(f(d,[c,c]),f(b,\bot),\bot) \to h(f(d,[c,c]),f(b,[b,b]),\bot) \\
& \to h(f(d,[c,c]),f(d,[b,b]),\bot) \to h(f(d,[c,c]),f(d,[c,b]),\bot) \\
& \to h(f(d,[c,c]),f(d,[c,c]),\bot) \\
& \to h(f(d,[c,c]),f(d,[c,c]),[\mathit{eq}(f(d,[c,c]),f(d,[c,c])),f(d,[c,c]),f(d,[c,c])]) \\
& \to h(f(d,[c,c]),f(d,[c,c]),[\mathit{true},f(d,[c,c]),f(d,[c,c])]) \\
& \to g(f(d,[c,c]),f(d,[c,c]),f(k,\bot),\bot) \to g(f(d,[e,c]),f(d,[c,c]),f(k,\bot),\bot) \\
& \to g(f(d,[e,e]),f(d,[c,c]),f(k,\bot),\bot) \to g(d,f(d,[c,c]),f(k,\bot),\bot) \\
& \to g(d,f(m,[c,c]),f(k,\bot),\bot) \to g(d,f(m,[l,c]),f(k,\bot),\bot) \\
& \to g(d,f(m,[l,l]),f(k,\bot),\bot) \to g(d,f(m,[l,l]),f(k,[k,k]),\bot) \\
& \to g(d,f(m,[l,l]),f(m,[k,k]),\bot) \to g(d,f(m,[l,l]),f(m,[l,k]),\bot) \\
& \to g(d,f(m,[l,l]),f(m,[l,l]),\bot) \\
& \to g(d,f(m,[l,l]),f(m,[l,l]),[\mathit{eq}(f(m,[l,l]),f(m,[l,l])),f(m,[l,l]),f(m,[l,l])]) \\
& \to g(d,f(m,[l,l]),f(m,[l,l]),[\mathit{true},f(m,[l,l]),f(m,[l,l])]) \\
& \to B
\end{array}$$

Hence, Viry's transformation is *not sound*. Using $\mathcal{R}_t$ instead of $\mathcal{R}_s$, whose corresponding TRS just adds rule $B \to A$ to that of $\mathcal{R}_s$, we can notice that it *does*

not preserves termination either. Let us now transform the CTRS in Example 2 to $\{f(g(x), \bot) \to f(g(x), [x, x]), f(x, [0, y]) \to y, g(g(x)) \to g(x)\}$; note that $\mathcal{R}'$ is *not confluent* [ABH03] (with or without Viry's conditional eagerness [Vir99]) $f(g(g(0)), \bot)$ can be reduced to both 0 and $f(g(0), [g(0), (g(0))])$. Therefore, this transformation does not fulfill the requirements of computational equivalence.

Antoy,Brassel&Hanus proposed in [ABH03] a simple fix to Viry's technique, namely to restrict the input CTRSs to *constructor-based* (i.e., the lhs of each rule is a term of the form $f(t_1, \ldots, t_n)$, where f is *defined* and $t_1, \ldots, t_n$ are all *constructor* terms) and *left-linear* ones. Under these restrictions, they also show that the substitution needed by Viry's transformation is not necessary anymore, so they drop it an prove that the new transformation is sound and complete; moreover, if the original CTRS is additionally weakly orthogonal, then the resulting CTRS is confluent on reachable terms. It is suggested in [ABH03] that what Viry's transformation (or their optimized version of it) needs to generate computationally equivalent TRSs is to reduce its applicability to only constructor-based, weakly orthogonal and left-linear CTRSs. While constructor-baseness and left linearity are easy to check automatically, we believe that it is an unnecessary strong restriction on the input CTRS, which may make the translation useless in many situations of practical interest (see, e.g., the bubble-sort algorithm in Section 4). In this paper we show that our transformation, which can also be regarded as a variant of Viry's, needs only the *ground confluence* of the CTRS, a property expected to hold in practice, including in logical functional programming that the transformation in [ABH03] is aimed at.

Roşu. The transformation in [Ros04] requires the rewrite engine to support some simple *contextual rewriting strategies*, namely an $\mathit{if}(\_, \_, \_)$ eager on the condition and an *equal?* eager on both arguments. As in Viry's transformation, additional arguments are added to each operation σ for each conditional rule $\rho_{\sigma,i}$, but they only need to keep truth values. The distinctive feature of this transformation is the introduction of the $\{\_\}$ operation, which allows the rewriting process to continue after a condition got stuck provided changes occur in subterms. A rule $\rho_{\sigma,i} : l \to r \ \mathit{if}\ cl{\downarrow}cr$ is encoded by $\overline{\rho}_{\sigma,i} : \widetilde{l}[c_{\sigma,i} \leftarrow \mathit{true}] \to \mathit{if}(\mathit{equal?}(\{\overline{cl}\}, \{\overline{cr}\}), \{\overline{r}\}, \widetilde{l}[c_{\sigma,i} \leftarrow \mathit{false}])$. The bracket clears the failed conditions on the path to the top: $\sigma(x_1, .., \{x_i\}, .., x_{\mathit{arity}(\sigma)}, y_1, .., y_m) \to \{\sigma(x_1, .., x_i, .., x_{\mathit{arity}(\sigma)}, \mathit{true}, .., \mathit{true})\}$ It is shown in [Ros04] that the transformation is sound and that operational termination is preserved and implies completeness and preservation of ground confluence, that is, computational equivalence. Left linearity needs not be assumed. Although most modern rewrite systems support the rewrite strategies required by the transformation in [Ros04], we argue that imposing restrictions on the order of evaluation makes a rewrite engine less friendly w.r.t parallelism and more complex; in some sense, contextual strategies can be seen as some sort of conditional rules: apply the rule *if* the context permits.

Our transformation basically integrates Rosu's $\{\_\}$ operation within Viry's

transformation, which allows us to also eliminate the need to carry a substitution. We recently found out[1] that a related approach was followed by Brassel in his master thesis [Bra99], but we can't relate our results since we were unable to obtain an English translation of his results.

3 Our Transformation

Like in the last three transformations above, auxiliary arguments are added to some operators to maintain the control context information. Let $\mathcal{R}$ be any Σ-CTRS. A *σ-conditional rule* [Vir99] is a conditional rule with σ at the top of its lhs, i.e., one of the form $\sigma(t_1,\ldots,t_n) \to r$ *if* $cl \to cr$. Let k_σ be the number of σ-conditional rules and let $\rho_{\sigma,i}$ denote the i^{th} σ-conditional rule in $\mathcal{R}$.

The signature transformation. Let $\overline{\Sigma}$ be the signature containing: a fresh constant $\bot$; a fresh unary operator $\{\_\}$; for any $\sigma \in \Sigma_n$ (i.e., $\sigma \in \Sigma$ has n arguments), an operation $\overline{\sigma} \in \overline{\Sigma}_{n+k_\sigma}$ (the additional k_σ arguments of $\overline{\sigma}$ are written to the right of the other n arguments). An important step in our transformation is to replace Σ-terms by corresponding $\overline{\Sigma}$-terms. The reason for the additional arguments is to pass the control context (due to conditional rules) into data context: the additional i-th argument of $\overline{\sigma}$ at some position in a term maintains the status of appliance of $\rho_{\sigma,i}$; if $\bot$ then that rule was not tried, otherwise the condition is being under evaluation or is already evaluated. Thus, the corresponding $\overline{\Sigma}$-term of a Σ-term is obtained by replacing each operator σ by $\overline{\sigma}$ with the k_σ additional arguments all $\bot$. Formally, let $\mathcal{X}$ be an infinite set of *variables* and let $\overline{\cdot} : T_\Sigma(\mathcal{X}) \to T_{\overline{\Sigma}}(\mathcal{X})$ be defined inductively as: $\overline{x} = x$ for any $x \in \mathcal{X}$ and $\overline{\sigma(t_1,\ldots,t_n)} = \overline{\sigma}(\overline{t_1},\ldots,\overline{t_n},\bot,\ldots,\bot)$ for any $\sigma \in \Sigma_n$ and any $t_1,\ldots,t_n \in T_\Sigma(\mathcal{X})$. Let us define another map, $\widetilde{\cdot}^X : T_\Sigma(X) \to T_{\overline{\Sigma}}(\mathcal{X})$, this time indexed by a *finite* set of variables $X \subseteq \mathcal{X}$, as $\widetilde{x}^X = x$ for any $x \in X$, and as $\widetilde{\sigma(t_1,\ldots,t_n)}^X = \overline{\sigma}(\widetilde{t_1}^X,\ldots,\widetilde{t_n}^X,b_1,\ldots,b_{k_\sigma})$ for any $\sigma \in \Sigma_n$ and $t_1,\ldots,t_n \in T_\Sigma(X)$, where $b_1,\ldots,b_{k_\sigma} \in \mathcal{X} - X$ are some arbitrary but fixed different fresh variables that do not occur in X or in $\widetilde{t_1}^X,\ldots,\widetilde{t_n}^X$. Therefore, $\widetilde{t}^X$ transforms the Σ-term t into a $\overline{\Sigma}$-term by replacing each operation $\sigma \in \Sigma$ by $\overline{\sigma} \in \overline{\Sigma}$ and adding some distinct fresh variables for the additional arguments, chosen arbitrarily but deterministically.

The rewrite rules transformation. Given a Σ-CTRS $\mathcal{R}$, let $\overline{\mathcal{R}}$ be the $\overline{\Sigma}$-TRS obtained as follows. For each conditional rule $\rho_{\sigma,i}$: $l \to r$ *if* $cl \to cr$ over variables X in $\mathcal{R}$, add to $\overline{\mathcal{R}}$ two rules, namely $\overline{\rho}_{\sigma,i} : \widetilde{l}^X[c_{\sigma,i} \leftarrow \bot] \to \widetilde{l}^X[c_{\sigma,i} \leftarrow \{\overline{cl}\}]$ and $\overline{\rho}'_{\sigma,i} : \widetilde{l}^X[c_{\sigma,i} \leftarrow \{cr\}] \to \{\overline{r}\}$, where $c_{\sigma,i}$ is the number $arity(\sigma)+i$ corresponding to the i^{th} conditional argument of σ. For each unconditional rule $l \to r$ in $\mathcal{R}$, add rule $\widetilde{l}^X \to \{\overline{r}\}$ to $\overline{\mathcal{R}}$. For each $\sigma \in \Sigma_n$ and each $1 \le i \le n$, add to $\overline{\mathcal{R}}$ a rule $\overline{\sigma}(x_1,..,x_{i-1},\{x_i\},x_{i+1},..,x_n,b_1,..,b_{k_\sigma}) \to \{\overline{\sigma}(x_1,..,x_{i-1},x_i,x_{i+1},..,x_n,\bot$

[1] from a private communication with Brent Brassel.

$, .., \bot)\}$, intuitively stating that a condition tried and potentially failed in the past at some position may hold once an immediate subterm changes; the operation $\{\_\}$, symbolizing the change, also needs to be propagated bottom-up. The applicability information of an operation can be updated from several of its subterms; to keep this operation idempotent, we add $\{\{x\}\} \to \{x\}$ to $\overline{\mathcal{R}}$. The size of $\overline{\mathcal{R}}$ is $1+u+2\times c+\sum_{n\geq 0} n \times |\Sigma_n|$, where u is the number of unconditional rewrite rules and c is the number of conditional rewrite rules in $\mathcal{R}$.

4 Examples and Experiments

We next illustrate our transformation on several examples.

Confluence is preserved. Let us transform the CTRS in Example 2:

$$\begin{array}{lll} \overline{f}(\overline{g}(x), \bot) \to \overline{f}(\overline{g}(x), \{x\}) & \overline{f}(\overline{g}(x), \{\overline{0}\}) \to \{x\} & \overline{g}(\overline{g}(x)) \to \{\overline{g}(x)\} \\ \overline{g}(\{x\}) \to \{\overline{g}(x)\} & \overline{f}(\{x\}, b) \to \{\overline{f}(x, \bot)\} & \{\{x\}\} \to \{x\} \end{array}$$

The problem that appeared in Viry's transformation is avoided in our transformation by the rules of $\{\cdot\}$, which allow the evaluation of a condition to be restarted at the top of a term once a modification occurs in a subterm. Thus, given the $\overline{\Sigma}$-term $\{\overline{f}(\overline{g}(\overline{g}(\overline{0})), \bot)\}$, even if a rewrite engine first tries to evaluate the condition at the top, a "correct" rewriting sequence is eventually obtained: $\{\overline{f}(\overline{g}(\overline{g}(\overline{0})), \bot)\} \to_{\overline{\mathcal{R}}} \{\overline{f}(\overline{g}(\overline{g}(\overline{0})), \{\overline{g}(\overline{0})\})\} \to_{\overline{\mathcal{R}}} \{\overline{f}(\{\overline{g}(\overline{0})\}, \{\overline{g}(\overline{0})\})\} \to_{\overline{\mathcal{R}}} \{\{\overline{f}(\overline{g}(\overline{0}), \bot)\}\}$, and now the condition can be tried again and this time will succeed.

