Proceedings of the

13th International Workshop on Confluence

July 9, 2024

Tallinn, Estonia

Foreword

The 13th International Workshop on Confluence (IWC 2024) is held on July 9, 2024, in Tallinn, Estonia, affiliated with the 9th International Conference on Formal Structures for Computation and Deduction (FSCD 2024).

Confluence, as a general notion of determinism, is an essential property of rewrite systems and has emerged as a crucial concept for many applications. However, the confluence property is also relevant to various further areas of rewriting, such as completion, commutation, termination, modularity, and complexity. The International Workshop on Confluence was created as a forum to discuss all these aspects, as well as related topics, implementation issues, and new applications.

IWC 2024 continues this tradition. The present report comprises nine regular submissions and the abstract of an invited talk, as well as descriptions of tools participating in the 13th Confluence Competition (CoCo 2024). In the invited talk, Aart Middeldorp highlights confluence of Logically Constrained Term Rewrite Systems, which has been attracting attention recently, and for which a new category of CoCo has started this year. The contributions in these proceedings reflect the wide scope of current research on confluence, ranging from new confluence criteria and novel confluence-related properties over formalization of confluence results to implementation aspects and applications. At the same time, the spectrum of rewrite formalisms (first- as well as higher-order, conditional rewriting, rewriting under strategies) used to model problems from different application areas underlines the importance of confluence for various domains.

The renewed interest in confluence research in the last decade resulted in a variety of novel approaches, which were also implemented in powerful tools that compete in the annual confluence competition. The second part of this report devoted to CoCo 2024 provides a general overview as well as system descriptions of all competition entrants.

IWC 2024 was made possible by the commitment of many people who contributed to the submissions, the preparation and the program of the workshop, as well as the confluence competition. These include authors of papers and tools, committee members, external reviewers, and the organizers of CoCo, the organizers of FSCD, as well as the local organizers. Their hard work is very much appreciated.

Cyrille Chenavier and Naoki Nishida — Limoges and Nagoya, 26 June 2024

Organization

IWC Steering Committee

- Mauricio Ayala-Rincón, Brasilia University, Brasil
- Sarah Winkler, Free University of Bozen-Bolzano, Italy

IWC Program Committee

- Sandra Alves, Universidade do Porto, Portugal
- Cyrille Chenavier, Université de Limoges, France (co-chair)
- Nao Hirokawa, JAIST, Japan
- Dohan Kim, University of Innsbruck, Austria
- Maja Hanne Kirkeby, Roskilde University, Denmark
- Salvador Lucas, Universitat Politècnica de València, Spain
- Naoki Nishida, Nagoya University, Japan (co-chair)
- Femke van Raamsdonk, Vrije Universiteit Amsterdam, Netherlands

Additional Reviewers

- Jörg Endrullis, Vrije Universiteit Amsterdam, Netherlands
- Jan Willem Klop, Vrije Universiteit Amsterdam, Netherlands
- Johannes Niederhauser, University of Innsbruck, Austria
- René Thiemann, University of Innsbruck, Austria

CoCo Steering Committee

- Raúl Gutiérrez, Universidad Politécnica de Madrid, Spain
- Aart Middeldorp, University of Innsbruck, Austria
- Naoki Nishida, Nagoya University, Japan
- Teppei Saito, JAIST, Japan
- René Thiemann, University of Innsbruck, Austria

Contents

Confluence of Logically Constrained Rewrite Systems

Aart Middeldorp

Department of Computer Science, University of Innsbruck, Austria

Logically constrained rewrite systems [1] are a natural extension of plain term rewrite systems with native support for constraints that are handled by SMT solvers. In this talk, which is based on joint work with Jonas Schöpf and Fabian Mitterwallner, we provide an overview of the confluence results for logically constrained rewrite systems reported in the literature [1–4].

References

[1] Cynthia Kop and Naoki Nishida. Term Rewriting with Logical Constraints. In *Proc. 9th International Symposium on Frontiers of Combining Systems*, volume 8152 of *LNCS (LNAI)*, pages 343–358, 2013. doi: 10.1007/978-3-642-40885-4_24.

[2] Jonas Schöpf and Aart Middeldorp. Confluence Criteria for Logically Constrained Rewrite Systems. In *Proc. 29th International Conference on Automated Deduction*, volume 14132 of *LNCS (LNAI)*, pages 474–490, 2023. doi: 10.1007/978-3-031-38499-8_27.

[3] Jonas Schöpf, Fabian Mitterwallner and Aart Middeldorp. Confluence of Logically Constrained Rewrite Systems Revisited. In *Proc. 12th International Joint Conference on Automated Reasoning*, *LNCS (LNAI)*, 2024. To appear.

[4] Sarah Winkler and Aart Middeldorp. Completion for Logically Constrained Rewriting. In *Proc. 3rd International Conference on Formal Structures for Computation and Deduction*, volume 108 of Leibniz International Proceedings in Informatics, pages 30:1–30:18, 2018. doi: 10.4230/LIPIcs.FSCD.2018.30.

On Proving Confluence of Concurrent Programs by All-Path Reachability of LCTRSs*

Misaki Kojima[1]†and Naoki Nishida[1]

Nagoya University, Nagoya, Japan
k-misaki@nagoya-u.jp nishida@i.nagoya-u.ac.jp

Abstract

A logically constrained term rewrite system (LCTRS, for short) modeling a concurrent program is usually non-terminating, left-linear, and overlapping, but sometimes confluent w.r.t. an initial ground term which corresponds to an initial configuration of the program. In this paper, we show how to prove confluence of such an LCTRS w.r.t. the initial term. To be more precise, we add the rule from the initial term to a fresh constant to the LCTRS, and construct a proof tree for the all-path reachability problem from the initial term to the constant. If the directed graph obtained from the proof tree by considering it as a cyclic proof is strongly connected and each node corresponds to a single term, then all terms reachable from the initial term can be reduced back to the initial one and thus the original LCTRS is confluent w.r.t. the initial one.

1 Introduction

Recently, approaches to program verification by means of *logically constrained term rewrite systems* (LCTRSs, for short) [12] are well investigated [5, 15, 4, 13, 6, 7, 11, 8, 9, 3, 10]. LCTRSs are extensions of *term rewrite systems* (TRSs, for short) by allowing rewrite rules to have guard constraints which are evaluated under built-in theories. A constrained rewrite rule is applied to a term if the matching substitution satisfies the guard constraint of the rule. LCTRSs are reasonable and useful as computation models of not only functional but also imperative programs. For instance, equivalence checking by means of LCTRSs is useful to ensure the correctness of terminating functions (cf. [5]). Transformations [5, 6] of sequential programs into LCTRSs have been extended to concurrent programs with semaphore-based exclusive control [11], and the extended transformation is applicable to *asynchronous integer transition systems with shared variables* (AITS, for short) [8].

The LCTRS obtained from a concurrent program by the aforementioned transformation is usually non-terminating, left-linear, and overlapping, but sometimes confluent w.r.t. an initial ground term which corresponds to an initial configuration of the program (or an initial state of the transition system). Here, an LCTRS $\mathcal{R}$ is said to be *confluent w.r.t. a term t* if the ARS $(T, \rightarrow_{\mathcal{R}} \cap (T \times T))$ is confluent, where T is the set of reachable terms from t w.r.t. $\rightarrow_{\mathcal{R}}$.

Let us consider the program graph in Figure 1 for *Peterson's mutual exclusion algorithm* [1, Example 2.25]. Two processes P_0, P_1 share Boolean variables b_0, b_1 and an integer variable x; b_i indicates that P_i wants to enter the critical section crit_i; x stores the identifier—0 or 1—of the process that has priority for the critical section at that time. The LCTRS $\mathcal{R}_1$ for the AITS consisting of PG_0 and PG_1 is illustrated in Figure 2.[1] The initial state $\langle \mathsf{noncrit}_0, \mathsf{noncrit}_1, \{b_0 \mapsto$

*This work was partially supported by JSPS KAKENHI Grant Number 24K02900 and Grant-in-Aid for JSPS Fellows Grant Number 24KJ1240.

†JSPS Research Fellow

[1]In the first rule of $\mathcal{R}_1$, we may replace b'_0 and x' by true and $\mathsf{1}$, respectively. However, for the correspondence with the program graphs PG_0, PG_1, we remain the label of the corresponding edge as the constraint "$b'_0 \wedge x'{=}1$" of the rule.

$$\begin{tikzcd}
{} \arrow[r] & \boxed{\mathsf{noncrit}_i} \arrow[r, "\langle b_i, x\rangle := \langle \mathsf{true}, 1-i\rangle"] & \boxed{\mathsf{wait}_i} \arrow[r, "x = i \lor \neg b_{1-i}"] & \boxed{\mathsf{crit}_i} \arrow[ll, bend left, "b_i := \mathsf{false}"]
\end{tikzcd}$$

Figure 1: Program graph PG_i $(i = 0, 1)$ for Peterson's mutual exclusion algorithm.

$$\mathcal{R}_1 = \left\{ \begin{array}{llllllll}
\mathsf{cnfg}(\mathsf{noncrit}_0, & p_1, & b_0, b_1, x) \to \mathsf{cnfg}(\mathsf{wait}_0, & p_1, & b_0', b_1, x') & [\ b_0' \land x' = 1\ ] \\
\mathsf{cnfg}(\mathsf{wait}_0, & p_1, & b_0, b_1, x) \to \mathsf{cnfg}(\mathsf{crit}_0, & p_1, & b_0, b_1, x\ ) & [x = 0 \lor \neg b_1] \\
\mathsf{cnfg}(\mathsf{crit}_0, & p_1, & b_0, b_1, x) \to \mathsf{cnfg}(\mathsf{noncrit}_0, & p_1, & b_0', b_1, x\ ) & [\quad \neg b_0' \quad] \\
\mathsf{cnfg}(p_0, & \mathsf{noncrit}_1, & b_0, b_1, x) \to \mathsf{cnfg}(p_0, & \mathsf{wait}_1, & b_0, b_1', x') & [\ b_1' \land x' = 0\ ] \\
\mathsf{cnfg}(p_0, & \mathsf{wait}_1, & b_0, b_1, x) \to \mathsf{cnfg}(p_0, & \mathsf{crit}_1, & b_0, b_1, x\ ) & [x = 1 \lor \neg b_0] \\
\mathsf{cnfg}(p_0, & \mathsf{crit}_1, & b_0, b_1, x) \to \mathsf{cnfg}(p_0, & \mathsf{noncrit}_1, & b_0, b_1', x\ ) & [\quad \neg b_1' \quad]
\end{array} \right\}$$

Figure 2: LCTRS $\mathcal{R}_1$ for the AITS consisting of PG_0 and PG_1 in Figure 1.

$\mathsf{false}, b_1 \mapsto \mathsf{false}, x \mapsto 0\}\rangle$ is represented by the term $\mathsf{cnfg}(\mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, 0)$. The LCTRS $\mathcal{R}_1$ is non-terminating, left-linear, and overlapping, and has non-trivial critical pairs. Note that $\mathcal{R}_1$ has no normal form that is rooted by cnfg and reachable from the initial term.

One of the simplest sufficient conditions for confluence of an LCTRS $\mathcal{R}$ w.r.t. an initial (ground) term is (ground) confluence of $\mathcal{R}$. However, it is not so easy to prove (ground) confluence of the above example because, e.g., $\mathcal{R}_1$ does not satisfy some well-known existing criteria—"(weak) orthogonality" [12] and "termination and joinability of critical pairs" [14]—for confluence of LCTRSs. In addition, non-termination often makes it difficult to prove joinability and its related properties (e.g., strong closedness) of critical pairs (see Example 6).

In this paper, for the LCTRS $\mathcal{R}$ obtained from a concurrent program, we show how to prove confluence of $\mathcal{R}$ w.r.t. an initial ground term s_0 which corresponds to an initial configuration of the program. To be more precise, we first add to $\mathcal{R}$ the rule from the initial term s_0 to a fresh constant init, obtaining the LCTRS $\mathcal{R}' = \mathcal{R} \cup \{s_0 \to \mathsf{init}\}$. Secondly, using a proof system for APR problems, we try to construct a proof tree for the *all-path reachability problem* (APR problem, for short) from the initial term to the constant w.r.t. $\mathcal{R}'$: $\{s_0\} \Rightarrow \{\mathsf{init}\}$. An APR problem of $\mathcal{R}'$ considered as a transition system is a pair $P \Rightarrow Q$ of state sets P, Q and is *demonically valid* (i.e., solved) if every finite *execution path* of $\mathcal{R}'$—a transition sequence starting with a state in P and ending with a terminating state—includes a state in Q [4]. Thirdly, we construct the directed graph from the proof tree by considering it as a *cyclic proof* [2] and by removing the nodes labeled with the $\mathsf{axiom}/\mathsf{c\text{-}subs}$ rule of the proof system. Finally, we show that if the directed graph is strongly connected and each node corresponds to a single term, then all terms reachable from the initial one can be reduced back to the initial one and thus the original LCTRS is confluent w.r.t. the initial one.

2 All-path Reachability Problems of LCTRSs

A *logically constrained term rewrite system* (LCTRS, for short) [12, 5] is a set $\mathcal{R}$ of constrained rewrite rules defined over two kinds of signatures, Σ_{theory} and Σ_{terms}, and a set $\mathcal{V}$ of variables. The theory signature Σ_{theory} equipped with interpretations $\mathcal{I}$ and $\mathcal{J}$ defines built-in objects which are either values in $\mathcal{V}al$ $(\subseteq \Sigma_{theory})$ or operators: For each theory sort ι, $\mathcal{I}$ specifies the universe of ι; $\mathcal{J}$ assigns an interpretation to each built-in operator, assigning to a ground term with sort ι an element in $\mathcal{I}(\iota)$. The non-theory signature Σ_{terms} is a set of function symbols for user-defined functions and constructors. Constrained rewrite rules in $\mathcal{R}$ specify the behavior of some function symbols in Σ_{terms}, and $\mathcal{R}$ implicitly includes rules for calculation symbols such as $+$: $x + y \to z\ [z = x + y]$.

We typically choose a theory-sort set $\mathcal{S}_{theory}$ and a theory signature Σ_{theory} such that $\mathcal{S}_{theory} \supseteq \{bool\}$, $\Sigma_{theory} \supseteq \Sigma^{core}_{theory} = \{\mathsf{true}, \mathsf{false} : bool,\ \wedge, \vee, \implies, \iff : bool \times bool \Rightarrow bool,\ \neg : bool \Rightarrow bool\} \cup \{=_\iota, \neq_\iota : \iota \times \iota \Rightarrow bool \mid \iota \in \mathcal{S}_{theory}\}$, $\mathcal{I}(bool) = \{\top, \bot\}$, and $\mathcal{J}$ interprets these symbols as expected: $\mathcal{J}(\mathsf{true}) = \top$ and $\mathcal{J}(\mathsf{false}) = \bot$. We omit the sort subscripts ι from $=_\iota$ and $\neq_\iota$ when they are clear from context. A theory term with sort *bool* is called a *constraint*. The *standard integer signature* Σ^{int}_{theory} is $\Sigma^{core}_{theory} \cup \{+, -, \times, \mathsf{exp}, \mathsf{div}, \mathsf{mod} : int \times int \Rightarrow int\} \cup \{\geq, > : int \times int \Rightarrow bool\} \cup \mathcal{V}al_{int}$ where $\mathcal{S}_{theory} \supseteq \{int, bool\}$, $\mathcal{V}al_{int} = \{\mathsf{n} \mid n \in \mathbb{Z}\}$, $\mathcal{I}(int) = \mathbb{Z}$, and $\mathcal{J}(\mathsf{n}) = n$ for any integer $n \in \mathbb{Z}$. Note that we use n (in sans-serif font) as the function symbol for $n \in \mathbb{Z}$ (in *math* font). We define $\mathcal{J}$ in a natural way. An LCTRS with the standard integer signature is called an *integer LCTRS*.

Example 1. *Let $\mathcal{S} = \{int, bool\}$, $\Sigma = \Sigma_{terms} \cup \Sigma^{int}_{theory}$ and $\Sigma_{terms} = \{\mathsf{fact} : int \Rightarrow int\} \cup \{\mathsf{n} : int \mid n \in \mathbb{Z}\}$. To implement an integer LCTRS calculating the* factorial *function over $\mathbb{Z}$, we use the signature Σ above and the following LCTRS:*

$$\mathcal{R}_2 = \{\ \mathsf{fact}(x) \to \mathsf{1}\ [x \leq \mathsf{0}] \qquad \mathsf{fact}(x) \to x \times \mathsf{fact}(x - \mathsf{1})\ [x > \mathsf{0}]\ \}$$

Note that $\mathcal{R}_2$ implicitly includes rules for calculation symbols such as $-$ and $\times$: $x - y \to z\ [z = x - y]$ and $x \times y \to z\ [z = x \times y]$. The term $\mathsf{fact}(\mathsf{3})$ is reduced by $\mathcal{R}_2$ to $\mathsf{6}$: $\mathsf{fact}(\mathsf{3}) \to_{\mathcal{R}_2} \mathsf{3} \times \mathsf{fact}(\mathsf{3}-\mathsf{1}) \to_{\mathcal{R}_2} \mathsf{3} \times \mathsf{fact}(\mathsf{2}) \to_{\mathcal{R}_2} \mathsf{3} \times (\mathsf{2} \times \mathsf{fact}(\mathsf{2}-\mathsf{1})) \to_{\mathcal{R}_2} \mathsf{3} \times (\mathsf{2} \times \mathsf{fact}(\mathsf{1})) \to_{\mathcal{R}_2} \mathsf{3} \times (\mathsf{2} \times (\mathsf{1} \times \mathsf{fact}(\mathsf{1}-\mathsf{1}))) \to_{\mathcal{R}_2} \mathsf{3} \times (\mathsf{2} \times (\mathsf{1} \times \mathsf{fact}(\mathsf{0}))) \to_{\mathcal{R}_2} \mathsf{3} \times (\mathsf{2} \times (\mathsf{1} \times \mathsf{1})) \to_{\mathcal{R}_2} \mathsf{3} \times (\mathsf{2} \times \mathsf{1}) \to_{\mathcal{R}_2} \mathsf{3} \times \mathsf{2} \to_{\mathcal{R}_2} \mathsf{6}$.

A *constrained term* is a pair $\langle t \mid \phi\rangle$ of a term t and a constraint ϕ, which can be considered the set of all instances of t w.r.t. substitutions that *respect* ϕ:[2] $\langle t \mid \phi\rangle = \{t\gamma \mid \gamma \text{ respects } \phi\}$. An *all-path reachability problem* (APR problem, for short) of an LCTRS $\mathcal{R}$ is a pair $\langle s \mid \phi\rangle \Rightarrow \langle t \mid \psi\rangle$ of constrained terms $\langle s \mid \phi\rangle, \langle t \mid \psi\rangle$ for state sets. We call $\langle s \mid \phi\rangle \Rightarrow \langle t \mid \psi\rangle$ *constant-directed* if $\langle t \mid \psi\rangle$ is a singleton set of a constant normal form c [8]. A (possibly infinite) reduction sequence of $\mathcal{R}$ is called an *execution path* if the reduction sequence either is infinite or ends with an irreducible term. If $\langle t \mid \psi\rangle = \{c\}$ for some constant normal form c, then we abbreviate $\langle s \mid \phi\rangle \Rightarrow \langle t \mid \psi\rangle$ to $\langle s \mid \phi\rangle \Rightarrow c$. A constant-directed APR problem $\langle s \mid \phi\rangle \Rightarrow c$ is *demonically valid* w.r.t. $\mathcal{R}$ if every *finite* execution path starting from a term in $\langle s \mid \phi\rangle$ ends with c.

The proof system c-DCC [8, 9] for constant-directed APR problems is a weakened but easily-implementable variant of the proof system DCC [4] for arbitrary APR problems.

Definition 2 (c-DCC [8, 9])**.** *Let $\mathcal{R}$ be an LCTRS, and G a finite set of constant-directed APR problems. The set of* derivatives *of a constrained term w.r.t. an LCTRS $\mathcal{R}$ is defined as follows: $\Delta_{\mathcal{R}}(\langle s \mid \phi\rangle) = \bigcup_{\ell \to r\ [\varphi] \in \mathcal{R}} \Delta_{\ell,r,\varphi}(\langle s \mid \phi\rangle)$, where $\ell \to r\ [\varphi]$ has no shared variable with $\langle s \mid \phi\rangle$*[3] *and $\Delta_{\ell,r,\varphi}(\langle s \mid \phi\rangle) = \{\langle (s[r]_p)\gamma \mid (\phi \wedge \varphi)\gamma\rangle \mid p \in \mathcal{P}os(s),\ s|_p \notin \mathcal{V},\ s|_p$ and ℓ are unifiable, $\gamma = mgu(s|_p, \ell)$, $\mathcal{R}an(\gamma|_{\mathcal{V}ar(\phi,\varphi)}) \subseteq \mathcal{V}al \cup \mathcal{V}$, $(\phi \wedge \varphi)\gamma$ is satisfiable$\}$. The proof system $\mathsf{c\text{-}DCC}(\mathcal{R}, G)$ consists of the following proof rules:*

$$\frac{\phi \text{ is unsatisfiable or } s = c}{\langle s \mid \phi\rangle \Rightarrow c}\ (\mathsf{axiom/c\text{-}subs}) \qquad \frac{\langle s_1 \mid \phi_1\rangle \Rightarrow c \qquad \ldots \qquad \langle s_n \mid \phi_n\rangle \Rightarrow c}{\langle s \mid \phi\rangle \Rightarrow c}\ (\mathsf{c\text{-}der})$$

where $\Delta_{\mathcal{R}}(\langle s \mid \phi\rangle) = \{\langle s_i \mid \phi_i\rangle \mid 1 \leq i \leq n\}$ for some $n > 0$.

$$\frac{\exists(\langle s' \mid \phi'\rangle \Rightarrow c) \in G.\ \langle s \mid \phi\rangle \subseteq \langle s' \mid \phi'\rangle}{\langle s \mid \phi\rangle \Rightarrow c}\ (\mathsf{weak\ circ})$$

[2]A substitution γ is said to *respect* a constraint ϕ if $\gamma(x) \in \mathcal{V}al$ for all $x \in \mathcal{V}ar(\phi)$, and $\mathcal{J}(\phi\gamma) = \top$.
[3]When there exists a shared variable, we rename the variables in $\ell \to r\ [\varphi]$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(2)}\,\text{(w. circ)} \quad \dfrac{}{(3)}\,\text{(w. circ)}}{(9)}\,\text{(c-der)} \quad \dfrac{\dfrac{}{(3)}\,\text{(w. circ)}}{(10)}\,\text{(c-der)}}{(5)}\,\text{(c-der)} \quad \dfrac{\dfrac{}{(10)}\,\text{(w. circ)}}{(6)}\,\text{(c-der)}}{(2)}\,\text{(c-der)} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{(2)}\,\text{(w. circ)}}{(11)}\,\text{(c-der)}}{(7)}\,\text{(c-der)} \quad \dfrac{\dfrac{}{(11)}\,\text{(w. circ)} \quad \dfrac{}{(1)}\,\text{(w. circ)}}{(8)}\,\text{(c-der)}}{(3)}\,\text{(c-der)} \quad \dfrac{}{(4)}\,\text{(a./c-subs)}}{(1)}\,\text{(c-der)}$$

where

(1) $\langle \mathsf{cnfg}(\mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, 0) \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$
(2) $\langle \mathsf{cnfg}(\mathsf{wait}_0, \mathsf{noncrit}_1, \mathsf{true}, \mathsf{false}, 1) \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$
(3) $\langle \mathsf{cnfg}(\mathsf{noncrit}_0, \mathsf{wait}_1, \mathsf{false}, \mathsf{true}, 0) \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$
(4) $\langle \mathsf{init} \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$
(5) $\langle \mathsf{cnfg}(\mathsf{crit}_0, \mathsf{noncrit}_1, \mathsf{true}, \mathsf{false}, 1) \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$
(6) $\langle \mathsf{cnfg}(\mathsf{wait}_0, \mathsf{wait}_1, \mathsf{true}, \mathsf{true}, 0) \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$
(7) $\langle \mathsf{cnfg}(\mathsf{wait}_0, \mathsf{wait}_1, \mathsf{true}, \mathsf{true}, 1) \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$
(8) $\langle \mathsf{cnfg}(\mathsf{noncrit}_0, \mathsf{crit}_1, \mathsf{false}, \mathsf{true}, 0) \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$
(9) $\langle \mathsf{cnfg}(\mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, 1) \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$
(10) $\langle \mathsf{cnfg}(\mathsf{crit}_0, \mathsf{wait}_1, \mathsf{true}, \mathsf{true}, 0) \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$
(11) $\langle \mathsf{cnfg}(\mathsf{wait}_0, \mathsf{crit}_1, \mathsf{true}, \mathsf{true}, 1) \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$

Figure 3: A proof tree for the APR problem $\langle \mathsf{cnfg}(\mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, 0) \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$.

Theorem 3 (soundness of c-DCC [9]). *Let $\mathcal{R}$ be an LCTRS, and G a finite set of constant-directed APR problems. Suppose that for each problem $(\langle s \mid \phi\rangle \Rightarrow c) \in G$, there exists a proof tree $\mathcal{T}$ under* c-DCC$(\mathcal{R}, G)$, *each* circ *node of which has a* c-der *node as an ancestor. Then, $\mathcal{R}$ demonically satisfies all APR problems in G.*

For an APR problem $\langle s \mid \phi\rangle \Rightarrow c$, a construction of a single proof tree in the style of *cyclic proofs* [2] has been proposed [10].

3 Confluence w.r.t. Ground Initial Terms

Confluence of an LCTRS $\mathcal{R}$ w.r.t. a initial (ground) term is equivalent to the property that any two of the reachable terms are joinable. If a term can be reduced to the initial term, then it is joinable with any other term. Thus, a trivial sufficient condition for the property is that all the reachable terms can be reduced back to the initial one. It is not so easy to prove that $\mathcal{R}$ satisfies this sufficient condition in the case where the set of reachable terms is infinite. In this section, we show how to prove it by using proof trees of APR problems.

Let us consider the LCTRS $\mathcal{R}_1$ and the initial term $s_0 = \mathsf{cnfg}(\mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, 0)$ in Section 1 again. We prepare a fresh constant init and add the rule $s_0 \to \mathsf{init}$ to $\mathcal{R}_1$, obtaining $\mathcal{R}_1' = \mathcal{R}_1 \cup \{s_0 \to \mathsf{init}\}$. Then, the APR problem $\langle \mathsf{cnfg}(\mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, 0) \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$ is demonically valid w.r.t. $\mathcal{R}_1'$ because of the proof tree illustrated in Figure 3. Since neither $\mathcal{R}_1$ nor $\mathcal{R}_1'$ has a normal form rooted by cnfg, there is no finite execution path starting from the initial term. For this reason, it is trivial that the APR problem above is demonically valid w.r.t. $\mathcal{R}_1'$. Though, the proof tree can be used to prove confluence w.r.t. the initial term.

By considering the proof tree as a cyclic proof and identifying weak circ nodes with other nodes having the same APR problems, we obtain the *proof graph* [2], which is a directed graph. All axiom/c-subs nodes have the APR problem $\langle \mathsf{init} \mid \mathsf{true}\rangle \Rightarrow \mathsf{init}$ which is not relevant to the reduction of the original LCTRS $\mathcal{R}$. Thus, we remove the axiom/c-subs nodes from the axiom/c-subs nodes from the proof graph, obtaining a directed graph. If the directed graph is strongly connected and each node corresponds to a single term, then all reachable nodes are reachable back to the root one, i.e., all reachable terms from the initial one can be reduced back to the initial one. For example, we remove node (4) from the proof tree in Figure 3, obtaining a strongly connected directed graph such that each node corresponds to a single term. Therefore, $\mathcal{R}_1$ is confluent w.r.t. the initial term.

$$\begin{tikzcd}
(9) \arrow[ddrr, bend right] \arrow[ddrrrr, bend left] & & (10) \arrow[ddrr] & & (11) \arrow[ddll] & \\
& (5) \arrow[ul] \arrow[ur] & (6) \arrow[u] & & (7) \arrow[u] & (8) \arrow[ul] \arrow[ddll, bend left] \\
& & (2) \arrow[ul] \arrow[u] & & (3) \arrow[u] \arrow[ur] & \\
& & & (1) \arrow[ul] \arrow[ur] & &
\end{tikzcd}$$

Figure 4: The proof graph obtained from the proof tree in Figure 3 by removing all axiom/c-subs nodes.

Theorem 4. *Let $\mathcal{R}$ be the LCTRS obtained from a concurrent program, s_0 be an initial term corresponding to an initial configuration of the program, and* init *a fresh constant. Suppose that we have a proof tree for the APR problem $\langle s_0 \mid \mathsf{true} \rangle \Rightarrow \mathsf{init}$. Let G be the directed graph obtained from the proof graph of the proof tree by removing all* axiom/c-subs *nodes. Suppose that G is strongly connected and one of the following holds:*

(a) for every node of G, the constrained term $\langle s \mid \phi \rangle$ of the attached APR problem $\langle s \mid \phi \rangle \Rightarrow \mathsf{init}$ is singleton, or

(b) the directed graph obtained from G by removing the root is acyclic.

Then, $\mathcal{R}$ is confluent w.r.t. s_0.

Proof. Recall that by assumption, thee is no reachable normal form rooted by cnfg. For reachable terms t_1, t_2 with $s_0 \to^*_{\mathcal{R}} t_1$ and $s_0 \to^*_{\mathcal{R}} t_2$, we show that t_1 and t_2 are joinable. Let v_1, v_2 be nodes of G such that APR problems $\langle s_1 \mid \phi_1 \rangle \Rightarrow \mathsf{init}$ and $\langle s_2 \mid \phi_2 \rangle \Rightarrow \mathsf{init}$ are the attached APR problems of v_1 and v_2, respectively, $t_i \in \langle s_i \mid \phi_i \rangle$ for $i = 1, 2$, and the paths from the root node to v_1 and v_2 correspond to $s_0 \to^*_{\mathcal{R}} t_1$ and $s_0 \to^*_{\mathcal{R}} t_2$, respectively. We make a case analysis depending on which premise holds, (a) or (b).

(a) In this case, we have that $\{t_i\} = \langle s_i \mid \phi_i \rangle$ for $i = 1, 2$, Since every node corresponds to a single term, paths of G coincide with reduction sequences. In addition, since G itself is strongly connected, there exist paths from v_1 and v_2 to the root node, respectively, and hence we have that $t_1 \to^*_{\mathcal{R}} s_0$ and $t_2 \to^*_{\mathcal{R}} s_0$.

(b) Since the graph without the root node is acyclic, there exist unique paths from v_1 and v_2 to the root node, respectively. Thus, we have that $t_1 \to^*_{\mathcal{R}} s_0$ and $t_2 \to^*_{\mathcal{R}} s_0$.

Therefore, t_1 and t_2 are joinable. □

Example 5. *We obtain the proof graph in Figure 4 from the proof tree in Figure 3 by removing all* axiom/c-subs *nodes. The proof graph is strongly connected, and for every node, the attached constrained term is singleton because every term rooted by* cnfg *has a unique reduct. Therefore, by Theorem 4, $\mathcal{R}_1$ is confluent w.r.t.* $\mathsf{cnfg}(\mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, 0)$.

Example 6. *Consider the LCTRS $\mathcal{R}_1$ in Figure 2 again. This LCTRS is linear and all its critical pairs are strongly closed and thus, $\mathcal{R}_1$ is strongly confluent (i.e., confluent) [14]. In fact,* crest,[4] *Constrained REwriting Software Tool, succeeds in proving confluence of $\mathcal{R}_1$. However,* crest *failed to prove confluence of the LCTRS obtained from $\mathcal{R}_1$ by adding the rules*

[4]http://cl-informatik.uibk.ac.at/software/crest/

$\{\mathsf{cnfg}(s_0, p_1, b_0, b_1, x) \to \mathsf{cnfg}(s_0, p_1, b_0, b_1, x) \mid s_0 \in \{\mathsf{noncrit}_0, \mathsf{wait}_0, \mathsf{crit}_0\},\ p_1, b_0, b_1, x \in \mathcal{V}\} \cup \{\mathsf{cnfg}(p_0, s_1, b_0, b_1, x) \to \mathsf{cnfg}(p_0, s_1, b_0, b_1, x) \mid s_1 \in \{\mathsf{noncrit}_1, \mathsf{wait}_1, \mathsf{crit}_1\},\ p_0, b_0, b_1, x \in \mathcal{V}\}$. *On the other hand, in our approach, we succeed in constructing a proof tree which is almost the same as the tree in Figure 3, and the corresponding graph satisfies the condition shown in Theorem 4. Therefore, the LCTRS is confluent w.r.t. the initial term* $\mathsf{cnfg}(\mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, 0)$.

The following examples shows the reason why the condition (a) is assumed in Thereom 4.

