Operational Confluence of Conditional Term Rewrite Systems*

Karl Gmeiner

UAS Technikum Wien, Vienna, Austria
gmeiner@technikum-wien.at

Abstract

In conditional term rewrite systems certain properties change their intuitive meaning when compared to unconditional rewriting. This is due to the fact that in rewrite steps, the evaluation of the conditions is left implicit. For instance, termination of a conditional term rewrite system does not imply the absence of infinitely many steps in the conditions. For this reason, the notion of operational termination has been introduced in the past that also considers the evaluation of conditions. This paper investigates the case of confluence in conditional term rewrite systems and introduces the notion of operational confluence, a confluence property that also implies confluence of the evaluations of conditions.

1 Introduction

Conditional term rewriting is an intuitive extension of term rewriting that plays an important role in e.g. functional-logic programming languages. In Conditional Term Rewrite Systems (CTRSs) the applicability of rules is bound to certain conditions that usually depend on the rewrite relation itself. Although this type of conditions is well-known from functional programming, it causes various problems when compared to unconditional rewriting.

In the case of termination, a CTRS may be terminating meaning that there are no infinite reduction sequences, yet there still might be infinite recursion. Hence, termination of a CTRS does not imply termination of the actual interpretation of the CTRS. Therefore, different termination properties have been introduced, like *effective termination* [5, 7], but most notably *operational termination* [4] that also covers termination of the interpretation of the CTRS.

Concerning confluence, many criteria of unconditional rewriting do not hold anymore. For instance, neither orthogonality nor the critical pair lemma imply confluence in general. Nonetheless, if we impose some syntactic restrictions we can still prove confluence via e.g. level-confluence or conditional critical pairs [3, 2, 8].

Yet, there is another aspect of confluence in conditional rewriting: Confluence implies that a term that is rewritten in multiple ways always yields the same result after some rewrite steps. Extending this notion to conditional rewriting and also applying it to the condition would mean that it should not matter what strategy we use to evaluate the conditions in confluent CTRSs, we still will always yield the same result. Unfortunately this is not the case in general as the following example shows:

$$\mathcal{R} = \left\{ \begin{array}{lll} a \to c & A \to C & B \to C \Leftarrow a = b \\ \downarrow \quad \downarrow & \downarrow & \\ b \to d & B & \end{array} \right\}$$

Equality here means reducibility. Since the condition is satisfied, the CTRS is confluent. Nonetheless, if we start rewriting from the left-hand side of the condition a, two rules can be

*The research is supported by FWF (Austrian Science Fund) project I963.

applied, yielding the reducts b and c. The latter reduct does not rewrite to b, hence a rewrite sequence starting with $a \to c$ yields a negative result for the condition, although the condition is satisfied. In the worst case we might have to investigate all possible rewrite sequences starting from the left-hand side of a condition, even though the CTRS is confluent.

Now, consider another CTRS:

$$\mathcal{R}' = \left\{ \begin{array}{lll} a \to c & A \to C & B \to D \Leftarrow a \to^* e \\ \downarrow \quad \downarrow & \downarrow \nearrow & \\ b \to d & B & \end{array} \right\}$$

This CTRS is also confluent because the condition $a \to^* e$ is not satisfiable. In this example, it does not matter whether a is reduced to b or c because in both cases the condition is unsatisfiable. Hence, this latter CTRS is not only confluent but satisfies a stronger confluence property that also considers conditional evaluations. This paper introduces *operational confluence* as notion for this stronger confluence property.

2 Preliminaries

We assume basic knowledge of terms and contexts and will use notions and notations similar to the ones in [7]. A conditional term rewrite system $\mathcal{R}$ consists of conditional rules. These rules are triples $\langle l, r, c\rangle$, usually denoted as $l \to r \Leftarrow c$, where l, r are terms and c is a conjunction of conditions. The underlying unconditional rewrite system is the set of the unconditional part of the conditional rules $\mathcal{R}_u = \{l \to r \mid l \to r \Leftarrow c \in \mathcal{R}\}$.