Odd/Even [Ros04]. Let us consider natural numbers with 0 and successor s, constants *true* and *false* and the following on purpose inefficient conditional rules defining *odd* and *even* operators on natural numbers:

$$\begin{array}{lll} o(0) \to \mathit{false} & o(s(x)) \to \mathit{true} \text{ if } e(x) \to \mathit{true} & o(s(x)) \to \mathit{false} \text{ if } e(x) \to \mathit{false} \\ e(0) \to \mathit{true} & e(s(x)) \to \mathit{true} \text{ if } o(x) \to \mathit{true} & e(s(x)) \to \mathit{false} \text{ if } o(x) \to \mathit{false} \end{array}$$

In order to check whether a natural number n, i.e., a term consisting of n successor operations applied to 0, is odd, a conditional rewrite engine may need $\mathcal{O}(2^n)$ rewrites in the worst case. Indeed, if $n > 0$ then either the second or the third rule of *odd* can be applied at the first step; however, in order to apply any of those rules one needs to reduce the even of the predecessor of n, twice. Iteratively, the evaluation of each even involves the reduction of two odds, and so on. Moreover, the rewrite engine needs to maintain a control context data-structure, storing the status of the application of each (nested) rule that is being tried in a reduction. It is the information stored in this control context that allows the rewriting engine to backtrack and find an appropriate rewriting sequence. As shown at the end of this section, current rewrite engines perform quite poorly on this system. Let us apply it our transformation. Since there are

two *odd*-conditional rules and two *even*-conditional rules, each of these operators will be enriched with two arguments. The new TRS is (for aesthetical reasons we overline only those operations that change; C_1 and C_2 are variables):

$$\begin{array}{ll}
\overline{o}(0, c_1, c_2) \to \{\mathit{false}\} & \overline{e}(0, c_1, c_2) \to \{\mathit{true}\} \\
\overline{o}(s(x), \{\mathit{false}\}, c_2) \to \{\mathit{false}\} & \overline{e}(s(x), \{\mathit{false}\}, c_2) \to \{\mathit{false}\} \\
\overline{o}(s(x), c_1, \{\mathit{true}\}) \to \{\mathit{true}\} & \overline{e}(s(x), c_1, \{\mathit{true}\}) \to \{\mathit{true}\} \\
\overline{o}(s(x), \bot, c_2) \to \overline{o}(s(x), \{\overline{e}(x, \bot, \bot)\}, c_2) & \overline{e}(s(x), \bot, c_2) \to \overline{e}(s(x), \{\overline{o}(x, \bot, \bot)\}, c_2) \\
\overline{o}(s(x), c_1, \bot) \to \overline{o}(s(x), c_1, \{\overline{e}(x, \bot, \bot)\}) & \overline{e}(s(x), c_1, \bot) \to \overline{e}(s(x), c_1, \{\overline{o}(x, \bot, \bot)\}) \\
s(\{x\}) \to \{s(x)\} & \overline{o}(\{x\}, c_1, c_2) \to \{\overline{o}(x, c_1, c_2)\} \\
\{\{x\}\} \to \{x\} & \overline{e}(\{x\}, c_1, c_2) \to \{\overline{e}(x, c_1, c_2)\}
\end{array}$$

If one wants to test whether a number n, i.e., n consecutive applications of successor on 0, is odd, one should reduce the term $\{\mathit{odd}(n, \bot, \bot)\}$.

Quotient/Reminder [Ohl02]. The CTRS below computes the quotient and reminder of two numbers:

$$\begin{array}{ll}
x < 0 \to \mathit{false} & s(x) - s(y) \to x - y \\
0 < s(x) \to \mathit{true} & s'(\langle x, y\rangle) \to \langle s(x), y\rangle \\
s(x) < s(y) \to x < y & \%(0, s(y)) \to \langle 0, 0\rangle \\
0 - s(y) \to 0 & \%(s(x), s(y)) \to \langle 0, s(x)\rangle \ \mathit{if}\ x < y \to \mathit{true} \\
x - 0 \to x & \%(s(x), s(y)) \to s'(\%(x - y, s(y))) \ \mathit{if}\ x < y \to \mathit{false}
\end{array}$$

The corresponding TRS is the following:

$$\begin{array}{ll}
x < 0 \to \{\mathit{false}\} & \overline{\%}(0, s(y), c_1, c_2) \to \{\langle 0, 0\rangle\} \\
0 < s(x) \to \{\mathit{true}\} & \overline{\%}(s(x), s(y), \bot, c_2) \to \overline{\%}(s(x), s(y), \{x < y\}, c_2) \\
s(x) < s(y) \to \{x < y\} & \overline{\%}(s(x), s(y), \{\mathit{true}\}, c_2) \to \{< 0, s(x) >\} \\
0 - s(y) \to \{0\} & \overline{\%}(s(x), s(y), c_1, \bot) \to \overline{\%}(s(x), s(y), c_1, \{x < y\}) \\
x - 0 \to \{x\} & \overline{\%}(s(x), s(y), c_1, \{\mathit{false}\}) \to \{s'(\overline{\%}(x - y, s(y), \bot, \bot))\} \\
s(x) - s(y) \to \{x - y\} & \overline{\%}(\{x\}, y, c_1, c_2) \to \{\overline{\%}(x, y, \bot, \bot)\} \\
s'(\langle x, y\rangle) \to \{\langle s(x), y\rangle\} & \overline{\%}(x, \{y\}, c_1, c_2) \to \{\overline{\%}(x, y, \bot, \bot)\}
\end{array}$$

$$\begin{array}{lll}
\{\{x\}\} \to \{x\} & s(\{x\}) \to \{s(x)\} & s'(\{x\}) \to \{s'(x)\} \\
\{x\} < y \to \{x < y\} & x < \{y\} \to \{x < y\} & \langle\{x\}, y\rangle \to \{\langle x, y\rangle\} \\
\{x\} - y \to \{x - y\} & x - \{y\} \to \{x - y\} & \langle x, \{y\}\rangle \to \{\langle x, y\rangle\}
\end{array}$$

Bubble sort. The following one-rule CTRS sorts lists of numbers (we assume appropriate rules for numbers) implementing the bubble sort algorithm.

$$\cdot(x, \cdot(y, l)) \to \cdot(y, \cdot(x, l)) \ \mathit{if}\ x < y \to \mathit{true}$$

This CTRS is ground confluent but *not* constructor-based. Its translation is:

$$\begin{array}{ll}
\cdot(x, \cdot(y, l, c), \bot) \to \cdot(x, \cdot(y, l, c), \{x > y\}) & \{\{l\}\} \to \{l\} \\
\cdot(x, \cdot(y, l, c), \{\mathit{true}\}) \to \{\cdot(y, \cdot(x, l, \bot), \bot)\} & \cdot(x, \{l\}, c) \to \{\cdot(x, l, \bot)\}
\end{array}$$

Experiments. Our major motivation to translate a CTRS into a computationally equivalent TRS that can run on any unrestricted unconditional rewrite engine was the potential to device highly parallelizable rewrite engines. It was therefore an unexpected and a pleasant surprise to note that our transformation can actually bring immediate benefits if implemented as a front-end to *existing* rewrite engines. Note, however, that current rewrite engines are optimized for *both* conditional and unconditional rewriting; an engine optimized for just unconditional rewriting could probably be more efficient.

We next give some numbers regarding the speed of the generated TRS. We used Elan and Maude as rewrite engines and the examples Odd/Even and Quotient/Reminder. We have tested how long it took for a term to be rewritten to a normal form. In the table below, Cond shows the results using the original system, Ucond those using the presented transformation and Ucond* those using a simple but practical optimization which is described below. Times were obtained on a machine with 2 GHz Pentium 4 CPU and 512MB RAM.

Odd/Even	Elan			Maude		
reducing odd(22)	Cond	Uncond	Uncond*	Cond	Uncond	Uncond*
	1286s	151s	∼0s	20s	6s	∼0s
Quotient/Reminder	Elan			Maude		
1275000/130	Cond	Uncond	Uncond*	Cond	Uncond	Uncond*
	5.8s	3.1s	3s	5.9s	7.5s	5s

Bubble-sort	100		100		10000	
(Maude)	Cond	Uncond	Cond	Uncond	Cond	Uncond
	18ms	11ms	1475ms	955ms	206s	139s

A relatively good computation speed-up is obtained for the odd/even example. The optimized transformation, overcame the speed of computation of the original CTRS in all our experiments. We actually expect our transformation to be significantly better on parallel rewrite engines.

The simple and practical optimization is as follows: if two or more σ-conditional rules have the same lhs and their conditions also have the same lhs, then we can add only one auxiliary argument to σ in $\overline{\sigma}$ for all of these and only one rule in the TRS for starting the condition. With this, e.g., the optimized TRS generated for the odd/even CTRS is:

$$\begin{array}{lll}
\overline{o}(0,c_1) \to \{\mathit{false}\} & \overline{o}(s(x),\{\mathit{false}\}) \to \{\mathit{false}\} & \overline{o}(s(x),\bot) \to \overline{o}(s(x),\{\overline{e}(x,\bot)\}) \\
\overline{e}(0,c_1) \to \{\mathit{true}\} & \overline{o}(s(x),\{\mathit{true}\}) \to \{\mathit{true}\} & \overline{e}(s(x),\bot) \to \overline{e}(s(x),\{\overline{o}(x,\bot)\}) \\
\{\{x\}\} \to \{x\} & \overline{e}(s(x),\{\mathit{false}\}) \to \{\mathit{false}\} & \overline{o}(\{x\},c_1) \to \{\overline{o}(x,c_1)\} \\
s(\{x\}) \to \{s(x)\} & \overline{e}(s(x),\{\mathit{true}\}) \to \{\mathit{true}\} & \overline{e}(\{x\},c_1) \to \{\overline{e}(x,c_1)\}
\end{array}$$

5 Theoretical aspects

5.1 Technical preliminaries.

We recall some basic notions of (conditional) rewriting we will use in the sequel, referring the interested reader to [Ohl02] for more details. An (unsorted) *signa-*

ture Σ is a finite set of operational symbols, each having zero or more arguments. We let $\Sigma_n \subseteq \Sigma$ denote the set of operations of n arguments. Operations *of zero arguments* in Σ_0 are called *constants*. We assume an infinite set of *variables* $\mathcal{X}$. Given a signature Σ and a set of variables $X \subseteq \mathcal{X}$, we let $T_\Sigma(X)$ denote the algebra of Σ-*terms* over variables in X. We let T_Σ denote the algebra $T_\Sigma(\emptyset)$ of *ground* terms. A map $\theta : \mathcal{X} \to T_\Sigma(\mathcal{X})$ can be uniquely extended to a morphism of algebras $\theta : T_\Sigma(\mathcal{X}) \to T_\Sigma(\mathcal{X})$, called *substitution*, replacing each $x \in X$ by a term $\theta(x)$. A conditional Σ-rewrite rule has the form $l \to r$ *if* $cl \to cr$, where l, r, cl and cr, are Σ-terms in $T_\Sigma(\mathcal{X})$. The term l is called the *left-hand-side (lhs)*, r is called the *right-hand-side (rhs)*, and $cl \to cr$ is called the *condition* of the rule. We disallow rewriting rules whose lhs is a variable and assume that the lhs contains all the variables that occur in the rule. Following the terminology in [MH94], our rules are *of type 1*. An *unconditional* rewrite rule has the form $l \to r$. A *conditional (unconditional)* Σ-*term rewrite system* $\mathcal{R} = (\Sigma, R)$, abbreviated *CTRS (TRS)*, consists of a finite set R of conditional (unconditional) Σ-rewrite rules. We here use only a restricted form of *normal* CTRSs (see [DOS88]), i.e., ones whose rhs of the condition is a constant not which is not a lhs for any of the rules. Section 6 discusses possible extensions. We say that a rewrite rule is left-linear if its lhs has no multiple occurrences of the same variable.

A *multi-context* is a term γ in $T_\Sigma(\{*\})$. If γ has n occurrences of $*$ then $\gamma[t_1, \ldots, t_n]$ denotes the term obtained by substituting $*$ with $t_1, \ldots, t_n$ from left to right in γ. If γ has a single occurrence of $*$ then we simply call it *context*

Any Σ-rewrite system $\mathcal{R} = (\Sigma, R)$ generates a relation $\to_\mathcal{R}$ on $T_\Sigma(\mathcal{X})$, defined as follows. For any $\theta(:)\mathcal{X} \to T_\Sigma(\mathcal{X})$, $\gamma[\theta(l)] \to_\mathcal{R} \gamma[\theta(r)]$ whenever $\theta(cl) \to^*_\mathcal{R} cr$, where γ is context, i.e., a term having one occurrence of a special variable $*$, $\gamma[\theta(l)]$ is the term obtained by substituting $*$ with $\theta(l)$ in γ, and $\to^*_\mathcal{R}$ is the reflexive and transitive closure of $\to_\mathcal{R}$. Note that $\to^*_\mathcal{R}$ is the least relation on $T_\Sigma(\mathcal{X})$ closed under reflexivity, transitivity, congruence and $\mathcal{R}$-substitution. We will also use the notion of *multi-context rewriting*, which allows a multi-context to be used in the definition above. We let $\Rightarrow_\mathcal{R}$ denote the relation associated with multi-context rewriting on system $\mathcal{R}$. Note that $\to^*_\mathcal{R} = \Rightarrow^*_\mathcal{R}$.

Positions are strings of numbers describing paths to subterms. Let $t|_\alpha$ denote the subterm at position α in a term t. We have that $\sigma(t_1, \ldots, \ldots, t_n)|_{i\alpha} = t_i|_\alpha$ and $t|_\varepsilon = t$. A rewrite step *occurred at position* α in a term t when γ is obtained from t by replacing its subterm at position α by $*$. We may let $t[\alpha \leftarrow s]$ denote the term obtained by substituting the subterm at position α in t by s.

$\overline{\Sigma}$-terms are more complex than the Σ-terms. There can even be some $\overline{\Sigma}$-terms that do not resemble any Σ-term. We next define and discuss several classes of $\overline{\Sigma}$-terms that will be used in the sequel. A $\overline{\Sigma}$-term t' is *structural* iff $t' = x$ where x is a variable, or $t' = \{t''\}$ where t'' is structural, or $t' = \overline{\sigma}(t'_1, \ldots, t'_n, C_1, \ldots, C_{k_\sigma})$ where $\sigma \in \Sigma_n$ and t'_i is structural for each $1 \leq i \leq n$. We say that a position α is *structural for* t', where t' is a structural term, iff α is empty, or $t' = \{t''\}$ and $\alpha = 1\alpha'$ with α' being structural for t'', or

$t' = \overline{\sigma}(t'_1, \ldots, t'_n, C_1, \ldots, C_{k_\sigma})$ with $\sigma \in \Sigma_n$ and $\alpha = i\alpha'$ where $1 \leq i \leq n$ and α' is structural for t'_i. A position α is *conditional for* t', where t' is a structural term, iff $\alpha = \alpha' c_{\sigma,i}$ such that α' is structural for t' and $t'_{\alpha'} = \overline{\sigma}(\overline{u})$. A ground $\overline{\Sigma}$-term t' is *reachable* iff there is some ground Σ-term t such that $\{\overline{t}\} \rightarrow^*_{\overline{\mathcal{R}}} \{t'\}$. Note that the lhs and rhs of any (unconditional) rule in $\overline{\mathcal{R}}$ are structural. The set of all conditions started for a reachable term t', written $cond(t')$, is defined as $\bigcup_C(\{s'|_\alpha\} \cup cond(s'|_\alpha))$ where C is the set of conditional positions α in s' such that $s'|_\alpha \neq \perp$.