Example 7. *Consider the following LCTRS:*

$$\mathcal{R}_2 = \left\{ \begin{array}{lll} \mathsf{cnfg}(\mathsf{f}(x)) \to \mathsf{cnfg}(\mathsf{g}(x,y)) & [\, y \geq x \,] \\ \mathsf{cnfg}(\mathsf{g}(x,y)) \to \mathsf{cnfg}(\mathsf{f}(x)) & [\, y \leq x \,] \\ \mathsf{cnfg}(\mathsf{g}(x,y)) \to \mathsf{cnfg}(\mathsf{g}(x,y')) & [\, y > x \wedge y' = y+1 \,] \\ \mathsf{cnfg}(\mathsf{f}(x)) \to \mathsf{cnfg}(\mathsf{h}(x,y)) & [\, y \geq x \,] \\ \mathsf{cnfg}(\mathsf{h}(x,y)) \to \mathsf{cnfg}(\mathsf{f}(x)) & [\, y \leq x \,] \\ \mathsf{cnfg}(\mathsf{h}(x,y)) \to \mathsf{cnfg}(\mathsf{h}(x,y')) & [\, y > x \wedge y' = y+1 \,] \end{array} \right\}$$

This LCTRS is not confluent w.r.t. $\mathsf{cnfg}(\mathsf{f}(0))$. *For the APR problem* $\langle \mathsf{cnfg}(\mathsf{f}(0)) \mid \mathsf{true} \rangle \Rightarrow \mathsf{init}$, *we obtain an APR proof, the proof graph of which is strongly connected. The root node (12) with* $\langle \mathsf{cnfg}(\mathsf{f}(0)) \mid \mathsf{true} \rangle \Rightarrow \mathsf{init}$ *has two children (13) and (14) with the APR problems* $\langle \mathsf{cnfg}(\mathsf{g}(0,y)) \mid y \geq 0 \rangle \Rightarrow \mathsf{init}$ *and* $\langle \mathsf{cnfg}(\mathsf{h}(0,y)) \mid y \geq 0 \rangle \Rightarrow \mathsf{init}$, *respectively, both of which are sets of two or more terms. There is a path from (13) to (12), but not all terms in (13) can be reduced to the initial term* $\mathsf{cnfg}(\mathsf{f}(0))$. *For example,* $\mathsf{cnfg}(\mathsf{g}(0,1))$ *cannot be reduced to the initial term, and all reduction sequences from* $\mathsf{cnfg}(\mathsf{g}(0,1))$ *follow the path (13),(13),...*

4 Conclusion

In this paper, we showed how to prove confluence of an LCTRS w.r.t. an initial term. The proposed method succeeded in proving confluence of concurrent programs w.r.t. their initial states, e.g., the AITS defined by the program graphs in Figure 1.

One may think that we do not have to use the APR setting in order to prove our confluence problems. In the case where the set of reachable terms from an initial one is finite, confluence w.r.t. the initial term is decidable and we do not have to reduce the confluence problem to an APR problem. In the other case, as for $\mathcal{R}_1$, our approach may work for the confluence problem. In addition, our approach can be automated by using the implementation of the proof system for APR problems, i.e., we do not have to implement a new tool from scratch.

A future work is the automation of the proposed method, i.e., to adapt our implementation for APR problems to confluence w.r.t. an initial one. Given an LCTRS $\mathcal{R}$ and its initial term s_0, using the proof system for constant-directed APR problems, we automatically construct a proof tree for the APR problem $\langle s_0 \mid \mathsf{true} \rangle \Rightarrow \mathsf{init}$ w.r.t. $\mathcal{R}$, obtaining the proof graph. Note that we may fail the construction even if $\mathcal{R}$ is confluent (w.r.t. s_0). The strong connectedness and the condition (b) in Theorem 4 are decidable. On the other hand, we have not had sufficient conditions for the condition (a) in Theorem 4 yet. It is also our future work to find a decidable sufficient condition for the condition (a).

References

[1] C. Baier and J. Katoen. *Principles of model checking.* MIT Press, 2008.

[2] J. Brotherston. Cyclic proofs for first-order logic with inductive definitions. In B. Beckert, editor, *Proceedings of the 14th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, volume 3702 of *Lecture Notes in Computer Science*, pages 78–92, Springer, 2005.

[3] Ş. Ciobâcă, D. Lucanu, and A. Buruiana. Operationally-based program equivalence proofs using LCTRSs. *Journal of Logical and Algebraic Methods in Programming*, 135:1–22, 2023.

[4] Ş. Ciobâcă and D. Lucanu. A coinductive approach to proving reachability properties in logically constrained term rewriting systems. In D. Galmiche, S. Schulz, and R. Sebastiani, editors, *Proceedings of the 9th International Joint Conference on Automated Reasoning*, volume 10900 of *Lecture Notes in Computer Science*, pages 295–311, Springer, 2018.

[5] C. Fuhs, C. Kop, and N. Nishida. Verifying procedural programs via constrained rewriting induction. *ACM Transactions on Computational Logic*, 18(2):14:1–14:50, 2017.

[6] Y. Kanazawa and N. Nishida. On transforming functions accessing global variables into logically constrained term rewriting systems. In J. Niehren and D. Sabel, editors, *Proceedings of the 5th International Workshop on Rewriting Techniques for Program Transformations and Evaluation*, volume 289 of *Electronic Proceedings in Theoretical Computer Science*, pages 34–52, Open Publishing Association, 2019.

[7] Y. Kanazawa, N. Nishida, and M. Sakai. On representation of structures and unions in logically constrained rewriting. IEICE Technical Report SS2018-38, Vol. 118, No. 385, pp. 67–72, the Institute of Electronics, Information and Communication Engineers, 2019 (in Japanese).

[8] M. Kojima and N. Nishida. From starvation freedom to all-path reachability problems in constrained rewriting. In M. Hanus and D. Inclezan, editors, *Proceedings of the 25th International Symposium on Practical Aspects of Declarative Languages*, volume 13880 of *Lecture Notes in Computer Science*, pages 161–179, Springer Nature Switzerland, 2023.

[9] M. Kojima and N. Nishida. Reducing non-occurrence of specified runtime errors to all-path reachability problems of constrained rewriting. *Journal of Logical and Algebraic Methods in Programming*, 135:1–19, 2023.

[10] M. Kojima and N. Nishida. A sufficient condition of logically constrained term rewrite systems for decidability of all-path reachability problems with constant destinations. *Journal of Information Processing*, 32:417–435, 2024.

[11] M. Kojima, N. Nishida, and Y. Matsubara. Transforming concurrent programs with semaphores into logically constrained term rewrite systems. In A. Riesco and V. Nigam, editors, *Informal Proceedings of the 7th International Workshop on Rewriting Techniques for Program Transformations and Evaluation*, pages 1–12, 2020.

[12] C. Kop and N. Nishida. Term rewriting with logical constraints. In P. Fontaine, C. Ringeissen, and R. A. Schmidt, editors, *Proceedings of the 9th International Symposium on Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Artificial Intelligence*, pages 343–358, 2013.

[13] N. Nishida and S. Winkler. Loop detection by logically constrained term rewriting. In R. Piskac and P. Rümmer, editors, *Proceedings of the 10th Working Conference on Verified Software: Theories, Tools, and Experiments*, volume 11294 of *Lecture Notes in Computer Science*, pages 309–321, Springer, 2018.

[14] J. Schöpf and A. Middeldorp. Confluence criteria for logically constrained rewrite systems. In B. Pientka and C. Tinelli, editors, *Proceedings of the 29th International Conference on Automated Deduction*, volume 14132 of *Lecture Notes in Computer Science*, pages 474–490, Springer, 2023.

[15] S. Winkler and A. Middeldorp. Completion for logically constrained rewriting. In H. Kirchner, editor, *Proceedings of the 3rd International Conference on Formal Structures for Computation and Deduction*, volume 108 of *Leibniz International Proceedings in Informatics*, pages 30:1–30:18, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

From Subtree Replacement Systems to Term Rewriting Systems: a Roadmap to Orthogonality*

Salvador Lucas

DSIC & VRAIN, Universitat Politècnica de València, Spain
slucas@dsic.upv.es

Abstract

Orthogonality plays an important role in the analysis of confluence of term rewriting systems, and also in related subfields like optimality, strategic rewriting, etc. However, the term "orthogonality" was introduced by Dershowitz and Jouannaud in 1990 after more than 20 years of developments, often "hidden" under terminologies which are not used anymore. This paper collects concepts, terminologies, and notations which have been relevant in this process, together with the main works that introduced them.

Man gave names to all the animals (Bob Dylan)

1 Motivation

Confluence is a central property of Term Rewriting Systems (TRSs). A helpful notion to prove confluence of TRSs is *orthogonality.* Since the nineties of the XXth century, the notion of an (almost, weakly) *orthogonal* TRS has been used and taught as presented in existing books on term rewriting systems, notably [1, 21, 24]. However, it took 20 years to fix the notion and denomination we use since then. In the meantime, different definitions and results were proposed and obtained by different authors whose research did *not* specifically focus on "term rewriting systems" or "orthogonality". This paper is an effort to provide a sequence of references where the evolution of the main involved notions and results can be found. Nowadays, the possibility of an electronic access to "old" documents makes this task easier than some years ago. Although considering all relevant references is difficult, we hope this paper can be a useful and sufficiently detailed roadmap to orthogonality (e.g., for students).

2 Confluence

Church and Rosser investigated reduction of λ-expressions [5]. They introduced several terms in use today: *conversion*, *reduction*, *reducible* (p. 472); *contract(ion)*, *peak*, *valley*, *residual* (p. 473). They proved that, if two λ-expressions A and B are convertible ($A \leftrightarrow^* B$ in the current notation), then there is a *valley* between A and B (we often say that A and B are *joinable*) [5, Theorem 1]. Nowadays, this result is known as the *Church-Rosser property* (of λ-calculus).

According to Newman, results like [5, Theorem 1] would be considered as a *theorem of confluence* [19, Introduction], in what seems to be the first use of the word "confluence" applied to the joinability of peaks. Such *confluence property* is further developed in [19, Section 3]. In particular Newman defines confluence as the property presented in item (A) of Section 3, which just formulates the Church-Rosser property in Newman's terminology. Newman's property

*Partially supported by MCIN/AEI project PID2021-122830OB-C42 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe" and by the grant CIPROM/2022/6 funded by Generalitat Valenciana.

(B) is presented as a consequence of his property (A), i.e., (A) $\Rightarrow$ (B) holds. Newman relies on notions and notations coming from the theory of partially ordered sets. However, only *transitivity* of the relation (denoted $>$, which actually coincides with $\to^+$, see the last paragraph of [19, Section 2]) is required. Property (B) is formulated as follows: *if x_1 and x_2 have an upper bound they have also a lower bound*, i.e.,

> *if there is x such that $x \geq x_1$ and $x \geq x_2$, then there is x' such that $x_1 \geq x'$ and $x_2 \geq x'$.*

Nowadays we would say that $\geq$ is $\to^*$; hence

> property (B) in [19, Section 3] is what we call confluence today.

See [13, Definition D7] and [20, Definition 24] for more explicit definitions, including the usual diagramatic representation. Curry and Feys' Property (B) in [6, page 110] is equivalent to Newman's property (B) and [6, Theorem 3] proves that (B) $\Rightarrow$ (A) holds,[1] thus establishing the well-known equivalence between confluence and Church-Rosser property.

3 Orthogonality of TRSs

In the development of orthogonality of TRSs as a syntactic criterion for confluence, there are two milestones: in 1970 Rosen proved confluence of (left-linear) closed Subtree Replacement Systems where no overlap is allowed (which correspond to orthogonal systems); then, in 1977 Huet and O'Donnell (independently) extended Rosen's result to TRSs with a restricted subclass of overlaps (weakly orthogonal TRSs).

Rosen (1970): Closed Subtree Replacement Systems. Following Brainerd's work [4, 3, 2], Rosen investigated confluence of *Subtree Replacement Systems* (SRSs [23, Definition 5.1], although [22, Definition 2.1] uses PBS, standing for *Post-Brainerd Systems*), consisting of a (possibly infinite) set of tree transformation rules. As in [10], each (finite) tree R with labels in a set V is a mapping $R : \mathsf{Dom}\,R \to V$ from a *tree domain* $\mathsf{Dom}\,R$, i.e., a nonempty subset of sequences in $\mathbb{N}^*_{>0}$, the set of sequences of positive numbers to a set V of *labels* (e.g., of symbols from an *alphabet*) associated to the node indicated by a sequence $i_1.i_2.\ldots.i_n$ of positive numbers. The tree domain $\mathsf{Dom}\,R$ must satisfy two closure properties: for all $p \in \mathbb{N}^*_{>0}$ and $i \in \mathbb{N}_{>0}$, (i) if $p.i \in \mathsf{Dom}\,R$, then $p \in \mathsf{Dom}\,R$ ('down' closed) and (ii) if $p.(i+1) \in \mathsf{Dom}\,R$, then $p.i \in \mathsf{Dom}\,R$ ('left' closed) [22, Section 2]. Although Rosen's notion of tree does *not* impose that nodes labeled with a given symbol have a *fixed* number of 'offsprings', in [22, Definition 3.1] he considers *properly ranked* trees where symbols have a fixed *rank* (i.e., an *arity*) and each k-ary symbol has exactly k associated subtrees.

Remark 1. *If terms t are viewed as properly ranked trees R, then $\mathsf{Dom}\,R$ is just the set $\mathcal{P}os(t)$ of positions of t.*

In the second column of [22, page 119], Rosen explains how to define an SRS (or PBS) by 'instantiating' a set of *rule schemata*. A *rule schema* is a pair $R \to S$ of trees[2] R and S where some *leaves* are labeled with the so-called *parameters* u from a set U, which play the role of *variables*. The left-hand sides R of a rule schema may contain at most *one* occurrence of each parameter u. Thus, Rosen's rule schemata correspond to left-linear TRSs. As done by

[1]Curry and Feys use (χ) to refer to Newman's property (A).
[2]The arrow was already used by Brainerd.

Table 1: Terms, substitutions, and term rewriting in [18]

[18]	*Operators with degrees*	*Words*	*Pure word*	*Nontrivial subword*
TRSs	Signature	Terms	Ground term	Non-variable subterm

[18]	*Relations* (λ, ϱ)	*Set of relations* R	α *reduces to* α' *w.r.t.* R	$\alpha \to \alpha'$ (R)
TRSs	Rules $\ell \to r$	TRS $\mathcal{R}$	α rewrites to α'	$\alpha \to_{\mathcal{R}} \alpha'$

Church and Rosser for the λ-calculus, Rosen developed an appropriate notion of *residual* to keep track of redexes that remain as redexes (and therefore can be reduced) after the application of a rewriting step using a rule 'above' the position of such a redex. He called such SRSs *closed* [22, Definition 2.3]. The main idea of closedness is depicted in [23, Figure 7]. Closed and *unequivocal* SRSs are confluent [22, Theorem 2.5]. Unequivocal SRSs are those which can be seen as mappings from trees to trees [22, Definition 2.1], i.e., no pair of rules share the same left-hand side [23, Definition 5.2]. Then, [22, Lemma 2.4] proves that SRSs obtained by *instantiating* the rules of a set of a *specific class* of rule schemata are closed. Such a class of rule schemata is defined by items (1) and (2) of [22, Lemma 2.4] (see also [23, Theorem 6.5]). Item (1) says that each rule in an SRS comes from the instantiation of a *single* rule in the rule schemata; item (2) says that, for all instances $\sigma(A)$ and $\sigma(C)$ of the left-hand sides A and C of rule schemata $A \to B$ and $C \to D$, and all nonempty positions $p \neq \Lambda$, if $\sigma(A)|_p = \sigma(C)$ holds, then $p \notin \mathcal{P}os_{\mathcal{F}}(A)$. They correspond to what we understand as an *orthogonal TRS* today. Orthogonal TRSs are clearly *unequivocal*. Thus,

confluence of orthogonal TRSs is a consequence of [22, Lemma 2.4 & Theorem 2.5].[3]

Knuth & Bendix (1970): Superposition. Knuth and Bendix followed a different approach to the analysis of confluence of TRSs [18]. Since their terminology was quite different from what is used today, Table 1 provides some hopefully useful connections to read the paper. The notion of *critical pair* is usually credited to [18], but they do *not* use this name which is due to Huet (see below). In [18, Section 5], Knuth and Bendix define a *superposition process* to check the *lattice condition* formulated in [18, Theorem 4] as follows: *"if $s \to t$ and $s \to t'$, then there exists u such that $t \to^* u$ and $t' \to^* u$"*. Of course, this is what we call *local confluence* today. In the description of the superposition process, the notion of critical pair arises from the final consideration, in page 274 (Case 2, equation (5.4)), of the situation when $\sigma(\ell') = \sigma(\ell|_p)$ for some rules $\ell \to r$ and $\ell' \to r'$ and $p \in \mathcal{P}os_{\mathcal{F}}(\ell)$, the only one which may lead to a failure in the "lattice condition". In [18, Theorem 5], this is called the "*superposition of ℓ' on p in ℓ*". Then, the corollary in page 275 rephrases Theorem 4 by replacing the lattice condition (i.e., local confluence) by the joinability of terms $C[\sigma(r')]_p$ and $\sigma(r)$ (see their equation (5.9)).[4]

The pair $\langle C[\sigma(r')]_p, \sigma(r)\rangle$, obtained from (5.9), is what we call *critical pair* today.

Knuth and Bendix pay *no attention* to *non-superposition*, i.e., the absence of critical pairs.

O'Donnell (1977): Non-overlappingness. Following Rosen's approach [22, 23], O'Donnell used *non-overlapping* to refer to *rule schemata* consisting of *left-linear* rules (in the current terminology, see condition (4) in [20, Definition 45]) where for all instances $\sigma(A)$ and $\sigma(C)$ of

[3]Huet remarks that Rosen's result concerns ground terms only [13, page 42, column 2, last paragraph].

[4]There is a typo the second member of (5.9), where ϱ_1 should be ϱ_2. Also, there is a typo in the joinability condition of the second line after (5.9): $\sigma'' \to^* \sigma_0''$ should be written instead of $\sigma_0'' \to^* \sigma_0''$.

the left-hand sides A and C of rule schemata $A \to B$ and $C \to D$, and all nonempty positions $p \neq \Lambda$ such that $\sigma(A)|_p = \sigma(C)$, we have $p \notin \mathcal{P}os_{\mathcal{F}}(A)$ [20, Definition 48]. Note, however, that this definition does *not* consider the case when $\sigma(A) = \sigma(C)$. Thus, in contrast to Rosen's approach, there can be non-overlapping rule schemata including rules $A \to B$ and $C \to D$ such that $\sigma(A) = \sigma(C)$ holds *and no further restriction is imposed on* $\sigma(C)$ *and* $\sigma(D)$. Actually, this situation is handled by introducing the notion of a consistent rule schemata [20, Definition 49]. A rule schemata is *consistent* if for all pairs of rule schemata $A \to B$ and $C \to D$, if $\sigma(A) = \sigma(C)$ holds, then B and D are identical up to renaming. Therefore,

non-overlapping and consistent rule schemata correspond to *weakly orthogonal TRSs.*

Accordingly, confluence of weakly orthogonal TRSs follows from [20, Theorems 17 and 6]. Note that there is no reference to any notion of critical pair.

Huet (1977): Critical Pairs of Term Rewriting Systems. Knuth and Bendix did *not* employ the terminology "term rewriting system" or "critical pair". Apparently, they were introduced by Huet,[5] possibly as part of his description of Knuth & Bendix superposition algorithm in [13, page 38].

Definition 2 (Critical pair [13, p. 38])**.** *Let $\mathcal{R}$ be a TRS, $\alpha : \ell \to r$ and $\alpha' : \ell' \to r'$ be rules of $\mathcal{R}$ sharing no variable (rename if necessary), and $p \in \mathcal{P}os_{\mathcal{F}}(\ell)$ be a nonvariable position of ℓ such that $\ell|_p$ and ℓ' unify with* mgu *θ. Then, $\langle\theta(\ell[r']_p), \theta(r)\rangle$ is a* critical pair *(CP for short) of $\mathcal{R}$. If α and α' are renamed versions of the same rule, then the case $p = \Lambda$ is* excluded.[6]

The set of CPs of $\mathcal{R}$ is $\mathsf{CP}(\mathcal{R})$. Huet also investigated conditions guaranteeing confluence of TRSs. In [13, Lemma 9] he proved that joinability of critical pairs characterizes local confluence[7] of a TRS. As remarked above, this was anticipated by Knuth & Bendix. Thus, terminating TRSs all whose critical pairs are joinable are confluent. In order to avoid the requirement of termination, in [13, Section 3] Huet investigates *free* TRSs. According to [13, Section II.1], free terms are what we call *linear* terms today. In [13, Section II.3], Huet introduces *left-free*, *right-free* and *free* TRSs, which, nowadays, we know as *left-linear*, *right-linear*, and *linear* TRSs, respectively. In the second column of page 41, Huet's corollary of [13, Lemma 11] establishes that *left-free parallel-0 closed term rewriting systems are confluent.* Parallel-0 closed TRSs are those whose critical pairs $\langle s, t\rangle$ satisfy that t can be obtained from s by a single step of *parallel rewriting*, written $s \mathrel{+\!\!\!\!\!\!\;\mid\!\!\!\!\!\;\mid\!\!\!\!\to} t$, which consists of the simultaneous contraction in s of a (possibly empty) set P of *disjoint* redexes, see the last paragraph of the second column in [13, page 40]; since P can be empty, $\mathrel{+\!\!\!\!\!\!\;\mid\!\!\!\!\!\;\mid\!\!\!\!\to}$ is *reflexive.* Weakly orthogonal TRSs only have trivial critical pairs $\langle t, t\rangle$ for which such a condition trivially holds, as parallel rewriting is *reflexive.* Thus,

confluence of *weakly orthogonal* TRSs follows from [13, Lemma 11].

Huet & Lévy (1979): Left-linearity. Non-ambiguity. Huet's *left-free* TRSs were called *left-linear* in [12, page 4]. The notions of *left-free*, *right-free* and *free* TRSs in [13] became *left-linear*, *right-linear*, and *linear* TRSs in [14]. Also, in [12, page 4],

[5]His notion of *term rewriting system*, introduced in [13, Section II.2], is essentially the current one, except that left-hand sides ℓ of rules $\ell \to r$ are *not* required to be non-variable terms

[6]In the first sentence of the last paragraph in [18, Section 5], Knuth and Bendix suggest omiting *superpositions of ℓ' on Λ in ℓ* if ℓ and ℓ' are renamed versions of the same term; however, they justify the saying that, in this case, the obtained critical pair would be trivial, for which the equality of *right-hand sides* r and r' of the involved rules should be required as well, as done by Huet.

[7]Local confluence coincides with the *Weak Church-Rosser* property in [16, Definition I.5.2(2)], where the usual acronym WCR is also introduced.

> a TRS is called *non-ambiguous* if for all (variable disjoint) left-hand sides ℓ and ℓ' of rules and non-variable position $p \in \mathcal{P}os_{\mathcal{F}}(\ell)$, $\ell|_p$ and ℓ' do *not* unify.

In [12, page 5, paragraph 2] they remark that this corresponds to TRSs *without critical pairs.*

Klop (1980): Regular Combinatory Reduction Systems. In his 1980 PhD Thesis, Klop investigated Combinatory Reduction Systems (CRSs), which properly include TRSs (in [16, Remark II.1.17], he cites [13] referring to the notion of TRS). When restricted to TRSs, his definition of

> *regular* CRS [16, Definition II.1.16] as a left-linear and non-ambiguous CRS

coincides with the current notion of orthogonality, see [17, Section 8].

Dershowitz (1981): Non-overlapping TRS. Dershowitz defined

> a TRS as *non-overlapping* if *no left-hand side ℓ overlaps a different left-hand side ℓ' and no proper subterm of ℓ overlaps all of ℓ* [8, last paragraph of page 449].

Here, a term t is said to *overlap* a term t' if t' can be unified with some (not necessarily proper) subterm s of t. He also remarks that this condition is equivalent to $\mathcal{R}$ having no critical pairs (as Huet & Lévy's non-ambiguity).

Dershowitz & Jouannaud (1990): Orthogonality. As suggested in the second paragraph of [15, Section 2], and kindly confirmed by Nachum Dershowitz [7],

> the word "orthogonal" was suggested by Dershowitz and Jouannaud and first introduced in [9, Section 7.2]

to *avoid the cumbersome phrase 'non-ambiguous and left-linear'* [17, Chapter 8]. However, Dershowitz and Jouannaud's definition of an orthogonal TRS in [9, Section 7.2] corresponds to what is called a *weakly orthogonal* TRS today. The current definition of (weakly) orthogonal TRSs was fixed by Klop.

Klop (1990): (Weakly) Orthogonal TRSs. Klop argues that providing a clear distinction between orthogonal and weakly orthogonal TRSs is important (see the footnote of [17, page 67]). According to the footnote in [17, page 67], the current definition of orthogonal and weakly orthogonal TRSs was first given in [17, Definition 8.1].

Definition 3 (Orthogonal TRS [17, Definition 8.1] & [15, Definition 3.1.1])**.** *A TRS $\mathcal{R}$ is orthogonal if $\mathcal{R}$ is left-linear and there are no critical pairs. $\mathcal{R}$ is weakly orthogonal if $\mathcal{R}$ is left-linear and $\mathcal{R}$ contains only trivial critical pairs.*

Such a definition is used in current books on term rewriting, e.g., [1, Definition 6.3.6 & page 155], [21, Definition 4.3.4], and [24, Definitions 4.1.1 & 4.1.2].

Hanus (1994): Almost orthogonality. Item 4 of [21, Definition 4.3.4] corresponds to the notion of *almost orthogonality*, which is missing in [1, 24] and provides an intermediate step between orthogonality and weak orthogonality. In [11], Hanus gave the following definition of an *almost* orthogonal TRS:

Definition 4 (Almost orthogonal TRS [11, page 669])**.** *A TRS is* almost orthogonal *if all rules are left-linear and for each pair of rules $\ell \to r$, $\ell' \to r'$, nonvariable subterm $\ell|_p$ of ℓ, and* mgu *θ for $\ell|_p$ and ℓ', p is the root position Λ and the terms $\theta(r)$ and $\theta(r')$ are identical.*

Some important properties of orthogonal TRSs are still valid for almost orthogonal TRSs but not for weakly orthogonal TRSs. For instance, descendants of redexes in almost orthogonal TRSs are still redexes, but this does *not* hold for weakly orthogonal TRSs [21, pager 54]. Almost orthogonality is closely related to the notion of *overlay* critical pairs, which are those obtained from overlaying rules: two (variable-disjoint) rules $\ell \to r$ and $\ell' \to r'$ *overlay* if ℓ and ℓ' unify [21, Definition 4.3.3]. Thus, Ohlebusch's book provides a complete picture of orthogonality.

Definition 5 (Orthogonality of TRSs [21, Definition 4.3.4])**.** *A left-linear TRS $\mathcal{R}$ is* orthogonal *if $\mathsf{CP}(\mathcal{R}) = \emptyset$; it is* almost orthogonal *if every critical pair in $\mathsf{CP}(\mathcal{R})$ originates from an overlay of rewrite rules of $\mathcal{R}$ and is trivial; it is* weakly orthogonal *if all CPs in $\mathsf{CP}(\mathcal{R})$ are trivial.*

4 Conclusions

We have traced the development of the notion of orthogonality of TRSs, since the introduction of the central property of *non-overlappingness*, coined by O'Donnell in 1977, but already present in previous works by Rosen (on *closed* and *unequivocal* subtree replacement systems) and Knuth & Bendix (introducing *superposition* of rewrite rules), to the name "orthogonality" suggested by Dershowitz and Jouannaud in 1990 (also considering the notion of *left-linearity* used by Huet and Lévy in 1979) and applied to TRSs in its usual form today by Klop, also in 1990.

Acknoledgments. I thank Nachum Dershowitz and Michael Hanus for kindly considering my requests about terminological and historical aspects of orthogonality.

References

[1] Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[2] Walter S. Brainerd. Semi-thue systems and representations of trees. In *10th Annual Symposium on Switching and Automata Theory, Waterloo, Ontario, Canada, October 15-17, 1969*, pages 240–244. IEEE Computer Society, 1969.

[3] Walter S. Brainerd. Tree generating regular systems. *Inf. Control.*, 14(2):217–231, 1969.

[4] Walter Scott Brainerd. *Tree Generating Generating Systems and Tree Automata.* PhD thesis, University of Purdue, 1967.

[5] Alonzo Church and J. B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39(3):472–482, 1936.

[6] Haskell B. Curry and Robert Feys. *CombinatoryLogic.* North-Holland Publishing Company, 1958.

[7] Nachum Dershowitz. Personal communication, March 13, 2024.

[8] Nachum Dershowitz. Termination of linear rewriting systems (preliminary version). In Shimon Even and Oded Kariv, editors, *Automata, Languages and Programming, 8th Colloquium, Acre (Akko), Israel, July 13-17, 1981, Proceedings*, volume 115 of *Lecture Notes in Computer Science*, pages 448–458. Springer, 1981.

[9] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite Systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 243–320. Elsevier and MIT Press, 1990.

[10] Saul Gorn. Explicit definitions and linguistic dominoes. In John F. Hart and Satoru Takasu, editors, *Systems and Computer Science Conference*, pages 77–115. University of Western Ontario, 1965.

[11] Michael Hanus. On extra variables in (equational) logic programming. In Leon Sterling, editor, *Logic Programming, Proceedings of the Twelfth International Conference on Logic Programming, Tokyo, Japan, June 13-16, 1995*, pages 665–679. MIT Press, 1995.

[12] Gérard Huet and Jean-Jacques Lévy. Call by need computations in non-ambiguous linear term rewriting systems. Technical report, IRIA Rocquencourt, France, 1979. Rapport Laboria 359.

[13] Gérard P. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 30–45. IEEE Computer Society, 1977.

[14] Gérard P. Huet. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. *J. ACM*, 27(4):797–821, 1980.

[15] J. W. Klop. Chapter 1: Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaurn, editors, *Handbook of Logic in Computer Science*, pages 1–116. Oxford University Press, 1992.

[16] Jan Willem Klop. *Combinatory Reduction Systems, PhD Thesis*. PhD thesis, University of Utrecht. ILLC Historical Dissertation (HDS) Series, HDS-33, 2022.

[17] Jan Willem Klop. Term rewriting systems. Technical report, Centrum voor Wiskunde en Informatica, CWI, 1990. Report CS-R9073.

[18] Donald E. Knuth and Peter E. Bendix. Simple Word Problems in Universal Algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[19] M. H. A. Newman. On Theories with a Combinatorial Definition of "Equivalence". *Annals of Mathematics*, 43(2):pp. 223–243, 1942.

[20] Michael J. O'Donnell. *Computing in Systems Described by Equations*, volume 58 of *Lecture Notes in Computer Science*. Springer, 1977.

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

[22] Barry K. Rosen. Tree-manipulating systems and church-rosser theorems. In Patrick C. Fischer, Robert Fabian, Jeffrey D. Ullman, and Richard M. Karp, editors, *Proceedings of the 2nd Annual ACM Symposium on Theory of Computing, May 4-6, 1970, Northampton, Massachusetts, USA*, pages 117–127. ACM, 1970.

[23] Barry K. Rosen. Tree-Manipulating Systems and Church-Rosser Theorems. *J. ACM*, 20(1):160–187, 1973.

[24] Terese. *Term rewriting systems*, volume 55 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 2003.

The problem of the calissons, by rewriting[*]

Vincent van Oostrom

University of Sussex, School of Engineering and Informatics, Brighton, UK
`Vincent.van-Oostrom@sussex.ac.uk`

Abstract

We show each of four confluence techniques: random descent, proof orders for decreasing diagrams, bricklaying, and local undercutting, serves to solve the problem of the calissons.

Introduction The problem of the calissons as presented in [2] is to show that if a *box*, a regular hexagonal, can be filled with *calissons*, so named after certain diamond-shaped sweets, then in the resulting filled box the numbers of calissons in each of their 3 orientations are the same; in a formula $r = g = b$ for r, g and b the numbers of red, green and blue[1] calissons in the box. For instance, for a box B with sides of length 2, there are 4 calissons for each of the

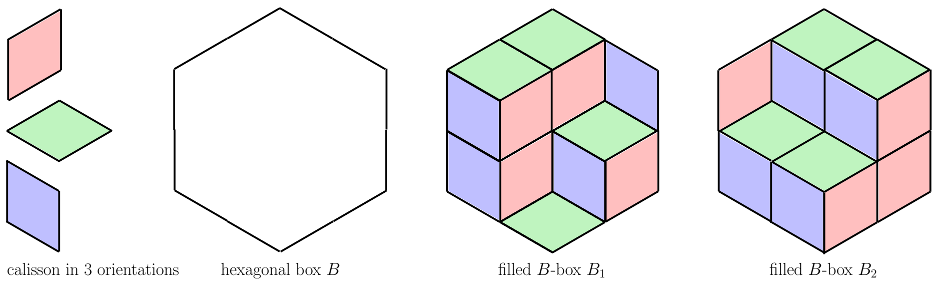


Figure 1: The Problem of the Calissons

3 orientations in both the filled B-boxes B_1 and B_2 in Figure 1; $r_i = g_i = b_i = 4$ for $i \in \{1, 2\}$.

The problem has received quite some attention since; the paper [2] currently has ≈150 citations. We refer the reader to that literature for descriptions, solutions, generalisations, applications and other discussions. The sole purpose here is to offer a rewriting perspective on the problem. We present four solutions, each based on a *confluence* technique.

Instead of requiring boxes to be equiangular hexagons that are also *equilateral* we relax the latter requirement to being *zonogonal*, i.e. to only having *opposite sides* of the same length. We show that if such a box is filled, then for each of their 3 orientations the number of calissons is always the same; if B_1 and B_2 fill the same box B, then $r_1 = r_2$, $g_1 = g_2$ and $b_1 = b_2$. This solves the original problem of the calissons since if B *is* equilateral, is a regular hexagon, the 3 numbers of calissons must in fact be the same as seen by rotational symmetry.

Looking at the filled boxes B_1 and B_2 in Figure 1 it's almost impossible not to see them as different stackings of small cubes inside a large *cube*. From that three-dimensional perspective, the generalisation considered here corresponds to stacking small cubes inside a large (rectangular) *cuboid*. That perspective suggests the number of calissons of a given orientation, is the product of the lengths of the sides of the hexagon parallel to the sides of calissons of that orientation; since B in Figure 1 'is' a cube with sides of length 2, the number of calissons of each type is $2 \times 2 = 4$ as indeed is the case for B_1 and B_2 in the figure.

[*]This note is under the Creative Commons Attribution 4.0 International License 🅭 🅯.

[1]We assign colours to the orientations for convenient referencing and reasons of aesthetics.