Depending on the relation used in the conditions we distinguish various types of CTRSs: Semi-equational CTRSs ($s \leftrightarrow^* t$), join CTRSs ($s \downarrow t$), oriented CTRSs ($s \to^* t$) and normal CTRSs ($s \to^* t$ and t is an irreducible ground term w.r.t. $\mathcal{R}_u$, denoted as $\to^!$ in the following). We can further classify CTRSs based on the distribution of (extra-)variables: 1-CTRSs ($\mathcal{V}ar(r, c) \subseteq \mathcal{V}ar(l)$), 2-CTRSs ($\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$), 3-CTRSs ($\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l, c)$) and 4-CTRSs (no restrictions). We will only consider oriented CTRSs. An oriented 3-rule $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_n \to^* t_n$ is deterministic if $\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(l, t_1, \ldots, t_{i-1})$ for all $i \in \{1, \ldots, n\}$. A 3-CTRS is a deterministic CTRS (DCTRS) if all its rules are deterministic. A CTRS $\mathcal{R}$ is strongly deterministic [2], if for all rules $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_n \to^* t_n$ and substitutions σ such that $x\sigma$ is irreducible w.r.t. $\mathcal{R}$, $t_i\sigma$ is also irreducible w.r.t. $\mathcal{R}$.

3 Operational Confluence

We first recall some definitions of [4]: Let G be a formula in some theory of some logic that is defined over some inference rules. A proof tree is either an open goal, G, or a derivation tree

$$\frac{T_1, \ldots, T_k}{G}(\Delta)$$

In the latter case, Δ is a derivation rule in the corresponding logic, the proof tree itself is an instance of Δ and $T_1, \ldots, T_n$ are proof trees themselves. We call the formula G the *head* of T, written $head(T)$. A proof tree T is closed if it does not contain any open goals.

We introduce the following notion of closeable proof trees:

Definition 1 (closeable proof trees)**.** *A proof tree T is closeable if it is a closed proof tree, or for all open goals U in T, there is a proof tree U' that is closed.*

The difference between closeable and closed proof trees is important if an inference rule can be applied infinitely many times.

Definition 2 (operational confluence)**.** *A theory $\mathcal{S}$ in a logic $\mathcal{L}$ defined by inference rules is operationally confluent for a formula F if for all instances $F\sigma$ of F, all proof trees T in $\mathcal{S}$ headed by $F\sigma$ the following holds:*

If there is a closed proof tree T with $head(T) = F\sigma$, then all proof trees T' with $head(T') = F\sigma$ are closeable.

To illustrate the previous definitions consider an inference system with rules (A) and (B), and a closed proof tree for some $F\sigma$:

$$\frac{\overline{T}(B) \qquad \overline{U}(B)}{F\sigma}(A)$$

If the given inference system is operationally confluent for F, we obtain a closed proof tree or a proof tree with open goals that is closeable, even if we apply a different inference rule (C) in one proof step.

$$\frac{\overline{T'}(B) \qquad \dfrac{\overline{V}(B)}{U'}(A)}{F\sigma}(C)$$

Hence, operational confluence implies that it does not matter which rule we apply, we will always yield a closed proof if the goal is proveable (up to non-termination in non-operationally terminating inference systems).

4 Operational Confluence of CTRSs

In this section we investigate the case of conditional rewriting for operational confluence. Lucas et. al. introduce inference rules in [4] for operational termination, yet the transitivity rule

$$\text{(Tran)} \qquad \frac{u \to u' \quad u' \to^* u''}{u \to^* u''}$$

cannot be used for operational confluence because u' may be an arbitrary term and thus immediately leads to non-confluence.[1] Therefore operational confluence for CTRSs will be defined using only two rules where *(Tran)*, *(Repl)* and *(Cong)* are merged into the single *(Appl)* rule which is additionally restricted to the satisfiability of the conditions of the applied rule.

$$\text{(Refl)} \qquad \frac{}{u \to^* u}$$

$$\text{(Appl)} \qquad \frac{c\sigma, u' \to^* v}{u \to^* v} \qquad \begin{array}{l}\text{where } l \to r \Leftarrow c \in \mathcal{R},\\ u = C[l\sigma],\ u' = C[r\sigma] \text{ and } c\sigma \text{ is satisfiable for } \mathcal{R}\end{array}$$

The *(Appl)* rule can only be applied if the condition of the rule is satisfied. It is undecidable in general whether a condition is satisfied, i.e., whether there is a closed proof tree, yet such an inference rule models the usual implementation of conditional rewrite engines to first verify the conditions and only then apply the rule.