In proofs, we will denote by $\underline{t}$ the linear variant of $\widetilde{t}^X$, that is the one also replacing the variables of t with fresh ones, giving distinct variables for distinct occurrences of the same variable.

Proposition 1 *1. Any subterm of a structural term on a structural position is also structural.*

2. *If t' is a structural term with variables on structural positions and θ is a substitution giving structural terms for variables of t' then $\theta(t')$ is also structural.*

3. *Structural terms are closed under $\overline{\mathcal{R}}$.*

4. *Any reachable term is also structural.*

5. *Reachable terms are closed under $\overline{\mathcal{R}}$.*

6. *Let t' be a reachable term and s be a Σ-term such that $\{\overline{s}\} \Rightarrow^k_{\overline{\mathcal{R}}} \{t'\}$. Let t_t be a Σ-term with variables and θ_t a $\overline{\Sigma}$-substitution giving for any variable in t_t a subterm of t' on a structural position. Then there exists a Σ-term s_t such that $\{\overline{s_t}\} \Rightarrow^{k'}_{\overline{\mathcal{R}}} \{\theta_t(\overline{t_t})\}$ with $k' \leq k$.*

7. *Let t' be a reachable term and t be a Σ-term such that $\{\overline{t}\} \Rightarrow^k_{\overline{\mathcal{R}}} \{t'\}$. Let s' be a subterm of t' in a structural position. Then there exists a Σ-term s such that $\{\overline{s}\} \Rightarrow^{k'}_{\overline{\mathcal{R}}} \{s'\}$ and $k' \leq k$.*

Proof:

1. We will prove that for all $\overline{\Sigma}$-terms t' and all structural positions α in t', the subterm s' at position α is also structural by induction over the length of α. If α is empty, then $s' = t'$ and thus it is structural. Now suppose that $\alpha = i\alpha'$. If $t' = \{t''\}$ then i is 1; since t'' is structural we apply the induction hypothesis for t'' and α' getting that s' is structural. Otherwise, assume that $t' = \overline{\sigma}(t'_1, \ldots, t'_n, C_1, \ldots, C_{k_\sigma})$ with $\sigma \in \Sigma_n$ and $t'_1, \ldots, t'_n$ structural, then apply the induction hypothesis for t'_i and α' getting that s' is structural.

2. Let $\mathcal{X}$ be a set of variables, t' a structural term with variables from X and θ a substitution from $\mathcal{X}$ to structural terms. We'll prove that $\theta(t)$ is structural by induction on the structure of t'. If t' is a variable then it's

obvious, since θ' substitutes variables by structural terms. Now suppose that $t' = \overline{\sigma}(t'_1, \ldots, t'_n, C_1, \ldots, C_{k_\sigma})$. Applying the induction hypothesis for each t'_i, $1 \leq i \leq n$. we get that $\theta'(\overline{t'_i})$ is structural. Then $\theta(t')$ must also be structural since

$$\theta(t') = \theta(\overline{\sigma}(t'_1, \ldots, t'_n, C_1, \ldots, C_{k_\sigma})) = \overline{\sigma}(\theta'(t'_1), \ldots, \theta'(t'_n), C_1, \ldots, C_{k_\sigma})$$

and the latter is obviously structural. Finally, if $t' = \{t'_1\}$ we can apply induction hypothesis for t'_1 and get that $\theta'(\overline{t'_1})$ is structural. Then obviously $\theta(\{t'_1\})$ must also be structural.

3. Let s' be a structural term, let t' a $\overline{\Sigma}$-term such that $s' \rightarrow_{\overline{\mathcal{R}}} t'$ and let α be the position of s' where the rewriting step occurred. If α is a non-structural position then t' must also be structural, since the definition of structural terms is done using only structural positions. Let us now prove that for any structural s' and any $\overline{\Sigma}$-term t' such that $s' \rightarrow_{\overline{\mathcal{R}}} t'$ the rewriting step occurring at the structural position α we have that t' is also structural. We do that by induction on the length of α. If $\alpha = i\alpha'$ then either $s' = \{s'_1\}$, $i = 1$ and $t' = \{t'_1\}$ or $s' = \overline{\sigma}(s'_1, \ldots, s'_n, C_1, \ldots, C_{k_\sigma})$, $1 \leq i \leq n$ and $t' = \overline{\sigma}(s'_1, \ldots, s'_{i-1}, t'_i, s'_{i+1}, \ldots, s'_n, C_1, \ldots, C_{k_\sigma})$. Also, we must have that s'_i is structural and $s'_i \rightarrow_{\overline{\mathcal{R}}} t'_i$ using the same rule and the same substitution a position α'. Applying the induction hypothesis for we get that t'_i is structural, whence t' is structural. If α is the empty word, then let $l' \rightarrow r'$ be the rule used and θ' be the substitution used. If $l' \rightarrow r'$ is of form:

   - $\{\{x\}\} \rightarrow \{x\}$ then since structural terms are closed under **SS**, $\theta'(x)$ is structural, whence $t' = \{\theta'(x)\}$ is also structural;
   - $\overline{\sigma}(x_1, \ldots, x_{i-1}, \{x_i\}, x_{i+1}, \ldots, x_n, C_1, \cdots, C_{k_\sigma}) \rightarrow$
     $\rightarrow \{\overline{\sigma}(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n, \bot, \cdots, \bot)\}$ then since variables x_j, $1 \leq j \leq n$ are in structural positions, $\theta'(x_j)$ is structural and we apply that structural terms are closed under $\mathbf{C\Sigma}$ for $\sigma(x_1, \ldots, x_n)$;
   - $\widetilde{l}[c_{\sigma,i} \leftarrow \bot] \rightarrow \widetilde{l}[c_{\sigma,i} \leftarrow \{\{\overline{cl}\}\}]$, then we don't have anything to prove since nothing changes on structural positions;
   - $\widetilde{l}[c_{\sigma,i} \leftarrow \{cr\}] \rightarrow \{\overline{r}\}$ or $\widetilde{l} \rightarrow \{\overline{r}\}$, then since all variables in l are on structural positions in $\widetilde{l}$ we have that $\theta'(x)$ is structural for each variable x occurring in l then we can apply that structural terms are closed under $\mathbf{C\Sigma}$ for r.

4. $\overline{s}$ is structural and structural terms are closed under $\overline{\mathcal{R}}$.

5. Obvious from the definition of reachable terms.

6. We prove the affirmation by well founded induction on k. Let t_t be a Σ-term with variables from $X_t = \{x_1, \ldots, x_n\}$ and θ_t a $\overline{\Sigma}$-substitution with variables from X_t such that $\theta_t(x_i) = t'_i$ for each $1 \leq i \leq n$. Then there exists a Σ-term s_t such that $\{\overline{s_t}\} \Rightarrow^{k'}_{\overline{\mathcal{R}}} \{\theta_t(\overline{t_t})\}$ with $k' \leq k$.

Let s' be a $\overline{\Sigma}$-term such that $\{\overline{s}\} \Rightarrow_{\overline{\mathcal{R}}}^{k-1} \{s'\} \Rightarrow_{\overline{\mathcal{R}}} \{t'\}$. We will show that there exist a Σ-term t_s with variables from X_s and a $\overline{\Sigma}$-substitution $\theta_{(}s$ giving for each variable in X_s a subterm of s' on a structural position and that $\theta_{(}s(\overline{t_s}) = \theta_t(\overline{t_t})$ or $\theta_{(}s(\overline{t_s}) \Rightarrow_{\overline{\mathcal{R}}} \theta_t(\overline{t_t})$.

Let c be a $\overline{\Sigma}$ multi-context, $l' \to r'$ be a rule in $\overline{\mathcal{R}}$ with variables in a set X and θ be a $\overline{\Sigma}$-substitution such that $s' = c[\theta(l')]$ and $t' = c[\theta(r')]$. Let us build the set X_s, a substitution θ' giving for any variable in X_t a Σ-term with variables form $X \cup X_t$ and a set of positions $\mathcal{P}$. For any $x \in X_t$, perform the following operation. Let β be the position of $\theta_t(x)$ in t'.

- If β is a position in c then
  - if $c|_\beta$ doesn't contain any variables, then $s'|_\beta = t'|_\beta$. Add x to X_s and let $\theta'(x) = x$.
  - otherwise, we have that $s'|_\beta = c|_\beta[\theta(l')]$ and $t'|_\beta = c|_\beta[\theta(r')]$. Add then x to X_s, let $\theta'(x) = x$ and for each position γ of x in t_t and each position δ of a variable in $c|_\beta$ add to $\mathcal{P}$ the position $\gamma\delta$.
- If β is not a position in c then
  - If $\beta = \alpha\gamma\delta$ where α is a position of a variable in c and γ is a position of a variable y in r', then let γ' be a position of y in l'. We have that $s'|_{\alpha\gamma'\delta} = t'|_{\alpha\gamma\delta}$. Add x to X_s and let $\theta'(x) = x$.
  - If $\beta = \alpha\gamma$ where α is a position of a variable in c and γ is a non-empty non-variable position in r', then let's analyze the possible cases for $l' \to r'$:
    (a) $\{\{x\}\} \to \{x\}$
    (b) $\overline{\sigma}(x_1, \ldots, x_{i-1}, \{x_i\}, x_{i+1}, \ldots, x_n, C_1, \ldots, C_{k_\sigma}) \to \to \{\overline{\sigma}(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n, \bot, \ldots, \bot)\}$
    (c) $\widetilde{l}[c_{\sigma,i} \leftarrow \bot] \to \widetilde{l}[c_{\sigma,i} \leftarrow cl]$
    (d) $\widetilde{l}[c_{\sigma,i} \leftarrow \{cr\}] \to \{\overline{r}\}$
    (e) $\widetilde{l} \to \{\overline{r}\}$
    (a) is impossible since the only position in $\{x\}$ are the one of x and the empty one. Since (c) changes only an auxiliary parameter of the operation on the top of $\widetilde{l}$ we will have that $s'_\beta = t'_\beta$. Then add x to X_s and let $\theta'(x) = x$.
    For (b), (d), and (e) we will have that $r'_\beta = \overline{u}$ for some Σ-term u with variables form X. Add then $var(u)$ to X_s and let $\theta'(x) = u$.

Let $t_s = \theta'(t_t)$ and let $\theta_{(}s$ be defined as follows:

$$\theta_{(}s(x) = \begin{cases} \theta_t(x) & \text{if } x \in X_t \\ \theta(x) & \text{if } x \in X \end{cases}$$

We then have that for any $x \in X_s$, $\theta_{(}s(x)$ is a subterm of s' on a structural position. Also if $\mathcal{P}$ is empty then $\theta_{(}s(t_s) = \theta_t(t_t)$; otherwise $\theta_{(}s(t_s) \Rightarrow_{\overline{\mathcal{R}}} \theta_t(t_t)$ using rule and substitution θ at any position in $\mathcal{P}$.

Applying the induction hypothesis, the proposition is proved.

7. Apply 6 for $t_t = x$ and $\theta_t(x) = s'$.

□

Before we formalize the relationship between CTRSs and their unconditional variants, let us define a partial map on terms, $\widehat{\cdot}: T_{\overline{\Sigma}}(X) \to T_{\Sigma}(X)$ only defined for $\overline{\Sigma}$-structural terms:

- $\widehat{x} = x$ for any variable x.
- $\widehat{\{t'\}} = \widehat{t'}$
- $\overline{\sigma}(t'_1, \ldots, \widehat{t'_n, C_1}, \ldots, C_{k_\sigma}) = \sigma(\widehat{t'_1}, \ldots \widehat{t'_n})$

Therefore, $\widehat{t'}$ forgets all the auxiliary arguments of each operation occurring in t'. Note in particular that $\widehat{\overline{t}} = t$ for any $t \in T_{\Sigma}$.

$\overline{\mathcal{R}}$ rewrite sequences. A $\overline{\Sigma}$-term s' is almost empty if all its conditional positions are $\bot$ but has proper subterms topped in $\{\cdot\}$. A $\overline{\Sigma}$-term s' is a *proof term* if $s' = \{\widehat{\overline{s'}}\}[\pi c_{\sigma,i} \leftarrow c']$ where $\pi c_{\sigma,i}$ is a conditional position in s' and c' is either $\bot$, a proof term or an almost empty term. A rewriting sequence $s' \to^*_{\overline{\mathcal{R}}} t'$ is called a *proof sequence* if: $s' = \{\overline{\widehat{s'}}\}$ and $t' = \{\overline{\widehat{s'}}\}$, all terms in the rewriting sequence are either almost empty or proof terms and a rule in $\overline{\mathcal{R}}$ encoding a rule from $\mathcal{R}$ is only applied at a position α in a term u when $u|_\alpha = \overline{\widehat{u|_\alpha}}$. Basically, proof sequences exactly encode the way a proof for $\widehat{s'} \to^*_{\mathcal{R}} \widehat{t'}$ is done as we shall see in the following section.

Let us next introduce the notion of *safe rewrite sequences*, a relaxation of the notion of proof sequences. We say that a rewrite sequence $\{\overline{s}\} \to^*_{\overline{\mathcal{R}}} t'$ is safe if for any rewriting step $u \to v$ occurring in the sequence using rule $\overline{\rho'}$ at position α we have that for any prefix $\beta c_{\sigma,i}$ of α such that $c_{\sigma,i}$ is a conditional position corresponding to $\rho_{\sigma,i}; l \to r \ \textit{if}\ cl \to cr$ in $u|_\beta$ we have that $u|_\beta[c_{\sigma,i} \leftarrow \bot] \to^*_{\overline{\mathcal{R}}} u|_\beta$ can be extracted from the original sequence $\{\overline{s}\} \to^*_{\overline{\mathcal{R}}} u$ and furthermore, each term w in the thus obtained sequence it is topped in σ and satisfies that $w|_j = \overline{\widehat{w|_j}}$ for each $1 \leq j \leq \textit{arity}(\sigma)$.