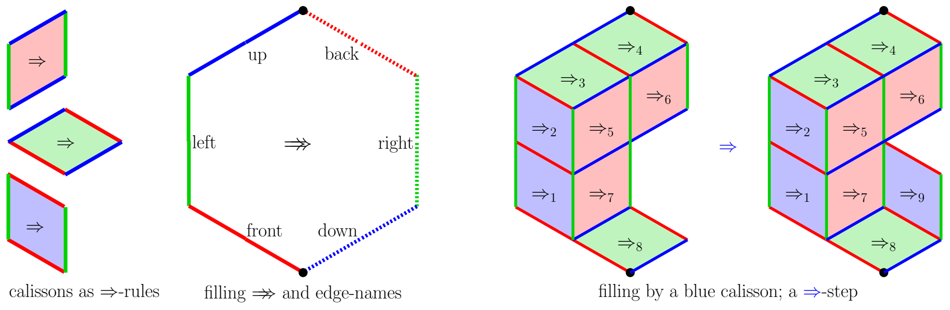


Figure 2: Solving the problem of the calissons by random descent

Random descent In the first approach to the problem by rewriting we view each calisson as a rewrite *rule* ⇒ used to transform the left leg up–left–front of the box into its right leg back–right–down, see Figure 2. A *filling* is a ⇒-reduction gradually transforming the *path* from the bullet • at the top to the • at the bottom of the box, into the dashed path, in the figure:

Definition 1. *Filling* ⇒ is the string rewrite system (SRS) over the alphabet {🟥, 🟩, 🟦} of edges and rewrite rules 🟦🟩 ⇒ 🟩🟦, 🟦🟥 ⇒ 🟥🟦, 🟩🟥 ⇒ 🟥🟩.

A possible filling F_1 from 🟦🟩🟥 to 🟥🟩🟦 is ⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒; it corresponds to the filled box B_1 above. *Other* fillings for the same filled box B_1 are possible, but does any filled box B have *some* filling? Any *partial* filling F of B ends in a path P allowing a filling step. If *none* of them would yield a partial filling of B again, then there could not be any occurrence of 🟦🟥 in P by the assumptions. Moreover, a rightmost occurrence of 🟦🟩 in P would then be filled in B by a pair of green,blue calissons and to the right of the blue one only other such could occur, giving a contradiction (to B being a filled box). Analogous reasoning pertains to a leftmost occurrence of 🟥🟩 in P, showing there's always *some* way to let filling make progress toward B. For instance, Figure 2 depicts that after the first 8 filling steps of F_1, progress toward B_1 is made by the further ⇒-step (not by the, also possible, ⇒-step!).

Having established adequacy of the modelling, the problem of the calissons resurfaces as a *quantitative* confluence question:

Does random descent hold for measure $\Rightarrow\ \mapsto (1,0,0)$, $\Rightarrow\ \mapsto (0,1,0)$, $\Rightarrow\ \mapsto (0,0,1)$?

Recall [9, 10, 13, 11] that a rewrite system having *random descent* (RD) means that for any object that is normalising, rewrites to *a* normal form, we have (i) maximally rewriting the former *always* ends in the latter, and (ii) that all such rewrite sequences have the same *measure*.

In this case, filling ⇒ is seen to be normalising (WN) for the same reason that sorting is; the ⇒-rules simply sort edges into red–green–blue order, cf. [10, Example 7], showing that filling *does* result in a filled box.

Also RD is easily seen to hold: *Because* the (only) critical peak 🟩🟦🟥 ⇐ 🟦🟩🟥 ⇒ 🟦🟥🟩 is joinable as 🟩🟦🟥 ⇒ 🟩🟥🟦 ⇒ 🟥🟩🟦 ⇐ 🟥🟦🟩 ⇐ 🟦🟥🟩 and both legs have measure $(1,1,1)$, *ordered local confluence* (OWCR) holds entailing RD by [13, Lem. 24].

Finally, to see that by having answered the quantitative confluence question in the affirmative we have solved the problem of the calissons, note that the measure given counts the respective numbers of red, green and blue steps while filling. Hence the triple of numbers of red, green and blue of calissons in a filled box, its *spectrum* [4], *is* the measure of its filling. Indeed, the measure $(4,4,4)$ of the filling F_1 is the same as the spectrum of the filled box B_1.

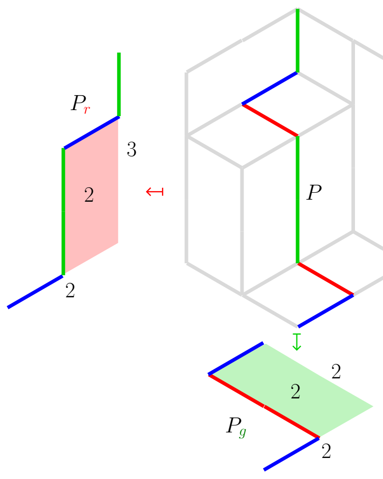


Figure 3: Volume of path P as areas of 3 projections P_r, P_g, P_b (by 'forgetting' colours)

Proof orders for decreasing diagrams Above we proceeded by (weak) normalisation (WN) and ordered local confluence (OWCR). Here we proceed instead by termination (SN) and local confluence (WCR) of filling $\Rightarrow$, as suggested by WN & OWCR $\iff$ SN & WCR [11].

Since the single critical peak of $\Rightarrow$ was shown to be joinable already, yielding WCR, it suffices to show termination of $\Rightarrow$ by a measure from which the numbers of calissons can be retrieved. To realise the idea that was given at the bottom of the first page we make use of the area measure on conversions, introduced for measuring decreasing diagrams in [5, Example 3].

Definition 2. The *area* measure of a conversion comprising ℓ forward and r backward steps, is a triple (ℓ, a, r) measuring how many *square* tiles a are needed to complete the conversion into a valley. The *volume* of a path P is the triple (r, g, b) of *area* measures of the conversions P_r, P_g and P_b obtained from P by forgetting respectively the red, green and blue edges.

Referring the reader to [5, Example 3] for formal details, we illustrate the definition by means of the path P given by 🟩🟦🟥🟩🟩🟥🟦 as depicted in Figure 3. Then P_r is the conversion $\rightarrow\leftarrow\rightarrow\rightarrow\leftarrow$ obtained by forgetting the 🟥-edges in P and orienting the 🟩- and 🟦-edges in opposite directions, yielding area $(3, 2, 2)$, P_g is the conversion $\leftarrow\rightarrow\rightarrow\leftarrow$ having area $(2, 2, 2)$ and P_b the conversion $\leftarrow\rightarrow\leftarrow\leftarrow\rightarrow$ with area $(3, 4, 2)$.

We claim that if V is the volume (r, g, b) of the initial path P of a filling F of box B, then the spectrum of the box is the triple V^2 of second components of V. For instance, the volume of the initial path for box B in Figure 1 is $((2, 4, 2), (2, 4, 2), (2, 4, 2))$, and indeed its triple $(4, 4, 4)$ of second components is the spectrum of B comprising 4 calissons of each colour.

To prove the claim we prove the property that for any filling step of a given colour only the second component of that colour is decremented in the volume, with the areas of the other colours being unchanged. This suffices, since for a filling yielding a filled box, the volume of its final path Q has second components that are all 0, since 'forgetting' then yields valleys: Q_r has shape $\twoheadrightarrow\twoheadleftarrow$, Q_g has shape $\twoheadrightarrow\twoheadleftarrow$ and Q_b shape $\twoheadrightarrow\twoheadleftarrow$. To see the property holds observe that a filling step of a given colour swaps *adjacent* edges of the other colours, so leaves the areas of *those other* colours unchanged, but decrements that of the *given* colour. Indeed, the filling $\Rightarrow$-step in Figure 2 transforms 🟥🟩🟦🟩🟥🟦 into 🟥🟩🟦🟥🟩🟦 and volume $((2, 1, 2), (2, 1, 2), (2, 2, 2))$ into $((2, 1, 2), (2, 1, 2), (2, 1, 2))$, decrementing (only) the blue area.

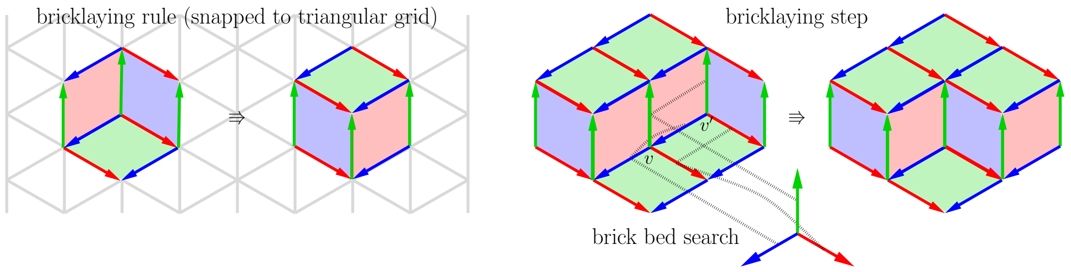


Figure 4: Solving the problem of the calissons by bricklaying

Bricklaying For the third approach to the problem of the calissons by rewriting we change the modelling; filled boxes are now the *objects* of a rewrite system $\Rrightarrow$ having *bricklaying* as *rule* [12] displayed on the left in Figure 4, allowing to locally rearrange the calissons in a box.

Definition 3. Linear combinations $r_r\begin{pmatrix}\frac{1}{2}\sqrt{3}\\-\frac{1}{2}\end{pmatrix} + r_g\binom{1}{0} + r_b\begin{pmatrix}-\frac{1}{2}\sqrt{3}\\-\frac{1}{2}\end{pmatrix}$ give the *grid* of *vertices* for *natural number* scalars r_r, r_g, r_b, a *box* when restricting them to *real* intervals $[0,w]$, $[0,h]$, $[0,d]$ for *natural number* width w, height h, depth d, and *calissons* when further restricting two among w, h, d to 1 and the other (its *colour*) to 0; reversing this yields *edges*. Box, diamond and edge *occurrences* arise by translation. We suppress writing 'occurrence'. A family $\mathcal{D}_I$ of diamonds is a *tiling* (of a box B) if $\mathcal{D}_i \cap \mathcal{D}_j$ is a subset of some edge for $i \neq j$ (and $\bigcup \mathcal{D}_I \subseteq B$).

W.l.o.g. we analyse only the *discrete* problem where calissons occur at vertices, B at the origin, and $B = \bigcup \mathcal{D}_I$. By the spectrum obviously being invariant under $\Rrightarrow$, the problem of the calissons resurfaces as the confluence / uniqueness of normal form question:

$$\text{Does } \bigcup \mathcal{D}_I = B = \bigcup \mathcal{D}'_J \text{ for box } B\text{, entail } \mathcal{D}_I,\ \mathcal{D}'_J \text{ have the same } \Rrightarrow\text{-normal form?}$$

Mapping calissons to their vertices and (coloured) edges, turns tilings into *bed-graphs* [12]: (i) every green tile is a tetragonal cycle of shape $\leftarrow\leftarrow\rightarrow\rightarrow$ and similarly for red and blue tiles; (ii) vertices have at most a single *in-*/*out*-edge of a given colour; (iii) there are no paths having edges of each of the 3 colours; (iv) every path $\rightarrow\rightarrow$ belongs to some tile and similarly for other colour-pairs; (v) if $\leftarrow a \rightarrow$ does not belong to a tile then a has a green in-edge and similarly for other colour-triples. Taking edges as vectors in the 3 colour-dimensions shows tilings even are *beds* [12], i.e. are *plane* bed-graphs under projection from viewpoint $\begin{pmatrix}\infty\\\infty\\\infty\end{pmatrix}$, as used in illustrations.

Let an *i-peak* for such a tiling $\mathcal{D}_I$ be a vertex in B having exactly i *out*-edges. Then $0 \leq i \leq 3$ by there being 3 colours, and we distinguish cases on whether or not there are 3-peaks:

If there are 3-peaks, then the bricklaying $\Rrightarrow$-rule is applicable to at at least one of them. This holds for any bed [12] as depicted in Figure 4: If v is a 3-peak but $\Rrightarrow$ does not apply, then by (v) it has an in-edge of colour c from v', which is a 3-peak by (iv) and *its* in-edge, if any, has colour c by (iii) from which we conclude by finiteness of tilings / monochrome paths in beds.

If there are no 3-peaks, then we have one large *brick* [12] generalising that in the rhs of the $\Rrightarrow$-rule in Figure 4. That is, at the top we *must* have a big green calisson composed of smaller such and *mutatis mutandis* the same for blue / red at the bottom–left / right. This holds in fact for any bed: Any $\leftarrow\rightarrow$-peak then *must* belong to a *green* tile since otherwise (v) and the above reasoning would give rise to a 3-peak contradicting the assumption. The big red, green, blue calissons share boundary paths and these 3 *rays* end up in the same *nexus*, the common reduct; this holds by monochrome paths being finite and (iii), with the former a consequence of the bed-graph being plane. We conclude by noting the 3 big calissons only depend on B.

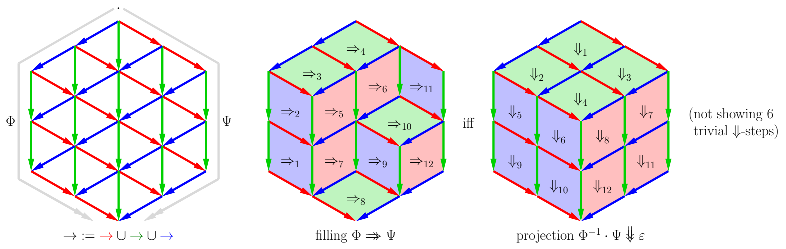


Figure 5: Solving the problem of the calissons by local undercutting; filling iff projection

Local undercutting Our fourth approach to the problem of the calissons by rewriting, inspired by [3, Proposition 4.16(4.18)], mixes the above modellings: We again view the grid for a box B as a rewrite system $\to := \to \cup \to \cup \to$ but with green $\to$-steps now oriented *downward* as displayed on the left in Figure 5. Calissons now are *diamonds* ${}^{\phi}_{\chi}\diamond^{\psi}_{\upsilon}$ inducing *filling* $\phi \cdot \chi \Rightarrow \psi \cdot \upsilon$ respectively *projection* $\phi^{-1} \cdot \psi \Downarrow \chi \cdot \upsilon^{-1}$ rules. Filling is modelled as $\Phi \Rrightarrow \Psi$ for left,right legs Φ, Ψ of B, and we claim it holds iff projecting Φ, Ψ is empty, i.e. iff $\Phi^{-1} \cdot \Psi \Ddownarrow \varepsilon$ with the spectra of filling and projection the same, from which we conclude as the spectrum of projection only depends on B, as seen before. Here we use $\Upsilon, \Phi, X, \Psi, \dots$ to range over *conversions*, elements of the free (typed) involutive monoid over $\to$-steps $\upsilon, \phi, \chi, \psi, \dots$, with $\cdot$ denoting *composition* and ${}^{-1}$ *reverse*; conversions are (possibly empty; ε) compositions of steps and reverse steps [5].

Definition 4. A *local undercutting*[2] (LUC) is a collection of diamonds $\mathcal{D}$ consisting of for every local peak $\phi^{-1} \cdot \psi$ at most one diamond of shape ${}^{\phi}_{X}\diamond^{\psi}_{\Upsilon}$ for reductions X, Υ such that: ${}^{\phi}_{\varepsilon}\diamond^{\phi}_{\varepsilon} \in \mathcal{D}$, and $(\phi \cdot X)^{-1} \cdot \psi \cdot \Upsilon \Ddownarrow \varepsilon$ if $\phi^{-1} \cdot \chi \cdot \chi^{-1} \cdot \psi \Ddownarrow X \cdot \Upsilon^{-1}$. $\mathcal{D}$ is spectrum-*preserving* if the spectra of the latter two projections are the same (they are unique by random descent [10] of $\Downarrow$).

Calissons induce a spectrum-preserving LUC after adjoining ${}^{\phi}_{\varepsilon}\diamond^{\phi}_{\varepsilon}$; per definition of $\to$ the only non-trivial case is $(\leftarrow \cdot \to \cdot \leftarrow \cdot \to) \Downarrow^{3} (\to \cdot \to \cdot \leftarrow \cdot \leftarrow)$ for which we indeed have $(\leftarrow \cdot \leftarrow \cdot \leftarrow \cdot \to \cdot \to \cdot \to) \Downarrow^{6} \varepsilon$, and both projections have spectrum (1, 1, 1). To prove the claim, it suffices that *filling iff projection* for spectrum-preserving LUCs, cf. Figure 5 right. To enable proving it *by inductions*, we rephrase filling using the notion of *foliage*[3] imaged on the left in Figure 6: a cyclic conversion $Z = Z_1 \cdot \ldots \cdot Z_n$ of length n, together with reductions Ξ_i for $0 \le i \le n$ with $\Xi_0 = \varepsilon = \Xi_n$, and fillings $\Xi_{i-1} \Rrightarrow Z_i \cdot \Xi_i$ if Z_i is a step and $Z_i^{-1} \cdot \Xi_{i-1} \Rrightarrow \Xi_i$ if Z_i is a reverse step.

Theorem 1. *If $\to$ is terminating and $\mathcal{D}$ LUC, then there is a foliage for conversion Z iff $Z \Ddownarrow \varepsilon$. If $\mathcal{D}$ moreover is spectrum-preserving then the foliage and the projection have the same spectra.*

See the appendix for a proof. Here we conclude by observing a filling $\Phi \Rrightarrow \Psi$ for left,right legs Φ, Ψ of a box B gives rise to a foliage for $\Phi^{-1} \cdot \Psi$ with the same spectrum, and *vice versa*.

Remark. The diagrammatic perspective originates with *Newman's II-Lemma* [9, Section 6]: If $\to$ is terminating, there is a diamond in $\mathcal{D}$ for every local peak, and $[\![\,]\!]$ is a typed involutive monoid homomorphism to a typed group mapping diamonds in $\mathcal{D}$ to 0, then $[\![\,]\!]$ maps every conversion cycle to 0. *Proof.* For any conversion Z there is a valley $X \cdot \Upsilon^{-1}$ with $[\![Z]\!] = [\![X \cdot \Upsilon^{-1}]\!]$, by enriching Newman's Lemma with that $[\![\,]\!]$ maps diamonds in $\mathcal{D}$ to 0. Hence if Z is a conversion *cycle*, say on a, then $[\![\Phi^{-1} \cdot Z \cdot \Phi]\!] = [\![\varepsilon]\!]$ for Φ a reduction from a to normal form, so $[\![Z]\!] = 0$.□ What Newman's Lemma is to the Critical Peak Lemma [6, Lemma 2.4] is Newman's II-Lemma to Squier's Finite Derivation Type method [14, 1]; it ought to be better-known.

[2]It expresses *cut*-elimination (transitivity-elimination) replacing two diamonds by a single one *under* them.
[3]Originally introduced for proving [12, Theorem 4].

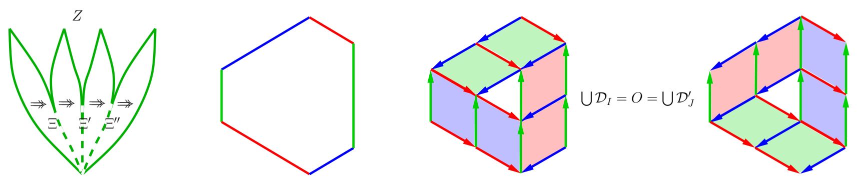

Z
$\Rrightarrow$ $\Rrightarrow$ $\Rrightarrow$ $\Rrightarrow$
Ξ Ξ' Ξ''
$\bigcup\mathcal{D}_I = O = \bigcup\mathcal{D}'_J$

Figure 6: foliage (left), non-fillable box (middle), non-convex non-$\Rrightarrow$-convertible (right)

Conclusion This note illustrates the power of modern confluence techniques: The first three provided solutions out of the box. The fourth, inspired by [3, Proposition 4.16(4.18)], is novel.

The equiangular hexagonal B in the middle in Figure 6 is not zonogonal; filling gets stuck. Still, as shown on the right, the spectra of both tilings $\mathcal{D}_I, \mathcal{D}'_J$ of B for $\bigcup\mathcal{D}_I = O = \bigcup\mathcal{D}'_J$ (a triangle is 'missing') *are* the same [4]. We leave it to future research to investigate whether the techniques presented here can be appropriately adapted (we expect the first and fourth can).

Acknowledgment. *Jan Willem Klop brought the problem of the calissons, it being amenable to rewrite techniques, and 'piling = tiling' (Theorem 1, cf. [7][15, Chapter 8]) to my attention.*

References

[1] D. Ara, A. Burroni, Y. Guiraud, P. Malbos, F. Métayer, and S. Mimram. Polygraphs: From rewriting to higher categories, 2023. doi:10.48550/arXiv.2312.00429.

[2] G. David and C. Tomei. The problem of the calissons. *The American Mathematical Monthly*, 96(5):429–431, 1989. doi:10.1080/00029890.1989.11972212.

[3] P. Dehornoy and alii. *Foundations of Garside Theory.* EMS, 2015. doi:10.4171/139.

[4] E.W. Dijkstra. On the problem of the calissons. Technical Report 1055, University of Texas, 1989. URL: https://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1055.html.

[5] B. Felgenhauer and V. van Oostrom. Proof orders for decreasing diagrams. In *RTA*, volume 21 of *LIPIcs*, pages 174–189, 2013. doi:10.4230/LIPIcs.RTA.2013.174.

[6] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, October 1980. doi:10.1145/322217.322230.

[7] J.W. Klop, V. van Oostrom, and R.C. de Vrijer. Course notes on braids, 1998. URL: http://www.javakade.nl/research/pdf/braids.pdf.

[8] J.-J. Lévy. *Réductions correctes et optimales dans le λ-calcul.* Thèse de doctorat d'état, Université Paris VII, 1978. URL: http://pauillac.inria.fr/~levy/pubs/78phd.pdf.

[9] M.H.A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43:223–243, 1942. doi:10.2307/2269299.

[10] V. van Oostrom. Random descent. In *RTA*, volume 4533 of *Lecture Notes in Computer Science*, pages 314–328, 2007. doi:10.1007/978-3-540-73449-9_24.

[11] V. van Oostrom. Uniform completeness. In *11th IWC*, pages 19–24, 2022. URL: http://cl-informatik.uibk.ac.at/iwc/2022/proceedings.pdf.

[12] V. van Oostrom. Residuation = skolemised confluence. In *12th IWC*, pages 20–25, 2023. URL: http://cl-informatik.uibk.ac.at/iwc/2023/proceedings.pdf.

[13] V. van Oostrom and Y. Toyama. Normalisation by Random Descent. In *FSCD*, volume 52 of *LIPIcs*, pages 32:1–32:18, 2016. doi:10.4230/LIPIcs.FSCD.2016.32.

[14] C.C. Squier, F. Otto, and Y. Kobayashi. A finiteness condition for rewriting systems. *Theoretical Computer Science*, 131(2):271–294, 1994. doi:10.1016/0304-3975(94)90175-9.

[15] Terese. *Term Rewriting Systems.* Cambridge University Press, 2003.

Appendix In the proof of Theorem 1 we *measure* a foliage for conversion Z by the multiset of pairs where the ith object a (*height*) of Z is paired with the number of $\Rightarrow$-root-steps in $Z_i^{-1} \cdot \Xi_{i-1} \Rrightarrow \Xi_i \Rrightarrow Z_{i+1} \cdot \Xi_{i+1}$ if a is the apex of a local peak and 0 otherwise (*width*). The multiset extension of the lexicographic product of $\leftarrow^+$ and $<$ well-foundedly orders measures.

Proof of Theorem 1. We prove the if-direction by induction on the number of steps p in $Z \Downarrow^p \varepsilon$, cf. [12, Theorem 4]. If $p = 0$, we trivially conclude as $Z = \varepsilon$. Otherwise, for some ${}^{\phi}_{X}\diamond^{\psi}_{\Upsilon}$ we have $Z = Z^l \cdot \phi^{-1} \cdot \psi \cdot Z^r$ and $Z \Downarrow Z'$ and $Z' \Downarrow^{p-1} \varepsilon$ for $Z' = Z^l \cdot X \cdot \Upsilon^{-1} \cdot Z^r$. By the IH there is a foliage for Z' (with spectrum that of $Z' \Downarrow^{p-1} \varepsilon$); its subconversions Z^l, Z^r combined with prefixing ϕ to the last reduction of Z^l then give a foliage for Z (with spectrum that of $Z \Downarrow^p \varepsilon$).

We prove the only–if-direction by induction on the measure of the foliage for Z and cases on Z. If Z is a valley, then by definition of foliage $Z = \varepsilon$ using that the legs of diamonds in $\mathcal{D}$ are non-empty, and we conclude. Otherwise, Z has shape $Z^\ell \cdot \phi^{-1} \cdot \psi \cdot Z^r$ with $\phi \cdot \Xi_{i-1} \Rrightarrow \Xi_i \Rrightarrow \psi \cdot \Xi_{i+1}$ and we distinguish cases on the width w of the apex of the local peak.

If $w = 0$ then $\phi = \psi$ and $\Xi_{i-1} \Rrightarrow \Xi_{i+1}$. Then $Z \Downarrow Z'$ for $Z' := Z^\ell \cdot Z^r$ by LUC. Replacing[4] Ξ_{i-1} by Ξ_{i+1} in the foliage for Z^ℓ renders Z' a foliage. We conclude by the IH for Z'.

If $w = 1$ then $\phi \cdot \Xi_{i-1} \Rrightarrow \phi \cdot X \cdot \Xi' \Rightarrow \psi \cdot \Upsilon \cdot \Xi' \Rrightarrow \psi \cdot \Xi_{i+1}$ for some diamond ${}^{\phi}_{X}\diamond^{\psi}_{\Upsilon} \in \mathcal{D}$ and some Ξ', where the displayed $\Rrightarrow$ do not have head-steps. Then $Z \Downarrow Z'$ for $Z' := Z^\ell \cdot X \cdot \Upsilon^{-1} \cdot Z^r$. Replacing (cf. footnote 4) Ξ_{i-1} by $X \cdot \Xi'$ in the foliage for Z^ℓ and replacing Ξ_{i+1} by $\Upsilon \cdot \Xi'$ in the foliage for Z^r, renders Z' a foliage again. We conclude by the IH for Z'.

If $w > 1$ then $\phi \cdot \Xi_{i-1} \Rrightarrow \phi \cdot \Phi \cdot \Xi' \Rightarrow \chi \cdot \Psi \cdot \Xi' \Rrightarrow \chi \cdot X \cdot \Xi'' \Rightarrow \psi' \cdot \Upsilon \cdot \Xi'' \Rrightarrow \psi \cdot \Xi_{i+1}$ for some diamonds ${}^{\phi}_{\Phi}\diamond^{\chi}_{\Psi}, {}^{\chi}_{X}\diamond^{\psi'}_{\Upsilon} \in \mathcal{D}$ and some Ξ', Ξ'', where the first two displayed horizontal reductions do not have head-steps (we may but need not have $\psi' = \psi$). The second induces a foliage for the peak $(\Psi \cdot \Xi')^{-1} \cdot X \cdot \Xi''$ to which the IH applies (by its apex being reached via χ), yielding $(\Psi \cdot \Xi')^{-1} \cdot X \cdot \Xi'' \Downarrow\!\!\!\Downarrow \varepsilon$. By random descent for $\Downarrow$, this projection factors as $\Psi^{-1} \cdot X \cdot \Downarrow\!\!\!\Downarrow X' \cdot \Psi'^{-1}$ and $\Xi'^{-1} \cdot X' \cdot \Psi'^{-1} \cdot \Xi'' \Downarrow\!\!\!\Downarrow \varepsilon$ for some reductions X', Ψ'.

The former $\Downarrow\!\!\!\Downarrow$ combined with two $\Downarrow$-steps for the diamonds gives $\phi^{-1} \cdot \chi \cdot \chi^{-1} \cdot \psi' \Downarrow\!\!\!\Downarrow \Phi \cdot X' \cdot (\Upsilon \cdot \Psi')^{-1}$ for which LUC entails $(\phi \cdot \Phi \cdot X')^{-1} \cdot \psi \cdot \Upsilon \cdot \Psi' \Downarrow\!\!\!\Downarrow \varepsilon$. The if-direction then yields a foliage for it, so $\phi \cdot \Phi \cdot X' \Rrightarrow \psi \cdot \Upsilon \cdot \Psi'$ having exactly 1 head-step (by a diamond for ϕ, ψ).

For the latter $\Downarrow\!\!\!\Downarrow$ the if-direction yields a foliage so $\Xi' \Rrightarrow X' \cdot \hat{\Xi}$ and $\Psi' \cdot \hat{\Xi} \Rrightarrow \Xi''$ for some $\hat{\Xi}$.

Combining both shows that $\phi \cdot \Phi \cdot \Xi' \Rrightarrow \psi' \cdot \Upsilon \cdot \Xi''$ using a single head-step, instead of the two before. Hence we conclude by the IH for the same Z but with this alternative foliage. □

Definition 5. *local semi-lattice* (LSL) is LUC with *commutativity* of $\mathcal{D}$: ${}^{\phi}_{\Phi}\diamond^{\psi}_{\Psi} \in \mathcal{D}$ iff ${}^{\psi}_{\Psi}\diamond^{\phi}_{\Phi} \in \mathcal{D}$.

Observe that to establish LUC it suffices to consider triples ϕ, ψ, χ where $\phi \neq \chi \neq \psi$ since if, say, $\phi = \chi$ then the assumption simplifies to $\phi^{-1} \cdot \psi \Downarrow\!\!\!\Downarrow X \cdot \Upsilon^{-1}$, which is seen to entail the conclusion $(\phi \cdot X)^{-1} \cdot \psi \cdot \Upsilon \Downarrow\!\!\!\Downarrow \varepsilon$ using that peaks between a step and itself were assumed trivial. LSL allows to also assume $\phi \neq \psi$, since if $\phi = \psi$ then $\phi^{-1} \cdot \chi \cdot \chi^{-1} \cdot \phi \Downarrow\!\!\!\Downarrow X \cdot \Upsilon^{-1}$ entails $X = \Phi = \Upsilon$ for ${}^{\phi}_{\Phi}\diamond^{\chi}_{\Psi}, {}^{\chi}_{\Psi}\diamond^{\phi}_{\Phi} \in \mathcal{D}$ so $(\phi \cdot X)^{-1} \cdot \phi \cdot \Upsilon \Downarrow\!\!\!\Downarrow \varepsilon$, using trivial peaks have trivial diamonds.

This resumes our attempts [7][15, Chapter 8] at a theory of *orthogonality* for rewriting and algebra: LSL holds for the $\lambda\beta$-calculus [8] (*local cube*) and for positive braids [3, Example 4.20]. Though neither $\rightarrow_\beta$ nor braids (Artin's σ_i) are terminating, that can be brought about (by *finiteness of family developments* [15] respectively *right-Noetherianity* [3]) making Theorem 1 applicable. From a rewriting / order perspective Theorem 1 aims at showing that *permutation equivalence* = *projection equivalence* [15] / reductions constitute a *semi-lattice* [8] (whence LSL). Contrapositively, it enables showing reductions do *not* have a common reduct (no *upperbound*) by showing projection of their peak does *not* terminate (no *least* upperbound) [3, Example 4.28].

[4] If $Z^\ell = \varepsilon$ replacing is not allowed but not needed: then $\Xi_{i-1} = \varepsilon = \Xi_{i+1}$ as legs of diamonds are non-ε.

A Roadmap to Orthogonality of Conditional Term Rewriting Systems*

Salvador Lucas

DSIC & VRAIN, Universitat Politècnica de València, Spain
slucas@dsic.upv.es

Abstract

There are *different* definitions of orthogonality of conditional term rewriting systems. This paper collects concepts, denominations, notions, and notations which have been important in their development, together with the main works that introduced them.

Intelligence, give me the exact name of things! (Juan Ramón Jiménez)

1 Motivation

A Term Rewriting System (TRS) is *orthogonal* if it is (i) *left-linear* and (ii) *non-overlapping* (or *non-ambiguous*), i.e., it has *no critical pairs* [11, Definition 3.1.1]. When orthogonality is ported to Conditional Term Rewriting Systems (CTRSs) $\mathcal{R}$, consisting of conditional rules $\alpha : \ell \to r \Leftarrow c$ for terms ℓ and r and conditions c, some particularities and difficulties arise:

1. The conditional part c of conditional rules α may have *different* shapes and *interpretations*.
2. Rewriting steps $\sigma(\ell) \to \sigma(r)$ for some substitution σ *require that* the instance $\sigma(c)$ of the conditional part c of the rule "holds". It is possible, though, that no substitution satisfies c, i.e., c is *infeasible* and the rule *useless*. However (in)feasibility is *undecidable*, in general.
3. Left-linearity is always required, but non-ovelappingness (ii) is required in different ways: (ii.1) by a direct definition of overlap without referring conditional critical pairs [13]; (ii.2) referred to orthogonality of the *unconditional part* $\mathcal{R}_u$ of $\mathcal{R}$ [2]; (ii.3) mimicking Klop's definition of orthogonality on the basis of the notion of *conditional critical pair* [18, 14]; (ii.4) as in (ii.3), but considering *feasible* conditional critical pairs only [4, 17, 19].
4. Orthogonality as a *purely syntactic criterion* for confluence *fails* to hold with CTRSs.

Concepts and results addressing such problems often came first. Later on, appropriate denominations were found and became stable and widely used. We briefly discuss this process.

2 Orthogonality of Conditional Term Rewriting Systems

A CTRS $\mathcal{R}$ consists of conditional rules $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n$, where conditions $s_i \approx t_i$ may have different *evaluation semantics* (*join*, *oriented*, *semi-equational*, etc. [11, Definition 4.0.2] and [14, Definition 7.1.3]). Rules are often classified according to the distribution of variables [12, Definition 6.1]: type 1, if $\mathcal{V}ar(r) \cup \mathcal{V}ar(c) \subseteq \mathcal{V}ar(\ell)$; type 2, if $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell)$; type 3, if $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell) \cup \mathcal{V}ar(c)$; and type 4, otherwise. A rule of type n is often called an n-rule. A CTRS $\mathcal{R}$ is called an n-CTRS if all its rules are of type n; if $\mathcal{R}$ contains at least one n-rule which is *not* an m-rule for some $m < n$, then we say that $\mathcal{R}$ is a *proper* n-CTRS.