[1] One of the anonymous referees pointed out this problem in the first submission of this paper.

Consider a conditional rule $l \to r \Leftarrow s \to^* t$ in an operationally confluent (for $\to^!$) CTRS. The normalform of a term can be obtained without traversing all possible rewrite paths from $s\sigma$. It is sufficient to determine one normalizing rewrite sequence $s\sigma \to^! u$ and test whether $u = t\sigma$ or not. In the latter case the condition is not satisfiable.

The following example resembles CTRSs that may be generated by the inversion procedure for term rewrite systems of [6] and shows that for some relevant cases confluence does not necessarily imply operational confluence for $\to^!$.

Example 3 (non-operationally confluent CTRS)**.** *Consider the following CTRS:*

$$\mathcal{R} = \left\{ \begin{array}{c} \mathit{half}(x) \to y \Leftarrow \mathit{pairs}(0,x) \to^* \mathit{pairs}(y,y) \\ \mathit{pairs}(x,s(y)) \to \mathit{pairs}(s(x),y) \end{array} \right\}$$

The CTRS is confluent, yet it is not operationally confluent for $\to^!$. Consider the rewrite sequence $\mathit{half}(s(s(0))) \to^* s(0)$. It gives rise to the following closed proof tree:

$$\dfrac{\dfrac{\dfrac{}{\mathit{pairs}(s(0),s(0)) \to^* \mathit{pairs}(s(0),s(0))}(\mathit{Refl})}{\mathit{pairs}(0,s(s(0))) \to^* \mathit{pairs}(s(0),s(0))}(\mathit{Appl})}{\mathit{half}(s(s(0))) \to^* s(0)}(\mathit{Appl})$$

Instead of the (Refl)*-inference rule also the* (Appl) *inference rule can be applied, leading to the following proof tree:*

$$\dfrac{\dfrac{\dfrac{\mathit{pairs}(0,s(s(0))) \to^* \mathit{pairs}(s(0),s(0))}{\mathit{pairs}(s(0),s(0)) \to^* \mathit{pairs}(s(0),s(0))}(\mathit{Appl})}{\mathit{pairs}(0,s(s(0))) \to^* \mathit{pairs}(s(0),s(0))}(\mathit{Appl})}{\mathit{half}(s(s(0))) \to^* s(0)}(\mathit{Appl})$$

The second proof tree contains one open goal $\mathit{pairs}(0,s(s(0))) \to^* \mathit{pairs}(s(0),s(0))$ to which no inference rule can be applied. Therefore, the second proof tree is not closeable so that the CTRS is not operationally confluent for $\to^!$.

In the previous condition, non-operational confluence is caused by an overlap of both inference rules. In normal 1-CTRSs such overlaps do not occur which leads to the following result:

Lemma 4 (operational confluence of normal 1-CTRSs)**.** *Every confluent normal 1-CTRS is also operationally confluent for $\to^!$.*

Proof Sketch. Let $u \to^* t$ be the head of a proof tree where t is an irreducible ground term w.r.t. the underlying TRS. If this condition is satisfiable, then there is a goal $t \to^* t$ in the proof tree such that the (Refl)-inference rule is applicable. Since t is irreducible w.r.t. underlying TRS, no other inference rule is applicable. Finally, since the CTRS is confluent, t is the unique normalform of u. □

A generalization of normal 1-CTRSs are strongly deterministic CTRSs. In this class of CTRSs, the right-hand sides of conditions are also irreducible, provided the image of the matcher only contains irreducible terms. Although this class of CTRSs plays an important role in proving confluence it is not sufficient for equivalence of operational confluence and confluence:

Example 5 (strongly deterministic CTRSs)**.** *Consider the CTRS $\mathcal{R}$ of Example 3. The CTRS is not strongly deterministic, but the union of $\mathcal{R}$ and $\{s(x) \to s(x)\}$ is, because every substitution that contains an s-rooted term in its image is not normalizing anymore. Hence, this (non-terminating) CTRS is strongly deterministic but not operationally confluent (for $\to^!$).*

5 Conclusion and Related Work

In this paper we introduced a new confluence property for conditional term rewrite systems, operational confluence. This property implies confluence but also adds the requirement that the evaluation of conditions does not give rise to different results depending on the evaluation strategy.

This notion is motivated from experiences in implementing rewrite engines for CTRSs because checking whether a condition is satisfied after each derivation step and using backtracking on dead ends adds significant overhead to such implementations. For operationally confluent CTRSs (for $\rightarrow^!$) it is sufficient to verify or disprove the condition for one evaluation branch. The notion of operational confluence hence better captures the intuitive meaning of confluence for conditional rewriting.

5.1 Relation to ultra-properties

In [5] Marchiori introduces a class of transformations from conditional term rewrite systems into unconditional ones. In the same paper, the notion of ultra-properties is introduced that defines properties of CTRSs using the transformed TRS. Informally, a CTRS $\mathcal{R}$ has the property ultra-$\mathcal{P}$ if the transformed system $\mathbb{U}(\mathcal{R})$ has the property $\mathcal{P}$.

It is plausible that there is a relation between ultra-confluence and operational confluence. Details to this are ongoing research. Furthermore, there are other transformations that better preserve confluence than unravelings (e.g. the transformations of [1]).

References

[1] Sergio Antoy, Bernd Brassel, and Michael Hanus. Conditional narrowing without conditions. In *Proc. 5th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, 27-29 August 2003, Uppsala, Sweden*, pages 20–31. ACM Press, 2003.

[2] Jürgen Avenhaus and Carlos Loría-Sáenz. On conditional rewrite systems with extra variables and deterministic logic programs. In Frank Pfenning, editor, *Proc. 5th Int. Conf. on Logic Programming and Automated Reasoning (LPAR'94), Kiev, Ukraine, July 16-22, 1994*, pages 215–229, 1994.

[3] Nachum Dershowitz, Mitsuhiro Okada, and G. Sivakumar. Confluence of conditional rewrite systems. In S. Kaplan and J.-P. Jouannaud, editors, *Proc. 1st Int. Workshop on Conditional Rewriting Systems (CTRS'87), Orsay, France, July 8-10, 1987*, volume 308 of *Lecture Notes in Computer Science*, pages 31–44. Springer-Verlag, 1988.

[4] Salvador Lucas, Claude Marché, and José Meseguer. Operational termination of conditional term rewriting systems. *Inf. Process. Lett.*, 95(4):446–453, 2005.

[5] Massimo Marchiori. Unravelings and ultra-properties. In Michael Hanus and Mario Mario Rodríguez-Artalejo, editors, *Proc. 5th Int. Conf. on Algebraic and Logic Programming, Aachen*, volume 1139 of *Lecture Notes in Computer Science*, pages 107–121. Springer, September 1996.

[6] Naoki Nishida, Masahiko Sakai, and Toshiki Sakabe. Partial inversion of constructor term rewriting systems. In Jürgen Giesl, editor, *Proc. 16th International Conference on Rewriting Techniques and Applications (RTA'05), Nara, Japan, April 19-21, 2005*, volume 3467 of *Lecture Notes in Computer Science*, pages 264–278. Springer, April 2005.

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

[8] Taro Suzuki, Aart Middeldorp, and Tetsuo Ida. Level-confluence of conditional rewrite systems with extra variables in right-hand sides. In Jieh Hsiang, editor, *Proc. 6th Int. Conf. on Rewriting Techniques and Applications (RTA'95), Kaiserslautern, Germany*, volume 914, pages 179–193, Kaiserslautern, Germany, April 1995. Springer-Verlag.