Another notion we will use in the sequel is that of *structural rewriting step*. A rewriting step $s' \to_{\overline{\mathcal{R}}} t'$ is structural if it occurs at a structural position in s' and either uses a rule of form $\overline{\rho}'_{\sigma,i}$ or one corresponding to an unconditional rule in $\mathcal{R}$. Note that this notion is not closed under any context. For example if $u|_\alpha = s'$, then $u \to_{\overline{\mathcal{R}}} u[\alpha \leftarrow t']$ is only structural if α is also structural. Therefore, let us distinguish these contexts as *structural contexts*.

The two notions introduced are related by the following proposition:

Proposition 2 *For any safe sequence $\{\overline{s}\} \to^*_{\overline{\mathcal{R}}} t'$ there exists a proof sequence $\{\overline{s}\} \to^*_{\overline{\mathcal{R}}} \{\overline{\widehat{t'}}\}$. having the same number of structural steps*

Proof: By induction on k, the length of the safe sequence. If $k = 0$ there is nothing to prove. Else, suppose $k > 0$ and let u be such that $\{\overline{s}\} \rightarrow_{\overline{\mathcal{R}}}^{k-1} u \rightarrow_{\overline{\mathcal{R}}} t'$. Applying the induction hypothesis for u we obtain the proof sequence $\{\overline{s}\} \rightarrow_{\overline{\mathcal{R}}}^{*} \{\widehat{\overline{u}}\}$. If $\widehat{u} = \widehat{t}$ there is nothing more to do. If the step applied corresponds to a unconditional rewrite step, we can simply apply the same rule to u, then propagate the resulting bracket until it is dissolved.

The only case left is when $u \rightarrow_{\overline{\mathcal{R}}} t'$ using a rule $\overline{\rho}'_{\sigma,i}$ corresponding to the rule $\rho_{\sigma,i} : l \rightarrow r \ \textit{if}\ cl \rightarrow cr$ using substitution θ' at position α. Since the sequence is safe, we must have that $\theta'(\{\overline{cl}\}) = \{\widehat{\theta'(\{\overline{cl}\})}\}$ and also that $\theta'(\{\overline{cl}\}) \rightarrow_{\overline{\mathcal{R}}}^{k'} \{cr\}$ with $k' < k$ whence we can apply the induction hypothesis to obtain a proof sequence for $\theta'(\{\overline{cl}\}) \rightarrow_{\overline{\mathcal{R}}}^{k'} \{cr\}$. We can now use this proof sequence to continue $\{\overline{s}\} \rightarrow_{\overline{\mathcal{R}}}^{*} \{\widehat{\overline{u}}\}$ by first applying rule $\overline{\rho}_{\sigma,i}$ and at the end applying rule $\overline{\rho}'_{\sigma,i}$. We then get the desired form by simply propagating the resulting bracket until it is dissolved. □

Lemma 1 *Let s' be a reachable term and s be a Σ-term such that $\{\overline{s}\} \Rightarrow_{\overline{\mathcal{R}}}^{k} \{s'\}$. Let α be a structural position in s', $\rho_{\sigma,i} : l \rightarrow r \ \textit{if}\ cl \rightarrow cr$ a rule in $\mathcal{R}$ and θ' a $\overline{\Sigma}$ substitution such that $s'|_\alpha = \theta'(\underline{l})$ and $s'|_{\alpha c_{\sigma,i}} \neq \bot$*

Then there exists a substitution θ_0 and Σ-terms s_0, s_x for each variable x in l such that $\{\overline{s_0}\} \Rightarrow_{\overline{\mathcal{R}}}^{k_0} \theta_0(\{\overline{cl}\}) \Rightarrow_{\overline{\mathcal{R}}}^{k_1} \theta'(\{\overline{cl}\})$, with $k_0 + k_1 < k$ and $\{\overline{s_x}\} \Rightarrow_{\overline{\mathcal{R}}}^{k_0^x} \theta_0(\{\overline{cl}\}) \Rightarrow_{\overline{\mathcal{R}}}^{k_1^x} \theta'(x)$ with $k_0^x + k_1^x < k$

Proof: Let us first show the following:

> Let s' and t' be two structural terms such that $s' \Rightarrow_{\overline{\mathcal{R}}} t'$ using the rule $l' \rightarrow r' \in \overline{\mathcal{R}}$, the multi-context c and the substitution θ. Let β be a structural position in t', $\rho_{\sigma,i} : l \rightarrow r \ \textit{if}\ cl \rightarrow cr$ a rule in $\mathcal{R}$ and θ'' a $\overline{\Sigma}$ substitution such that $t'|_\beta = \theta''(\underline{l})$ $t'|_{\beta c_{\sigma,i}} \neq \bot$.
>
> Then there exists a structural position α of s' and a $\overline{\Sigma}$-substitution θ' such that $s'|_\alpha = \theta'(\underline{l})$ and $\theta'(\underline{l}) = \theta''(\underline{l})$ or $\theta'(\underline{l}) \Rightarrow_{\overline{\mathcal{R}}} \theta''(\underline{l})$.

If β is a position in c then if $c|_\beta$ doesn't contain any variables, we can consider $\alpha = \beta$, $\theta'(=)\theta''$; otherwise, for any variable x of $\underline{l}$ on a structural position γ_x we must have that $\beta\gamma_x$ is also a position in c (otherwise the structure of $\underline{l}$ would have changed). We can then take $\alpha = \beta$ and $\theta'(x) = \theta''(c|_{\beta\gamma_x})$.

If β is not a position in c, then $\beta = \beta'\gamma$ where β' is a position of a variable in c. Then γ must be of the form $\gamma = \delta\epsilon$ where δ is a position of a variable x in r' (because otherwise we couldn't have $t'|_{\beta c_{\sigma,i}} \neq \bot$). Let then δ' be a position in l' of the same variable x, and take $\alpha = \beta'\delta'\epsilon$ and $\theta'(=)\theta''$.

We can now apply the statement above repeatedly as long as its hypothesis is satisfied, meaning as long as $t'|_{\beta c_{\sigma,i}} \neq \bot$. We know that it must reach an $\bot$ in at most k steps, since $\overline{s}$ has nothing but $\bot$s on any non-structural position. Let $s'_\bot$ be the term where the above proposition cannot be applied anymore and k_s its position in the rewriting sequence $\overline{s} \Rightarrow_{\overline{\mathcal{R}}}^{k} t'$. We have thus obtained a sequence of terms $l'_1 \Rightarrow_{\overline{\mathcal{R}}} l'_2 \Rightarrow_{\overline{\mathcal{R}}} \ldots \Rightarrow_{\overline{\mathcal{R}}} l'_{k'}$ with $k' \leq k - k_s$ and all terms matching $\underline{l}$.

Furthermore, $l'_{k'} = s'|_\alpha$ and l'_1 is the only one in sequence matching $\underline{l}[c_{\sigma,i} \leftarrow \bot]$. Since also $l'_1 \Rightarrow_{\overline{\mathcal{R}}} l'_2$ we must have that l'_1 matches $\widetilde{l}[c_{\sigma,i} \leftarrow \bot]$ since this is the only way to change the ith auxiliary variable of σ from $\bot$ to something different from $\bot$ - by applying the $\overline{\rho}_{\sigma,i}$ rule.

Let θ_0 be the substitution such that $\theta_0(\widetilde{l}) = l'_1$. Since $\theta_0(\widetilde{l}[c_{\sigma,i} \leftarrow \bot]) \Rightarrow^{k'}_{\overline{\mathcal{R}}} s'|_\alpha$ it must be that $\theta_0(()\{\overline{cl}\}) \Rightarrow^{k''}_{\overline{\mathcal{R}}} s'|_{\alpha c_{\sigma,i}}$ with $k'' < k'$. We can use Proposition 1.6 to verify the existence of an s_0 satisfying the needed property.

Also we have that for each variable x occurring in l $\theta_0(x) \Rightarrow^{k^x_1}_{\overline{\mathcal{R}}} \theta'(x)$ with $k^x_1 < k'$. We can use Proposition 1.7 to verify the existence of s_x. □

The following result gives a good intuition of why $\{\_\}$ is good.

Proposition 3 *Let $\{s'\}$ be a reachable term on which no bracket rule can be applied. Then there exists a safe rewriting sequence $\widehat{\overline{s'}} \rightarrow^*_{\overline{\mathcal{R}}} s'$.*

Proof: We will prove the affirmation by induction on the number of subterms on conditional positions which are not $\bot$. Take a minimal length conditional position $1\alpha c_{\sigma,i}$ such that $\{s'\}|_{1\alpha c_{\sigma,i}} \neq \bot$. Also consider $\rho_{\sigma,i} : l \rightarrow r \ if \ cl \rightarrow cr$ be the corresponding rule. Since no bracket operations can be applied on s', no rule on the structure could have been applied on any proper subterm of $\{s'\}|_{1\alpha}$ since the condition was started; otherwise, the bracket generated by a rule changing the structure would have propagated to dissolve the condition. That means we can consider the term $s'_\bot$ such that $\widehat{s'_\bot} = \widehat{s'|_\alpha}$, $s'_\bot|_{\alpha c_{\sigma,i}} = \bot$ and $\sigma'_\bot \rightarrow^k_{\overline{\mathcal{R}}} s'|_\alpha$, k being minimal with this property. Notice that $s'_\bot$ has strictly less conditions started than s'. Let $\theta'_\bot$ be the substitution for which $\theta'_\bot(\widetilde{l}[c_{\sigma,i} \leftarrow \bot]) = s'_\bot$. Define $\theta''_\bot$ by $\theta''_\bot(x) = \widehat{\overline{\theta'_\bot(x)}}$ for variables in l and $\bot$ for all conditional variables. We have that $\theta''(\widetilde{l}[c_{\sigma,i} \leftarrow \bot]) = \widehat{\overline{s'_\bot}}$, whence we can apply rule $\overline{\rho}_{\sigma,i}$ to obtain $\widehat{\overline{s'_\bot}}[c_{\sigma,i} \leftarrow \{\theta''_\bot(cl)\}]$. We can now apply the induction hypothesis for each variable x to get that $\widehat{\overline{\theta'_\bot(x)}} \rightarrow^*_{\overline{\mathcal{R}}} \theta'_\bot(x)$, hence $\{\theta''_\bot(cl)\} \rightarrow^*_{\overline{\mathcal{R}}} \{\theta'_\bot(cl)\} \rightarrow^*_{\overline{\mathcal{R}}} s'|_{c_{\sigma,i}}$. Same reasoning applies for any position $c_{\sigma,j}$ such that $s'|_{c_{\sigma,j}} \neq \bot$. Also, se can apply the induction hypothesis for any proper structural subterm of $s'|_j$ to obtain that $\widehat{\overline{s'|_j}} \rightarrow^*_{\overline{\mathcal{R}}} s'|_j$. Putting them all together, we can now start with $\widehat{\overline{s'}}$, rewrite its top conditions, then rewrite its proper structural subterms to finally obtain s'. □

5.2 Soundness and Completeness

Completeness means that rewriting that can be executed in the original CTRS can also be simulated on the corresponding TRS. This is a natural result; all transformations are defined with completeness as their primary goal. We show that "everything that can be done on a term s in $\mathcal{R}$ can also be done on the term $\{\overline{s}\}$ in $\overline{\mathcal{R}}$".

Theorem 1 ***Completeness.*** *$s \rightarrow^k_{\mathcal{R}} t$ iff there exists a proof sequence $\{\overline{s}\} \rightarrow_{\overline{\mathcal{R}}} \{\overline{t}\}$ having k structural steps.*

Proof: First, let us prove the direct implication. Rewriting relation can be defined as the least relation closed under R (reflexivity), T (transitivity), $C\Sigma$ (compatibility with the operations) and $Sub_{\mathcal{R}}$ ($\mathcal{R}$-substitution). We will show that the relation:

$$D = \{(s,t) \in T_\Sigma \times T_\Sigma \mid \{\overline{s}\} \rightarrow_{\overline{\mathcal{R}}}^{*} \{\overline{t}\} \text{ is a proof sequence}\}$$

is closed under the above rules whence it contains $\rightarrow_{\mathcal{R}}^{*}$. Closure under reflexivity and transitivity is obvious. For proving the closure under $Sub_{\mathcal{R}}$ let us first formally describe it:

> For any rule $l \rightarrow r$ *if* $cl \rightarrow cr$ in $\mathcal{R}$ and any Σ-substitution θ such that $(\theta(cl), cr) \in D$ we have that $(\theta(l), \theta(r)) \in D$.

Let $\rho_{\sigma,i} : l \rightarrow r$ *if* $cl \rightarrow cr$ be a rule in $\mathcal{R}$, θ be a Σ-substitution such that $(\theta(cl), cr) \in D$. Consider the $\overline{\Sigma}$-substitution θ' defined as $\theta'(x) = \overline{\theta(x)}$. for any variable x occurring in $\rho_{\sigma,i}$ and $\theta'(x) = \perp$ for any new variable occurring in $\overline{\rho_{\sigma,i}}$. We then have that $\theta'(\overline{t}) = \overline{\theta(t)}$ for $t \in \{l, r, cl\}$. We then can apply rule $\overline{\rho_{\sigma,i}}$ on $\{\overline{\theta(l)}\}$ using the substitution θ' and the context $\{\cdot\}$, getting $\{\overline{\theta(l)}[c_{\sigma,i} \leftarrow \{\overline{cl}\}]\}$. Now we use that $(\theta(cl), cr) \in D$ which means that $\{\overline{\theta(cl)}\} \rightarrow_{\overline{\mathcal{R}}}^{*} \{cr\}$, whence

$$\{\overline{\theta(l)}[c_{\sigma,i} \leftarrow \{\overline{\theta(cl)}\}]\} \rightarrow_{\overline{\mathcal{R}}}^{*} \{\overline{\theta(l)}[c_{\sigma,i} \leftarrow \{cr\}]\} \rightarrow_{\overline{\mathcal{R}}}^{*} \{\overline{\theta(r)}\}$$

Finally let us prove that D is closed under $C\Sigma$, that is: if $(s,t) \in D$ then for any operation $\sigma \in \Sigma_n$, for any Σ-terms t_j, $1 \leq j \leq n$ and for any $1 \leq i \leq n$ we have that: $(\sigma(t_1, \ldots, t_{i-1}, s, t_{i+1}, \ldots, t_n), \sigma(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n)) \in D$. In order to prove it we first have to observe that for any $\overline{\Sigma}$-terms s' and t', if $\{s'\} \rightarrow_{\overline{\mathcal{R}}}^{*} \{t'\}$, then $s' \rightarrow_{\overline{\mathcal{R}}}^{*} t'$ or $s' \rightarrow_{\overline{\mathcal{R}}}^{*} \{t'\}$. This is obvious since the only rule containing $\{\cdot\}$ which has effect on the term in the brackets is that dissolving another bracket.