*Partially supported by MCIN/AEI project PID2021-122830OB-C42 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe" and by the grant CIPROM/2022/6 funded by Generalitat Valenciana.

O'Donnell (1977): Non-overlapping and consistent generalized CTRSs. Regarding conditional rewriting systems, O'Donnell pioneered the analysis of confluence of *Rule Schemata* [13, Definition 45], which *generalize* Rosen's Rule Schemata [16], and can be viewed as sets $\mathcal{R}$ of conditional, left-linear rules $\ell \to r \Leftarrow P(x_1, \ldots, x_n)$, for terms ℓ and r and a predicate P, where all variables in r and $x_1, \ldots, x_n$ already occur in ℓ (only 1-rules are allowed).

Remark 1 (Generalized CTRSs). *In [11, Definition 4.0.2(4)], O'Donnell's rules were slightly generalized*[1] *to $\ell \to r \Leftarrow P_1(x_1, \ldots, x_m), \ldots, P_n(x_1, \ldots, x_m)$ with several predicates $P_1, \ldots, P_n$ and variables $x_1, \ldots, x_m$ which do not need to occur in ℓ. Sets of such rules are called* Generalized CTRSs. *We often use this denomination for O'Donnell systems as well.*

In [13, Definition 48], O'Donnell called *non-overlapping* to *rule schemata* such that, for all instances $\sigma(\ell)$ and $\sigma(\ell')$ of the left-hand sides ℓ and ℓ' of rules $\alpha : \ell \to r \Leftarrow P(x_1, \ldots, x_m)$ and $\alpha' : \ell' \to r' \Leftarrow P'(x'_1, \ldots, x'_n)$, and all nonempty positions $p \neq \Lambda$, if $\sigma(\ell)|_p = \sigma(\ell')$ for some substitution σ, then (NOv.1) $p = q.p'$ for some $q \in \mathcal{P}os_{\mathcal{X}}(\ell)$, e.g., $\ell|_q = x_i$ for some variable x_i, $1 \leq i \leq m$, and (NOv.2) $P(\sigma(x_1), \ldots, \sigma(x_i)[\sigma(r')]_{p'}, \ldots, \sigma(x_m))$ holds. Since the case $\sigma(\ell) = \sigma(\ell')$ is *not* considered, non-overlapping rule schemata may include rules α and α' such that $\sigma(\ell) = \sigma(\ell')$ holds. This situation is handled through the notion of *consistency*. A rule schemata is *consistent* if for all pairs of rule schemata $\ell \to r \Leftarrow P(x_1, \ldots, x_m)$ and $\ell' \to r' \Leftarrow P'(x'_1, \ldots, x'_n)$, if $\sigma(\ell) = \sigma(\ell')$ holds for some substitution σ, then r and r' are identical up to renaming [13, Definition 49]. As a consequence of [13, Theorems 17 and 6]:

Non-overlapping and consistent (generalized) rule schemata are confluent.

Note, however, that O'Donnell's non-overlappingness does *not* admit a purely syntactic checking due to (NOv.2). This was later addressed by Bergstra and Klop by imposing *additional* requirements on the shape and interpretation of the conditional part of rules.

Kaplan (1984): Conditional Critical Pairs for CTRSs. Kaplan introduced conditional critical pairs[2] to investigate confluence of CTRSs.

Definition 2 (Conditional critical pairs [9, 10, Definition 3.2]). *Let $\mathcal{R}$ be a CTRS, $\alpha : \ell \to r \Leftarrow c$ and $\alpha' : \ell' \to r' \Leftarrow c'$ be rules of $\mathcal{R}$ sharing no variable (rename if necessary), and $p \in \mathcal{P}os_{\mathcal{F}}(\ell)$ be a nonvariable position of ℓ such that $\ell|_p$ and ℓ' unify with* mgu *θ. Then,*

$$\langle \theta(\ell[r']_p), \theta(r) \rangle \Leftarrow \theta(c), \theta(c') \tag{1}$$

is a conditional critical pair *(CCP for short) of $\mathcal{R}$, often denoted $\pi_{\alpha,p,\alpha'}$.*

Kaplan distinguishes *feasible* and *infeasible* conditional critical pairs (roughly speaking a CCP $\langle s, t \rangle \Leftarrow c$ is feasible if $\sigma(c)$ holds for some substitution σ) and remarks that *"when testing confluence, only feasible (conditional) critical pairs are relevant"* [9, above Definition 3.2]. Definition 2 is essentially reproduced in, e.g., [4, Definition 3] and [11, Definition 4.0.14(1)]. A CCP $\langle s, t \rangle \Leftarrow c$ is called *trivial* if $s = t$.

Bergstra and Klop (1986): Orthogonality of the unconditional part. In [2, Section 3], Bergstra and Klop considered O'Donnell's conditional systems $\mathcal{R}$ with rules $\ell \to r \Leftarrow P(x_1, \ldots, x_n)$. However, they changed O'Donnell's non-overlappingness by a combination of

[1]Plaisted had already generalized O'Donnell's rules allowing for arbitrary *atoms* and *literals* (i.e., negated atoms) in the conditional part of rules [15]. However, he did not consider any orthogonality-like result.

[2]He named them *Contextual Critical Pairs*.

two properties: (i) the *unconditional part* $\mathcal{R}_u = \{\ell \to r \mid \ell \to r \Leftarrow c \in \mathcal{R}\}$ of $\mathcal{R}$ [2, page 334, item (ii)] must be of *type 0*, i.e., a *left-linear* and *non-ambiguous* (that is, *orthogonal*) TRS [2, page 324]; and (ii) the conditions in rules of $\mathcal{R}$ are *stable*, according to the following:

Definition 3 (Stable condition [2, Definition 3.1])**.** *Let $\mathcal{R}$ be a Generalized CTRS. A condition $P(x_1, \ldots, x_n)$ is called* stable *if for all substitutions σ and $1 \leq i \leq n$, if $\sigma(x_i) \to_{\mathcal{R}} t$ for some term t and $P(\sigma(x_1), \ldots, \sigma(x_i), \ldots, \sigma(x_n))$ holds, then $P(\sigma(x_1), \ldots, t, \ldots, \sigma(x_n))$ holds.*

Stability of conditions in rules, although undecidable, guarantees that (NOv.2) is fulfilled. Requirement (i) above is *more restrictive* than (NOv.1) (as it forbids overlaps at the root of lhs's of rules, not forbidden by (NOv.1)), but implicates O'Donnell's *consistency* as it forbids the situation $\sigma(\ell) = \sigma(\ell')$ for (different) rules $\alpha : \ell \to r \Leftarrow P(x_1, \ldots, x_m)$ and $\alpha' : \ell' \to r' \Leftarrow P'(x'_1, \ldots, x'_n)$. On the basis of O'Donnell results, Bergstra and Klop conclude:

> Generalized CTRSs $\mathcal{R}$ such that $\mathcal{R}_u$ is of type 0 (i.e., orthogonal) and all conditions in rules are stable are confluent [2, Theorem 3.2].

Since the undecidable notion of stability is used, in order to obtain a purely syntactic criterion for confluence, Bergstra and Klop considered 2-CTRSs $\mathcal{R}$ such that $\mathcal{R}_u$ is orthogonal and $\mathcal{R}$ consists of rules $\alpha : \ell \to r \Leftarrow c$ with c of the form $s_1 \approx t_1, \ldots, s_n \approx t_n$ for terms s_i, t_i, $1 \leq i \leq n$, and $\approx$ is given a fixed interpretation so that stability of c is guaranteed [2, Definition 2.1(i)]:

- *Type I CTRSs*, where conditions $s \approx t \in c$ are interpreted as *conversion* statements $s \leftrightarrow^* t$ (nowadays we call them *semi-equational CTRSs*).
- *Type III$_n$ CTRSs*, where for all conditions $s \approx t \in c$, t is a *ground* $\mathcal{R}_u$-irreducible term and $\approx$ is interpreted as *reachability* $\to^*$ (nowadays we call them *normal CTRSs*).

[2, Corollary 3.3 and Theorem 3.5] prove confluence of CTRSs of type I and III$_n$, respectively.

Dershowitz, Okada, and Sivakumar (1987): Non-overlappingness; overlays. In contrast to O'Donnell, [4] defines *non-overlappingness* using *conditional critical pairs* (1):

> A CTRS is non-overlapping if it has no feasible, non-trivial (conditional) critical pairs [4, Definition 4].

If $p = \Lambda$ in (1), then (1) is an *overlay* [4, Definition 8]. In contrast to [2], Dershowitz, Okada and Sivakumar also considered 3-CTRSs whose rules *may non-trivially overlay* themselves.

Example 1. *Consider the 3-CTRS $\mathcal{R}$ [4, Conclusion]:*

$$\mathsf{a} \to \mathsf{f}(x) \Leftarrow \mathsf{p}(x) \approx \mathsf{tt} \qquad \mathsf{p}(\mathsf{b}) \to \mathsf{tt} \qquad \mathsf{p}(\mathsf{c}) \to \mathsf{tt}$$

The CCP $\langle \mathsf{f}(x), \mathsf{f}(x') \rangle \Leftarrow \mathsf{p}(x) \approx \mathsf{tt}, \mathsf{p}(x') \approx \mathsf{tt}$ is feasible and not trivial. Thus, $\mathcal{R}$ is overlapping.

In contrast to [4, 9, 10], Avenhaus and Loría-Sáenz distinguish *proper* and *improper* CCPs.

Definition 4 (Proper and improper CCPs [1, Def. 4.2])**.** *Let $\mathcal{R}$ be a CTRS and $\pi_{\alpha,p,\alpha'}$ be a CCP of $\mathcal{R}$. If α and α' are renamed versions of the* same *rule and $p = \Lambda$, then $\pi_{\alpha,p,\alpha'}$ is called* improper*; otherwise, it is called* proper*.*

However, no notion of (weak) orthogonality of CTRSs is given in [4] or [1].

Klop (1992): (Weakly) Orthogonal 2-CTRSs. Klop adopted Dershowitz and Jouannaud's term *orthogonality* [3, Section 7.2] to provide the current definition of (weakly) orthogonal TRSs [11, Definition 2.1.1]. He also adopted the notions of semi-equational and normal CTRSs from [5] (among others). Following [2], Klop introduced the following.

Definition 5 ((Weakly) Orthogonal CTRS [11, Definition 4.0.5])**.** *A CTRS (of any type: semi-equational, normal,...) is (weakly) orthogonal if $\mathcal{R}_u$ is (weakly) orthogonal.*

Definition 5 implicitly restricts (weak) orthogonality to 2-CTRSs, as $\mathcal{R}_u$ must be a TRS and this is possible only if $\mathcal{R}$ is a 2-CTRS. Taking full power of O'Donnell's results for Generalized CTRSs, actually covering *weakly orthogonal* CTRSs, Klop establishes:

> Weakly orthogonal semi-equational CTRSs are confluent [11, Corollary 4.0.8].
> Weakly orthogonal normal CTRSs are confluent [11, Theorem 4.0.10].

Unfortunately, these results do *not* extend to other well-known evaluation modes for the conditions $s \approx t$ in rules (e.g., *joinability*, where $s \approx t$ is interpreted as joinability of terms s and t, written $s \downarrow t$, or *reachability*, where $s \approx t$ is interpreted as reachability $s \to^* t$).[3]

Gramlich (1994): A more restrictive definition of conditional critical pair. In [7, Definition 2], Gramlich defines a notion of conditional critical pair for 2-CTRSs so that improper conditional critical pairs in Definition 4 are explicitly *discarded.* Such a more restricted notion of conditional critical pair can be found in Ohlebusch's book for *arbitrary* CTRSs [14, Definition 7.1.8(1)].

Definition 6 (Conditional critical pair [7, Def. 2], [14, Def. 7.1.8(1)])**.** *Let $\mathcal{R}$ be a CTRS, $\alpha : \ell \to r \Leftarrow c$ and $\alpha' : \ell' \to r' \Leftarrow c'$ be rules of $\mathcal{R}$ sharing no variable (rename if necessary), and $p \in \mathcal{P}os_{\mathcal{F}}(\ell)$ be a nonvariable position of ℓ such that $\ell|_p$ and ℓ' unify with* mgu θ*. We call*

$$\langle \theta(\ell[r']_p), \theta(r)\rangle \Leftarrow \theta(c), \theta(c')$$

a conditional critical pair *of $\mathcal{R}$.* ***If α and α' are renamed versions of the same rule, we do not consider the case $p = \Lambda$.***

The **important change** with respect to Definition 4 is that *improper* CCPs are neglected.

Example 2. *The (improper) CCP in Example 1 obtained from the root overlap of renamed versions of the conditional rule is* dimissed *as a conditional critical pair by Definition 6.*

Hanus (1994): Almost orthogonal CTRSs. Hanus' definition of almost orthogonality of CTRSs follows the definition style of [2], i.e., the property is referred to the unconditional part $\mathcal{R}_u$. Thus, it is also restricted to 2-CTRSs, although this is not explicit in [8]. A left-linear TRS is *almost orthogonal* if for each pair of rules $\ell \to r$, $\ell' \to r'$, nonvariable subterm $\ell|_p$ of ℓ, and *mgu* θ for $\ell|_p$ and ℓ', p is the root position Λ and $\theta(r) = \theta(r')$ [8, page 669].

Definition 7. *A 2-CTRS is* almost orthogonal *if $\mathcal{R}_u$ is almost orthogonal [8, page 669].*

[3]Some authors include additional requirements regarding the conditions of rules to provide a more restrictive notion of orthogonality of CTRSs deriving more "direct" confluence results, e.g., [6, Definition 10.9].

Suzuki, Middeldorp, and Ida (1995): Confluence of orthogonal 3-CTRSs. Although [19] defines orthogonality of CTRSs $\mathcal{R}$ as orthogonality of $\mathcal{R}_u$, in [18], mimicking the usual definition of orthogonality of TRSs (based on the absence of critical pairs), but using Definition 6, where only *proper* CCPs are considered, they provided the following.

Definition 8 (Orthogonality of CTRSs [18, page 3])**.** *A left-linear CTRS without (*proper*) conditional critical pairs is called* orthogonal.

Accordingly, $\mathcal{R}$ in Example 1 is orthogonal. They prove new results for *oriented* and *join* CTRSs, where conditions $s \approx t$ are treated as reachability and joinability tests, respectively:

> Orthogonal properly oriented right-stable 3-CTRSs are confluent [19, Cor. 4.7].
> Orthogonal properly oriented right-stable join 3-CTRSs are confluent [19, Cor. 5.3].

Here, a 3-CTRS $\mathcal{R}$ is *properly oriented* if every rule $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n$ satisfies: if $\mathcal{V}ar(r) \not\subseteq \mathcal{V}ar(\ell)$, then $\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(\ell) \cup \bigcup_{j=1}^{i-1} \mathcal{V}ar(t_j)$ for all $1 \leq i \leq n$ [19, Definition 3.1]. A CTRS is *right-stable* if for every rule $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n \in \mathcal{R}$ and for all $1 \leq i \leq n$, (a) $(\mathcal{V}ar(\ell) \cup \bigcup_{j=1}^{i-1} \mathcal{V}ar(s_j \approx t_j) \cup \mathcal{V}ar(s_i)) \cap \mathcal{V}ar(t_i) = \emptyset$, and (b) t_i is either a linear constructor term or a ground $\mathcal{R}_u$-irreducible term [19, Definition 3.6]. The authors claim that their results apply to Hanus' almost orthogonal 2-CTRSs as well [19, last paragraph of Section 6].

Suzuki, Middeldorp, and Ida (1995): Infeasible CCPs in orthogonal CTRSs. In [19, Section 7], the authors consider the possibility of discarding *infeasible* critical pairs which (as anticipated by Kaplan) would be excluded from the analysis of confluence of CTRSs. They say that *orthogonality can be strengthened by allowing infeasible (conditional) critical pairs.*

Ohlebusch (2002): Almost and weak orthogonality as the absence of *proper* CCPs. Extending Definition 8, Ohlebusch provides the following.[4]

Definition 9 (Orthogonality of CTRSs [14, Definition 7.1.10])**.** *Let $\mathcal{R}$ be a left-linear CTRS. Then, $\mathcal{R}$ is* orthogonal *if it has no* proper *conditional critical pair; $\mathcal{R}$ is* almost orthogonal *if every* proper *conditional critical pair is a trivial overlay; $\mathcal{R}$ is* weakly orthogonal *if every* proper *conditional critical pair is trivial.*

Sternagel and Sternagel (2016): orthogonality *modulo feasibility.* Following [4, Definition 4] and [19, Section 7], Sternagel & Sternagel's definition of (almost) orthogonality of (oriented) CTRSs *modulo infeasibility* restricts the attention to *feasible* (proper) conditional critical pairs, thus making explicit the aforementioned idea of Suzuki, Middeldorp and Ida.

Definition 10 ([17, Definition 2])**.** *A left-linear CTRS $\mathcal{R}$ is orthogonal (modulo infeasibility) if for all pairs $\alpha : \ell \to r \Leftarrow c$ and $\alpha' : \ell' \to r' \Leftarrow c'$ of (possibly renamed) variable disjoint rules of $\mathcal{R}$, and $p \in \mathcal{P}os_{\mathcal{F}}(\ell)$ such that $\pi_{\alpha,p,\alpha'}$ exists, then $\pi_{\alpha,p,\alpha'}$ is improper or infeasible.*

Dershowitz et al.'s notion of *non-overlapping* CTRS, see above, anticipated this. Clearly, orthogonality according to Definition 8 implies orthogonality modulo infeasibility.

[4]In personal communication, Ohlebusch said that he origin of this definition is unclear.

3 Different notions of orthogonality of CTRSs

The previous definitions of orthogonality are *not* equivalent, in general, even for 1-CTRSs.

Example 3. *Consider the* oriented *1-CTRS* $\mathcal{R} = \{\mathsf{a} \to \mathsf{b} \Leftarrow \mathsf{b} \approx \mathsf{c}, \mathsf{a} \to \mathsf{b} \Leftarrow \mathsf{b} \approx \mathsf{a}, \mathsf{c} \to \mathsf{b}\}$*.*

- *By Definition 5,* $\mathcal{R}$ *is orthogonal, as* $\mathcal{R}_u = \{\mathsf{a} \to \mathsf{b}, \mathsf{c} \to \mathsf{b}\}$ *is.*
- *By Definition 8,* $\mathcal{R}$ *is* not *orthogonal: there is a CCP* $\langle \mathsf{b}, \mathsf{b} \rangle \Leftarrow \mathsf{b} \approx \mathsf{c}, \mathsf{b} \approx \mathsf{a}$*.*
- *By Definition 10,* $\mathcal{R}$ *is* orthogonal modulo infeasibility*: the CCP is* infeasible *as* $\mathsf{b} \not\to^*_{\mathcal{R}} \mathsf{c}$*.*

Now, if a join *semantics is assumed, then* $\mathcal{R}$ *is orthogonal according to Definition 5, but the CCP is* not *infeasible, as* $\mathsf{b} \downarrow \mathsf{c}$ *and hence* $\mathsf{b} \downarrow \mathsf{a}$*. Thus,* $\mathcal{R}$ *is* not *orthogonal modulo infeasibility.*

Finally, note that Definitions 8 and 10 depend on the notion of CCP in Definition 6, which correspond to *proper* CCPs in Definition 4. Do we obtain a different definition of (almost, weak) orthogonality if *every* CCP is considered? Since every rule $\ell \to r \Leftarrow c$ induces an improper CCP, no CTRS would be orthogonal. For instance, the 3-CTRS $\mathcal{R}$ in Example 1 would *not* be orthogonal now. As for almost and weak orthogonality, all CCPs should be trivial. Given variable-disjoint rules $\alpha : \ell \to r \Leftarrow c$ and $\alpha' : \ell \to r \Leftarrow c$, if $\pi_{\alpha,p,\alpha'}$ is trivial, then both α and α' are 2-rules, i.e., only 2-CTRSs would be almost or weakly orthogonal. Since improper CCPs of 2-rules are trivial, nothing new is obtained regarding almost/weak orthogonality.

4 Conclusions

We have traced the development of the notion of orthogonality of CTRSs, generalizing the corresponding notion for TRSs. Since the early research by O'Donnell, generalizing Rosen's work for subtree replacement systems by considering *conditional rule schemata*, different notions have been proposed. Bergstra and Klop referred orthogonality of CTRSs $\mathcal{R}$ to that of the *unconditional part* $\mathcal{R}_u$. Suzuki, Middeldorp, and Ida used (the absence of) *proper CCPs* as a suitable definition of orthogonality for left-linear CTRSs. Sternagel and Sternagel introduced orthogonality *modulo infeasibility*, exploiting an early observation by Kaplan (and then by Suzuki et al.) about the harmlessness of *infeasible CCPs* regarding confluence of CTRSs. Example 3 shows that such notions do *not* coincide, even for 1-CTRSs, although that of Definition 8 implies that of Definition 10. The *evaluation semantics* of the conditions also matters.

Acknoledgments. I thank Nachum Dershowitz, Michael Hanus, and Enno Ohlebusch for kindly attending my requests about terminological and historical aspects of orthogonality. I also thank Aart Middeldorp for providing access to [18]. Finally, I thank the reviewers for many suggestions for (future) improvements.

References

[1] Jürgen Avenhaus and Carlos Loría-Sáenz. On Conditional Rewrite Systems with Extra Variables and Deterministic Logic Programs. In Frank Pfenning, editor, *Logic Programming and Automated Reasoning, 5th International Conference, LPAR'94, Proceedings*, volume 822 of *Lecture Notes in Computer Science*, pages 215–229. Springer, 1994.

[2] Jan A. Bergstra and Jan Willem Klop. Conditional rewrite rules: Confluence and termination. *J. Comput. Syst. Sci.*, 32(3):323–362, 1986.

[3] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite Systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 243–320. Elsevier and MIT Press, 1990.

[4] Nachum Dershowitz, Mitsuhiro Okada, and G. Sivakumar. Confluence of Conditional Rewrite Systems. In Stéphane Kaplan and Jean-Pierre Jouannaud, editors, *Conditional Term Rewriting Systems, 1st International Workshop, Proceedings*, volume 308 of *Lecture Notes in Computer Science*, pages 31–44. Springer, 1987.

[5] Nachum Dershowitz, Mitsuhiro Okada, and G. Sivakumar. Canonical conditional rewrite systems. In Ewing L. Lusk and Ross A. Overbeek, editors, *9th International Conference on Automated Deduction, 1988, Proceedings*, volume 310 of *Lecture Notes in Computer Science*, pages 538–549. Springer, 1988.

[6] Nachum Dershowitz and David A. Plaisted. Rewriting. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 535–610. Elsevier and MIT Press, 2001.

[7] Bernhard Gramlich. On termination and confluence of conditional rewrite systems. In Nachum Dershowitz and Naomi Lindenstrauss, editors, *Conditional and Typed Rewriting Systems, 4th International Workshop, CTRS-94, Notes in Computer Science*, pages 166–185. Springer, 1994.

[8] Michael Hanus. On extra variables in (equational) logic programming. In Leon Sterling, editor, *Logic Programming, Proceedings of the Twelfth International Conference on Logic Programming, 13-16, 1995*, pages 665–679. MIT Press, 1995.

[9] Stéphane Kaplan. Fair conditional term rewriting systems: Unification, termination, and confluence. In Hans-Jörg Kreowski, editor, *Recent Trends in Data Type Specification, 3rd Workshop on Theory and Applications of Abstract Data Types, 1984, Selected Papers*, volume 116 of *Informatik-Fachberichte*, pages 136–155. Springer, 1984.

[10] Stéphane Kaplan. Simplifying Conditional Term Rewriting Systems: Unification, Termination and Confluence. *J. Symb. Comput.*, 4(3):295–334, 1987.

[11] J. W. Klop. Chapter 1: Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaurn, editors, *Handbook of Logic in Computer Science*, pages 1–116. Oxford University Press, 1992.

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

[13] Michael J. O'Donnell. *Computing in Systems Described by Equations*, volume 58 of *Lecture Notes in Computer Science*. Springer, 1977.

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

[15] David A. Plaisted. A Logic for Conditional Term Rewriting Systems. In Stéphane Kaplan and Jean-Pierre Jouannaud, editors, *Conditional Term Rewriting Systems, 1st International Workshop, Proceedings*, volume 308 of *Lecture Notes in Computer Science*, pages 212–227. Springer, 1987.

[16] Barry K. Rosen. Tree-Manipulating Systems and Church-Rosser Theorems. *J. ACM*, 20(1):160–187, 1973.

[17] Christian Sternagel and Thomas Sternagel. Certifying confluence of almost orthogonal ctrss via exact tree automata completion. In Delia Kesner and Brigitte Pientka, editors, *1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016 Proceedings in Informatics (LIPIcs)*, pages 29:1–29:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[18] Taro Suzuki, Aart Middeldorp, and Tetsuo Ida. Level-Confluence of Conditional Rewrite Systems with Extra Variables in Right-Hand Sides. 1995. revised version of [19], available: `http://cl-informatik.uibk.ac.at/users/ami/research/publications/proceedings/95rta.pdf`.

[19] Taro Suzuki, Aart Middeldorp, and Tetsuo Ida. Level-Confluence of Conditional Rewrite Systems with Extra Variables in Right-Hand Sides. In Jieh Hsiang, editor, *Rewriting Techniques and Applications, 6th International Conference, RTA-95, Proceedings*, volume 914 of *Lecture Notes in Computer Science*, pages 179–193. Springer, 1995.

On Non-triviality of the Hierarchy of Decreasing Church-Rosser Abstract Rewriting Systems

Ievgen Ivanov

Taras Shevchenko National University of Kyiv, Ukraine
ivanov.eugen@gmail.com

Abstract

We propose an answer to a question posed by J. Endrullis, J.W. Klop, and R. Overbeek about the existence of a confluent abstract rewriting system (ARS) that does not belong to the class DCR_2 of confluent ARS for which confluence can be proved with the help of the decreasing diagrams method using the label set $\{0, 1\}$ ordered in such a way that $0 < 1$.

We show that there exists an uncountable confluent ARS outside of the class DCR_2 and provide a formal proof of a formalized version of this statement in Isabelle proof assistant using HOL logic. We also show that the hierarchy of decreasing Church-Rosser ARS introduced by J. Endrullis, J.W. Klop, R. Overbeek does not collapse at the level 2.

1 Introduction

Perhaps the most widely known confluence criterion for abstract rewriting systems (ARS) is Newman's lemma [8, 4], which, in modern reformulations, establishes the equivalence between the confluence and the local confluence property for terminating ARS. An obvious limitation of this lemma is the termination assumption. A frequently cited example of an acyclic, inductive, locally confluent, but *not* confluent relation due to Newman [4, Figure 6a] (Newman's counterexample) and an example of a finite, locally confluent, but *not* confluent relation with cycles due to Hindley [4, Figure 6b] show that the termination assumption cannot be omitted from the statement of this lemma in the general case. Besides, there exists a strengthened version of Newman's counterexample [6, Theorem 2] that is an acyclic, locally confluent, but *not* confluent ARS $(A, \to)$ such that $(A, \to^*)$ is a *strictly inductive* poset, i.e. a partially ordered set where every nonempty chain has a *least* upper bound, where $\to^*$ denotes the reflexive-transitive closure of $\to$ on A (however, unlike the regular Newman's counterexample which is countable, the mentioned strengthened Newman's counterexample is necessarily uncountable).

Van Oostrom's *decreasing diagrams* method [10] provides one of the approaches to proving confluence of ARS that overcomes limitations of Newman's lemma. Semi-formally, to prove that an ARS is confluent using this method, one needs to find a set of labels I together with a well-founded partial order $\prec$ on it, and find a labeled version of an ARS that satisfies a condition reminiscent to the local confluence, but with special constraints on how labels associated with reduction steps that appear in this condition are related by $\prec$. In a rigorous way it can be formulated using the notion of a *decreasing Church-Rosser* (DCR) ARS [10].

Importantly, unlike the case of confluence proofs based on Newman's lemma, the decreasing diagrams method requires one to choose/guess a set of labels, a well-founded order on it, an assignment of labels to rewrite steps that can be thought of as method parameters. This leads to questions about the possible redundancy of these parameters and the possibility of minimization of the parameter space. In part, a setting for studying the latter questions can be formalized using a hierarchy of subclasses of DCR ARS indexed by ordinals introduced by J. Endrullis, J.W. Klop, and R. Overbeek in [2]:

$$DCR_0 \subseteq DCR_1 \subseteq DCR_2 \subseteq ... \tag{1}$$

Semi-formally, for each ordinal α, DCR_α is the class of confluent ARS for which confluence can be proved with the help of the decreasing diagrams method using a fixed set of labels $\{\beta \mid \beta < \alpha\}$ (ordinals less than α) and a fixed order on them that is a restriction of the usual order on ordinals to $\{\beta \mid \beta < \alpha\}$ (note that since a label set and an order on it are fixed, only an assignment of labels to rewrite steps remains a method parameter).

In the works [2, 3] it was shown that every confluent ARS with at most countable set of elements belongs to the class DCR_2, so the hierarchy of intersections $DCR_\alpha \cap CNT$, where CNT is the class of ARS with at most countable set of elements, collapses at the level 2. However, in [2, 3] various questions concerning the hierarchy (1) in the general case (without countability assumption) were not studied, and a number of open problems were formulated.

In particular, [3, Open Problem 6.3] asks whether there is a confluent uncountable ARS outside of the class DCR_2. In this work we show that the answer to this question is affirmative (Theorem 1), so the class DCR_2 does *not* coincide with the class of all confluent ARS. We provide a concrete example of such an ARS. Additionally, we show that this example belongs to the class DCR_3 (Theorem 2). Thus the hierarchy (1) does *not* collapse at the level 2, *unlike* the countable DCR ARS hierarchy.

2 Preliminaries

An abstract rewriting system (ARS) is a pair $(A, \rightarrow)$, where A is a set and $\rightarrow \subseteq A \times A$ is a binary relation (called reduction or rewrite relation). We will use the following notation:

- $\rightarrow^+$ is the transitive closure of $\rightarrow$
- $\rightarrow^=$ is the reflexive closure of $\rightarrow$ (on A)
- $\rightarrow^*$ is the reflexive-transitive closure of $\rightarrow$ (on A)
- $\neg$, $\vee$, $\wedge$, $\Rightarrow$ are logical negation, disjunction, conjunction, and implication respectively.

An ARS $(A, \rightarrow)$ is

- confluent, if $\forall a, b, c \in A\ (a \rightarrow^* b \wedge a \rightarrow^* c \Rightarrow \exists d \in A\ (b \rightarrow^* d \wedge c \rightarrow^* d))$.

For any ordinal α denote as $\curlyvee\alpha$ the set of all ordinals below β, i.e. $\{\beta \mid \beta < \alpha\}$, where $<$ is the usual order on ordinals. Note that in the case of usual set-theoretic definitions of ordinals, e.g. [7, Chapter 4], $<$ is the membership $\in$ and $\curlyvee\alpha$ is the ordinal α itself, however, we will use $\curlyvee\alpha$ to highlight that α is considered as a set of ordinals.

Remark 2.1. *A separate notation for $\{\beta \mid \beta < \alpha\}$ is also useful for some approaches to ordinals in non-set-theoretic contexts [1, 9], e.g. used in some proof assistants, where $<$ is not $\in$.*

The following definition is based on [3, Definition 4.4] and [3, Definition 4.2].

Definition 2.1. *Let γ be a (fixed) ordinal. An ARS $(A, \rightarrow)$ belongs to the class DCR_γ, if there exists an indexed family $(\rightarrow_\alpha)_{\alpha \in (\curlyvee\gamma)}$ of binary relations on A (indexed by ordinals $\alpha < \gamma$) such that for every ordinals $\alpha, \beta \in (\curlyvee\gamma)$ and for every $a, b, c \in A$ the following implication holds: if*

$$a \rightarrow_\alpha b \wedge a \rightarrow_\beta c, \tag{2}$$

then there exist elements $b', b'', c', c'', d \in A$ such that

$$\left(b \overset{*}{\underset{\curlyvee\alpha}{\dashrightarrow}} b' \overset{=}{\underset{\{\beta\}}{\dashrightarrow}} b'' \overset{*}{\underset{\curlyvee\alpha \cup \curlyvee\beta}{\dashrightarrow}} d\right) \wedge \left(c \overset{*}{\underset{\curlyvee\beta}{\dashrightarrow}} c' \overset{=}{\underset{\{\alpha\}}{\dashrightarrow}} c'' \overset{*}{\underset{\curlyvee\beta \cup \curlyvee\alpha}{\dashrightarrow}} d\right), \tag{3}$$

where symbols of the form $\overset{=}{\underset{K}{\dashrightarrow}}$ *and* $\overset{*}{\underset{K}{\dashrightarrow}}$ *, where* K *is an expression that denotes a set of ordinals, have the following meaning:*

$$\overset{=}{\underset{K}{\dashrightarrow}} = \{(a,a)\,|\, a \in A\} \cup \bigcup_{\kappa\in K} \to_\kappa \quad \textit{and} \quad \overset{*}{\underset{K}{\dashrightarrow}} = \{(a,a)\,|\, a \in A\} \cup \left(\bigcup_{\kappa\in K} \to_\kappa\right)^{+}. \tag{4}$$

An illustration for Definition 2.1 is given below.

$$\begin{tikzcd}
a \arrow[rrr, "\alpha"'] \arrow[ddd, "\beta"'] & & & b \arrow[d, dashed, "\curlyvee\alpha"', "*"] \\
& & & b' \arrow[d, dashed, "\{\beta\}"', "="] \\
& & & b'' \arrow[d, dashed, "\curlyvee\alpha\cup\curlyvee\beta"', "*"] \\
c \arrow[r, dashed, "*", "\curlyvee\beta"'] & c' \arrow[r, dashed, "=", "\{\alpha\}"'] & c'' \arrow[r, dashed, "*", "\curlyvee\beta\cup\curlyvee\alpha"'] & d
\end{tikzcd}$$

Figure 1: Illustration of the conditions (2), (3) from Definition 2.1. The meaning of arrows is described in (4). This figure is analogous to [3, Figure 9] and [10, Figure 2], however, note that we do *not* use the notation "$\twoheadrightarrow$" for the reflexive-transitive closure.

The following propositions make the definitions of the classes DCR_2, DCR_3 more explicit:

Proposition 1. *An ARS* $(A, \to)$ *belongs to the class* DCR_2 *if and only if there exist binary relations* $\to_0 \subseteq A\times A$ *and* $\to_1 \subseteq A \times A$ *such that* $\to = (\to_0 \cup \to_1)$ *and the following 3 conditions hold, where for* $i = 0, 1$, $\to_i^=$ *and* $\to_i^*$ *denote the reflexive closure of* $\to_i$ *and the reflexive-transitive closure of* $\to_i$ *on* A *respectively:*

1. $\forall a, b, c \in A\ ((a \to_0 b \wedge a \to_0 c) \Rightarrow \exists d \in A\ (b \to_0^= d \wedge c \to_0^= d))$
2. $\forall a, b, c \in A\ ((a \to_0 b \wedge a \to_1 c) \Rightarrow \exists b', d \in A\ (b \to_1^= b' \wedge b' \to_0^* d \wedge c \to_0^* d))$
3. $\forall a, b, c \in A\ ((a \to_1 b \wedge a \to_1 c) \Rightarrow$
   $\exists b', b'', c', c'', d \in A\ (b \to_0^* b' \wedge b' \to_1^= b'' \wedge b'' \to_0^* d \wedge c \to_0^* c' \wedge c' \to_1^= c'' \wedge c'' \to_0^* d)).$

Proof. Follows immediately from Definition 2.1 by assuming that $\gamma = 2$ and taking into account symmetry of the cases $\alpha = 0, \beta = 1$ and $\alpha = 1, \beta = 0$. □

Proposition 2. *An ARS* $(A, \to)$ *belongs to the class* DCR_3 *if and only if there exist relations* $\to_0, \to_1, \to_2 \subseteq A \times A$ *such that* $\to = (\to_0 \cup \to_1 \cup \to_2)$ *and the next 6 conditions hold, where*
– *for* $i = 0, 1, 2$, $\to_i^=$ *denotes the reflexive closure of* $\to_i$ *on* A
– *for* $i = 0, 1, 2$, $\to_i^*$ *denotes the reflexive-transitive closure of* $\to_i$ *on* A
– $\to_{01}^*$ *denotes the reflexive-transitive closure of* $\to_0 \cup \to_1$ *on* A*:*

1. $\forall a, b, c \in A\ ((a \to_0 b \wedge a \to_0 c) \Rightarrow \exists d \in A\ (b \to_0^= d \wedge c \to_0^= d))$

2. $\forall a, b, c \in A\ ((a \to_0 b \wedge a \to_1 c) \Rightarrow \exists b', d \in A\ (b \to_1^= b' \wedge b' \to_0^* d \wedge c \to_0^* d))$

3. $\forall a, b, c \in A\ ((a \to_1 b \wedge a \to_1 c) \Rightarrow$
   $\exists b', b'', c', c'', d \in A\ (b \to_0^* b' \wedge b' \to_1^= b'' \wedge b'' \to_0^* d \wedge c \to_0^* c' \wedge c' \to_1^= c'' \wedge c'' \to_0^* d))$.

4. $\forall a, b, c \in A\ ((a \to_0 b \wedge a \to_2 c) \Rightarrow \exists b', d \in A\ (b \to_2^= b' \wedge b' \to_{01}^* d \wedge c \to_{01}^* d))$

5. $\forall a, b, c \in A\ ((a \to_1 b \wedge a \to_2 c) \Rightarrow \exists b', b'', d \in A\ (b \to_0^* b' \wedge b' \to_2^= b'' \wedge b'' \to_{01}^* d \wedge c \to_{01}^* d))$

6. $\forall a, b, c \in A\ ((a \to_2 b \wedge a \to_2 c) \Rightarrow$
   $\exists b', b'', c', c'', d \in A\ (b \to_{01}^* b' \wedge b' \to_2^= b'' \wedge b'' \to_{01}^* d \wedge c \to_{01}^* c' \wedge c' \to_2^= c'' \wedge c'' \to_{01}^* d))$.

Proof. Follows from Definition 2.1 by assuming that $\gamma = 3$ and taking into account symmetry of the cases $(\alpha, \beta) \in \{(0,1), (1,0)\}$, $(\alpha, \beta) \in \{(0,2), (2,0)\}$, and $(\alpha, \beta) \in \{(1,2), (2,1)\}$. □

3 Main Result

Let $\mathbb{R}$ denote the set of real numbers and $2^{\mathbb{R}}$ denote the power set of $\mathbb{R}$.

Let $\mathcal{P}_f(\mathbb{R})$ denote the set of all finite subsets of $\mathbb{R}$.

Remark 3.1. *The set $\mathbb{R}$ is used here for concreteness and simplicity. In the example given below it can be replaced with any other uncountable set.*

Example 1. *Let $(E, \to_E)$ be an ARS, where $E = \{0,1,2,3,4,5,6,7,8\} \times 2^{\mathbb{R}} \times 2^{\mathbb{R}} \times 2^{\mathbb{R}}$ and $\to_E$ is a binary relation on E such that for every $(n, A, B, C), (n', A', B', C') \in E$ a rewrite step $(n, A, B, C) \to_E (n', A', B', C')$ holds if and only if one of the following 9 conditions holds:*

1. $n = 0 \wedge A = \emptyset \wedge B = \emptyset \wedge C = \emptyset \wedge n' = 1 \wedge A' \in \mathcal{P}_f(\mathbb{R}) \wedge B' = \emptyset \wedge C' = \emptyset$

2. $n = 1 \wedge A \in \mathcal{P}_f(\mathbb{R}) \wedge B = \emptyset \wedge C = \emptyset \wedge n' = 2 \wedge A' = A \wedge B' = \emptyset \wedge C' = \emptyset$

3. $n = 2 \wedge A \in \mathcal{P}_f(\mathbb{R}) \wedge B = \emptyset \wedge C = \emptyset \wedge n' = 3 \wedge A' = A \wedge B' \in \mathcal{P}_f(\mathbb{R}) \wedge C' = \emptyset$

4. $n = 3 \wedge A \in \mathcal{P}_f(\mathbb{R}) \wedge B \in \mathcal{P}_f(\mathbb{R}) \wedge C = \emptyset \wedge n' = 4 \wedge A' = A \wedge B' = B \wedge C' = \emptyset$

5. $n = 4 \wedge A \in \mathcal{P}_f(\mathbb{R}) \wedge B \in \mathcal{P}_f(\mathbb{R}) \wedge C = \emptyset \wedge n' = 5 \wedge A' = A \wedge B' = B \wedge C' \in \mathcal{P}_f(\mathbb{R})$

6. $n = 5 \wedge A \in \mathcal{P}_f(\mathbb{R}) \wedge B \in \mathcal{P}_f(\mathbb{R}) \wedge C \in \mathcal{P}_f(\mathbb{R}) \wedge n' = 6 \wedge A' = A \wedge B' = B \wedge C' = C$

7. $n = 6 \wedge A \in \mathcal{P}_f(\mathbb{R}) \wedge B \in \mathcal{P}_f(\mathbb{R}) \wedge C \in \mathcal{P}_f(\mathbb{R}) \wedge n' = 7 \wedge A' = A \cup B \cup C \wedge B' = A' \wedge C' = A'$

8. $n = 7 \wedge A \in \mathcal{P}_f(\mathbb{R}) \wedge B = A \wedge C = A \wedge n' = 8 \wedge A' = A \wedge B' = A' \wedge C' = A'$

9. $n = 8 \wedge A \in \mathcal{P}_f(\mathbb{R}) \wedge B = A \wedge C = A \wedge n' = 7 \wedge A \subset A' \wedge A' \in \mathcal{P}_f(\mathbb{R}) \wedge B' = A' \wedge C' = A'$.

Theorem 1. *The ARS $(E, \to_E)$ defined in Example 1 is confluent, but it does not belong to the class DCR_2.*

Proof. A machine-checked proof of a formalized version of Theorem 1 in Isabelle proof assistant is available in the supplementary material for this paper [5, lines 1142–1145, theorem `thm_1`].

More specifically, the formal proof shows that $(E, \to_E)$ is a confluent ARS, but there is no pair of relations $\to_0 \subseteq E \times E$ and $\to_1 \subseteq E \times E$ such that $\to_E = (\to_0 \cup \to_1)$ and $\to_0, \to_1$ satisfy the conditions 1-3 of Proposition 1.

Instructions on re-checking the formal proof using Isabelle 2023 software are included in the supplementary material [5].

□

Semi-formally, the construction of the ARS in Example 1 is based on viewing the set E as a union of 9 "layers" (where a layer consists of elements of the form (n, A, B, C) for a fixed $n \in \{0, 1, ..., 8\}$) and joining elements of the mentioned layers in accordance with the pattern illustrated in Figure 2: elements with uncountably many direct successors are succeeded by elements with a single direct successor, elements with a single direct successor are succeeded by elements with uncountably many successors. Reduction sequences of elements with $n \in \{0, ..., 5\}$ only diverge, but their continuations using elements with $n \in \{6, 7, 8\}$ can converge. Details of the definition of $\to_E$ ensure that $(E, \to_E)$ is confluent, but does not have the cofinality property.

The main steps of the proof of Theorem 1 can be described as follows. First, it is assumed that there exist $\to_0 \subseteq E \times E$ and $\to_1 \subseteq E \times E$ such that $\to_E = (\to_0 \cup \to_1)$ and $\to_0, \to_1$ satisfy the conditions 1-3 of Proposition 1. Then using the condition 1 of Proposition 1 and the definition of $\to_E$ it can be shown that any element of E has no more than one direct successor w.r.t. $\to_0$ ([5, formal lemma `lem_rE_diamsubr_un`]). Note that this implies that the set of elements reachable from any given element using $\to_0$ is at most countable. Using this fact it can be shown [5, formal lemma `lem_rE_one`] that the relation $\to_0$ does not contain rewrite steps of the form $(n, A, B, C) \to_0 (n', A', B', C')$, where $n \in \{0, 2, 4\}$ (note that such elements (n, A, B, C) have uncountably many direct successors w.r.t. $\to_E$). Then all such rewrite steps belong to $\to_1$. Using this fact a contradiction with the condition 3 of Proposition 1 can be obtained.

Remark 3.2. *Reviewers Jörg Endrullis, Femke van Raamsdonk, and Jan Willem Klop in their review of the initial version of this paper proposed the following simplification of the example of a confluent ARS outside of the class DCR_2:*

$(A, \to)$, where $A = \{1, 2, 3\} \times \mathcal{P}_{fin}(\mathbb{R})$ and $\to$ is defined by the following rules:

$(1, P) \to (2, P \uplus \{p, q\})$
$(2, P) \to (3, P)$
$(3, P) \to (2, P \uplus \{p\})$

for all $p, q \in \mathbb{R} \backslash P$ with $p \neq q$.

Theorem 2. *The ARS $(E, \to_E)$ defined in Example 1 belongs to the class DCR_3.*

Proof. A machine-checked proof of a formalized version of Theorem 2 in Isabelle proof assistant is available in the supplementary material for this paper [5, lines 1147–1150, theorem `thm_2`].

More specifically, the formal proof shows that the following binary relations $\to_i$ on E satisfy the equality $\to_E = (\to_0 \cup \to_1 \cup \to_2)$ and the conditions 1-6 of Proposition 2:

- $\to_0 = \{((n, A, B, C), (n', A', B', C')) \in \to_E \mid n \in \{1, 3, 5, 6, 7\}\}$
- $\to_1 = \{((n, A, B, C), (n', A', B', C')) \in \to_E \mid n \in \{4, 8\}\}$
- $\to_2 = \{((n, A, B, C), (n', A', B', C')) \in \to_E \mid n \in \{0, 2\}\}$
- $\to_{01} = \to_0 \cup \to_1$.

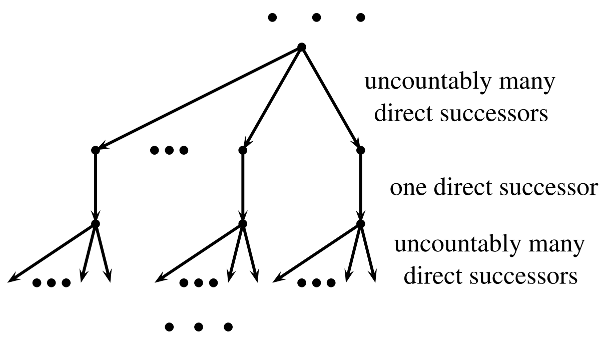

uncountably many direct successors
one direct successor
uncountably many direct successors

Figure 2: Illustration of the main rewrite pattern used in the ARS $(E, \to_E)$ from Example 1.

□

Remark 3.3. *Theorems 1 and 2 together imply that the answer to [3, Open Problem 6.4] is affirmative.*

References

[1] Jasmin Christian Blanchette, Andrei Popescu, and Dmitriy Traytel. Cardinals in Isabelle/HOL. volume 8558 of *Lecture Notes in Computer Science*, pages 111–127, 2014.

[2] Jörg Endrullis, Jan Willem Klop, and Roy Overbeek. Decreasing diagrams with two labels are complete for confluence of countable systems. In *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*, pages 14:1–14:15, 2018. https://doi.org/10.4230/LIPIcs.FSCD.2018.14.

[3] Jörg Endrullis, Jan Willem Klop, and Roy Overbeek. Decreasing diagrams for confluence and commutation. *Logical Methods in Computer Science*, 16:23:1–23:25, 2020. https://doi.org/10.23638/LMCS-16(1:23)2020.

[4] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980.

[5] Ievgen Ivanov. Supplementary material for this paper. https://doi.org/10.5281/zenodo.11571490 .

[6] Ievgen Ivanov. On Newman's lemma and non-termination. In *CEUR-WS.org*, volume 3624, pages 14–24, 2024.

[7] Elliott Mendelson. *Introduction to Mathematical Logic*. CRC Press, 2015.

[8] Maxwell Herman Alexander Newman. On theories with a combinatorial definition of "equivalence". *Annals of mathematics*, pages 223–243, 1942.

[9] Chuangjie Xu Nicolai Kraus, Fredrik Nordvall Forsberg. Type-theoretic approaches to ordinals. *Theoretical computer science*, 957, 2023.

[10] Vincent Van Oostrom. Confluence by decreasing diagrams. *Theoretical computer science*, 126(2):259–280, 1994.

Orthogonality of Generalized Term Rewriting Systems*

Salvador Lucas

DSIC & VRAIN, Universitat Politècnica de València, Spain
slucas@dsic.upv.es

Abstract

We introduce the notion of V-orthogonality for *Generalized Term Rewriting Systems* (GTRSs), which generalize Conditional Term Rewriting Systems by (i) restricting rewritings on arguments of function symbols and (ii) allowing for more general conditions in rules, namely, atoms defined by a set of Horn clauses. The usual left-linearity requirement for rules is relaxed. However, besides the absence of proper conditional critical pairs, the infeasibility of the conditional variable pairs introduced by conditional rules is also required. We prove that V-orthogonal GTRSs are confluent. This new confluence criterion has been implemented in the confluence tool CONFident improving its ability to deal with context-sensitive and conditional term rewriting systems. We briefly report on this.

1 Introduction

For Term Rewriting Systems (TRSs), orthogonality is a *syntactic* criterion guaranteeing confluence by requiring (i) *left-linearity* of rules and (ii) the *absence of critical pairs* [4, Definition 3.1.1]. In this paper we extend this approach to *Generalized Term Rewriting Systems (GTRSs [6])* $\mathcal{R} = (\mathcal{F}, \Pi, \mu, H, R)$, where $\mathcal{F}$ is a signature of *function symbols*; Π is a signature of *predicate symbols*, including $\to$ and $\to^*$; μ is a *replacement map*, i.e., a mapping such that, for all k-ary function symbols $f \in \mathcal{F}$, $\mu(f) \subseteq \{1, \ldots, k\}$ is the set of *active* arguments on which rewriting steps are allowed [5]; H is a (possibly empty) set of definite Horn clauses $A \Leftarrow c$, where the predicate symbol of A is not $\to$ or $\to^*$; and R is a set of rewrite rules $\ell \to r \Leftarrow c$ such that $\ell \notin \mathcal{X}$ [6, Definition 51]. In both cases, c is a sequence of atoms.

Example 1. *Consider the GTRS* $\mathcal{R} = (\mathcal{F}, \Pi, \mu_\bot, H, R)$, *where* $\mathcal{F} = \{\mathsf{0}, \mathsf{s}\}$, $\Pi = \{\to, \to^*, \geq, \mathsf{peven}, \mathsf{odd}, \mathsf{zero}\}$, $\mu_\bot(f) = \emptyset$ *for all* $f \in \mathcal{F}$, $H = \{(1), (2), (3), (4), (5)\}$, *and* $R = \{(6)\}$, *with:*

$$\begin{array}{rclr@{\qquad}rclr} x \geq \mathsf{0} & & & (1) & \mathsf{odd}(x) & \Leftarrow & x \to^* \mathsf{s}(\mathsf{0}) & (4) \\ \mathsf{s}(x) \geq \mathsf{s}(y) & \Leftarrow & x \geq y & (2) & \mathsf{zero}(x) & \Leftarrow & x \to^* \mathsf{0} & (5) \\ \mathsf{peven}(x) & \Leftarrow & x \to^* \mathsf{s}(\mathsf{s}(\mathsf{0})) & (3) & \mathsf{s}(\mathsf{s}(x)) \to x & \Leftarrow & x \geq \mathsf{s}(\mathsf{0}) & (6) \end{array}$$

Predicate $\geq$ *is defined by the Horn clauses (1) and (2); clauses (3), (4), and (5) define tests to check whether a number (in Peano notation) is* positive *and* even *(*peven*),* odd, *or* zero.

The FO-theory of a GTRS $\mathcal{R}$ is $\overline{\mathcal{R}} = \{(\text{Rf}), (\text{Co})\} \cup \{(\text{Pr})_{f,i} \mid f \in \mathcal{F}, i \in \mu(f)\} \cup \{(\text{HC})_\alpha \mid \alpha \in H \cup R\}$, where (see Table 1), (Rf) expresses *reflexivity* of many-step rewriting; (Co) expresses *compatibility* of one-step and many-step rewriting; for each k-ary function symbol f and $i \in \mu(f)$, $(\text{Pr})_{f,i}$ enables the *propagation* of rewriting steps in the i-th immediate *active* subterm $t|_i$ of a term t with root symbol f; finally, for each Horn clause $\alpha \in H \cup R$, $(\text{HC})_\alpha$ provides the usual *implicative* form.

*Partially supported by MCIN/AEI project PID2021-122830OB-C42 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe" and by the grant CIPROM/2022/6 funded by Generalitat Valenciana.

Table 1: Generic sentences of the first-order theory of a GTRS

Label	Sentence
(Rf)	$(\forall x)\ x \to^* x$
(Co)	$(\forall x,y,z)\ x \to y \wedge y \to^* z \Rightarrow x \to^* z$
$(\mathrm{Pr})_{f,i}$	$(\forall x_1,\ldots,x_k,y_i)\ x_i \to y_i \Rightarrow f(x_1,\ldots,x_i,\ldots,x_k) \to f(x_1,\ldots,y_i,\ldots,x_k)$
$(\mathrm{HC})_{A\Leftarrow A_1,\ldots,A_n}$	$(\forall x_1,\ldots,x_p)\ A_1 \wedge \cdots \wedge A_n \Rightarrow A$ where $x_1,\ldots,x_p$ are the variables occurring in $A_1,\ldots,A_n$ and A

Example 2. *We have $\overline{\mathcal{R}} = \{(Rf),(Co),(HC)_{(1)},(HC)_{(2)},(HC)_{(3)},(HC)_{(4)},(HC)_{(5)},(HC)_{(6)}\}$ for $\mathcal{R}$ in Example 1. Note that no propagation rule is necessary as $\mu_\perp(f)=\emptyset$ for all $f\in\mathcal{F}$.*

For all terms s and t, we write $s \to_{\mathcal{R}} t$ (resp. $s \to^*_{\mathcal{R}} t$) iff $\overline{\mathcal{R}} \vdash s \to t$ (resp. $\overline{\mathcal{R}} \vdash s \to^* t$). A GTRS $\mathcal{R}$ is (locally) confluent (resp. terminating) iff $\to_{\mathcal{R}}$ is (locally) confluent (resp. terminating). A sequence of atoms $A_1,\ldots,A_n$ is *$\overline{\mathcal{R}}$-feasible* (or just *feasible* if no confusion arises) if there is a substitution σ such that, for all $1\leq i\leq n$, $\overline{\mathcal{R}} \vdash \sigma(A_i)$ holds. A rule $\ell \to r \Leftarrow c$ is feasible if c is feasible. Only feasible rules contribute to $\to_{\mathcal{R}}$-rewriting steps and hence to confluence. As in [8, Definition 6.1], rules $\ell \to r \Leftarrow c$ in GTRSs are classified according to the distribution of variables: type 1, if $\mathcal{V}ar(r)\cup\mathcal{V}ar(c)\subseteq\mathcal{V}ar(\ell)$; type 2, if $\mathcal{V}ar(r)\subseteq\mathcal{V}ar(\ell)$; type 3, if $\mathcal{V}ar(r)\subseteq\mathcal{V}ar(\ell)\cup\mathcal{V}ar(c)$; and type 4, otherwise. A rule of type n is often called an n-rule. A GTRS $\mathcal{R}$ is called an n-GTRS if all its rules are of type n; if $\mathcal{R}$ contains at least one n-rule which is *not* an m-rule for some $m<n$, then we say that $\mathcal{R}$ is a *proper* n-GTRS.

2 V-orthogonality of GTRSs

In the following, we deal with *conditional pairs*, $\pi : \langle s,t\rangle \Leftarrow c$, where s and t are terms and c is a sequence of atomic conditions; π is *trivial* if $s=t$; it is *$\overline{\mathcal{R}}$-(in)feasible* if c is.

Definition 1 (Conditional critical and variable pairs of a GTRS [6, Def. 59])**.** *Let $\mathcal{R}$ be a GTRS, and $\alpha : \ell \to r \Leftarrow c, \alpha' : \ell' \to r' \Leftarrow c'$ be variable disjoint rules of $\mathcal{R}$ (rename if necessary).*

- *Let $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(\ell)$ be a nonvariable active position of ℓ such that $\ell|_p$ and ℓ' unify with* mgu *θ. Then, $\langle\theta(\ell[r']_p),\theta(r)\rangle \Leftarrow \theta(c),\theta(c')$, labelled $\pi_{\alpha,p,\alpha'}$, is a* conditional critical pair *(CCP). If $p=\Lambda$ and α and α' are renamed versions of the same rule, then $\pi_{\alpha,p,\alpha'}$ is an* improper *CCP; otherwise, it is a* proper *CCP. Let $\mathsf{pCCP}(\mathcal{R})$ be the set of* feasible *proper CCPs of $\mathcal{R}$. Let $\mathsf{iCCP}(\mathcal{R})$ be the set of* feasible *improper CCPs of proper 3- or 4-rules in $\mathcal{R}$.*
- *Let $p \in \mathcal{P}os^{\mu}_{x}(\ell)$ be an active position of a variable x in ℓ, and x' be a fresh variable. Then, $\langle\ell[x']_p,r\rangle \Leftarrow x \to x', c$, labelled $\pi_{\alpha,x,p}$, is a* conditional variable pair *(CVP). Let $\mathsf{CVP}(\alpha)$ be the set of feasible CVPs of a rule α.*

Our next definition introduces V-orthogonality of GTRSs $\mathcal{R}$. For *unconditional* rules $\ell \to r$ in $\mathcal{R}$, the requirement of *left-linearity* is weakened to benefit from replacement restrictions and the consideration of CVPs: for unconditional rules, we require that *active* variables occur at most once in the left-hand sides and that they *remain active* in the right-hand sides. On the other hand, besides requiring that only trivial and proper critical pairs are present, we also require the *infeasibility* of conditional variable pairs for proper *conditional rules.*

Definition 2 (V-orthogonal GTRS)**.** *Let $\mathcal{R} = (\mathcal{F}, \Pi, \mu, H, R)$ be a GTRS such that:*

1. *for all unconditional rules $\ell \to r \in \mathcal{R}$, and all active variables $x \in \mathcal{V}ar^{\mu}(\ell)$ in ℓ, (a) x occurs nowhere else in ℓ and (b) x is* not *frozen in r;*
2. *all feasible improper CCPs $\pi \in \mathsf{iCCP}(\mathcal{R})$ are* trivial*; and*
3. *for all rules $\alpha : \ell \to r \Leftarrow c$ with c nonempty there is no feasible CVP: $\mathsf{CVP}(\alpha) = \emptyset$.*

If $\mathsf{pCCP}(\mathcal{R}) = \emptyset$, then $\mathcal{R}$ is called V-orthogonal. If for all $\pi \in \mathsf{pCCP}(\mathcal{R})$, π is a trivial overlay, then $\mathcal{R}$ is called almost V-orthogonal. If for all $\pi \in \mathsf{pCCP}(\mathcal{R})$, π is trivial, then $\mathcal{R}$ is called weakly V-orthogonal.

V-orthogonal GTRSs are almost V-orthogonal, hence weakly V-orthogonal. For TRSs, Definition 2 and *(almost, weak) orthogonality* coincide. And (almost, weakly) orthogonal 2-CTRSs *modulo infeasibility* [10, Definition 2] (properly including (weakly) orthogonal 2-CTRSs [9, Definition 7.1.10]) are (weakly) V-orthogonal. The GTRS $\mathcal{R}$ in Example 1 is V-orthogonal.

3 Confluence of weakly V-orthogonal GTRSs

Two positions p and q are *disjoint* (written $p \parallel q$ [9, page 39]) if $p \not\leq q$ and $q \not\leq p$; otherwise, we write $p \not\parallel q$. Given a position p and a set of positions P, we write $p \parallel P$ if $p \parallel p'$ for all $p' \in P$. Given sets of positions P and Q, we write $P \parallel Q$ if for all $p \in P$, $p \parallel Q$ (equivalently $q \parallel P$ for all $q \in Q$). As before, we write $p \not\parallel P$ (resp. $P \not\parallel Q$) if $p \parallel P$ (resp. $P \parallel Q$) does *not* hold. Given a position p and a set of positions Q, let $\mathsf{A}(p, Q) = \{q \in Q \mid q < p\}$ be positions in Q (strictly) *above* p, and $\mathsf{B}(p, Q) = \{q \in Q \mid p < q\}$ be positions in Q (strictly) *below* p. Note that $p \notin \mathsf{B}(p, Q)$ and $p \parallel p'$ implies $\mathsf{B}(p, Q) \parallel \mathsf{B}(p', Q)$. We say that P is a set of *mutually disjoint positions* if positions in P are pairwise disjoint: for all $p, p' \in P$, $p \neq p'$ implies $p \parallel p'$.

Proposition 3. *Let p be a position and Q be a set of mutually* disjoint *positions. Then, (1) $\mathsf{A}(p, Q) \neq \emptyset$ implies that $\mathsf{A}(p, Q) = \{q\}$ is a* singleton*; (2) if $p \notin Q$, then $\mathsf{A}(p, Q) = \mathsf{B}(p, Q) = \emptyset$ implies $p \parallel Q$; and (3) it is* not *possible to simultaneously have $\mathsf{A}(p, Q) \neq \emptyset$* and *$\mathsf{B}(p, Q) \neq \emptyset$.*

Rewriting with GTRSs $\mathcal{R}$ has a *position-based* view [6, Proposition 58]: $s \to_{\mathcal{R}} t$ iff there is $p \in \mathcal{P}os^{\mu}(s)$ and $\ell \to r \Leftarrow c \in R$ such that (i) $s|_p = \sigma(\ell)$ for some substitution σ, (ii) for all $A \in c$, $\overline{\mathcal{R}} \vdash \sigma(A)$ holds, and (iii) $t = s[\sigma(r)]_p$. Accordingly, given a term s, a set $P = \{p_1, \ldots, p_n\} \subseteq \mathcal{P}os^{\mu}(s)$ of disjoint (active) positions in s, for some $n \geq 0$, rules $\alpha_i : \ell_i \to r_i \Leftarrow c_i \in \mathcal{R}$ and substitutions σ_i, $1 \leq i \leq n$, such that $s|_{p_i} = \sigma_i(\ell_i)$ and $\sigma(c_i)$ holds, we write $s \xrightarrow{P}_{\mathcal{R}} t$ if $t = s[\sigma_1(r_1)]_{p_1} \cdots [\sigma_n(r_n)]_{p_n}$ (also $s \mathrel{+\!\!\!\!\!\!\Vdash\!\!\!\!\to}_{\mathcal{R}} t$, or even $s \mathrel{+\!\!\!\!\!\!\Vdash\!\!\!\!\to} t$, if no confusion arises).

Proposition 4. *Let $\mathcal{R} = (\mathcal{F}, \Pi, \mu, H, R)$ be a weakly V-orthogonal GTRSs. For all terms $s, t, t' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, if $s \mathrel{+\!\!\!\!\!\!\Vdash\!\!\!\!\to} t$ and $s \mathrel{+\!\!\!\!\!\!\Vdash\!\!\!\!\to} t'$, there is u such that $t \mathrel{+\!\!\!\!\!\!\Vdash\!\!\!\!\to} u$ and $t' \mathrel{+\!\!\!\!\!\!\Vdash\!\!\!\!\to} u$.*

Proof. Let $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $P, P' \subseteq \mathcal{P}os^{\mu}(s)$ be such that $s \xrightarrow{P}_{\mathcal{R}} t$ and $s \xrightarrow{P'}_{\mathcal{R}} t'$ for some terms t, t'. Let $P = \{p_1, \ldots, p_m\}$ and $P' = \{p'_1, \ldots, p'_n\}$ for some $m, n > 0$ be nonempty sets of mutually disjoint redex positions of s. Thus, there are (pairwise variable disjoint) rules $\alpha_i : \ell_i \to r_i \Leftarrow c_i$ for all $1 \leq i \leq m$ and $\alpha'_j : \ell'_j \to r'_j \Leftarrow c'_j$ for all $1 \leq j \leq n$ such that $s = s[\sigma(\ell_1)]_{p_1} \cdots [\sigma(\ell_m)]_{p_m} = s[\sigma(\ell'_1)]_{p'_1} \cdots [\sigma(\ell'_n)]_{p'_n}$ and then

$$t = s[\sigma(r_1)]_{p_1} \cdots [\sigma(r_m)]_{p_m} \quad \text{and} \quad t' = s[\sigma(r'_1)]_{p'_1} \cdots [\sigma(r'_n)]_{p'_n} \tag{7}$$

Since P and P' are sets of pairwise disjoint redex positions, the order in which replacements in (7) are made is not important. We partition P as $P = A \uplus B \uplus E \uplus D$, where $A = \{p \in P \mid \mathsf{B}(p, P') \neq \emptyset\}$ is the subset of positions in P which are strictly *above* some positions of P' (and hence *not below* any position in P', due to Proposition 3.(3)); $B = \{p \in P \mid \mathsf{A}(p, P') \neq \emptyset\}$ is the subset of positions in P which are strictly *below* positions of P'; $E = P \cap P'$, and $D = P - (A \cup B \cup E) = \{p \in P - P' \mid \mathsf{A}(p, P') = \emptyset \wedge \mathsf{B}(p, P') = \emptyset\}$ is the set of positions of P which are *disjoint* from P' (Proposition 3(2)). Proposition 3(3) guarantees that $A \cap B = \emptyset$. Note that, since P is mutually disjoint, A, B, D, and E are pairwise mutually disjoint.

1. If $p \in E$, then $p \in P \cap P'$ and $s|_p = \sigma(\ell) = \sigma(\ell')$ for some (variable disjoint) rules $\alpha : \ell \to r \Leftarrow c$ and $\alpha' : \ell' \to r' \Leftarrow c'$, where both $\sigma(c)$ and $\sigma(c')$ are satisfied. Thus, ℓ and ℓ' overlap at the root position Λ, i.e., ℓ and ℓ' unify with *mgu* θ, defining a (feasible) conditional (overlay) critical pair $\pi : \langle \theta(\ell)[\theta(r')]_\Lambda, \theta(r)\rangle \Leftarrow \theta(c), \theta(c')$, i.e., $\pi : \langle \theta(r'), \theta(r)\rangle \Leftarrow \theta(c), \theta(c')$. Also, $t|_p = \sigma(r)$ and $t'|_p = \sigma(r')$. By Definition 2, whether proper or improper, π is trivial. Then, $t'|_p = \sigma(r') = \sigma(r) = t|_p$, see the diagram in Figure 1(1).