Since $\overline{\sigma(t_1, \ldots, t_{i-1}, s, t_{i+1}, \ldots, t_n)} = \overline{\sigma}(\overline{t_1}, \ldots, \overline{t_{i-1}}, \overline{s}, \overline{t_{i+1}}, \ldots, \overline{t_n}, \perp, \ldots, \perp)$, it follows that one can rewrite $\{\overline{\sigma(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n)}\}$ to one of this two: $\{\overline{\sigma}(\overline{t_1}, \ldots, \overline{t_{i-1}}, \overline{t}, \overline{t_{i+1}}, \ldots, \overline{t_n}, \perp, \ldots, \perp)\}$ or $\{\overline{\sigma}(\overline{t_1}, \ldots, \overline{t_{i-1}}, \{\overline{t}\}, \overline{t_{i+1}}, \ldots, \overline{t_n}, \perp, \ldots, \perp)\}$. In the first case our proof is complete, in the second we just need to apply the rule for propagating $\{\cdot\}$ up and then the one for dissolving one $\{\cdot\}$ at the top.

For the converse, let us do an induction on k, the length of the sequence $\{\overline{s}\} \rightarrow_{\overline{\mathcal{R}}}^{k} \{\overline{t}\}$. If $k = 0$ then obviously, $s = t$. Suppose now that $k \neq 0$ and consider s' such that $\{\overline{s}\} \rightarrow_{\overline{\mathcal{R}}} s' \rightarrow_{\overline{\mathcal{R}}}^{k-1} \{\overline{t}\}$. The first rule applied must be either corresponding to an unconditional rule, or of the form $\overline{\rho}_{\sigma,i}$. If the rule applied is corresponding to an unconditional one, then we can also apply it to s to obtain $s \rightarrow_{\mathcal{R}} \widehat{s'}$. Also, since it is a proof sequence, we have that $s' \rightarrow_{\overline{\mathcal{R}}}^{k'} \{\overline{\widehat{s'}}\} \rightarrow_{\overline{\mathcal{R}}}^{k-k'-1} \{\overline{t}\}$, thus we can apply the induction hypothesis for $\{\overline{\widehat{s'}}\} \rightarrow_{\overline{\mathcal{R}}}^{k-k'-1} \{\overline{t}\}$ and get that $\widehat{s'} \rightarrow_{\mathcal{R}}^{*} t$. If it is corresponding to $\rho_{\sigma,i} : l \rightarrow r$ *if* $cl \rightarrow cr$ and is applied at position 1α using substitution θ' then it must also be that $s|_\alpha = \widehat{\theta'}(l)$. Also, $s'|_{1\alpha c_{\sigma,i}} = \{\theta'(\overline{cl})\}$. Since $\{\overline{s}\} \rightarrow_{\overline{\mathcal{R}}}^{*} \{\overline{t}\}$ is a proof sequence,

we can infer that we can extract a subsequence $s' \to_{\overline{\mathcal{R}}}^{k'} s'[1\alpha c_{\sigma,i} \leftarrow \{cr\}] \to_{\overline{\mathcal{R}}} s'' \to_{\overline{\mathcal{R}}}^{k''} \{\overline{\widehat{s''}}\} \to_{\overline{\mathcal{R}}}^{k-k'-k''-1} \{\bar{t}\}$. This further says that $\{\theta'(\overline{cl})\} \to_{\overline{\mathcal{R}}}^{k'} \{cr\}$ is a proof rewrite sequence and since $k' < k$ we can apply the induction hypothesis and obtain that $\widehat{\theta'}(cl) \to_{\mathcal{R}}^{*} cr$ whence we can apply the rule $\rho_{\sigma,i}$ and obtain $s \to_{\mathcal{R}} \widehat{s''}$. Also we can apply the induction hypothesis for $\{\overline{\widehat{s''}}\} \to_{\overline{\mathcal{R}}}^{k-k'-k''-1} \{\bar{t}\}$ and get that $\widehat{s''} \to_{\mathcal{R}}^{*} t$. □

Although it may not seem so, $\{\bar{s}\} \to_{\overline{\mathcal{R}}}^{*} \{\bar{t}\}$ does not generally imply that $s \to_{\mathcal{R}}^{*} t$:

Example 4 Consider the transformation for $\mathcal{R}_s$ from Example 1:

$$
\begin{array}{lll}
A \to \{h(f(a,\bot),f(b,\bot))\} & f(\{x\},y) \to \{f(x,\bot)\} & a \to \{c\} \\
h(x,x) \to \{g(x,x,f(k,\bot))\} & h(\{x\},y) \to \{h(x,y)\} & a \to \{d\} \\
g(d,x,x) \to \{B\} & h(x,\{y\}) \to \{h(x,y)\} & b \to \{c\} \\
f(x,\bot) \to f(x,\{x\}) & g(\{x\},y,z) \to \{g(x,y,z)\} & b \to \{d\} \\
f(x,\{e\}) \to \{x\} & g(x,\{y\},z) \to \{g(x,y,z)\} & c \to \{e\} \\
\{\{x\}\} \to \{x\} & g(x,y,\{z\}) \to \{g(x,y,z)\} & c \to \{l\}
\end{array}
\qquad
\begin{array}{l}
k \to \{l\} \\
k \to \{m\} \\
d \to \{m\}
\end{array}
$$

Then the following rewrite sequence can be obtained in $\overline{\mathcal{R}}$:

$$
\begin{array}{ll}
\{A\} & \to_{\overline{\mathcal{R}}}^{*} \{h(f(a,\bot),f(b,\bot)\} \to_{\overline{\mathcal{R}}}^{*} \{h(f(\{d\},\{c\}),f(b,\bot))\} \\
& \to_{\overline{\mathcal{R}}}^{*} \{h(f(\{d\},\{c\}),f(\{d\},\{c\}))\} \\
& \to_{\overline{\mathcal{R}}}^{*} \{g(f(\{d\},\{c\}),f(\{d\},\{c\}),f(k,\bot))\} \\
& \to_{\overline{\mathcal{R}}}^{*} \{g(f(\{d\},\{e\}),f(\{d\},\{c\}),f(k,\bot))\} \\
& \to_{\overline{\mathcal{R}}}^{*} \{g(d,f(\{d\},\{c\}),f(k,\bot))\} \to_{\overline{\mathcal{R}}}^{*} \{g(d,f(\{m\},\{l\}),f(k,\bot))\} \\
& \to_{\overline{\mathcal{R}}}^{*} \{g(d,f(\{m\},\{l\}),f(\{m\},\{l\}))\} \\
& \to_{\overline{\mathcal{R}}}^{*} \{B\},
\end{array}
$$

but it is not the case that $A \to_{\mathcal{R}}^{*} B$. □

Even though Theorem 1 is too weak to give us a procedure in $\overline{\mathcal{R}}$ to test reachability in $\mathcal{R}$, it still gives us a technique to test whether a term t is *not* reachable from a term s in $\mathcal{R}$: if it is not true that $\{\bar{s}\} \to_{\overline{\mathcal{R}}}^{*} \{\bar{t}\}$ then it is also not true that $s \to_{\mathcal{R}}^{*} t$. Of course, in order for this to work, the set of terms reachable from $\{\bar{s}\}$ must be finite. This does not give us much, but it is the most we can get without additional restrictions on $\mathcal{R}$.

Soundness means that any rewrite in $\overline{\mathcal{R}}$ of a $\overline{\Sigma}$-term of the form $\{\bar{s}\}$ where s is a Σ-term, corresponds to a rewrite of s in $\mathcal{R}$. Unfortunately, as shown by Example 4, this result does not hold without restricting $\mathcal{R}$. We show that ground confluence *or* left linearity of $\mathcal{R}$ suffices.

Theorem 2 *If $\mathcal{R}$ is ground confluent and s', t' are reachable terms such that $s' \to_{\overline{\mathcal{R}}}^{*} t'$, then $\widehat{s'} \to_{\mathcal{R}}^{*} \widehat{t'}$. Moreover, if $s' \to_{\overline{\mathcal{R}}}^{*} t'$ has k structural steps, then $\widehat{s'} \to^{k} \widehat{t'}$.*

Proof: Since $\to_{\overline{\mathcal{R}}}^{*} = \Rightarrow_{\overline{\mathcal{R}}}^{*}$ and $\to_{\mathcal{R}}^{*} = \Rightarrow_{\mathcal{R}}^{*}$ we can change the affirmation in the proposition the following way:

Let s', t' be reachable terms such that $s' \Rightarrow^*_{\overline{\mathcal{R}}} t'$. Then $\widehat{s'} \Rightarrow^*_{\mathcal{R}} \widehat{t'}$.

Since s' is reachable there is some Σ-term s such that $\{\overline{s}\} \rightarrow^*_{\overline{\mathcal{R}}} \{s'\}$ equivalent to $\{\overline{s}\} \Rightarrow^*_{\overline{\mathcal{R}}} \{s'\}$. We will prove that for any Σ-term s and any $\overline{\Sigma}$ terms s' and t' such that $\{\overline{s}\} \Rightarrow^p_{\overline{\mathcal{R}}} \{s'\} \Rightarrow^k_{\overline{\mathcal{R}}} \{t'\}$ we have that $\widehat{s'} \Rightarrow^*_{\mathcal{R}} \widehat{t'}$, by induction on $m = p + k$.

If $k = 0$ then $s' = t'$ and there is nothing more to prove. If $k \neq 0$ let s'_t be a $\overline{\Sigma}$-term such that $s' \Rightarrow^{k-1}_{\overline{\mathcal{R}}} s'_t \Rightarrow_{\overline{\mathcal{R}}} t'$. We then can apply the induction hypothesis for s, s' and s'_t and get that $\widehat{s'} \Rightarrow^*_{\mathcal{R}} \widehat{s'_t}$. It suffices now to show that $\widehat{s'_t} \Rightarrow^*_{\mathcal{R}} \widehat{t'}$.

If $\widehat{s'_t} = \widehat{t'}$ then our proof is done.If $\widehat{s'_t} \neq \widehat{t'}$ then it must be the case that was applied either a rule of the form $\widetilde{l} \rightarrow \{\overline{r}\}$ or one of the form $\overline{\rho}'_{\sigma,i} : \widetilde{l}[c_{\sigma,i} \leftarrow \{cr\}] \rightarrow \{\overline{r}\}$ using a substitution θ and a multi-context c having at least one variable in a structural position. If a rule of the first type was applied, $l \rightarrow r$ can also be applied to $\widehat{s'_t}$ using the context $\widehat{c}$ and substitution $\widehat{\theta}$ to obtain $\widehat{t'}$.

Suppose now that a rule $\overline{\rho}'_{\sigma,i}$ was applied, at position α using the substitution θ'. Applying Lemma 1 we obtain the substitution θ_0 such that $\theta_0(\{\overline{cl}\}) \Rightarrow_{\overline{\mathcal{R}}} **\{cr\}$ verifies the induction hypothesis, whence $\widehat{\theta_0}(cl) \rightarrow^*_{\mathcal{R}} cr$. Also, $\theta_0(x) \Rightarrow^*_{\overline{\mathcal{R}}} \theta'(x)$ satisfies the induction hypothesis for each variable x of l, whence $\widehat{\theta_0}(x) \rightarrow^*_{\mathcal{R}} \widehat{\theta'}(x)$. But this implies that $\widehat{\theta_0}(cl) \rightarrow^*_{\mathcal{R}} \widehat{\theta'}(cl)$ and using the confluence and the fact that cr is a normal form we get that $\widehat{\theta'}(cl) \rightarrow cr$ whence we can apply rule $\rho_{\sigma,i}$ on $\widehat{s'}$ using substitution $\widehat{\theta'}$ to get $\widehat{t'}$. □

The above claim may not hold if the original CTRS is not ground confluent:

Example 5 Consider the following CTRS and its corresponding TRS:

$$(\mathcal{R}) \begin{cases} a \rightarrow \mathit{true} \\ a \rightarrow \mathit{false} \\ f(x) \rightarrow \mathit{true} \ \mathit{if}\ x \rightarrow \mathit{true} \end{cases} \qquad (\overline{\mathcal{R}}) \begin{cases} \overline{a} \rightarrow \{\overline{\mathit{true}}\} & \overline{a} \rightarrow \{\overline{\mathit{false}}\} \\ \overline{f}(x, \bot) \rightarrow \overline{f}(x, \{x\}) & \overline{f}(x, \{\overline{\mathit{true}}\}) \rightarrow \{\overline{\mathit{true}}\} \\ \overline{f}(\{x\}, y) \rightarrow \{\overline{f}(x, \bot)\} & \{\{x\}\} \rightarrow \{x\} \end{cases}$$

The following sequence is valid in $\overline{\mathcal{R}}$, but it is *not* the case that $f(\mathit{false}) \rightarrow_{\mathcal{R}} \mathit{true}$:

$$\{\overline{f}(\overline{a}, \bot)\} \rightarrow_{\overline{\mathcal{R}}} \{\overline{f}(\overline{a}, \{\overline{a}\})\} \rightarrow_{\overline{\mathcal{R}}} \{\overline{f}(\{\overline{\mathit{false}}\}, \{\overline{a}\})\} \rightarrow_{\overline{\mathcal{R}}} \{\overline{f}(\{\overline{\mathit{false}}\}, \{\overline{\mathit{true}}\})\} \rightarrow^+_{\overline{\mathcal{R}}} \{\overline{\mathit{true}}\}$$

□

The next result shows that our transformation is also sound when the original CTRS is left linear instead of ground confluent:

Proposition 4 *$\overline{\mathcal{R}}$ is left linear iff $\mathcal{R}$ is.*

Proof: Obvious from the transformation (since the conditions' rhs have no variables) □

Proposition 5 *If $\{\overline{s}\} \rightarrow^*_{\overline{\mathcal{R}}} t'$ is a safe sequence having k structural steps, then $s \rightarrow^k_{\mathcal{R}} \widehat{t'}$.*

Proof: Applying Proposition 2 we get a proof sequence $\{\overline{s}\} \rightarrow^*_{\overline{\mathcal{R}}} \overline{\widehat{t'}}$ on which we can apply Theorem 1.

Lemma 2 *If $\overline{\mathcal{R}}$ is left linear and $\{\overline{s}\} \rightarrow^*_{\overline{\mathcal{R}}} t'$, then there exists a safe rewrite sequence $\{\overline{s}\} \rightarrow^*_{\overline{\mathcal{R}}} t'$ having at least the same number of structural steps.*

Proof: The idea is to replay the rewritings from the transformed system in the original one. What can happen is that while a condition is being evaluated, rewriting is permitted in the transformed system but not in the original one. To resyncronize, all one has to do is that whenever a condition that will eventually be successfully solved is started, to record the rewriting steps occurring in variable subterms and only replay them after the condition was solved and the rule applied. The term on which condition has been started might be multiplied by using non-right-linear rules but for each of the copies we know exactly if the condition will be fullfill or not. For those copies for which we know that it will be not rewritten, we can at this point apply the recording of rewritings in variables.

The fact that the rewriting process continues as before is guaranteed by the left-linearity of $\overline{\mathcal{R}}$ - there is no need of synchronizing subterms; the rules will match whatever subterms. □

Theorem 3 *If $\overline{\mathcal{R}}$ is left linear and $\overline{s} \rightarrow^*_{\overline{\mathcal{R}}} t'$ then $s \rightarrow^*_{\mathcal{R}} \widehat{t'}$. Moreover, if $\overline{s} \rightarrow^*_{\overline{\mathcal{R}}} t'$ has k structural steps, then $s \rightarrow^{k'}_{\mathcal{R}} \widehat{t'}$ with $k' \geq k$.*

Proof: Apply Lemma 2, then Proposition 5. □

Thus, our transformation is sound for Example 5. However, Example 4 shows that soundness may not hold if $\mathcal{R}$ is neither ground confluent nor left linear.

Corollary 1 *If $\mathcal{R}$ is ground confluent* ***or*** *left linear, then our transformation is sound and complete, i.e., $s \rightarrow^*_{\mathcal{R}} t$ iff $\{\overline{s}\} \rightarrow^*_{\overline{\mathcal{R}}} \{\overline{t}\}$ for any $s, t \in T_\Sigma$.*

Therefore, we can semi-decide reachability, $s \rightarrow^*_{\mathcal{R}}$, in a ground confluent or left linear CTRS: (1) transform $\mathcal{R}$ to the TRS $\overline{\mathcal{R}}$; (2) do a breadth-first search in $\overline{\mathcal{R}}$ starting with $\{\overline{s}\}$; (3) if $\{\overline{t}\}$ is reached then return true. The breadth-first search may loop forever if there is no solution for the original problem. However, it will return true *iff* the original problem has a solution. This important reachability result is operationally important, since searching is very difficult in CTRSs and it can sometimes lead to defectious implementations.

Example 6 Consider the following three-rule CTRS: $a \rightarrow c$ *if* $a \rightarrow b$, $a \rightarrow b$ and $c \rightarrow b$. A rewrite engine sensitive to the order in which rules are given, such as Maude, may crash (an indeed, it does so if the rules are give in the order above) when asked to verify $a \rightarrow^*_{\mathcal{R}} c$. The reason is that although Maude does breadth-first search in general, it chooses not to do it within conditions. □

This CTRS is transformed to: $a(\bot) \to a(\{a(\bot)\})$, $c \to \{b\}$, $a(\{b\}) \to \{c\}$, $a(x) \to \{b\}$, $\{\{x\}\} \to \{x\}$. Although this TRS does not terminate either, we can use any rewrite engine which supports breadth-first searching, including Maude, to verify any reachability problem which has solutions in the original system.

5.3 Simulation and confluence

The next result shows that if $\mathcal{R}$ is left-linear, due to soundness and completeness, ground confluence of $\overline{\mathcal{R}}$ on reachable terms yield confluence of $\mathcal{R}$.

Proposition 6 *If $\overline{\mathcal{R}}$ is left linear and ground confluent on reachable terms, then $\mathcal{R}$ is ground confluent.*

Proof: Consider ground Σ-terms s, u, v such that $s \to^*_{\mathcal{R}} u$ and $s \to^*_{\mathcal{R}} v$. Then we have that $\{\overline{s}\} \to^*_{\overline{\mathcal{R}}} \{\overline{u}\}$ and $\{\overline{s}\} \to^*_{\overline{\mathcal{R}}} \{\overline{v}\}$. Since $\overline{\mathcal{R}}$ is confluent on reachable terms, there exists t' such that $\{\overline{u}\} \to^*_{\overline{\mathcal{R}}} t'$ and $\{\overline{v}\} \to^*_{\overline{\mathcal{R}}} t'$. By Proposition 4 and Theorem 3 we obtain that $u \to^*_{\mathcal{R}} \widehat{t'}$ and $v \to^*_{\mathcal{R}} \widehat{t'}$. □

Even though a transformation is sound and complete, one cannot simulate $\mathcal{R}$ through $\overline{\mathcal{R}}$. We show that if $\mathcal{R}$ is ground confluent and left linear then $\widehat{\cdot}$ defines a simulation relation between $\overline{\mathcal{R}}$ and $\mathcal{R}$:

Proposition 7 ***(Simulation)*** *If $\mathcal{R}$ is ground confluent and $\overline{\mathcal{R}}$ is left linear then $\overline{\mathcal{R}}^*$ restricted to reachable terms weakly simulates $\mathcal{R}^*$, that is,*
for any reachable term s' such that $\widehat{s'} \to^*_{\mathcal{R}} t$, there exists a $\overline{\Sigma}$-term t' such that $\widehat{t'} = t$ and $s' \to^*_{\overline{\mathcal{R}}} t'$. Also, if $\widehat{s'} \to_{\mathcal{R}} t$, t' can be chosen such that $s' = s'_0 \to_{\overline{\mathcal{R}}} \ldots \to_{\overline{\mathcal{R}}} s'_n \to_{\overline{\mathcal{R}}} t'_0 \to_{\overline{\mathcal{R}}} \ldots \to_{\overline{\mathcal{R}}} t'_n = t'$ with $\widehat{s'_i} = \widehat{s'}$ and $\widehat{t'_i} = \widehat{t'}$.
Moreover, $cond(t') \subseteq cond(s')$.

Proof: Induction on the size $|cond(s')|$ of the set of started conditions in s'. If s' has no started conditions, we can apply Theorem 1.

If $\widehat{s'} = t$ there is nothing to prove. Suppose now that the $\widehat{s'}$ rewrites in one step to t. If it is a non-conditional one, we can also apply it to s'. Else, we have two cases: (1) if the evaluation of the corresponding condition has been started for the rule applied in s' and (2) if it has not.

1. In the first case, we can find $s_\bot$ as in Lemma 1 and we will have that $\theta_0(cl) \to_{\overline{\mathcal{R}}} \theta'(C)$ and $\theta_0(()x) \to_{\overline{\mathcal{R}}} \theta'(x)$, whence $\theta_0(cl) \to_{\overline{\mathcal{R}}} \theta'(cl)$. Applying Theorem 2 we get that $\widehat{\theta_0(cl)} \to_{\mathcal{R}} \widehat{\theta'(C)}$ and $\widehat{\theta_0(cl)} \to_{\mathcal{R}} \widehat{\theta'(cl)}$. Also, we have that $\widehat{\theta'(cl)} \to^*_{\mathcal{R}} cr$ whence, using confluence, it must be that $\widehat{\theta'(C)} \to_{\mathcal{R}} cr$. Since set of conditions for $\theta'(C)$ is strictly included in the one for s', we can apply the induction hypothesis on it and obtain that $\theta'(C) \to_{\mathcal{R}} \{cr\}$, whence it follows that the conditional rule can be potentially applied on s' to obtain t' with the desired properties.

2. Suppose now that the condition evaluation has not been started. Then we can start it now and replay in $\overline{\mathcal{R}}$ the rewriting process in $\mathcal{R}$ for satisfying

the condition. As long as the conditions which need to be satisfied are not already started for the given term we can start them. However when we need to fulfill an already started condition, we use the same reasoning as for the first case and apply the induction hypothesis, since the condition is the same as the one from s', thus has less conditions started than s'.

We can see that the set of started conditions after any of the previous steps is included in the set of conditions of s'. Thus, if the number of rewrite steps from $\widehat{s'}$ to t is bigger than 1 we can iterate the above reasoning.

Note that left linearity is needed here to assure that a matching that can be performed on a term in the original system can be "lifted" to one on a enriched term of the transformed one. Without left linearity, one would need first to join the terms corresponding to the same variable, by joining their started conditions. □

Although the result above is interesting result by itself, we will also use in the sequel one of its corallaries.

Corollary 2 *If s' is reachable and $\widehat{s'} \to^*_{\mathcal{R}} c$ where c is a constant which is not a lhs for any of the rules in $\mathcal{R}$, then $s' \to^*_{\overline{\mathcal{R}}} \{c\}$*

Proof: We can apply the Theorem 7 to obtain a term c' such that $\widehat{c'} = c$ and $s' \to^*_{\overline{\mathcal{R}}} c'$. Since c' is reachable it must be that it has brackets at top. also, since $\widehat{c'} = c$, c' can only consists on a bonded number of brackets over c. Then we can apply the idempotency for the brackets to obtain $\{c\}$. □

It is worthwhile noticing that the confluence of $\mathcal{R}$ does *not* imply the confluence of $\overline{\mathcal{R}}$, as the following (counter-)example shows.

Example 7 Consider the confluent CTRS $\mathcal{R}$ consisting of one rule, $f(x) \to x$ *if* $g(x) \to \mathit{false}$ and its corresponding TRS $\overline{\mathcal{R}}$: $\overline{f}(x, \bot) \to \overline{f}(x, \{\overline{g}(x)\})$, $\overline{f}(\{x\}, y) \to \{\overline{f}(x, \bot)\}$, $\overline{f}(x, \{\overline{\mathit{false}}\}) \to \{x\}$, $\{\{x\}\} \to \{x\}$. Then $\overline{f}(\{\overline{\mathit{false}}\}, \{\overline{\mathit{false}}\})$ rewrites in one step to $\{\overline{\mathit{false}}\}$, which is in normal form, and $\overline{f}(\{\overline{\mathit{false}}\}, \{\overline{\mathit{false}}\}) \to_{\overline{\mathcal{R}}} \{\overline{f}(\overline{\mathit{false}}, \bot)\} \to_{\overline{\mathcal{R}}} \{\overline{f}(\overline{\mathit{false}}, \{\overline{g}(\overline{\mathit{false}})\})\}$, which is also in normal form. Therefore, $\overline{\mathcal{R}}$ is not confluent. □

In fact, for computational equivalence purposes, $\overline{\mathcal{R}}$ does *not* need to be confluent. What is needed is its confluence on *reachable* terms. The next result shows that in the presence of linearity, (ground) confluence is preserved.

Theorem 4 *Suppose $\overline{\mathcal{R}}$ is left linear. If $\mathcal{R}$ is ground confluent then $\overline{\mathcal{R}}$ is ground confluent on reachable terms, or, even stronger,*
for any reachable terms s'_1, s'_2, If $\widehat{s'_1}$ and $\widehat{s'_2}$ are joinable in t then s'_1 and s'_2 are joinable in t' such that $\widehat{t'} = t$.

Proof: Induction on $|\mathit{cond}(s'_1)| + |\mathit{cond}(s'_2)|$. First, we apply Proposition 7 to obtain s''_1 and s''_2 such that $s'_j \to^*_{\overline{\mathcal{R}}} s''_j$, $\widehat{s''_j} = t$ and $\mathit{cond}(s''_j) \subseteq \mathit{cond}(s'_j)$ for $j = \overline{1,2}$. If $|\mathit{cond}(s''_1)| + |\mathit{cond}(s''_2)| = 0$, the problem is already solved. Suppose now that $|\mathit{cond}(s''_1)| + |\mathit{cond}(s''_2)| \neq 0$.

Induction on the number of conditional positions s_1'' and s_2'' differ on (they are the same on the structure). First, we can assume that no bracket propagating rules can be applied. Let $1\alpha c_{\sigma,i}$ be a conditional position such that $s_1''|_{\alpha c_{\sigma,i}} \neq s_2''|_{\alpha c_{\sigma,i}}$. If both conditions are started then we have that $|cond(s_1''|_{\alpha c_{\sigma,i}})| \leq |cond(s_1'')| - 1$ and $|cond(s_2''|_{\alpha c_{\sigma,i}})| \leq |cond(s_2'')| - 1$. Note that we can anyway consider that both conditions are started because, if only one of them is started, say for s_2'', then in one rewriting step we can start the other one too (the fact that there are no brackets implies that $s_2''|_\alpha$ matches $\widetilde{l}$ whence $s_1''|_\alpha$ must also). The only difference is that in this case we will have that $|cond(s_1''|_{\alpha c_{\sigma,i}})| \leq |cond(s_1')|$, i.e., the inequality is not strict anymore. However, summing the number of conditions fro both of them we still get a strict inequality, so if we can verify that their coresponding Σ-terms are joinable, we can apply the induction hypothesis.