2. If $p \in A$, then for all $p' \in \mathsf{B}(p, P') \subseteq P'$, $p' = p.q'$ for some nonempty position q'. Thus, $\mathsf{B}(p, P') = p.\{q'_1, \ldots, q'_m\}$ for some $m \geq 1$ and nonempty positions $q'_1, \ldots, q'_m$. We have

$$s|_p = \sigma(\ell) = \sigma(\ell)[\sigma(\ell'_1)]_{q'_1} \cdots [\sigma(\ell'_k)]_{q'_m}$$

for some substitution σ and rules $\alpha : \ell \to r \Leftarrow c$ and $\alpha'_i : \ell'_i \to r'_i \Leftarrow c_i$ for $1 \leq i \leq m$ such that $\sigma(c)$ and $\sigma(c'_i)$ hold for all $1 \leq i \leq m$. Let $\mathcal{V}ar(\ell) = \{x_1, \ldots, x_k\}$ and, for each $1 \leq j \leq k$, $V_j = \mathcal{P}os_{x_j}(\ell)$ be the set of positions of ℓ where x_j occurs. We also let $V^\mu_j = V_j \cap \mathcal{P}os^\mu(\ell)$ be the (possibly empty) subset of active positions of x_j in ℓ and $\overline{V}^\mu_j = V_j \cap \overline{\mathcal{P}os^\mu}(\ell) = V_j - V^\mu_j$ be the (possibly empty) subset of frozen positions of x_j in ℓ. Then, $\ell = \ell[x_1]_{V_1} \cdots [x_k]_{V_k} = \ell[x_1]_{V^\mu_1}[x_1]_{\overline{V}^\mu_1} \cdots [x_k]_{V^\mu_k}[x_k]_{\overline{V}^\mu_k}$ and we can write

$$\begin{array}{lcll} s|_p & = & \sigma(\ell) = \ell[\sigma(x_1)]_{V^\mu_1}[\sigma(x_1)]_{\overline{V}^\mu_1} \cdots [\sigma(x_k)]_{V^\mu_k}[\sigma(x_k)]_{\overline{V}^\mu_k} & (8) \\ s|_p & \xrightarrow{\{\Lambda\}}_\alpha & \sigma(r) = t|_p & (9) \\ s|_p & \xrightarrow{\{q'_1,\ldots,q'_m\}}_\mathcal{R} & \sigma(\ell)[\sigma(r'_1)]_{q'_1} \cdots [\sigma(r'_m)]_{q'_m} = t'|_p & (10) \end{array}$$

We consider cases *2.a* and *2.b*:

2.a. No rule α_i overlaps α, i.e., for all $1 \leq i \leq m$, $q'_i \notin \mathcal{P}os_\mathcal{F}(\ell)$. Thus, $\mathcal{V}ar^\mu(\ell) \neq \emptyset$ and there is a variable index mapping $\nu : \{1, \ldots, m\} \to \{1, \ldots, k\}$ such that for all $1 \leq i \leq m$, q'_i is below an *active* position of a variable $x_{\nu(i)}$ in ℓ, i.e., $q'_i = u_i.u'_i$ for some $u_i \in V^\mu_{\nu(i)}$ and active position u'_i. Let $Q' = \{q'_1, \ldots, q'_m\}$. We can partition Q' as $Q' = Q'_1 \uplus \cdots \uplus Q'_k$ where, for all $1 \leq j \leq k$, $Q'_j = \{q' \in Q' \mid u \leq q' \text{ for some } u \in V^\mu_j\}$ collects the positions in Q' which are *below* a position of an occurrence of variable x_j in ℓ. Define the substitution σ' to be such that $\sigma(x_j) \xrightarrow{Q'_j}_\mathcal{R} \sigma'(x_j)$ for all $1 \leq j \leq k$. We distinguish two cases:

- If α is an unconditional rule $\ell \to r$, then, by item 1.(a) in Definition 2, for all $1 \leq j \leq k$, either (I) $V^\mu_j = \emptyset$ and hence $Q'_j = \emptyset$ and $\sigma'(x_j) = \sigma(x_j)$; or (II) $V^\mu_j = \{u_j\}$ is a singleton and then $\overline{V}^\mu_j = \emptyset$ and $u_j \leq Q'_j$. Hence, (10) can be written as follows:

$$s|_p \xrightarrow{\{q'_1,\ldots,q'_m\}}_\mathcal{R} \ell[\sigma'(x_1)]_{V^\mu_1}[\sigma'(x_1)]_{\overline{V}^\mu_1} \cdots [\sigma'(x_k)]_{V^\mu_k}[\sigma'(x_k)]_{\overline{V}^\mu_k} = t'|_p \quad (11)$$

  for all $1 \leq i \leq m$, let $W_{\nu(i)}$ be the (possibly empty) set of positions of $x_{\nu(i)}$ in r: $W_{\nu(i)} = \mathcal{P}os_{x_{\nu(i)}}(r)$. Let $W = \bigcup_{i=1}^m p.W_{\nu(i)} = \mathcal{P}os_\mathcal{X}(r)$. By item 1.(b) of Definition 2, for all

$$\begin{tikzcd} s|_p \arrow[r, "\{\Lambda\}"] \arrow[d, "\{\Lambda\}"'] & t|_p \arrow[d, equal] \\ t'|_p \arrow[r, equal] & u \end{tikzcd}$$

1. $p \in E$

$$\begin{tikzcd} s|_p \arrow[r, "\{\Lambda\}"] \arrow[d, "\{q'_1,\ldots,q'_m\}"'] & t|_p \arrow[d, "W"] \\ t'|_p \arrow[r, "\{\Lambda\}"] & u \end{tikzcd}$$

2.(a) $p \in A$

$$\begin{tikzcd} s|_p \arrow[r, "\{\Lambda\}"] \arrow[d, "\{q'_1,\ldots,q'_m\}"'] & t|_p \arrow[d, "\{q'_{\nu+1},\ldots,q'_m\}"] \\ t'|_p \arrow[r, equal] & u \end{tikzcd}$$

2.(b) $p \in A$

$$\begin{tikzcd} s|_{p'} \arrow[r, "\{q_1,\ldots,q_n\}"] \arrow[d, "\{\Lambda\}"'] & t|_{p'} \arrow[d, "\{\Lambda\}"] \\ t'|_{p'} \arrow[r, "W'"] & u \end{tikzcd}$$

3.(a) $p \in B$

$$\begin{tikzcd} s|_{p'} \arrow[r, "\{q_1,\ldots,q_n\}"] \arrow[d, "\{\Lambda\}"'] & t|_{p'} \arrow[d, equal] \\ t'|_{p'} \arrow[r, "\{q_{\nu'+1},\ldots,q_n\}"] & u \end{tikzcd}$$

3.(b) $p \in B$

$$\begin{tikzcd} s|_p \arrow[r, "\{\Lambda\}"] \arrow[d, equal] & t|_p \arrow[d, equal] \\ t'|_p \arrow[r, "\{\Lambda\}"] & u \end{tikzcd}$$

4. $p \in D$

Figure 1: Different cases in the proof of Proposition 4

$1 \leq i \leq m$, x_i does *not* occur frozen in r. Hence, $W = \mathcal{P}os^{\mu}_{\mathcal{X}}(r)$ is a set of *disjoint* active variable positions of r. Therefore,

$$t|_p = \sigma(r) = r[\sigma(x_1)]_{W_1} \cdots [\sigma(x_k)]_{W_k} \xrightarrow{W}_{\mathcal{R}} r[\sigma'(x_1)]_{W_1} \cdots [\sigma'(x_k)]_{W_k} = u \quad (12)$$

$$t'|_p = \ell[\sigma'(x_1)]_{V_1^{\mu}}[\sigma'(x_1)]_{\overline{V}_1^{\mu}} \cdots [\sigma'(x_k)]_{V_k^{\mu}}[\sigma'(x_k)]_{\overline{V}_k^{\mu}} \xrightarrow{\{\Lambda\}}_{\alpha} = u \quad (13)$$

The parallel rewriting step (12) is possible because (10) is possible. The rewriting step (13) is possible due to (I) and (II) above. Overall, we obtain diagram 2.(a) in Figure 1.

- If α is a conditional rule $\ell \to r \Leftarrow c$, and $x_j \in V_j^{\mu}$ for some $1 \leq j \leq k$ such that $Q'_j \neq \emptyset$, then, by item (3) in Definition 2, the CVP $\langle \ell[x'_j]_{u_j}, r\rangle \Leftarrow x_j \to x'_j, c$ for a fresh variable x'_j is *infeasible*. Since, due to (9), $\sigma(c)$ holds it must be $\sigma(x_j) \not\to_{\mathcal{R}} \sigma(x'_j)$, which contradicts that $\sigma(x_j) \xrightarrow{Q'_j}_{\mathcal{R}} \sigma'(x_j)$ due to (11). Thus, the peak is not possible.

2.b. If some α_i overlaps α, i.e., $q'_i \in \mathcal{P}os_{\mathcal{F}}(\ell)$, then without loss of generality, there is ν, $1 \leq \nu \leq m$ partitioning $\{q'_1, \ldots, q'_m\}$ into $\{q'_1, \ldots, q'_\nu\}$, representing *critical peaks*, and $\{q'_{\nu+1}, \ldots, q'_m\}$, representing *variable peaks*. Accordingly, one of the following two conditions hold: for all $1 \leq i \leq \nu$, $q'_i \in \mathcal{P}os_{\mathcal{F}}(\ell)$, i.e., $\ell|_{q'_i}$ and ℓ' *unify* with *mgu* θ_i and there is a conditional critical pair $\pi_i : \langle \sigma(\ell)[\sigma(r'_i)]_{q'_i}, \theta_i(r)\rangle \Leftarrow \theta_i(c), \theta_i(c'_i)$. Since $\sigma(c)$ and $\sigma(c'_i)$ hold, π_i is feasible, i.e., $\pi_i \in \mathsf{pCCP}(\mathcal{R})$. By strong orthogonality, π_i is trivial for all $1 \leq i \leq \nu$. Hence,

$$\begin{aligned} t|_p &= \sigma(r) = \sigma(\ell)[\sigma(r'_1)]_{q'_1} = \cdots = \sigma(\ell)[\sigma(r'_\nu)]_{q'_\nu} \\ &= \sigma(\ell)[\sigma(r'_1)]_{q'_1} \cdots [\sigma(r'_\nu)]_{q'_\nu}[\sigma(\ell'_{\nu+1})]_{q'_{\nu+1}} \cdots [\sigma(\ell'_m)]_{q'_m} \xrightarrow{\{q'_{\nu+1},\ldots,q'_m\}}_{\mathcal{R}} t'|_p \end{aligned}$$

and we obtain the diagram 2.(b) in Figure 1.

3. If $p \in B$, then, by Proposition 3(1), $\mathsf{A}(p, P') = \{p'\}$ returns the (only) position $p' \in P'$ which is *above* p. Then, $B' = \mathsf{B}(p', P)$ is the set of positions in P which are *below* p'. By reasoning as in the previous item, we obtain diagrams 3.(a) and 3.(b) in Figure 1.

4. If $p \in D$, then $s|_p \xrightarrow{\{p\}}_{\alpha} t|_p$ but, since $p \parallel P'$, $t'|_p = s|_p$, see diagram 4 in Figure 1. $\square$

Proposition 4 proves the *diamond property* [1, Definition 2.7.8] of $\nparallel\!\!\to_{\mathcal{R}}$, which implies strong confluence of $\to_{\mathcal{R}}$ [1, Definition 2.7.3]. Since $\to_{\mathcal{R}} \subseteq \nparallel\!\!\to_{\mathcal{R}} \subseteq \to^*_{\mathcal{R}}$, by [1, Corollary 2.7.7], we have:

Theorem 5. *Weakly V-orthogonal GTRSs are confluent.*

The V-orthogonal GTRS $\mathcal{R}$ in Example 1 is confluent.

Application to CS-TRSs. A Context-Sensitive Term Rewriting System (CS-TRS) $\mathcal{R} = (\mathcal{F}, \mu, R)$, where R is a set of unconditional rules [5], can be seen as a GTRS $\mathcal{R} = (\mathcal{F}, \Pi, \mu, H, R)$, where $\Pi = \{\to, \to^*\}$ and H is empty. When applied to CS-TRSs, Definition 2 collapses into the requirement (1) for unconditional rules, as (2) and (3) concern conditional rules only.

Example 3. *Consider the following CS-TRS $\mathcal{R}$ from COPS (1336.trs):*

$$\mathsf{f}(x, x) \to \mathsf{plus}(\mathsf{plus}(x, x), x) \qquad\qquad \mathsf{plus}(x, y) \to \mathsf{plus}(y, x)$$

Since all variables in the left-hand sides of rules are frozen, $\mathcal{R}$ is V-orthogonal and confluent.

Left-linear CS-TRSs $\mathcal{R}$ without μ-critical pairs (i.e., critical pairs in $\mathsf{CP}(\mathcal{R})$ whose critical position is active, i.e., they coincide with $\mathsf{pCCP}(\mathcal{R})$) and such that for all $\ell \to r$ in $\mathcal{R}$ and active variable x in ℓ the variable occurs nowhere else in ℓ and is not frozen in r, are called μ-orthogonal [7, Definition 35]. V-orthogonal CS-TRSs properly *include* μ-orthogonal CS-TRSs. For instance, $\mathcal{R}$ in Example 3 is V-orthogonal but not μ-orthogonal.

Application to Conditional Term Rewriting Systems. A CTRS $\mathcal{R} = (\mathcal{F}, R)$, with R a set of rules $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n$ for a predicate symbol $\approx$, can be seen as a GTRS $\mathcal{R} = (\mathcal{F}, \Pi, \mu_\top, H, R)$, where $\Pi = \{\to, \to^*, \approx\}$, $\mu_\top(f) = \{1, \ldots, k\}$ for all k-ary symbols $f \in \mathcal{F}$ (i.e., no replacement restriction is considered), and H depends on the (fixed) *evaluation semantics* (*join*, *oriented*, or *semi-equational*, see, e.g., [9, Definition 7.1.3]) for the conditions $s \approx t$ in rules. Accordingly, H is either $\{(\mathrm{J})\}$, or $\{(\mathrm{O})\}$, or $\{(\mathrm{SE}_1), (\mathrm{SE}_2), (\mathrm{SE}_3)\}$, see [6, Figure 2]. Since no replacement restriction is considered, when applied to CTRSs, requirement (1) in Definition 2 just becomes *left-linearity* (of unconditional rules).

Example 4. *The following (non-terminating) 2-CTRS $\mathcal{R}$:*

$$\mathsf{a} \to \mathsf{b} \qquad\qquad \mathsf{c} \to \mathsf{g}(\mathsf{c}) \qquad\qquad \mathsf{f}(x, x) \to x \Leftarrow \mathsf{h}(\mathsf{b}, y) \approx \mathsf{h}(x, \mathsf{g}(y)), y \approx \mathsf{c}$$

is not *(almost, weakly) orthogonal [9, Definition 7.1.10], as it is not left-linear due to the conditional rule. Thus, the usual results guaranteeing confluence of (weakly) orthogonal CTRSs (see, e.g., [9, Section 7.4]) do not apply. However, the unconditional rules are left-linear, $\mathsf{pCCP}(\mathcal{R}) = \emptyset$, and the only improper CCP obtained from the conditional (2-)rule is trivial. The two CVPs, $\mathsf{f}(x', x) \to x \Leftarrow x \to x', \mathsf{h}(\mathsf{b}, y) \approx \mathsf{h}(x, \mathsf{g}(y)), y \approx \mathsf{c}$ and $\mathsf{f}(x, x') \to x \Leftarrow x \to x', \mathsf{h}(\mathsf{b}, y) \approx \mathsf{h}(x, \mathsf{g}(y)), y \approx \mathsf{c}$ are* infeasible*: if a substitution σ would satisfy the (common) conditional part then $\sigma(x) \to_{\mathcal{R}} \sigma(x')$ holds. However, $\sigma(\mathsf{h}(\mathsf{b}, y)) = \mathsf{h}(\mathsf{b}, \sigma(y)) \to^*_{\mathcal{R}} \mathsf{h}(\sigma(x), \mathsf{g}(\sigma(y))) = \sigma(\mathsf{h}(x, \mathsf{g}(y)))$ requires $\mathsf{b} = \sigma(x)$, which contradicts reducibility of $\sigma(x)$. Therefore, $\mathcal{R}$ is almost V-orthogonal and hence confluent.*

V-orthogonality, alone, does *not* subsume existing results for orthogonal CTRSs.

Example 5. *The following orthogonal 3-CTRS $\mathcal{R}$ [11, Example 4.4]*

$$\begin{array}{rclcrclcrcl} \mathsf{a} & \to & \mathsf{b} & \qquad & \mathsf{b} & \to & \mathsf{c} & \qquad & \mathsf{f}(x) & \to & \mathsf{g}(x, y) \Leftarrow x \approx \mathsf{i}(y) \\ & & & & \mathsf{k}(x) & \to & \mathsf{j}(x) & & \mathsf{h}(x) & \to & \mathsf{i}(y) \Leftarrow x \approx \mathsf{j}(y) \end{array}$$

is confluent by [11, Corollary 4.7]. However, $\mathcal{R}$ is not *weakly V-orthogonal: the improper CCP $\langle \mathsf{i}(y), \mathsf{i}(y') \rangle \Leftarrow x \approx \mathsf{j}(y), x \approx \mathsf{j}(y')$ obtained from the last rule is feasible and not trivial.*

Such results, though, require *additional* restrictions on the conditional rules (e.g., a particular evaluation semantics for the conditional part [2], or conditions of a given shape [11]).

4 Implementation

The results in this paper were implemented as part of the confluence tool CONFident to prepare its participation in the 2023 edition of the Confluence Competition, CoCo 2023[1]. CONFident and ConfCSR participated in the CSR category of CoCo 2023, where CS-TRSs and CS-CTRSs are mixed (there were 176 CS-TRSs and 287 CS-CTRSs). Since ConfCSR does not handle CS-CTRSs (whose confluence is (dis)proved by CONFident by using the results in this paper together with [6, 3], as CS-CTRSs are particular GTRSs [6, Section 7.3]), the following table reproduces the information for CS-TRSs extracted in [3, Table 10 (left)] for the CoCo 2023 *full run* benchmarks set (full details in [3, Section 8]):

Tool	Yes	No	Maybe	Solved	Total
ConfCSR	28	83	65	111	176
CONFident 2024	49	83	44	132	176

For instance, the CS-TRS $\mathcal{R}$ in Example 3 was solved by using the results in this paper.[2]

Acknoledgements. I thank Miguel Vítores for implementing these results in CONFident.

References

[1] Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[2] Jan A. Bergstra and Jan Willem Klop. Conditional rewrite rules: Confluence and termination. *J. Comput. Syst. Sci.*, 32(3):323–362, 1986.

[3] Raúl Gutiérrez, Salvador Lucas, and Miguel Vítores. Proving Confluence in the Confluence Framework with CONFident. *Fundamenta Informaticae*, to appear, 2024.

[4] J. W. Klop. Chapter 1: Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaurn, editors, *Handbook of Logic in Computer Science*, pages 1–116. Oxford University Press, 1992.

[5] Salvador Lucas. Context-sensitive Rewriting. *ACM Comput. Surv.*, 53(4):78:1–78:36, 2020.

[6] Salvador Lucas. Local confluence of conditional and generalized term rewriting systems. *Journal of Logical and Algebraic Methods in Programming*, 136:paper 100926, pages 1–23, 2024.

[7] Salvador Lucas, Miguel Vítores, and Raúl Gutiérrez. Proving and disproving confluence of context-sensitive rewriting. *Journal of Logical and Algebraic Methods in Programming*, 126:100749, 2022.

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

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

[10] Christian Sternagel and Thomas Sternagel. Certifying confluence of almost orthogonal ctrss via exact tree automata completion. In Delia Kesner and Brigitte Pientka, editors, *1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016 Proceedings in Informatics (LIPIcs)*, pages 29:1–29:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[11] Taro Suzuki, Aart Middeldorp, and Tetsuo Ida. Level-Confluence of Conditional Rewrite Systems with Extra Variables in Right-Hand Sides. In Jieh Hsiang, editor, *Rewriting Techniques and Applications, 6th International Conference, RTA-95, Proceedings*, volume 914 of *Lecture Notes in Computer Science*, pages 179–193. Springer, 1995.

[1] https://project-coco.uibk.ac.at/2023/

[2] The report produced by CONFident for CoCo 2023 displays "strong orthogonality" rather than V-orthogonality, see http://cops.uibk.ac.at/results/2023/CSR/CONFident/1336.trs/615314450.txt.

Second-order Church-Rosser modulo, without normalization

Thiago Felicissimo

Université Paris-Saclay, INRIA, Deducteam, Laboratoire de Méthodes Formelles, ENS Paris-Saclay
thiago.felicissimo@inria.fr

Abstract

We investigate criteria for proving Church-Rosser modulo of second-order rewrite systems without relying on normalization.

1 Introduction

Rewriting modulo is an alternative to standard rewriting in which one considers not only rewriting rules $l \longmapsto r \in \mathcal{R}$ but also undirected equations $t \approx u \in \mathcal{E}$, allowing to handle theories defined by axioms that cannot be oriented in a well-behaved manner, such as commutativity. In this setting, the Church-Rosser property must be adapted into *Church-Rosser modulo*, stating that $t\ (\longrightarrow \cup \longleftarrow \cup \simeq)^*\ u$ implies $t \xrightarrow{\sim}{}^* \circ \simeq \circ {}^*\!\xleftarrow{\sim} u$, where $\simeq$ is the congruence generated by $\mathcal{E}$, $\longrightarrow$ is the rewrite relation generated by $\mathcal{R}$, and $\xrightarrow{\sim}$ is a relation that can be $\longrightarrow$ [9], or $\simeq \circ \longrightarrow$, or something in between [10] depending on the considered variant of this definition.

Unfortunately, most criteria for Church-Rosser modulo rely on normalization [9, 10, 13, 6], posing a challenge in an ongoing line of study [2] to extend Dedukti [5], a rewriting-based framework for defining type theories, with rewriting modulo. Indeed, confluence proofs for dependent type theories are usually carried out on untyped terms, for which normalization does not hold due to rules such as β-reduction. The situation is made worse by the fact that type theories rely not on first- but second-order rewriting, for which criteria for Church-Rosser modulo are much rarer.

Example 1. Consider the following second-order rewrite system modulo, defining (a fragment of) the conversion of a type theory with an addition satisfying commutativity and assocativity. Function symbols are written in blue, and metavariables in `typewriter font`. We would like to show this example (or a variant of it) to be Church-Rosser modulo.[1]

$$+(\mathtt{t}, 0) \approx \mathtt{t} \qquad +(\mathtt{t}, S(\mathtt{u})) \approx S(+(\mathtt{t},\mathtt{u})) \qquad +(\mathtt{t},\mathtt{u}) \approx +(\mathtt{u},\mathtt{t}) \qquad +(+(\mathtt{t},\mathtt{u}),\mathtt{v}) \approx +(\mathtt{t}, +(\mathtt{u},\mathtt{v}))$$

$$@(\lambda(x.\mathtt{t}\{x\}), \mathtt{u}) \longmapsto \mathtt{t}\{\mathtt{u}\} \qquad\qquad \mathbb{N}_{\mathsf{rec}}(0, \mathtt{p0}, xy.\mathtt{pS}\{x,y\}) \longmapsto \mathtt{p0}$$

$$\mathbb{N}_{\mathsf{rec}}(S(\mathtt{n}), \mathtt{p0}, xy.\mathtt{pS}\{x,y\}) \longmapsto \mathtt{pS}\{\mathtt{n}, \mathbb{N}_{\mathsf{rec}}(\mathtt{n}, \mathtt{p0}, xy.\mathtt{pS}\{x,y\})\}$$

In this work, we investigate criteria for proving Church-Rosser modulo of second-order rewrite systems without relying on normalization. We start by proposing a first criterion, which relies on a notion of *unblocked* term. We then discuss a second criterion, which can be shown almost directly from a well-known result. The proofs not given here can be found in a technical report [7].

2 Preliminaries

We work in the setting of (untyped) second-order rewriting.[2] An *arity* is a natural number n and a *binding arity* is a list of natural numbers $(n_1, \ldots, n_k)$. Given a set $\mathcal{V}$ of variables $x, y, \ldots,$

[1] Among other things, this is needed to establish the *injectivity of* Π*-types*: $\Pi(A_1, x.A_2) \equiv \Pi(A'_1, x.A'_2)$ should imply $A_i \equiv A'_i$. This is then used to establish subject reduction of β-reduction.

[2] This can be seen as Hamana's Second-order Computational Systems [8], or a simply-typed version of Klop's Combinatory Reduction Systems (CRSs) [12] with a single base type, or Nipkow's Pattern Rewrite

a set $\mathcal{M}$ of metavariables $\mathsf{t}, \mathsf{u}, \mathsf{x}, \ldots$ equipped with arities, and a set $\mathcal{F}$ of (function) symbols $f, g, \ldots$ equipped with binding arities, we define the terms by the following grammar, where we write $|\vec{x}|$ for the number k of variables in a list of variables $\vec{x} = x_1 \ldots x_k$. Here, the (simple) arity $\mathsf{arity}(\mathsf{x}) = k$ of x specifies that it takes k arguments, and the binding arity $\mathsf{arity}(f) = (n_1, \ldots, n_k)$ of f specifies that it takes k arguments and binds n_i variables $\vec{x}_i$ in its i-th argument t_i. Given $\mathcal{F}' \subseteq \mathcal{F}$, we write $\mathcal{T}(\mathcal{F}')$ for the set of terms containing only symbols in $\mathcal{F}'$.

$$\begin{array}{llll} t, u, v ::= & \mid x & x \in \mathcal{V} & \\ & \mid \mathsf{x}\{t_1, \ldots, t_k\} & \mathsf{x} \in \mathcal{M} & \text{with } \mathsf{arity}(\mathsf{x}) = k \\ & \mid f(\vec{x}_1.t_1, \ldots, \vec{x}_k.t_k) & f \in \mathcal{F} & \text{with } \mathsf{arity}(f) = (n_1, \ldots, n_k) \text{ and } |\vec{x}_i| = n_i \end{array}$$

We write $\mathsf{fv}(t)$ for the set of free variables of t and $\mathsf{mv}(t)$ for its set of metavariables. We often abbreviate $t_1, \ldots, t_k$ as $\vec{t}$, and $\vec{x}_1.t_1, \ldots, \vec{x}_k.t_k$ as $\mathbf{t}$. A *substitution* σ is a set of pairs either of the form t/x or $\vec{x}.t/\mathsf{x}$ where $\mathsf{arity}(\mathsf{x}) = |\vec{x}|$. We write $t[\sigma]$ for the application of σ to t, the interesting cases being $x[\sigma] = t$ if $t/x \in \sigma$ and $\mathsf{x}\{\vec{t}\}[\sigma] = u[\vec{t}[\sigma]/\vec{x}]$ if $\vec{x}.u/\mathsf{x} \in \sigma$.

A term t is said to be a *$\vec{x}$-pattern* if $\mathsf{fv}(t) \subseteq \vec{x}$ and all metavariables occurrences are of the form $\mathsf{x}\{\vec{x}, \vec{y}\}$ where $\vec{y}$ are the variables bound between the root of the term and this occurrence of x[3]. A *pattern* is then just a ε-pattern, where ε is the empty list.

A *rewrite system* $\mathcal{R}$ is a set of *rewrite rules* $l \longmapsto r$, where l is a pattern headed by a symbol and $\mathsf{mv}(r) \subseteq \mathsf{mv}(l)$ and $\mathsf{fv}(r) = \emptyset$ (we already have $\mathsf{fv}(l) = \emptyset$ from the definition of pattern). An *equational system* $\mathcal{E}$ is a set of *equations* $t \approx u$, where t, u are both patterns.[4] Given a rewrite system $\mathcal{R}$, we write $\longrightarrow$ for the rewrite relation generated by $\mathcal{R}$, and given an equational system $\mathcal{E}$ we write $\simeq$ for the congruence on terms generated by $\mathcal{E}$. We write $\mathcal{F}_{\mathcal{R}} \subseteq \mathcal{F}$ for the symbols appearing in some left-hand side of $\mathcal{R}$, and $\mathcal{F}_{\mathcal{E}} \subseteq \mathcal{F}$ for the symbols appearing in some equation of $\mathcal{E}$.

A *rewrite system modulo* is a pair $(\mathcal{R}, \mathcal{E})$, and we write $\equiv$ for the least equivalence relation containing $\longrightarrow$ and $\simeq$. As previously mentioned, there are many different definitions of Church-Rosser modulo in the literature, so here we will say that $(\mathcal{R}, \mathcal{E})$ is *weak* (resp. *strong*) *Church-Rosser modulo* when $t \equiv u$ implies $t \ (\simeq \circ \longrightarrow)^* \circ \simeq \circ {}^*(\longleftarrow \circ \simeq) \ u$ (resp. $t \longrightarrow^* \circ \simeq \circ {}^*\!\longleftarrow u$).

A *context* C is a linear pattern on distinguished metavariables $\square_1, \square_2, \ldots$ called *holes*. Given a context C with k holes and $\mathbf{t} = \vec{x}_1.t_1, \ldots, \vec{x}_k.t_k$ with $\mathsf{arity}(\square_i) = |\vec{x}_i|$ for all i, we write $C[\mathbf{t}]$ for $C[\vec{x}_1.t_1/\square_1, \ldots, \vec{x}_k.t_k/\square_k]$.

Finally, we will also consider the definition of *orthogonal rewriting* $\Longrightarrow$ (also known as *developements*, or *simultaneous reduction* [3]), given by the following rules. Here we write $\vec{t_1} \Longrightarrow \vec{t_2}$ when $\vec{t_i} = t_{i,1}, \ldots, t_{i,k}$ and $t_{1,j} \Longrightarrow t_{2,j}$, and $\mathbf{t}_1 \Longrightarrow \mathbf{t}_2$ when $\mathbf{t}_i = \vec{x}_1.t_{i,1}, \ldots, \vec{x}_k.t_{i,k}$ and $t_{1,j} \Longrightarrow t_{2,j}$, and $\sigma_1 \Longrightarrow \sigma_2$ when $\mathsf{dom}(\sigma_1) = \mathsf{dom}(\sigma_2)$ and $t_1 \Longrightarrow t_2$ whenever $t_i/x \in \sigma_i$ or $\vec{x}.t_i/\mathsf{x} \in \sigma_i$.

$$\begin{array}{c} l \longmapsto r \in \mathcal{R} \\ \mathsf{mv}(l) = \mathsf{dom}(\sigma) \end{array} \frac{\sigma \Longrightarrow \sigma'}{l[\sigma] \Longrightarrow r[\sigma']} \qquad \frac{\mathbf{t} \Longrightarrow \mathbf{t}'}{f(\mathbf{t}) \Longrightarrow f(\mathbf{t}')} \qquad \frac{\vec{t} \Longrightarrow \vec{t}'}{\mathsf{x}\{\vec{t}\} \Longrightarrow \mathsf{x}\{\vec{t}'\}} \qquad \frac{}{x \Longrightarrow x}$$

3 The first criterion

In order to explain our criterion and the intuition behind it, it is instructive to start with the following simple criterion for abstract rewriting. Recall that an abstract rewrite system modulo is given by a set $\mathcal{A}$ equipped with a binary relation $\succ$ and an equivalence relation $\sim$.

Systems (PRSs) [13] over a single base type but with only function symbols of order at most two and variables of order at most one. In particular, this allows us to use results developed for any of these formalisms.

[3]Compared with Miller's original definition, we require patterns to be fully applied.

[4]Note that we allow equations such as $+(\mathsf{n}, 0) \approx \mathsf{n}$, in which one of the sides is a metavariable.

Definition 1 (Unblocking subset of $\mathcal{A}$)**.** Given an abstract rewrite system modulo, we say that a subset $\mathcal{U} \subseteq \mathcal{A}$ is *unblocking* for it if:

(A) For all $a \in \mathcal{A}$ there is some $b \in \mathcal{U}$ with $a \sim b$.

(B) For all $a, a' \in \mathcal{A}$ and $b \in \mathcal{U}$ with $b \sim a \succ a'$ we have $b \succ b' \sim a'$ for some $b' \in \mathcal{A}$.

Condition **(B)** intuitively states that in elements $b \in \mathcal{U}$ redexes are maximally unblocked with respect to $\sim$, so that if $b \sim a$, then any redex that is available in a is also available in b. **(A)** then imposes that for every $\sim$ equivalence class one can always find an element in $\mathcal{U}$.

An interesting consequence of having an unblocking subset $\mathcal{U}$ is that we do not need to search the whole $\sim$ equivalence class for redexes, but instead we can just appeal to **(A)** to obtain a term in which all redexes are available. This motivates us to define the relation $\rhd_{\mathcal{U}}$ by $a \rhd_{\mathcal{U}} b$ iff $a \sim c \succ b$ for some $c \in \mathcal{U}$. We then have the following easy result — in the statement, we think of $\succ$ as orthogonal rewriting, which is why supposing the diamond property is reasonable.

Proposition 1. *Suppose that $\succ$ satisfies the diamond property and that we have an unblocking subset $\mathcal{U} \subseteq \mathcal{A}$. Then $a\ (\succ \cup \prec \cup \sim)^*\ b$ implies $a \rhd^*_{\mathcal{U}} \circ \sim \circ {}^*_{\mathcal{U}}\!\lhd b$.*

The main insight of our criterion is then that, under suitable conditions (satisfied for instance by Example 1), we can find an unblocking subset of terms. The intuition is that the only two ways that a redex can be blocked in Example 1 is if (1) some collapsable term is inserted in the middle of the redex (as in $@(+(\lambda(x.t), 0), u)$, where we have $+(x, 0) \simeq x$), or (2) some symbol in $\mathcal{E}$ matched by a rule left-hand side is not maximally exposed (as in $\mathbb{N}_{\mathsf{rec}}(+(x, \mathsf{S}(y)), p, xy.q)$, where S could be exposed using $+(x, \mathsf{S}(y)) \simeq \mathsf{S}(+(x, y))$). This motivates the following definition:

Definition 2 (Unblocked terms)**.** A context E is an *$\mathcal{E}$-fragment* of t if $E \in \mathcal{T}(\mathcal{F}_{\mathcal{E}})$ and $E[\mathbf{u}]$ is a subterm of t for some $\mathbf{u}$. A term t is said to be *unblocked* if, for all $\mathcal{E}$-fragments E of t:

(1) $E \simeq \square_i$ implies $E = \square_i$.

(2) $E \simeq f(\mathbf{t})$ with $f \in \mathcal{F}_{\mathcal{E}} \cap \mathcal{F}_{\mathcal{R}}$ implies $E = f(\mathbf{t}')$ with $\mathbf{t} \simeq \mathbf{t}'$.

Let us now state the assumptions of our criterion:

(i) Equations $t_1 \approx t_2 \in \mathcal{E}$ are linear, and we have $\mathsf{mv}(t_1) = \mathsf{mv}(t_2)$.

(ii) Symbols in $\mathcal{F}_{\mathcal{E}}$ have a binding arity of the form $(0, \ldots, 0)$, the list being possibly empty.

(iii) For every context $E \in \mathcal{T}(\mathcal{F}_{\mathcal{E}})$, there is some unblocked $E' \in \mathcal{T}(\mathcal{F}_{\mathcal{E}})$ with $E \simeq E'$.