For each $j = 1, 2$ do the following. Let θ^j be such that $\theta^j(\widetilde{(l)}) = s_j''|_\alpha$. Since $s_j''|_{\alpha c_{\sigma,i}} \neq \perp$ we can apply Lemma 1 to obtain that there exists a reachable substitution θ_0^j such that $\theta_0^j(\{\overline{cl}\}) \rightarrow_{\overline{\mathcal{R}}}^* s_j''|_{\alpha c_{\sigma,i}}$ and $\theta_0^j(x) \rightarrow_{\overline{\mathcal{R}}}^* \theta^j(x)$, whence $\theta_0^j(\{\overline{cl}\}) \rightarrow_{\overline{\mathcal{R}}}^* \theta^j(\{\overline{cl}\})$. Applying Theorem 2 we get that $\widehat{\theta_0^j(\{\overline{cl}\})} \rightarrow_{\mathcal{R}}^* \widehat{s_j''|_{\alpha c_{\sigma,i}}}$ and $\widehat{\theta_0^j(\{\overline{cl}\})} \rightarrow_{\overline{\mathcal{R}}}^* \widehat{\theta^j(\{\overline{cl}\})}$.

Now, noticing that $\widehat{\theta^1(\{\overline{cl}\})} = \widehat{\theta^2(\{\overline{cl}\})}$ and applying confluence three times we obtain that $\widehat{s_1''|_{\alpha c_{\sigma,i}}}$ and $\widehat{s_2''|_{\alpha c_{\sigma,i}}}$ are joinable. We can now apply the induction hypothesis to obtain that $s_1''|_{\alpha c_{\sigma,i}}$ and $s_2''|_{\alpha c_{\sigma,i}}$ are also joinable.

Note that since we are joining conditions in a top-bottom manner, whenever I have to join two conditional all of the conditional above are already joined, so joining them does not affect the condition sets of the others not yet joined. □

5.4 Termination and Computational Equivalence

Next, we show that termination of $\overline{\mathcal{R}}$ on reachable terms implies operational termination of $\mathcal{R}$, and that the other implication does not hold without additional requirements on $\mathcal{R}$. Moreover, we show that confluence or left linearity of $\mathcal{R}$ suffices for the other implication to hold.

Proposition 8 *If $\overline{\mathcal{R}}$ terminates on $\{\overline{s}\}$, then $\mathcal{R}$ operationally terminates on s.*

Proof: Obvious, since, to any proof tree we can associate a proof sequence in $\overline{\mathcal{R}}$, each deduction rule in the proof tree corresponding to a rewriting step in $\overline{\mathcal{R}}$. □

The other implication does not hold without additional restrictions on $\mathcal{R}$.

Example 8 Consider the system $\mathcal{R}_t$ in Example 1. Since $\mathcal{R}_t = \mathcal{R}_s \cup \{B \rightarrow A\}$, its transformed version will be the same as the one in example 4, except adding one more rule, $B \rightarrow \{A\}$. Remember that, with the system $\overline{\mathcal{R}}$ in example 4 we have obtained that $\{A\} \rightarrow_{\overline{\mathcal{R}}}^* \{B\}$. With the new rule we therefore get that $\{A\} \rightarrow_{\overline{\mathcal{R}}}^+ \{A\}$, thus the transformed version is not terminating. However, the original system is decreasing, so it is operationally terminating. □

However, confluence *or* left-linearity of $\mathcal{R}$ preserves termination.

Theorem 5 *If $\mathcal{R}$ is confluent and operationally terminates on s, then $\overline{\mathcal{R}}$ terminates on $\{\overline{s}\}$.*

Proof: Suppose that $\overline{\mathcal{R}}$ is not terminating on $\overline{s}$, that is, that there is some infinite rewriting sequence starting with $\{\overline{s}\}$. We will show that we can build an infinite proof tree in $\mathcal{R}$.

For any non-terminating sequence $s'_0 \to_{\overline{\mathcal{R}}} s'_1 \to \overline{\mathcal{R}} \ldots \to_{\overline{\mathcal{R}}} s'_n \to_{\overline{\mathcal{R}}} \ldots$ it must be that there exists a minimal index k such that for all $n > k$, $\widehat{s'_n} = \widehat{s'_k}$. That is because otherwise we could use confluence to obtain, by Theorem 2, an operational non-terminating sequence in $\mathcal{R}$. Therefore, any nonterminating sequence eventually stabilizes on structure.

For a reachable term s' that can start a non-terminating sequence in $\overline{\mathcal{R}}$, let $k_{s'}$ be *the smallest* natural number at which some non-terminating sequence starting with s' stabilizes on structure after $k_{s'}$ rewrite steps in $\overline{\mathcal{R}}$. In other words, for any infinite sequence in $\overline{\mathcal{R}}$ starting with s' whose minimal index at which it stabilizes on structure is k, it is the case that $k_{s'} \leq k$, and, moreover, $k_{s'}$ is the largest with this property.

Consider prefix sequence $(s'_{0,n})_{0 \leq n \leq k_0}$ with $s'_{0,0} = \overline{s}$ and $k_0 = k_{\overline{s}}$. We then have that:

- all conditions which have been started are terminating; otherwise, once one of these sequences has been started we could have obtained a k lower k_0;
- since no further rewriting step can occur on the structure, it should be that a non terminating condition will be eventually started.

Consider a rewrite sequence $(s'_{1,n})_{k_0 \leq n \leq k_1}$ with $s'_{1,k_0} = s'_{0,k_0}$ and k_1 is minimal such that s'_{1,k_1} starts a non-terminating condition[2]. Consider now the proof tree associated to $s \to_{\mathcal{R}} \widehat{s'_{0,k_0}}$. Also, let s_1 be the term which could be obtained from $\widehat{s'_{1,k_1}}$ applying the rule corresponding to the non-terminating condition started.

Consider the (incomplete) proof tree for $s \to^+_{\mathcal{R}} s_1$ having the final rule a transitivity rule and above the line the tree for $s \to^+_{\mathcal{R}} \widehat{s'_{1,k_1}}$ and the tree having $\widehat{s'_{1,k_1}} \to_{\mathcal{R}} s_1$ as root and $\widehat{\theta}(cl) \to^+_{\mathcal{R}} cr$.

Iterating on the above construction and using minimality to guarantee that once a non-terminating condition has been started, no rewriting occurs elsewere but inside it, we can keep on expanding the proof tree so that it can grow arbitrary large. But this contradicts the operational-termination assumption.

□

Theorem 6 *If $\mathcal{R}$ is left-linear and operationally terminates on s, then $\overline{\mathcal{R}}$ terminates on $\{\overline{s}\}$.*

[2]One could potentially prove that $k_1 = k_0 + 1$ but, since this is not essential to our proof, we chose not to.

Proof: The proof proceeds along the same lines of for the confluence case, using Lemma 2 each time soundness is needed to ensure conditions of Theorem 3 are fulfilled. □

Finally, we prove that ground confluence yields computational equivalence:

Theorem 7 *(Computational equivalence) If $\mathcal{R}$ is finite, ground confluent and operationally terminates on s then $\overline{\mathcal{R}}$ is ground confluent and terminates on terms reachable from $\{\overline{s}\}$. That is, $\overline{\mathcal{R}}$ is computationally equivalent to $\mathcal{R}$.*

Proof: That $\overline{\mathcal{R}}$ terminates on $\{\overline{s}\}$ we know from Theorem 5. This implies that for each s' reachable from $\{\overline{s}\}$, the set of terms s' can be rewritten to is finite. Let $|s'|$ denote the number of elements in this set.

Let us now state the confluence this way: for any triple s', s'_1, s'_2 such that s' is reachable from $\{\overline{s}\}$, $s' \to^*_{\overline{\mathcal{R}}} s'_1$ and $s' \to^*_{\overline{\mathcal{R}}} s'_2$ we have that there exists t' such that $s'_1 \to^*_{\overline{\mathcal{R}}} t'$ and $s'_2 \to^*_{\overline{\mathcal{R}}} t'$. Moreover, if $\widehat{s'_1} = \widehat{s'_2}$ then t' can be chosen such that $\widehat{s'_1} = \widehat{s'_2} = \widehat{t'}$. We will prove this by induction on $|\{\overline{s}\}|$. Notice that for any s' reachable from $\{\overline{s}\}$ we have that (by soundness and completeness) $\{\widehat{\overline{s'}}\}$ is also reachable from $\{\overline{s}\}$, thus, $\overline{\mathcal{R}}$ terminates on $\{\widehat{\overline{s'}}\}$ and $\left|\{\widehat{\overline{s'}}\}\right| \leq |\{\overline{s}\}|$.

Consider s, s', s'_1, s'_2 as above. We can assume that no bracket rules can be applied to s'_1 and s'_2 (otherwise, we can apply them). By Soundness Theorem we have that $\widehat{s'} \to^*_{\mathcal{R}} \widehat{s'_1}$ and $\widehat{s'} \to^*_{\mathcal{R}} \widehat{s'_2}$ whence there exists t such that $\widehat{s'_1} \to^*_{\mathcal{R}} t$ and $\widehat{s'_2} \to^*_{\mathcal{R}} t$. Let's prove that t' can be chosen such that $\widehat{t'} = t$ by induction on the sum of rewriting steps for meeting $\widehat{s'_1}$ and $\widehat{s'_2}$.

The base case is $\widehat{s'_1} = \widehat{s'_2}$. Consider a conditional position $\alpha c_{\sigma,i}$ with minimal length such that $s'_1|_{\alpha c_{\sigma,i}} \neq s'_2|_{\alpha c_{\sigma,i}}$.

- Suppose first that neither of them is $\perp$. Using Soundness and Completeness theorems we get that $\{\overline{s}\} \to^*_{\overline{\mathcal{R}}} \{\widehat{\overline{s'_1}}\} \to_{\overline{\mathcal{R}}} \{\widehat{\overline{s'_1}}\}[\alpha c_{\sigma,i} \leftarrow \overline{\theta}(\{\overline{cl}\})]$ and also $\{\overline{\theta}(\{\overline{cl}\})\} \to^*_{\overline{\mathcal{R}}} \{\widehat{\overline{s'_1|_{\alpha c_{\sigma,i}}}}\}$. Applying Proposition 3 we also get that $\{\widehat{\overline{s'_1|_{\alpha c_{\sigma,i}}}}\} \to^*_{\overline{\mathcal{R}}} s'_1|_{\alpha c_{\sigma,i}}$, whence $\{\overline{\theta}(\{\overline{cl}\})\} \to^*_{\overline{\mathcal{R}}} s'_1|_{\alpha c_{\sigma,i}}$. Using the same reasoning for s'_2 we get that $\{\overline{\theta}(\{\overline{cl}\})\} \to^*_{\overline{\mathcal{R}}} s'_2|_{\alpha c_{\sigma,i}}$. Noticing that $\left|\{\overline{\theta}(\{\overline{cl}\})\}\right| < \left|\{\widehat{\overline{s'_1}}\}\right| \leq |\{\overline{s}\}|$ we can apply the induction hypothesis for $\theta(cl), \{\overline{\theta}(\{\overline{cl}\})\}, s'_1|_{\alpha c_{\sigma,i}}, s'_2|_{\alpha c_{\sigma,i}}$ to join the two conditions.

- If one of them, let's say $s'_1|_{\alpha c_{\sigma,i}}$ is $\perp$ then we again have two cases: either $\rho_{\sigma,i}$ is left-linear or it is not. If it is, then we can start the condition right away, and then proceed as above. If it is not, take for example to positions $\alpha\beta$ and $\alpha\gamma$ which should represent the same variable x. We have that (by Proposition 3) $\widehat{\overline{s'_1|_{\alpha\beta}}} \to^*_{\overline{\mathcal{R}}} s'_1|_{\alpha\beta}$ and $\widehat{\overline{s'_1|_{\alpha\gamma}}} \to^*_{\overline{\mathcal{R}}} s'_1|_{\alpha\gamma}$. Also, $\widehat{s'_1|_{\alpha\beta}} = \widehat{s'_1|_{\alpha\gamma}}$ and $\left|\{\widehat{\overline{s'_1|_{\alpha\beta}}}\}\right| < \left|\{\widehat{\overline{s'_1}}\}\right| \leq |\{\overline{s}\}|$ whence we can apply the induction hypothesis for $\widehat{s'_1|_{\alpha\beta}}, \{\widehat{\overline{s'_1|_{\alpha\beta}}}\}, \{s'_1|_{\alpha\beta}\}, \{s'_2|_{\alpha\gamma}\}$ to obtain a term

$\{s'_x\}$ such that $\{s'_1|_{\alpha\beta}\} \to^*_{\overline{\mathcal{R}}} \{s'_x\}$, $\{s'_1|_{\alpha\gamma}\} \to^*_{\overline{\mathcal{R}}} \{s'_x\}$ and $\widehat{s'_x} = \widehat{s'_1|_{\alpha\beta}}$. Since $\mathcal{R}$ is terminating, it follows that $s'_1|_{\alpha\beta} \to^*_{\overline{\mathcal{R}}} s'_x$ and $s'_1|_{\alpha\gamma} \to^*_{\overline{\mathcal{R}}} s'_x$. Using this iteratively, we can rewrite s'_1 to s''_1 such that all equal variables in l have corresponding equal subterms. This allows us to start the condition applying rule $\overline{\rho}_{\sigma,i}$ and then to proceed as above.

We can iterate this process until all conditions are joined and thus our affirmation holds for $\widehat{s'_1} = \widehat{s'_2}$.

Suppose now that $\widehat{s'_1} \neq \widehat{s'_2}$. Then, assume for simplicity that $\widehat{s'_1} \neq t$ (one of them should satisfy this) and consider s_1 such that $\widehat{s'_1} \to_{\overline{\mathcal{R}}} s_1 \to^*_{\overline{\mathcal{R}}} t$ and let ρ be the rule applied at position α using substitution θ. If ρ is linear, Then we can "lift" the substitution θ directly. If ρ is not left-linear we can use the result proven above to do the same thing. Let θ' be the substitution obtained.