(iv) $\mathcal{R}$ is left-linear and no left-hand side is headed by a symbol in $\mathcal{F}_{\mathcal{E}} \cap \mathcal{F}_{\mathcal{R}}$.

(v) Orthogonal rewriting ($\Longrightarrow$) with $\mathcal{R}$ satisfies the diamond property.

The goal of the rest of this section is then to show the following theorem, where we write $\overset{\sim}{\longrightarrow}$ for the relation defined by $t \overset{\sim}{\longrightarrow} t'$ iff $t \simeq u \longrightarrow t'$ for some unblocked u.

Theorem 1. *Suppose* **(i)**-**(v)**. *Then $t \equiv u$ implies $t \overset{\sim}{\longrightarrow}^* \circ \simeq \circ {}^*\overset{\sim}{\longleftarrow} u$ for all t, u. In particular, $(\mathcal{R}, \mathcal{E})$ is weak Church-Rosser modulo.*

Before showing the proof of Theorem 1, let us see how it can be applied to Example 1. Conditions **(i)**, **(ii)** and **(iv)** can be directly verified, whereas **(v)** follows from the fact that $\mathcal{R}$ is *orthogonal* [14, Theorem 4.8]. To show **(iii)**, let us first note that each context $E \in \mathcal{T}(\mathcal{F}_{\mathcal{E}})$ can be put in a normal-form. Indeed, writing $\langle \vec{t} \rangle$ for $+(t_1, \ldots +(t_{k-1}, t_k) \ldots)$ or 0 when $\vec{t}$ is empty, then for each context $E \in \mathcal{T}(\mathcal{F}_{\mathcal{E}})$ we have $E \simeq \mathsf{S}^k(\langle \square_1, \ldots, \square_n \rangle)$ for some unique k when E has n holes. We can easily see that this normal form is unblocked, showing **(iii)**.

Proof of Theorem 1

To prove Theorem 1, we set out to show that unblocked terms satisfy properties **(A)** and **(B)** of Definition 1 for the abstract rewrite system modulo defined by $\Longrightarrow$ and $\simeq$, which will be achieved by Propositions 2 and 3 respectively, and then apply Proposition 1.

A central tool of our proof is the fact that every term t can be decomposed as $t = \overline{E}[\vec{t}]$ where $\overline{E}$ is a context with $\overline{E} \in \mathcal{T}(\mathcal{F}_{\mathcal{E}})$, and none of the terms in $\vec{t}$ is headed by a symbol in $\mathcal{F}_{\mathcal{E}}$. We call $\overline{E}[\vec{t}]$ the *$\mathcal{E}$-decomposition* of t, which is unique modulo renaming of $\overline{E}$. A main property of $\mathcal{E}$-decompositions is that $\mathcal{E}$-conversions can be split along them, in the sense of the following lemma. Let us write $t_1 \cong t_2$ when we have $t_i = x$ or $t_i = \mathsf{x}\{\vec{v}_i\}$ and $\vec{v}_1 \simeq \vec{v}_2$ or $t_i = f(\mathbf{v}_i)$ and $\mathbf{v}_1 \simeq \mathbf{v}_2$. Let us then write $\vec{t}_1 \cong \vec{t}_2$ for its pointwise extension to the elements of $\vec{t}_i$.

Lemma 1 (Splitting of an $\mathcal{E}$-conversion)**.** *Suppose* **(i)**, **(ii)**, *and $t_1 \simeq t_2$ for some t_1, t_2. Writing $\overline{E}_i[\vec{u}_i]$ for the $\mathcal{E}$-decomposition of t_i, we then must have $\overline{E}_1 \simeq \overline{E}_2$ and $\vec{u}_1 \cong \vec{u}_2$.*

Proof. By induction on $t_1 \simeq t_2$. For the case $t_i = u_i[\sigma]$ with $u_1 \approx u_2 \in \mathcal{E}$ we crucially rely on the fact that equations are linear, which ensures that matching is completely local. □

Lemma 2. *Suppose* **(i)**, **(ii)** *and that we have $t_1 \simeq t_2$ with t_1 an unblocked term and t_2 a term not headed by a symbol in $\mathcal{F}_{\mathcal{E}}$. Then we have $t_1 \cong t_2$.*

Proof. We consider the $\mathcal{E}$-decompositions $t_i = \overline{E}_i[\vec{u}_i]$ and apply Lemma 1 to get $\overline{E}_1 \simeq \overline{E}_2$ and $\vec{u}_1 \cong \vec{u}_2$. But because t_2 is not of the form $f(\mathbf{t})$ for some $f \in \mathcal{F}_{\mathcal{E}}$, we must have $\overline{E}_2 = \square$, and because t_1 is unblocked we get $\overline{E}_1 = \square$, so we conclude $\vec{u}_i = t_i$ and thus $t_1 \cong t_2$. □

We can now show that the set of unblocking terms satisfies property **(A)** of Definition 1.

Proposition 2. *Suppose* **(i)**-**(iii)**. *For each t, there is some unblocked term u with $t \simeq u$.*

Proof. By induction on t. Case $t = x$ is trivial. Cases $t = \mathsf{x}\{\vec{t}\}$ and $t = f(\mathbf{t})$ with $f \notin \mathcal{F}_{\mathcal{E}}$ follow directly by the ih. For t headed by a symbol in $\mathcal{E}$, consider its decomposition $t = \overline{E}[\vec{t}]$. By ih we have $\vec{u}$ unblocked with $\vec{t} \simeq \vec{u}$, so by Lemma 2 we have $\vec{t} \cong \vec{u}$. Finally, by **(iii)** we get $\overline{E}' \in \mathcal{T}(\mathcal{F}_{\mathcal{E}})$ unblocked with $\overline{E} \simeq \overline{E}'$, and so $\overline{E}'[\vec{u}]$ is unblocked and $\overline{E}'[\vec{u}] \simeq t$. □

To show that the set of unblocking terms satisfies property **(B)** of Definition 1 we now only need the following technical lemma, shown by induction on t_1 and using Lemma 2.

Lemma 3. *Suppose* **(i)**-**(iv)**, *and $t_1[\sigma] \simeq t_2$ with t_2 unblocked and $t_1 \in \mathcal{T}(\mathcal{F}_{\mathcal{R}})$ a linear $\vec{x}$-pattern and $\mathsf{dom}(\sigma) = \mathsf{mv}(t_1)$. Then we have $t_2 = t_1[\sigma']$ for some $\sigma' \simeq \sigma$.*

In the following, let us write $\rho(t \Longrightarrow u)$ for the number of redexes contracted in $t \Longrightarrow u$ (even if there might be many derivations of $t \Longrightarrow u$, the relevant one can be inferred from the context).

Proposition 3. *Suppose* **(i)**-**(v)**. *If u is unblocked and $u \simeq t \Longrightarrow t'$ then $u \Longrightarrow u' \simeq t'$ and $\rho(t \Longrightarrow t') = \rho(u \Longrightarrow u')$.*

Proof. By induction on $t \Longrightarrow t'$. Almost all cases are either trivial or follow directly by applying Lemma 2 and the ih, and case $l[\sigma] \Longrightarrow r[\sigma']$ also uses Lemma 3. The case $f(\mathbf{t}) \Longrightarrow f(\mathbf{t}')$ is more interesting, and follows using Lemma 2 when $f \notin \mathcal{F}_{\mathcal{E}}$, or using Lemma 1 when $f \in \mathcal{F}_{\mathcal{E}}$. □

Corollary 1. *Suppose* **(i)**-**(v)**. *If u is unblocked and $u \simeq t \longrightarrow t'$ then $u \longrightarrow u' \simeq t'$.*

We can now prove Theorem 1. Let us write $t \overset{\sim}{\Longrightarrow} u$ when $t \simeq t' \Longrightarrow u$ with some t' unblocked.

Proof of Theorem 1. By Propositions 2 and 3, the set of unblocked terms is unblocking for $\Longrightarrow$ and $\simeq$. So by Proposition 1, and the fact that $\equiv$ equals $(\Longrightarrow \cup \Longleftarrow \cup \simeq)^*$, we get that $\equiv$ is included in $\overset{\sim}{\Longrightarrow}^* \circ \simeq \circ {}^*\overset{\sim}{\Longleftarrow}$. To conclude it now suffices to replace the $\overset{\sim}{\Longrightarrow}$ by $\overset{\sim}{\longrightarrow}$. To do this, we first show that $\overset{\sim}{\Longrightarrow}$ is included in $\overset{\sim}{\longrightarrow}^* \circ \simeq$ using Corollary 1, and then we show that $\overset{\sim}{\Longrightarrow}^* \circ \simeq$ is included in $\overset{\sim}{\longrightarrow}^* \circ \simeq$ by induction on $\overset{\sim}{\Longrightarrow}^*$. □

Before concluding this section, let us mention that Theorem 1 can probably be strengthen. More precisely, we think the requirement that $\mathsf{mv}(t_1) = \mathsf{mv}(t_2)$ in **(i)** can be dropped, and that it might be possible to weaken **(v)** to only require $\mathcal{R}$ to be confluent. Assumption **(ii)** also does not seem to be essential, and was made here to simplify proofs, because all our use cases satisfy it. On the other hand, all linearity assumptions are essential in our proof.

4 The second criterion

We now discuss an alternative and much easier criterion, which follows almost directly from a well-known result. In the following, let us write $\mathsf{CP}^{\pm}(\mathcal{R}_1, \mathcal{R}_2)$ for the set of *critical pairs* between rules of $\mathcal{R}_1$ and $\mathcal{R}_2$ (see for instance [13] for a definition), and $\mathsf{CP}(\mathcal{R})$ for $\mathsf{CP}^{\pm}(\mathcal{R}, \mathcal{R})$. Moreover, given $\mathcal{E}$ for which both sides of all equations are headed by symbols and have the same metavariables[5], let us write $\mathcal{E}^{\pm}$ for the rewrite system $\mathcal{E} \cup \mathcal{E}^{-1}$, and note that we have $\simeq_{\mathcal{E}} = \longrightarrow^*_{\mathcal{E}^{\pm}}$. Then, given a second-order rewrite system modulo $(\mathcal{R} \cup \mathcal{S}, \mathcal{E})$, consider the assumptions:

- **(a)** $\mathcal{R} \cup \mathcal{S}$ is left-linear
- **(b)** For all equations $t \approx u \in \mathcal{E}$, we have t, u linear and headed by symbols, and $\mathsf{mv}(t) = \mathsf{mv}(u)$
- **(c)** $\mathcal{R}$ is confluent
- **(d)** $(\mathcal{S}, \mathcal{E})$ is strong Church-Rosser modulo
- **(e)** $\mathsf{CP}^{\pm}(\mathcal{R}, \mathcal{S}) = \emptyset$ and $\mathsf{CP}^{\pm}(\mathcal{R}, \mathcal{E}^{\pm}) = \emptyset$

Theorem 2. *Suppose* **(a)**-**(e)**. *Then $(\mathcal{R} \cup \mathcal{S}, \mathcal{E})$ is strong Church-Rosser modulo.*

So to apply Theorem 2 we still have to prove strong Church-Rosser modulo of a smaller system. The point is that in some cases of interest this smaller system can be strongly-normalizing, enabling the use of criteria that rely on this property. Let us see an example of this.

Example 2. Consider a version of Example 1 in which equations $+(\mathsf{t}, 0) \approx \mathsf{t}$ and $+(\mathsf{t}, \mathsf{S}(\mathsf{u})) \approx \mathsf{S}(+(\mathsf{t}, \mathsf{u}))$ are removed from $\mathcal{E}$, and in which we consider an additional rewrite system $\mathcal{S}$ given by

$$\mathcal{S} := \quad +(\mathsf{t}, 0) \longmapsto \mathsf{t}, \quad +(\mathsf{t}, \mathsf{S}(\mathsf{u})) \longmapsto \mathsf{S}(+(\mathsf{t}, \mathsf{u})), \quad +(0, \mathsf{t}) \longmapsto \mathsf{t}, \quad +(\mathsf{S}(\mathsf{u}), \mathsf{t}) \longmapsto \mathsf{S}(+(\mathsf{t}, \mathsf{u}))$$

Then conditions **(a)**, **(b)** and **(e)** can be directly verified, while **(c)** follows by orthogonality and **(d)** follows by [13, Theorem 5.11][6], using the fact that $\longrightarrow_{\mathcal{S}} \circ \simeq$ is strongly normalizing and that $\mathsf{CP}^{\pm}(\mathcal{E}^{\pm}, \mathcal{S}) \cup \mathsf{CP}(\mathcal{S}) \subseteq \longrightarrow^!_{\mathcal{S}} \circ \simeq_{\mathcal{E}} \circ {}^!_{\mathcal{S}}\!\longleftarrow$, where $\longrightarrow^!$ denotes reduction to a normal form.

Let us now move to the proof of Theorem 2. As previously mentioned, it can be shown almost directly from a well-known result, more precisely the following:

Theorem 3 (Shown locally in the proof of Theorem 6.8 in [15])**.** *If $\mathcal{R}$ and $\mathcal{S}$ are left-linear Pattern Rewrite Systems (PRSs) with $\mathsf{CP}^{\pm}(\mathcal{R}, \mathcal{S}) = \emptyset$ then they commute.*

[5]This extra condition on $\mathcal{E}$ is important for the definition to make sense. For instance, if $\mathcal{E} := \{+(\mathsf{x}, 0) \approx \mathsf{x}\}$, then $\mathcal{E} \cup \mathcal{E}^{-1}$ does *not* define a rewrite system, as left-hand sides of rewrite rules should be headed by symbols.

[6]Actually, [13, Theorem 5.11] is a criterion for *confluence modulo*, but by [3, Exercise 14.3.7] in this case this is equivalent to strong Church-Rosser modulo, given that $\mathcal{S}$ is strongly normalizing.

Proof of Theorem 2. Let us start with the following claim:

Claim 1. *We have* $\simeq_{\mathcal{E}} \circ {}^{*}_{\mathcal{RS}}\!\longleftarrow \circ \longrightarrow^{*}_{\mathcal{S}} \circ \simeq_{\mathcal{E}} \subseteq \longrightarrow^{*}_{\mathcal{S}} \circ \simeq_{\mathcal{E}} \circ {}^{*}_{\mathcal{RS}}\!\longleftarrow$.

Proof of Claim 1. First of all, because $(\mathcal{S}, \mathcal{E})$ is strong Church-Rosser modulo, it is easy to see that it suffices to prove ${}^{*}_{\mathcal{RS}}\!\longleftarrow \circ \longrightarrow^{*}_{\mathcal{S}} \circ \simeq_{\mathcal{E}} \subseteq \longrightarrow^{*}_{\mathcal{S}} \circ \simeq_{\mathcal{E}} \circ {}^{*}_{\mathcal{RS}}\!\longleftarrow$. To do this, we now show that $t \; {}^{n}_{\mathcal{RS}}\!\longleftarrow \circ \longrightarrow^{*}_{\mathcal{S}} \circ \simeq_{\mathcal{E}} u$ implies $t \longrightarrow^{*}_{\mathcal{S}} \circ \simeq_{\mathcal{E}} \circ {}^{*}_{\mathcal{RS}}\!\longleftarrow u$ by induction on n, the base case being trivial. For the induction step, we have $t \; {}_{\mathcal{RS}}\!\longleftarrow t' \; {}^{n}_{\mathcal{RS}}\!\longleftarrow \circ \longrightarrow^{*}_{\mathcal{S}} \circ \simeq_{\mathcal{E}} u$, so by ih we get $t \; {}_{\mathcal{RS}}\!\longleftarrow t' \longrightarrow^{*}_{\mathcal{S}} \circ \simeq_{\mathcal{E}} \circ {}^{*}_{\mathcal{RS}}\!\longleftarrow u$. We now do a case analysis on $t \; {}_{\mathcal{RS}}\!\longleftarrow t'$. For the case $t \; {}_{\mathcal{R}}\!\longleftarrow t'$ this follows because $\mathcal{R}$ commutes with $\mathcal{S}$ and $\mathcal{E}^{\pm}$ (by Theorem 3) and the fact that $\simeq = \longrightarrow^{*}_{\mathcal{E}^{\pm}}$. For the case $t \; {}_{\mathcal{S}}\!\longleftarrow t'$ this follows because $(\mathcal{S}, \mathcal{E})$ is strong Church-Rosser modulo. ☐

We now proceed with the proof of Theorem 2. We have $\equiv$ equal to $(\longrightarrow_{\mathcal{RS}} \cup {}_{\mathcal{RS}}\!\longleftarrow \cup \simeq_{\mathcal{E}})^{*}$, so we prove the theorem by showing that $t \; (\longrightarrow_{\mathcal{RS}} \cup {}_{\mathcal{RS}}\!\longleftarrow \cup \simeq_{\mathcal{E}})^{n} \; u$ implies $t \longrightarrow^{*}_{\mathcal{RS}} \circ \simeq_{\mathcal{E}} \circ {}^{*}_{\mathcal{RS}}\!\longleftarrow u$ by induction on n. The base case is trivial, and for the induction step we have $t \; (\longrightarrow_{\mathcal{RS}} \cup {}_{\mathcal{RS}}\!\longleftarrow \cup \simeq_{\mathcal{E}})^{n} \; u' \; (\longrightarrow_{\mathcal{RS}} \cup {}_{\mathcal{RS}}\!\longleftarrow \cup \simeq_{\mathcal{E}}) \; u$, and by ih we have $t \longrightarrow^{*}_{\mathcal{RS}} \circ \simeq_{\mathcal{E}} \circ {}^{*}_{\mathcal{RS}}\!\longleftarrow u'$. We conclude by a case analysis on $u' \; (\longrightarrow_{\mathcal{RS}} \cup {}_{\mathcal{RS}}\!\longleftarrow \cup \simeq_{\mathcal{E}}) \; u$. Case $u' \; {}_{\mathcal{RS}}\!\longleftarrow u$ is trivial. Cases $u' \longrightarrow_{\mathcal{S}} u$ and $u' \simeq_{\mathcal{E}} u$ follow by Claim 1, and case $u' \longrightarrow_{\mathcal{R}} u$ follows because $\mathcal{R}$ commutes with $\mathcal{S}$ and $\mathcal{E}^{\pm}$ (by Theorem 3) and with itself (by confluence of $\mathcal{R}$). ☐

5 Discussion

In this work, we discussed two criteria for proving Church-Rosser modulo without normalization.

As previously mentioned, Theorem 2 can be shown almost directly from the fact that two left-linear PRSs with no critical pairs between them commute, so it might not be original (maybe it was already clear for others that it could be derived)[7]. However, until finding Theorems 1 and 2, it was unclear for us how to prove Church-Rosser modulo for second-order systems with non-terminating rules, and we think that this problem should be more discussed in the literature. With Theorem 2 in hand, showing this property can be possible by isolating a terminating subsystem $(\mathcal{S}, \mathcal{E})$, proving it strong Church-Rosser modulo with the criteria available in the literature (such as [13, Theorem 5.11]), and trying to verify the other hypotheses of Theorem 2 (as illustrated in Example 2). Because of this, we consider important to document here the existence of Theorem 2 and the strategy of how it can be applied, which will certainly be useful in the future for showing Church-Rosser modulo in an upcoming version of Dedukti with rewriting modulo [2].

A natural question is then how the two criteria compare. First of all, both of them require equations to be linear and rewrite rules to be left-linear, two important limitations, but which seem reasonable: indeed, non-terminating higher-order rules (such as β-reduction) are known to interact badly with non-left-linearity (see Klop's countexample [11])[8]. Then, some of the other hypotheses seem difficult to compare, and indeed each of our examples (Examples 1 and 2) only works with one of the two criteria. However, looking closer at the examples, we remark an interesting point: the unblocked terms in Example 1 turn out to be exactly the normal forms for the system $\mathcal{S}$ in Example 2, so the move from Example 1 to Example 2 replaces the search for an unblocked term by rewriting with $\mathcal{S}$. The latter option seems more desirable, as it yields the strong variant of Church-Rosser modulo, instead of just the weak. It is an open question for us whether there are interesting examples that can be covered by Theorem 1, but which do not admit a variant covered by Theorem 2, or if the second criterion is always more useful in practice.

[7]Let us mention that a similar result is given by Blanqui in [4, Theorem 15], yet it is neither stronger nor weaker than ours (e.g., it only shows confluence modulo, which is weaker than strong Church-Rosser modulo).

[8]An alternative is to consider a setting with *confinement* to avoid interactions between the higher-order rules and the equations, as done in [1]. Unfortunately, this would forbid interesting examples like Examples 1 and 2.

References

[1] Ali Assaf, Gilles Dowek, Jean-Pierre Jouannaud, and Jiaxiang Liu. Untyped Confluence In Dependent Type Theories. working paper or preprint, April 2017.

[2] Bruno Barras, Thiago Felicissimo, and Theo Winterhalter. Towards a logical framework modulo rewriting and equational theories. In *30th International Conference on Types for Proofs and Programs (TYPES 2024)*. https://types2024.itu.dk/abstracts.pdf#section.0.40.

[3] Marc Bezem, Jan Willem Klop, and Roel de Vrijer. *Term rewriting systems*. 2003.

[4] Frédéric Blanqui. Rewriting modulo in deduction modulo. In Robert Nieuwenhuis, editor, *Rewriting Techniques and Applications*, pages 395–409, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[5] Frédéric Blanqui, Gilles Dowek, Emilie Grienenberger, Gabriel Hondet, and François Thiré. A modular construction of type theories. *Logical Methods in Computer Science*, 2023.

[6] Alexandre Boudet and Evelyne Contejean. About the confluence of equational pattern rewrite systems. In Claude Kirchner and Hélène Kirchner, editors, *Automated Deduction — CADE-15*, pages 88–102, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[7] Thiago Felicissimo. Second-order church-rosser modulo, without normalization. Technical report, 2024. Can be found at http://www.lsv.fr/~felicissimo/.

[8] Makoto Hamana. Complete algebraic semantics for second-order rewriting systems based on abstract syntax with variable binding. *Mathematical Structures in Computer Science*, 32(4):542–573, 2022.

[9] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems: Abstract properties and applications to term rewriting systems. *Journal of the ACM (JACM)*, 1980.

[10] Jean-Pierre Jouannaud and Hélene Kirchner. Completion of a set of rules modulo a set of equations. In *11th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, 1984.

[11] Jan Willem Klop. *Combinatory reduction systems*. PhD thesis, Rijksuniversiteit Utrecht, 1963.

[12] Jan Willem Klop, Vincent Van Oostrom, and Femke Van Raamsdonk. Combinatory reduction systems: introduction and survey. *Theoretical computer science*, 121(1-2):279–308, 1993.

[13] Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical computer science*, 192(1):3–29, 1998.

[14] Tobias Nipkow. Orthogonal higher-order rewrite systems are confluent. In *International Conference on Typed Lambda Calculi and Applications*, pages 306–317. Springer, 1993.

[15] Vincent van Oostrom and Femke van Raamsdonk. Weak orthogonality implies confluence: The higher-order case. Technical report, 1994. http://www.javakade.nl/research/ps/ISRL-94-5.ps.

Rule Removal for Confluence[*]

Nao Hirokawa[1] and Kiraku Shintani[2]

[1] JAIST, Nomi, Japan
hirokawa@jaist.ac.jp
[2] Japan
s.kiraku@gmail.com

Abstract

We show how rule removal criteria for confluence can be derived from compositional confluence criteria.

1 Introduction

Rule removal for confluence is a method to remove unnecessary rules from term rewrite systems (TRSs). The method is characterized as a criterion that identifies an equi-confluent subsystem $\mathcal{S}$ of a TRS $\mathcal{R}$. Here a TRS $\mathcal{S}$ is a *subsystem* of $\mathcal{R}$ if $\mathcal{S} \subseteq \mathcal{R}$, and $\mathcal{R}$ and $\mathcal{S}$ are *equi-confluent* if confluence of $\mathcal{R}$ is equivalent to confluence of $\mathcal{S}$. Confluence criteria often demand termination or severe syntactic conditions for TRSs. Therefore, rule removal can improve applicability of confluence criteria and automated confluence analysis may benefit from this method. However, techniques for rule removal still remain to be explored. Up to our knowledge, the following trivial criterion is the only known result.

Theorem 1 ([7, 9])**.** *A TRS $\mathcal{R}$ and a subsystem $\mathcal{S}$ of $\mathcal{R}$ are equi-confluent if $\mathcal{R} \subseteq \to_{\mathcal{S}}^*$.*

Example 2. *Consider the TRS $\mathcal{R}$:*

$$1\colon ((\mathsf{S} \circ x) \circ y) \circ z \to (x \circ z) \circ (y \circ z) \qquad 2\colon (\mathsf{K} \circ x) \circ y \to x \qquad 3\colon ((\mathsf{S} \circ \mathsf{K}) \circ \mathsf{K}) \circ x \to x$$

For the subsystem $\mathcal{S} = \{1, 2\}$ of $\mathcal{R}$ the relation $((\mathsf{S} \circ \mathsf{K}) \circ \mathsf{K}) \circ x \to_{\mathcal{S}}^* x$ holds. According to Theorem 1, the TRSs $\mathcal{R}$ and $\mathcal{S}$ are equi-confluent. Since $\mathcal{S}$ is orthogonal, $\mathcal{S}$ is confluent. Hence, $\mathcal{R}$ is confluent.

In this note we show how to obtain rule removal criteria. Our idea is to exploit *compositional* confluence criteria [6, 7, 10]. Here a compositional criterion means a sufficient condition that confluence of a subsystem implies confluence of the original rewrite system. For instance, consider the following compositional criterion:

Theorem 3 ([7, Corollary 9])**.** *Suppose that $\mathcal{S}$ is a subsystem of a TRS $\mathcal{R}$ with $\mathcal{R} \subseteq \leftrightarrow_{\mathcal{S}}^*$. If $\mathcal{S}$ is confluent then $\mathcal{R}$ is confluent.*

Unfortunately, the converse does not hold (consider, e.g., $\mathcal{R} = \{\mathsf{a} \to \mathsf{b}, \mathsf{a} \to \mathsf{c}, \mathsf{b} \to \mathsf{c}\}$ and $\mathcal{S} = \{\mathsf{a} \to \mathsf{b}, \mathsf{a} \to \mathsf{c}\}$). However, if the converse is somehow guaranteed, the criterion can be used as a rule removal method. In the next section, we present a condition for the converse. The condition is combinable with any compositional confluence criterion. Using the fact, we demonstrate rule removal based on compositional versions of orthogonality, rule labeling, and Knuth and Bendix' criterion (Sections 3, 4, and 5). Throughout the note, we assume familiarity with term rewriting and the notions of parallel/extended critical pair [1, 3, 5, 11]. The contents of Sections 2 and 3this note are partly based on our recent publication [10].

[*]Supported by JSPS KAKENHI Grant Number JP22K11900.

2 A Condition for the Converse

Let $\mathcal{R}$ be a TRS and $\mathcal{S}$ a subsystem of $\mathcal{R}$. We show that if $\mathcal{R}\restriction_{\mathcal{S}} \subseteq \to_{\mathcal{S}}^*$ then confluence of $\mathcal{R}$ implies confluence of $\mathcal{S}$. Here $\mathcal{R}\restriction_{\mathcal{S}}$ stands for the TRS $\{\ell \to r \in \mathcal{R} \mid \mathcal{F}\mathsf{un}(\ell) \subseteq \mathcal{F}\mathsf{un}(\mathcal{S})\}$, and $\mathcal{F}\mathsf{un}(\ell)$ and $\mathcal{F}\mathsf{un}(\mathcal{S})$ for the sets of all function symbols in ℓ and in $\mathcal{S}$, respectively. The following auxiliary lemma explains the role of the condition $\mathcal{R}\restriction_{\mathcal{S}} \subseteq \to_{\mathcal{S}}^*$.

Lemma 1. *Suppose $\mathcal{R}\restriction_{\mathcal{S}} \subseteq \to_{\mathcal{S}}^*$.*

(1) If $s \to_{\mathcal{R}} t$ and $s \in \mathcal{T}(\mathcal{F}\mathsf{un}(\mathcal{S}), \mathcal{V})$ then $s \to_{\mathcal{S}}^* t$ and $t \in \mathcal{T}(\mathcal{F}\mathsf{un}(\mathcal{S}), \mathcal{V})$

(2) If $s \to_{\mathcal{R}}^* t$ and $s \in \mathcal{T}(\mathcal{F}\mathsf{un}(\mathcal{S}), \mathcal{V})$ then $s \to_{\mathcal{S}}^* t$.

As a consequence of Lemma 1(2), confluence of $\mathcal{R}$ carries over to confluence of $\mathcal{S}$, when the inclusion $\mathcal{R}\restriction_{\mathcal{S}} \subseteq \to_{\mathcal{S}}^*$ holds and the signature of $\mathcal{S}$ is $\mathcal{F}\mathsf{un}(\mathcal{S})$. The restriction against the signature of $\mathcal{S}$ can be lifted by the fact that confluence is preserved under signature extensions:

Proposition 4. *A TRS $\mathcal{S}$ is confluent if and only if the implication*

$$t \;{}_{\mathcal{S}}^{*}\!\leftarrow s \to_{\mathcal{S}}^* u \implies t \to_{\mathcal{S}}^* \cdot {}_{\mathcal{S}}^{*}\!\leftarrow u$$

holds for all terms $s, t, u \in \mathcal{T}(\mathcal{F}\mathsf{un}(\mathcal{S}), \mathcal{V})$.

Proof. Immediate from the modularity of confluence [13]. ☐

Theorem 5 ([10, Theorem 8.3])**.** *Let $\mathcal{R}$ be a TRS and $\mathcal{S}$ a subsystem with $\mathcal{R}\restriction_{\mathcal{S}} \subseteq \to_{\mathcal{S}}^*$. If $\mathcal{R}$ is confluent then $\mathcal{S}$ is confluent.*

Proof. Suppose that $\mathcal{R}$ is confluent. It is enough to show the implication in Proposition 4 for all $s, t, u \in \mathcal{T}(\mathcal{F}\mathsf{un}(\mathcal{S}), \mathcal{V})$. Suppose $t \;{}_{\mathcal{S}}^{*}\!\leftarrow s \to_{\mathcal{S}}^* u$. By confluence of $\mathcal{R}$ we have $t \to_{\mathcal{R}}^* v \;{}_{\mathcal{R}}^{*}\!\leftarrow u$ for some v. Therefore, Lemma 1(2) yields $t \to_{\mathcal{S}}^* v \;{}_{\mathcal{S}}^{*}\!\leftarrow u$. ☐

Theorem 5 enables us to use Theorem 3 as a rule removal method.

Example 6. *We show the confluence of the TRS $\mathcal{R}$:*

$$1\colon\ \mathsf{f}(x) \to x \qquad 2\colon\ \mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(x) \qquad 3\colon\ \mathsf{g}(x, x) \to \mathsf{g}(\mathsf{f}(x), \mathsf{f}(x))$$

For the subsystem $\mathcal{S} = \{1, 2\}$ the inclusion $\mathcal{R} \subseteq \leftrightarrow_{\mathcal{S}}^*$ holds. Because $\mathcal{F}\mathsf{un}(\mathcal{S}) = \{\mathsf{f}\}$, we obtain $\mathcal{R}\restriction_{\mathcal{S}} = \{1, 2\} = \mathcal{S}$. Thus, $\mathcal{R}\restriction_{\mathcal{S}} \subseteq \to_{\mathcal{S}}^*$ holds. Therefore, $\mathcal{R}$ and $\mathcal{S}$ are equi-confluent. Since $\mathcal{S}$ is terminating and all critical pairs are joinable, Knuth and Bendix' criterion applies. Thus, $\mathcal{S}$ is confluent, and hence $\mathcal{R}$ is confluent.

3 A Generalization of Orthogonality

In this section we illustrate rule removal based on parallel critical pairs [3]. There is a compositional confluence criterion that originates from orthogonality (see [10, Section 5]), and it employs a subsystem that can close all parallel critical pairs.

Let t be a term and let P be a set of (pairwise) parallel positions in t. By $t[u_p]_{p \in P}$ we denote the term that results from replacing in t the subterm at p by a term u_p for all $p \in P$. We write $s \overset{P}{\twoheadrightarrow\!\!\!\!\!\shortmid\;\,} t$ if the parallel step $s \twoheadrightarrow\!\!\!\!\!\shortmid\;\, t$ contracts redexes at positions in P.

Definition 7. *Let $\mathcal{R}$ and $\mathcal{S}$ be TRSs, $\ell \to r$ a variant of an $\mathcal{S}$-rule, and $\{\ell_p \to r_p\}_{p \in P}$ a family of variants of $\mathcal{R}$-rules, where P is a set of positions. A local peak $(\ell\sigma)[r_p\sigma]_{p\in P}\ {}_{\mathcal{R}}\!\!\mathrel{\leftarrow\!\!\!\!\shortmid\!\shortmid}\ \ell\sigma \xrightarrow{\epsilon}_{\mathcal{S}} r\sigma$ is called a* parallel critical peak *between $\mathcal{R}$ and $\mathcal{S}$ if the following conditions hold:*

- *P is a non-empty set of parallel function positions in ℓ;*
- *none of rules $\ell \to r$ and $\ell_p \to r_p$ for $p \in P$ shares a variable with other rules;*
- *σ is a most general unifier of $\{\ell_p \approx (\ell|_p)\}_{p\in P}$; and*
- *if $P = \{\epsilon\}$ then $\ell_\epsilon \to r_\epsilon$ is not a variant of $\ell \to r$.*

When $t\ {}_{\mathcal{R}}\!\!\overset{P}{\mathrel{\leftarrow\!\!\!\!\shortmid\!\shortmid}}\ s \xrightarrow{\epsilon}_{\mathcal{S}} u$ is a parallel critical peak, the pair (t,u) is called a parallel critical pair *between $\mathcal{R}$ and $\mathcal{S}$. The set of all parallel critical pairs between $\mathcal{R}$ and $\mathcal{R}$ is denoted by* $\mathsf{PCP}(\mathcal{R})$.