Now, if ρ was unconditional, then we can apply the corresponding rule to s'_1 at position 1α using substitution θ'. Suppose now that the rule applied was conditional, say $\rho_{\sigma,i}$. First, we can assume that $s'_1|_{1\alpha c_{\sigma,i}} \neq \bot$ (otherwise, we can now apply rule $\overline{\rho_{\sigma,i}}$ at position 1α using substitution θ'). Using Soundness and Completeness theorems we get that $\{\overline{s}\} \to^*_{\overline{\mathcal{R}}} \{\widehat{\overline{s'_1}}\} \to_{\overline{\mathcal{R}}} \{\widehat{\overline{s'_1}}\}[1\alpha c_{\sigma,i} \leftarrow \overline{\theta}(\{\overline{cl}\})]$ and also $\{\overline{\theta}(\{\overline{cl}\})\} \to^*_{\overline{\mathcal{R}}} \{\overline{\widehat{s'_1|_{1\alpha c_{\sigma,i}}}}\}$. Applying Proposition 3 we also get that $\{\overline{\widehat{s'_1|_{1\alpha c_{\sigma,i}}}}\} \to^*_{\overline{\mathcal{R}}} s'_1|_{1\alpha c_{\sigma,i}}$. Also we know that $\{\overline{\theta}(\{\overline{cl}\})\} \to^*_{\overline{\mathcal{R}}} \{cr\}$. Since $\left|\{\overline{\theta}(\{\overline{cl}\})\}\right| < |\{\overline{s}\}|$ (it is a subterm of a term reachable from $\{\overline{s}\}$ in more than one step), we ca apply the induction hypothesis for $\theta(cl), \{\overline{\theta}(\{\overline{cl}\})\}, s'_1|_{1\alpha c_{\sigma,i}}, \{cr\}$. Because $\{cr\}$ is normal form in $\overline{\mathcal{R}}$ it follows that $s'_1|_{1\alpha c_{\sigma,i}} \to^*_{\overline{\mathcal{R}}} \{cr\}$. This means that $s'_1 \to^*_{\overline{\mathcal{R}}} s'_1[1\alpha c_{\sigma,i} \leftarrow \{cr\}]$. We can now apply $\overline{\rho}'_{\sigma,i}$ to $s'_1[1\alpha c_{\sigma,i} \leftarrow \{cr\}]$ at position $i\alpha$ using substitution θ'. □

Then one can simulate reduction in a confluent CTRS $\mathcal{R}$ by using the transformed TRS $\overline{\mathcal{R}}$. Reducing a Σ-term t to its normal form in $\mathcal{R}$ can be done as follows: start reducing $\{\overline{t}\}$; if it does not terminate, there exists a way t might have not terminate in $\mathcal{R}$; if it terminates and $fn(\{\overline{t}\})$ is its normal form, then $\widehat{fn(\{\overline{t}\})}$ is the normal form of t in $\mathcal{R}$.

6 Transforming more complex CTRSs

It is shown in [DO90] that join CTRSs can be transformed into equivalent normal ones. This is done by adding one rule $equal?(x, x) \to true$ and translating condition $t{\downarrow}t'$ into $equal?(t, t') \to true$; multiple join conditions can be translated to one normal condition by adding an $\wedge$ operator and one rule $true \wedge true \to true$. Left-linearity of the original system is destroyed, but if the system is constructor-based, *equal?* can be defined on constructors and left linearity is preserved (see [ABH03], for example). A CTRS is *deterministic* if any rule $r_0 \to l_{n+1}$ *if* $l_1 \to r_1 \wedge \ldots \wedge l_n \to r_n$ satisfies $Var(l_j) \subseteq \bigcup_{k=0}^{j-1} Var(r_k)$. A rewrite engine supporting DCTRS solves conditions from left to right, accumulating the substitution. Usually, conditional rhs are restricted to strong normal

forms (strongly DCTRS) or to being constructor terms (syntactically DCTRS). To simplify presentation, let us assume that input systems are syntactically DCTRS. The conditional arguments now need to carry on the accumulated substitution; second, we now need to allow evaluation of multiple conditions for the same rule and we need to keep track which condition is currently under evaluation. Basically, each rule $\rho_{\sigma,i} : r_0 \to l_{n+1}$ *if* $l_1 \to r_1 \wedge \ldots \wedge l_n \to r_n$ is now transformed into the following $n+1$ rules (where $\mathcal{V}_j = \bigcup_{k=1}^{j-1} Var(r_j)$):

$$\widetilde{r_0}[c_{\sigma,i} \leftarrow \bot] \to \widetilde{r_0}[c_{\sigma,i} \leftarrow \delta_1(\{\overline{l_1}\}, \bot)] \qquad \widetilde{r_0}[c_{\sigma,i} \leftarrow \delta_n(\{r_n\}, \mathcal{V}_n)] \to \{\overline{l_{n+1}}\}$$
$$\widetilde{r_0}[c_{\sigma,i} \leftarrow \delta_j(\{r_j\}, \mathcal{V}_j)] \to \widetilde{r_0}[c_{\sigma,i} \leftarrow \delta_{j+1}(\{\overline{l_{j+1}}\}, \mathcal{V}_{j+1})], \text{ for } 1 \le j < n$$

Example 9 The following syntactically DCTRS (inspired form [Ohl02]):

$$\begin{array}{l}
split(x, y : L) \to \langle L_{<x}, y : L_{\ge x}\rangle \ \textit{if}\ split(x, L) \to \langle L_{<x}, L_{\ge x}\rangle \wedge x \le y \to true \\
split(x, y : L) \to \langle y : L_{<x}, L_{\ge x}\rangle \ \textit{if}\ split(x, L) \to \langle L_{<x}, L_{\ge x}\rangle \wedge x \le y \to false \\
split(x, nil) \to \langle nil, nil\rangle \qquad\qquad qsort(nil) \to nil \\
qsort(x : L) \to qsort(L_{<x}) : x : qsort(L_{\ge x}) \ \textit{if}\ split(x, L) \to \langle L_{<x}, L_{\ge x}\rangle
\end{array}$$

is transformed (using the optimization mentioned in Section 4) to the TRS

$$\begin{array}{l}
split(x, y : L, \bot) \to split(x, y : L, [\{split(x, L, \bot)\}, \bot]) \\
split(x, y : L, [\{\langle L_{<x}, L_{\ge x}\rangle\}, \bot]) \to split(x, y : L, [\{x \le y\}, (L_{<x}, L_{\ge x})]) \\
split(x, y : L, [\{true\}, (L_{<x}, L_{\ge x})]) \to \{\langle L_{<x}, y : L_{\ge x}\rangle\} \\
split(x, y : L, [\{false\}, (L_{<x}, L_{\ge x})]) \to \{\langle y : L_{<x}, L_{\ge x}\rangle\} \\
split(x, nil, C) \to \{\langle nil, nil\rangle\} \qquad\qquad qsort(nil, C) \to \{nil\} \\
qsort(x : L, \bot) \to qsort(x : L, [\{split(x, L, \bot)\}, \bot]) \\
qsort(x : L, [\{\langle L_{<x}, L_{\ge x}\rangle\}, \bot]) \to qsort(L_{<x}, \bot) : x : qsort(L_{\ge x}, \bot) \\
\{\{x\}\} \to \{x\} \qquad\qquad split(\{x\}, L, C) \to \{split(x, L, \bot)\}) \\
split(x, \{L\}, C) \to \{split(x, L, \bot)\}) \quad qsort(\{L\}, C) \to \{qsort(L, \bot)\}
\end{array}$$

□

Operational termination is now equivalent to quasi-decreasingness and the left linearity of $\overline{\mathcal{R}}$ is equivalent to the *semilinearity* [Mar97] of DCTRSs. The proofs of the results in Section 5 were engineered to also work for this extended transformation.

7 Discussion and Future Work

We presented a technique to eliminate conditional rules by replacing them with unconditional rules. The generated TRS is computationally equivalent with the original CRTS provided that the CTRS is ground confluent. Besides the theoretical results, we have also empirically shown that the proposed transformation may lead to the development of faster conditional rewrite engines. In the case of constructor-based CTRSs, the operation $\{\_\}$ is not needed, so our transformation becomes the same as the one in [ABH03]; thus, our theoretical results

imply that the transformation in [ABH03] preserves ground confluence, which is a stronger result than the one proved in [ABH03].

One should *not* use our transformation for equational theorem proving, because it is not sound for this purpose. Indeed, consider the equational variant of Example 7. Then one can deduce $\{\overline{false}\} = \overline{f}(\{\overline{false}\}, \{\overline{false}\}) = \{\overline{f}(\overline{false}, \bot)\}$ in the transformed specification, which has no counterpart in the original specification. However, if the original equational specification is terminating and confluent as a rewrite system, then one can use the transformed system to prove equalities in the original system by reducing them to their normal forms. We postulate here that, for obvious reasons, no transformation resembling ours can escape this fact. And by resembling we mean one that adds control arguments to the operations and use it to control the rewriting sequence. By obvious we mean that such a system cannot avoid having a term with undesirable values on those control arguments, i.e., non-reachable terms and from that undesirable equalities may be proved as shown in the above example.

Techniques to compact the generated TRS are worthwhile investigating in detail. Also, propagation rules for $\{\_\}$ can destroy useful partial reductions; can one adapt our transformation to restart only the conditions that are invalidated when a rewrite step occurred? We believe that confluence is preserved even in the absence of left linearity or termination but we have not been able to prove it.

We have not considered here rewrite systems *modulo equations*, such as associativity, commutativity and/or idempotency; extending our transformation to such CTRSs is expected to be a non-trivial task. While commutativity and idempotency can easily be eliminated by multiplicating the rules containing these kind of operations, delaing with associativity seems like highly non-trivial. As it is, this transformation could be used to correctly transforms those systems for which associative operations don't occur in the lhs of any conditional rule, or if they do occur, there is only one way to match the lhs at a given position. However, this is a rather strong assumption on the given rewriting system and one should consider some way to deal with the systems not satisfying it, as well.

References

[ABH03] Sergio Antoy, Bernd Brassel, and Michael Hanus. Conditional narrowing without conditions. In *PPDP'03*, pages 20–31. ACM Press, 2003.

[AGM90] Hitoshi Aida, Joseph A. Goguen, and José Meseguer. Compiling concurrent rewriting onto the rewrite rule machine. In *CTRS'90*, volume 516 of *Lecture Notes in Computer Science*, pages 320–332. Springer, 1990.

[BCD+00] Peter Borovanský, Horatiu Cirstea, Hubert Dubois, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, Christophe Ringeis-

sen, and Marian Vittek. ELAN *V 3.4 User Manual*. LORIA, Nancy (France), fourth edition, January 2000.

[BK86] Jan A. Bergstra and Jan Willem Klop. Conditional rewrite rules: Confluence and termination. *Journal of Computer and System Sciences*, 32(3):323–362, 1986.

[Bra99] Bernd Brassel. Bedingte narrowing-verfahren mit verzögerter auswertung. Master's thesis, RWTH Aachen, 1999. In german.

[CDE+03] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn L. Talcott. *Maude 2.0 Manual*, 2003. `http://maude.cs.uiuc.edu/manual`.

[DF98] Răzvan Diaconescu and Kokichi Futatsugi. *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*. World Scientific, 1998. AMAST Series in Computing, volume 6.

[DO90] Nachum Dershowitz and Mitsuhiro Okada. A rationale for conditional equational programming. *Journal of Theoretical Computer Science*, 75(1&2):111–138, 1990.

[DOS88] Nachum Dershowitz, Mitsuhiro Okada, and G. Sivakumar. Canonical conditional rewrite systems. In Ewing L. Lusk and Ross A. Overbeek, editors, *CADE'98*, volume 310 of *Lecture Notes in Computer Science*, pages 538–549. Springer, 1988.

[DP88] Nachum Dershowitz and David A. Plaisted. Equational programming. In J. E. Hayes, Donald Michie, and J. Richards, editors, *Machine Intelligence 11*, pages 21–56. Oxford University Press, 1988.

[GM87] Elio Giovannetti and Corrado Moiso. Notes on the elimination of conditions. In Stéphane Kaplan and Jean-Pierre Jouannaud, editors, *CTRS'87*, volume 308 of *Lecture Notes in Computer Science*, pages 91–97. Springer, 1987.

[GWM+00] Joseph Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. In *Software Engineering with OBJ: algebraic specification in action*, pages 3–167. Kluwer, 2000.

[Han94] Michael Hanus. The integration of functions into logic programming: From theory to practice. *The Journal of Logic Programming*, 19 & 20:583–628, 1994.

[Hin94] Claus Hintermeier. How to transform canonical decreasing CTRSs into equivalent canonical TRSs. In *CTRS'94*, volume 968 of *Lecture Notes in Computer Science*, pages 186–205. Springer, 1994.

[LMM05] Salvador Lucas, Claude Marché, and José Meseguer. Operational termination of conditional term rewriting systems. *Information Processing Letters*, 95(4):446–453, August 2005.

[Mar96] Massimo Marchiori. Unravelings and ultra-properties. In Michael Hanus and Mario Rodríguez-Artalejo, editors, *ALP'96*, volume 1139 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 1996.

[Mar97] Massimo Marchiori. On deterministic conditional rewriting. Computation Structures Group, Memo 405, MIT Laboratory for Computer Science, 1997.

[MH94] Aart Middeldorp and Erik Hamoen. Completness results for basic narrowing. *Appl. Algebra Eng. Commun. Comput.*, 5:213–253, 1994.

[NSS04] Naoki Nishida, Masahiko Sakai, and Toshiki Sakabe. On simulation-completeness of unraveling for conditional term rewriting systems. In *LA Symposium 2004 Summer*, volume 2004-7 of *LA Symposium*, pages 1–6, 2004.

[Ohl99] Enno Ohlebusch. Transforming conditional rewrite systems with extra variables into unconditional systems. In Harald Ganzinger, David A. McAllester, and Andrei Voronkov, editors, *LPAR'99*, volume 1705 of *Lecture Notes in Computer Science*, pages 111–130. Springer, 1999.

[Ohl02] Enno Ohlebusch. *Advanced Topics in Term Rewriting*. Springer, 2002.

[Ros04] Grigore Rosu. From conditional to unconditional rewriting. In *WADT'04*, volume 3423 of *Lecture Notes in Computer Science*, pages 218–233. Springer, 2004.

[Vir99] Patrick Viry. Elimination of conditions. *Journal of Symbolic Computation*, 28:381–401, September 1999.