Theorem 8 ([10, Theorem 5.4])**.** *Let $\mathcal{R}$ be a left-linear TRS and $\mathcal{S}$ a subsystem of $\mathcal{R}$ with $\mathsf{PCP}(\mathcal{R}) \subseteq \leftrightarrow_{\mathcal{S}}^*$. If $\mathcal{S}$ is confluent then $\mathcal{R}$ is confluent.*

Theorem 8 coincides with weak orthogonality when $\mathcal{S} = \varnothing$, and generalizes Theorem 3 for left-linear TRSs as the inclusions $\mathsf{PCP}(\mathcal{R}) \subseteq \leftrightarrow_{\mathcal{R}}^* \subseteq \leftrightarrow_{\mathcal{S}}^*$ hold.

Example 9. *Consider the left-linear TRS $\mathcal{R}$:*

$$1\colon\ ((\mathsf{S} \circ x) \circ y) \circ z \to (x \circ z) \circ (y \circ z) \quad 2\colon\ (\mathsf{K} \circ x) \circ y \to x \quad 3\colon\ \mathsf{I} \circ x \to x \quad 4\colon\ \mathsf{I} \to (\mathsf{S} \circ \mathsf{K}) \circ \mathsf{K}$$

Let $\mathcal{S} = \{1,2\}$. The TRS $\mathcal{R}$ admits only one parallel critical pair, and it is joinable by $\mathcal{S}$-steps:

$$\begin{tikzcd}
 & \mathsf{I} \circ x \arrow[dl, bend right, "\{1\}", "\mathcal{R}"'] \arrow[dr, bend left, "\epsilon", "\mathcal{R}"'] & \\
((\mathsf{S} \circ \mathsf{K}) \circ \mathsf{K}) \circ x \arrow[r, "\mathcal{S}"'] & (\mathsf{K} \circ x) \circ (\mathsf{K} \circ x) \arrow[r, "\mathcal{S}"'] & x
\end{tikzcd}$$

Because of $\mathcal{F}\mathsf{un}(\mathcal{S}) = \{\circ, \mathsf{S}, \mathsf{K}\}$ we obtain $\mathcal{R}\restriction_{\mathcal{S}} = \{1,2\} = \mathcal{S}$. Thus, according to Theorems 5 and 8, the TRSs $\mathcal{R}$ and $\mathcal{S}$ are equi-confluent. The latter TRS $\mathcal{S}$ is orthogonal, so it is confluent. Hence, the confluence of $\mathcal{R}$ follows. Note that the combination of Theorem 3 and Theorem 5 is not available for $\mathcal{R}$ because no proper subsystem $\mathcal{S}$ of $\mathcal{R}$ satisfies $\mathcal{R} \subseteq \leftrightarrow_{\mathcal{S}}^*$ and $\mathcal{R}\restriction_{\mathcal{S}} \subseteq \to_{\mathcal{S}}^*$.

4 Rule Labeling

Rule labeling [8, 15] is a powerful confluence method for left-linear TRSs and its compositional version is known [10]. Let $\mathcal{R}$ be a TRS. *Labeling functions* are functions from $\mathcal{R}$ to $\mathbb{N}$. Given a labeling function ϕ and an integer k, we write $\mathcal{R}_{\phi,k}$ for the TRS $\{\alpha \in \mathcal{R} \mid \phi(\alpha) \leqslant k\}$. The rewrite relation of $\mathcal{R}_{\phi,k}$ is referred to as $\to_{\phi,k}$. Note that $\mathcal{R}_{\phi,-1} = \varnothing$ as $\phi(\alpha) \geqslant 0$. Suppose ϕ and ψ are labeling functions and $k, m \in \mathbb{N}$. A peak $t\ {}_{\phi,k}\!\!\overset{P}{\mathrel{\leftarrow\!\!\!\!\shortmid\!\shortmid}}\ s \to_{\psi,m} u$ is (ψ,ϕ)-*decreasing* if

$$t \xleftrightarrow[k-1]{*} \cdot \underset{\psi,m}{\mathrel{\shortmid\!\shortmid\!\!\!\!\rightarrow}}{}^{=} \cdot \xleftrightarrow[\max\{m,n\}-1]{*} v \;{}^{=}\!\underset{\phi,k}{\overset{P'}{\mathrel{\leftarrow\!\!\!\!\shortmid\!\shortmid}}} \cdot \xleftrightarrow[m-1]{*} u$$

and $\mathcal{V}\mathsf{ar}(v, P') \subseteq \mathcal{V}\mathsf{ar}(s, P)$ for some v and set P' of parallel positions in v. Here $\leftrightarrow_i$ stands for the union of ${}_{\phi,i}\!\leftarrow$ and $\to_{\psi,i}$, and $\mathcal{V}\mathsf{ar}(w, Q)$ for the union of $\mathcal{V}\mathsf{ar}(w|_p)$ for all $p \in Q$.

Theorem 10 ([10, Theorem 6.5])**.** *Let $\mathcal{R}$ be a left-linear TRS and ϕ, ψ labeling functions for $\mathcal{R}$ with $\mathcal{R}_{\phi,0} = \mathcal{R}_{\psi,0} =: \mathcal{S}$. Assume that the following conditions hold for all $(k,m) \in \mathbb{N}^2 \setminus \{(0,0)\}$:*

- *Every parallel critical peak of form* $t \underset{\phi,k}{\xleftarrow{\;\Vert\;}} s \xrightarrow[\psi,m]{\epsilon} u$ *is* (ψ,ϕ)*-decreasing.*
- *Every parallel critical peak of form* $t \underset{\psi,m}{\xleftarrow{\;\Vert\;}} s \xrightarrow[\phi,k]{\epsilon} u$ *is* (ϕ,ψ)*-decreasing.*

If $\mathcal{S}$ *is confluent then* $\mathcal{R}$ *is confluent.*

It is known that Theorem 10 subsumes Theorem 8 and the parallel closedness results by Huet [4] and Toyama [12, 14], see [10]. We now use this powerful theorem for rule removal.

Example 11. *We show the confluence of the left-linear TRS* $\mathcal{R}$*:*

$1\colon\ \mathsf{s}(x) + y \to \mathsf{s}(x+y)$	$3\colon\ (x+y)+z \to x+(y+z)$	$5\colon\ \mathsf{d}(x) \to x+x$
$2\colon\ x+\mathsf{s}(y) \to \mathsf{s}(x+y)$	$4\colon\ \infty \to \mathsf{s}(\infty)$	$6\colon\ \mathsf{d}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{d}(x)))$

Define the labeling functions ϕ *and* ψ *as follows:*

$$\phi(1)=\phi(2)=\phi(3)=0 \qquad \phi(4)=\phi(5)=1 \qquad \phi(6)=2$$
$$\psi(1)=\psi(2)=\psi(3)=0 \qquad \psi(4)=\psi(5)=1 \qquad \psi(6)=2$$

Then $\mathcal{R}_{\phi,0} = \mathcal{R}_{\psi,0} = \{1,2,3\} := \mathcal{S}$ *and the decreasingness conditions in Theorem 10 are satisfied. For instance, the parallel critical peak between rules* 5 *and* 6 *admits the diagram:*

$$\begin{tikzcd}
 & & \mathsf{d}(\mathsf{s}(x)) \arrow[dll, bend right=15, "\{\epsilon\}"', "\phi{,}1", "\Vert" description] \arrow[dr, bend left=15, "\epsilon", "\psi{,}2"'] & \\
\mathsf{s}(x)+\mathsf{s}(x) \arrow[r, "\psi{,}0"'] & \mathsf{s}(x+\mathsf{s}(x)) \arrow[r, "\psi{,}0"'] & \mathsf{s}(\mathsf{s}(x+x)) & \mathsf{s}(\mathsf{s}(\mathsf{d}(x))) \arrow[l, "\{1\}"', "\phi{,}1", "\Vert" description]
\end{tikzcd}$$

Because $\mathcal{V}\mathsf{ar}(\mathsf{s}(\mathsf{s}(x+x))), \{1\}) = \{x\} = \mathcal{V}\mathsf{ar}(\mathsf{d}(\mathsf{s}(x)), \{\epsilon\})$*, the peak is* (ψ,ϕ)*-decreasing. Since* $\mathcal{F}\mathsf{un}(\mathcal{S}) = \{\mathsf{s},+\}$*, we obtain* $\mathcal{R}{\restriction_{\mathcal{S}}} = \mathcal{S} \subseteq \to_{\mathcal{S}}^{*}$*. Therefore, by Theorems 5 and 10 the TRSs* $\mathcal{R}$ *and* $\mathcal{S}$ *are equi-confluent. Since* $\mathcal{S}$ *is terminating and all critical pairs are joinable, the confluence of* $\mathcal{S}$ *is shown by Knuth and Bendix' criterion. Hence,* $\mathcal{R}$ *is confluent.*

5 A Generalization of Knuth and Bendix' Criterion

Finally, we illustrate rule removal based on the generalization of Knuth and Bendix' criterion [6], which is available for non-left-linear TRSs. Given a TRS $\mathcal{R}$, we write $\mathcal{R}^{\circ}$ for the extended TRS $\{\ell^{\circ} \to r \mid \ell \to r \in \mathcal{R}\}$. Here ℓ° stands for an arbitrary but fixed linear term that admits a variable substitution σ with $\ell^{\circ}\sigma = \ell$. TRSs $\mathcal{R}$ and $\mathcal{S}$ are *strongly non-overlapping* if there is no overlap between $\mathcal{R}^{\circ}$ and $\mathcal{S}^{\circ}$. For example, the TRSs $\{\mathsf{f}(x,x) \to \mathsf{a}\}$ and $\{\mathsf{g}(x) \to x\}$ are strongly non-overlapping, and the TRSs $\{\mathsf{f}(x,x) \to \mathsf{a}\}$ and $\{\mathsf{f}(\mathsf{s}(x),x) \to \mathsf{b}\}$ are *not* because $\mathsf{f}(x,x)^{\circ} = \mathsf{f}(x_1,x_2)$ and $\mathsf{f}(\mathsf{s}(x),x)^{\circ} = \mathsf{f}(\mathsf{s}(x_3),x_4)$ are unifiable. When $\to_{\mathcal{S}}^{*} \cdot \to_{\mathcal{R}} \cdot \to_{\mathcal{S}}^{*}$ is terminating, the pair of TRSs $\mathcal{R}$ and $\mathcal{S}$, denoted by $\mathcal{R}/\mathcal{S}$, is called *relatively terminating.*

Definition 12 ([5])**.** *Let* $\mathcal{R}$ *and* $\mathcal{S}$ *be TRSs and let* $\ell_1 \to r_1$ *and* $\ell_2 \to r_2$ *be variants of* $\mathcal{R}$*-rules having no common variables. Suppose that the following conditions hold:*

- p *is a function position in* ℓ_2*;*
- σ *is a substitution in a complete set of* $\mathcal{S}$*-unifiers of* ℓ_1 *and* $\ell_2|_p$*; and*
- *if* $p = \epsilon$ *then* $\ell_1 \to r_1$ *is not a variant of* $\ell_2 \to r_2$*.*

Then the pair $((\ell_2\sigma)[r_1\sigma]_p, r_2\sigma)$ *is called an* extended $\mathcal{S}$-critical pair *of* $\mathcal{R}$. *The set of all extended* $\mathcal{S}$*-critical pairs of* $\mathcal{R}$ *is denoted by* $\mathsf{CP}_{\mathcal{S}}(\mathcal{R})$.

Theorem 13 ([6, Theorem 2])**.** *Let* $\mathcal{R}$ *and* $\mathcal{S}$ *be strongly non-overlapping TRSs such that* $\mathcal{R}/\mathcal{S}$ *is terminating and* $\mathsf{CP}_{\mathcal{S}}(\mathcal{R}) \subseteq {\downarrow_{\mathcal{R}\cup\mathcal{S}}}$. *If* $\mathcal{S}$ *is confluent then* $\mathcal{R}\cup\mathcal{S}$ *is confluent.*

When $\mathcal{S} = \varnothing$, the theorem coincides with Knuth and Bendix' criterion.

Example 14. *We show the confluence of the non-left-linear TRS [6, Example 10]:*

$$
\begin{array}{rlrl}
1: & \mathsf{eq}(\mathsf{s}(n), x : xs, x : ys) \to \mathsf{eq}(n, xs, ys) & 3: & \mathsf{nats} \to \mathsf{0} : \mathsf{inc}(\mathsf{nats}) \\
2: & \mathsf{eq}(n, xs, xs) \to \mathsf{T} & 4: & \mathsf{inc}(x : xs) \to \mathsf{s}(x) : \mathsf{inc}(xs)
\end{array}
$$

Let $\mathcal{R} = \{1, 2\}$ *and* $\mathcal{S} = \{3, 4\}$. *The TRSs* $\mathcal{R}$ *and* $\mathcal{S}$ *are strongly non-overlapping, and the termination of* $\mathcal{R}/\mathcal{S}$ *is shown by, e.g., the termination tool* $\mathsf{T_TT_2}$. *Except for the symmetric version, the TRS* $\mathcal{R}$ *admits the only one* $\mathcal{S}$*-critical pair originating from* $\mathsf{eq}(\mathsf{s}(n), x : xs, x : xs)$, *and it is joinable by* $\mathcal{R}\cup\mathcal{S}$*:*

$$\begin{tikzcd}
 & \mathsf{eq}(\mathsf{s}(n), x : xs, x : xs) \arrow[dl, bend right, "\epsilon"', "\mathcal{R}"] \arrow[dr, bend left, "\epsilon", "\mathcal{R}"'] & \\
\mathsf{eq}(n, xs, xs) \arrow[rr, "*"', "\mathcal{R}\cup\mathcal{S}"'] & & \mathsf{T}
\end{tikzcd}$$

Since $\mathcal{F}\mathsf{un}(\mathcal{S}) = \{\mathsf{0}, \mathsf{s}, :, \mathsf{nat}, \mathsf{inc}\}$, *we obtain* $(\mathcal{R}\cup\mathcal{S})\restriction_{\mathcal{S}} = \{3, 4\} = \mathcal{S}$. *Therefore, the equi-confluence of* $\mathcal{R}\cup\mathcal{S}$ *and* $\mathcal{S}$ *follows by Theorems 5 and 13. As* $\mathcal{S}$ *is orthogonal,* $\mathcal{S}$ *is confluent. Hence, the original TRS* $\mathcal{R}\cup\mathcal{S}$ *is confluent.*

6 Conclusion

We have presented how rule removal criteria can be derived from compositional confluence criteria. We are currently preparing a next version of the confluence tool Hakusan [10], where confluence is analyzed by successive application of rule removal criteria. Experimental data based on the tool will be reported during the workshop. As we have seen, Theorem 5 is a key for rule removal. We anticipate that applicability of the theorem is improved by incorporating modularity results such as *layer systems* [2].

References

[1] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998. doi:10.1017/CBO9781139172752.

[2] B. Felgenhauer, A. Middeldorp, H. Zankl, and V. van Oostrom. Layer systems for proving confluence. *ACM Trans. Comput. Logic*, 16(2):1–32, 2015. doi:10.1145/2710017.

[3] B. Gramlich. Confluence without termination via parallel critical pairs. In *Proceedings of 21st International Colloquium on Trees in Algebra and Programming*, volume 1059 of *LNCS*, pages 211–225, 1996. doi:10.1007/3-540-61064-2_39.

[4] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980. doi:10.1145/322217.322230.

[5] J.-P. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal on Computing*, 15(4):1155–1194, 1986. doi:10.1137/0215084.

[6] D. Klein and N. Hirokawa. Confluence of non-left-linear TRSs via relative termination. In *Proceedings of 18th International Conference on Logic Programming and Automated Reasoning*, volume 7180 of *LNCS*, pages 258–273, 2012. doi:10.1007/978-3-642-28717-6_21.

[7] J. Nagele, B. Felgenhauer, and A. Middeldorp. Improving automatic confluence analysis of rewrite systems by redundant rules. In *Proceedings of 26th International Conference on Rewriting Techniques and Applications*, volume 36 of *LIPIcs*, pages 257–268, 2015. doi:10.4230/LIPIcs.RTA.2015.257.

[8] V. van Oostrom. Confluence by decreasing diagrams, converted. In *Proceedings of 19th International Conference on Rewriting Techniques and Applications*, volume 5117 of *LNCS*, pages 306–320, 2008. doi:10.1007/978-3-540-70590-1_21.

[9] K. Shintani and N. Hirokawa. CoLL: A confluence tool for left-linear term rewrite systems. In *Proceedings of 25th International Conference on Automated Deduction*, volume 9195 of *LNCS (LNAI)*, pages 127–136, 2015. doi:10.1007/978-3-319-21401-6_8.

[10] K. Shintani and N. Hirokawa. Compositional confluence criteria. *Logical Methods in Computer Science*, 20:6:1–6:28, 2024. doi:10.46298/lmcs-20(1:6)2024.

[11] Terese. *Term Rewriting Systems.* Cambridge University Press, 2003. doi:10.1017/S095679680400526X.

[12] Y. Toyama. On the Church–Rosser property of term rewriting systems. In *NTT ECL Technical Report*, volume No. 17672. NTT, 1981. Japanese.

[13] Y. Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM*, 34(1):128–143, 1987. doi:10.1145/7531.7534.

[14] Y. Toyama. Commutativity of term rewriting systems. In *Programming of Future Generation Computers II*, pages 393–407. North-Holland, 1988.

[15] H. Zankl, B. Felgenhauer, and A. Middeldorp. Labelings for decreasing diagrams. *Journal of Automated Reasoning*, 54(2):101–133, 2015. doi:10.1007/s10817-014-9316-y.

Confluence by the Z-property for De Bruijn's λ-calculus with nameless dummies, based on PLFA*

Vincent van Oostrom

University of Sussex, School of Engineering and Informatics, Brighton, UK
`Vincent.van-Oostrom@sussex.ac.uk`

Abstract

We discuss the Agda formalisation of a proof of confluence of the untyped $\lambda\beta$-calculus, more precisely, of the Z-property for De Bruijn's λ-calculus notation with nameless dummies, based on the latter's formalisation in Programming Language Foundations in Agda.

Introduction We consider confluence of the rewrite system $\to_\beta$ of the $\lambda\beta$-calculus [3].

A sufficient condition for a rewrite system $\to$ to be confluent is the *Z-property*, the existence of a map $\bullet$ on its objects such that if $a \to b$ then $b \twoheadrightarrow a^\bullet \twoheadrightarrow b^\bullet$. The Z-property originates with Loader [14, Section 4.1] for the $\lambda\beta$-calculus with the *Gross–Knuth* bullet map, and with Dehornoy [9] for self-distributivity with the *full distribution* bullet map. See [15] for more on it and on its theory, e.g. that it is not *necessary* for confluence.

In [15, Remark 52][10, Conclusions] we claimed that the proof of confluence of $\to_\beta$ by means of the Z-property given there was (a bit) shorter than the proof of the same due to Takahashi [18] via the *angle* property, the existence of a map $\bullet$ on the objects of $\to$ together with a rewrite system $\mathrel{-\!\!\ominus\!\!\rightarrow}$ with $\to \subseteq \mathrel{-\!\!\ominus\!\!\rightarrow} \subseteq \twoheadrightarrow$ and such that if $a \mathrel{-\!\!\ominus\!\!\rightarrow} b$ then $b \mathrel{-\!\!\ominus\!\!\rightarrow} a^\bullet$ [19, 15].

Remark. In the same paper we showed [15, Lemma 8] that the Z-property and the angle property are *equivalent*. That the proof using the former is (a bit) shorter nonetheless is due to that it dispenses with introducing the auxiliary rewrite system $\mathrel{-\!\!\ominus\!\!\rightarrow}$; the $\bullet$-map suffices.

Here we investigate that claim specialised to a *formalisation* of the λ-calculus, in particular to the formalisation of it in Agda in Programming Language Foundations in Agda [20]. There, in the module Confluence, a proof of confluence of De Bruijn's λ-calculus with nameless dummies [7] was formalised via the angle property. We present an alternative formalisation[1] of confluence via the Z-property; it again is (a bit) shorter than the one via the angle property. Our alternative formalisation is based on the October 2021 snapshot[2] of [20]; it may replace its Confluence module but is still based on its modules Untyped (formalising De Bruijn's λ-calculus with nameless dummies) and Substitution (formalising the so-called Substitution Lemma).

We first describe and comment on the key ingredients of the Agda code from [20] we took as the basis of our formalisation, the *objects* and *steps* of the rewrite system, and motivate why we did so. We then do the same for our supplementary code, the alternative formalisation of Confluence, split into: (i) supplementary code (50 loc) showing the system to be a *term* rewrite system, (ii) code (65 loc) for *the Z-property*, and (iii) code (25 loc) for *inferring confluence* from Z, supplemented with comments on **design** decisions. We assume knowledge of rewriting [19] and the λ-calculus [3] with nameless dummies [7].

*This work is licensed under the Creative Commons Attribution 4.0 International License 🅭 🅯. The Agda code was developed in the 1st week of October 2021 based on the then-current version of Programming Language Foundations in Agda [20] on a MacBook Pro 2019 with Agda 2.6.1.1, Emacs 26.3.

[1]HTML source at http://www.javakade.nl/research/agda/plfa.part2.ConfluenceZ4.html (external hyperlinks do not work; see PLFA) and pure Agda code at http://www.javakade.nl/research/agda/ConfluenceZ4.agda.

[2]That snapshot already had the proof via the angle property of the 2022 version of [20], instead of the Tait–Martin-Löf proof via the *diamond* property found in its 2020 version, i.e. based on the existence of a rewrite system $\mathrel{-\!\!\ominus\!\!\rightarrow}$ with $\to \subseteq \mathrel{-\!\!\ominus\!\!\rightarrow} \subseteq \twoheadrightarrow$, such that for $a \mathrel{-\!\!\ominus\!\!\rightarrow} b$ and $a \mathrel{-\!\!\ominus\!\!\rightarrow} c$ there is a d with $b \mathrel{-\!\!\ominus\!\!\rightarrow} d$ and $c \mathrel{-\!\!\ominus\!\!\rightarrow} d$.

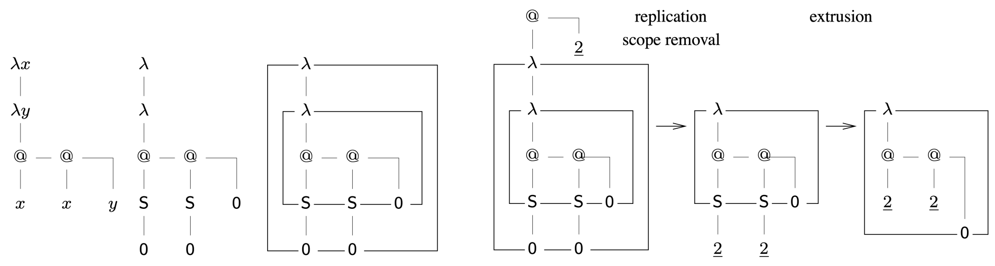


Figure 1: Church numeral $\underline{2}$ with names, nameless and scopes (left), and factoring $\underline{2}\,\underline{2} \to_\beta \underline{4}$ (right)

Objects of the rewrite system The objects of the rewrite system are the λ-terms with nameless dummies of [7], obtained by restricting *generalised* such terms [5, 12] $t ::= \mathsf{0} \,|\, \mathsf{S}t \,|\, \lambda t \,|\, t\,t$ by requiring t in $\mathsf{S}t$ to be an *index*, a unary natural number generated from $\mathsf{0}$ and S only. Figure 1 (left) exemplifies the correspondence between named [3] and nameless [7] (better: *uni*-named) λ-terms for the Church numeral $\underline{2}$:=$\lambda\lambda(\mathsf{S0})\,((\mathsf{S0})\,\mathsf{0})$ (we employ the notational conventions of [3]).

It is a matter of hygiene to factor a notion for *open* λ-terms through the same for *closed* λ-terms, e.g. it is hygienic to define an *open* term M to be *solvable* [3] if its *closure* $\hat{M}$ is, i.e. if $\hat{M}\,\vec{P} =_\beta \mathsf{I}$ for some $\vec{P}$. The formalisation in Untyped follows suit and carves out from the above λ-terms the *closed* ones [2, 16] by means of the inference rules (to be read bottom–up):

$$\frac{\mathsf{S}i \vdash \mathsf{0}}{}\,\mathsf{0} \qquad \frac{\mathsf{S}i \vdash \mathsf{S}t}{i \vdash t}\,\mathsf{S} \qquad \frac{i \vdash \lambda t}{\mathsf{S}i \vdash t}\,\lambda \qquad \frac{i \vdash t_1 t_2}{i \vdash t_1 \quad i \vdash t_2}\,@$$

Here the *scoping* judgment $i \vdash t$ asserts that t is a λ-term having i as *upper bound* on its free indices, in particular $0 \vdash t$ asserts t is closed. See [16, example 3] for a derivation showing $\underline{2}$ is indeed closed. Scoping judgments are implemented in the module Untyped of [20], there split into separate judgments for *indices* i and λ-*terms* t built from them by means of abstraction (λ) and application (@). As is good engineering practice, we *reuse* it for our formalisation.

Remark. The reason for calling $i \vdash t$ a *scoping* judgment in [20] is that making the scopes of λ-abstractions explicit [12, 16] as boxes, as done in Figure 1 for the Church numeral $\underline{2}$, makes it clear that each S represents an *end-of-scope* of, and each $\mathsf{0}$ a *reference* to, its λ-abstraction. The notion of *binding* for named λ-terms is recovered for nameless λ-terms by the context-free notion of *matching parentheses* along paths of the abstract syntax tree of the λ-term, viewing each λ as an opening parenthesis (and each end-of-scope S or reference $\mathsf{0}$ as a closing parenthesis) [12]. A box is formed by an abstraction together with its matching end-of-scopes and references.

Design. Since in the $\lambda\beta$-calculus terms are constructed just from abstractions and applications, in the literature [3] one usually abstains from formalising the concept of a *symbol* / *signature* [19] to construct terms from. The formalisation in the module Untyped follows suit. For applied λ-calculi this design choice leads to *redundancy*, e.g. an inference rule for each construct.

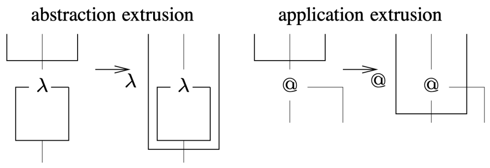
Steps of the rewrite system The $\to_\beta$-steps are generated by closing the β-rule scheme [7]:

$$\boldsymbol{\beta} : (\lambda t)\,s \to t[s]$$

under contexts, where $t[s]$ denotes substitution of s for the free occurrences of 0 in t; it must be hygienic in that it should *preserve* scoping: if $i \vdash (\lambda t)\,s$, i.e. if $\mathsf{S}i \vdash t$ and $i \vdash s$, then $i \vdash t[s]$.

$$\begin{aligned} \mathsf{0}[s]^{\mathsf{0}} &= s \\ \mathsf{0}[s]^{\mathsf{S}i} &= \mathsf{0} \\ (\mathsf{S}t)[s]^{\mathsf{0}} &= t \\ (\mathsf{S}t)[s]^{\mathsf{S}i} &= \mathsf{S}t[s]^{i} \\ (\lambda t)[s]^{i} &= \lambda t[s]^{\mathsf{S}i} \\ (t_1 t_2)[s]^{i} &= t_1[s]^{i} t_2[s]^{i} \end{aligned}$$

$$\begin{aligned} \mathsf{S}\lambda t &\to_{\lambda} \lambda t^{\mathsf{S0}} \\ \mathsf{S}(t_1 t_2) &\to_{@} \mathsf{S}t_1 \mathsf{S}t_2 \end{aligned}$$



$$\begin{aligned} t^{\mathsf{0}} &= \mathsf{S}t \\ \mathsf{0}^{\mathsf{S}i} &= \mathsf{0} \\ (\mathsf{S}t)^{\mathsf{S}i} &= \mathsf{S}t^{i} \\ (\lambda t)^{\mathsf{S}i} &= \lambda t^{\mathsf{SS}i} \\ (t_1 t_2)^{\mathsf{S}i} &= t_1^{\mathsf{S}i} t_2^{\mathsf{S}i} \end{aligned}$$

Figure 2: Alternative substitution (left), scope extrusion (middle) and minimal lifting (right)

Contracting $(\lambda t)\, s$ can be split into two phases. In the first phase the λ together with its box are removed and each $\mathsf{0}$ on it is replaced by s, see Figure 1 (right). This yields a *generalised* λ-term, upon which the second phase *extrudes* any offending scopes, to yield the λ-term $t[s]$.

Contracting $(\lambda t)\, s$ to $t[s]$ in the module Untyped of [20] factors through subst-zero yielding a *parallel* substitution σ mapping $\mathsf{0} \mapsto s$ and $\mathsf{S}i \mapsto i$ implementing the first phase, with the second *lifting* phase brought about during the parallel substitution process $t[s] = t^{\sigma}$ via subst.[3] That subst satisfies the so-called *Substitution Lemma* [3] is shown in the module Substitution of [20]. Since our code is modular in it, we *reuse* also that module.

Remark. The Substitution Lemma is unavoidable in *any* proof of confluence as it captures the resolution of the *critical peak* of the β-rule arising from *nested* redex-patterns $(\lambda y.(\lambda x.M)\, N))\, L$:

$$(\lambda y.M[x := N])\, L \leftarrow (\lambda y.(\lambda x.M)\, N))\, L \to ((\lambda x.M)\, N)[y := L] = (\lambda x.M[y := L])(N[y := L])$$

in a canonical way by means of a valley, having the Substitution Lemma (SL) in its middle:

$$(\lambda y.M[x := N])\, L \to M[x := N][y := L] =_{SL} M[y := L][x := N[y := L]] \leftarrow (\lambda x.M[y := L])(N[y := L])$$

Design. To allow for an algebraic proof of the Substitution Lemma, the module Substitution of [20] first defines the basic substitution *operations* corresponding to the explicit substitution *operators* of [17], based on [1] but having laws that are *complete* w.r.t. *De Bruijn* algebras in the sense that explicit substitution expressions are equivalent iff they are provably equal by means of the laws. Then these operations are linked to substitutions, the algebraic laws are proven to hold for substitutions, and finally the Substitution Lemma is proven using the algebraic laws.

The result of taking that indirect approach is that the module Substitution is largish. As it essentially only provides (both to [20] and to our code) the Substitution Lemma subst-commute, it should be feasible and interesting to replace it by a smaller module on a different basis.

E.g., [13] employs only 2 operations, substitution (`subst_rec`) and lifting (`lift_rec`), and 6 laws governing their interaction to formalise the Substitution Lemma in Coq. That development can be seen as implementing $t[s]$ as $t[s]^{\mathsf{0}}$ followed by *maximal scope extrusion* as sketched above, defined in [16], and repeated here in Figure 2. We formalised[4] the Substitution Lemma for *generalised* λ-terms in Coq, based on *minimal* scope extrusion, only extruding scopes as far as needed to make β-redexes visible [12]. Somewhat surprisingly Huet's 6 laws still hold.

Design. In the module Untyped of [20] steps are generated as in [3] via the *compatible* closure, via clauses conventionally called ζ for abstraction and ξ_1 and ξ_2 for (the arguments of) application. Already here the absence of a signature leads to redundancy (between ξ_1 and ξ_2). The redundancy will be worse for applied λ-calculi having larger signatures.

[3]Now lifting the substitution into a box top–down via its λ, instead of bottom–up via an S as earlier.
[4]HTML at http://www.javakade.nl/research/coq/ConfluencebyZofGeneralisedLocalDeBruijninCoq.html.

```
app-cong : ∀{Γ} {K L M N : Γ ⊢ ★} → K —↠ L → M —↠ N → K · M —↠ L · N
rew-rew : ∀{Γ} {M N : Γ , ★ ⊢ ★ } {K L : Γ ⊢ ★ }
  → M —↠ N
  → K —↠ L
    --------------------
  → M [ K ] —↠ N [ L ]
```

Figure 3: Closure of reduction under application (app-cong) and substitution (rew-rew)

Term rewrite system To say that we have a *term* rewrite system [19] is to say that steps (and reductions) are closed under *contexts* and *substitutions* [19]. We supplemented the results in [20] with the remaining ones needed to show that, with the main ones being closure of reduction under application (app-cong) and under substitution (rew-rew), displayed in Figure 3. The 50 loc needed for it constitute Part I of our formalisation and confirm sanity of the module Untyped.

Remark. For rew-rew, we first showed closure of *steps* under substitution (stp-subst) after which we pointwise extended the definition of many-step reduction —↠ to many-step reduction of substitutions —↠s and showed the latter to be closed under term-contexts trm-subst, finally allowing us to show reductions are closed under substitution for rew-rew, all missing from [20].

Design. Since in the λ-calculus there is only a single rule β, one usually abstains from formalising the concept of a *rule*, instead giving the corresponding ad hoc *rule scheme* directly (one can think of the scheme as obtained from the rule by taking all its substitution instances). Since in the module Untyped there is only one such rule scheme, conventionally [3] called $\boldsymbol{\beta}$, this is manageable. However, for applied λ-calculi this would again lead to redundancy.

It should be feasible and interesting to give an alternative formalisation of the untyped λ-calculus in [20] as a higher-order term rewrite systems [19, Chapter 9], proceeding in the spirit of [19, Chapter 8] and [6] via a signature of both *function* and *rule* symbols, with (*multi*)steps simply being terms over that signature, so-called *proofterms*.

Z To establish the Z-property for $\to_\beta$ we follow the approach pioneered by Loader [14] with key properties outlined in [15, Sections 3.3.1, 3.4] for orthogonal term rewrite systems. We take as bullet map • the *full development* map [15, Definition 42], recursively mapping a term M to the target obtained by contracting all β-redexes in M; its Agda code is presented in Figure 4.

Remark. For the history of this bullet map for the λ-calculus, going back to Gross, Knuth, and Takahashi among others, see [4, 15].

Design. I initially tried formalising the Z-property for the *full superdevelopment* [15, Definition 42] bullet map going back to Aczel, van Raamsdonk and others, cf. [11], inside–out contracting β-redexes starting from M, but failed. (*Defining* the full superdevelopment map was unproblematic but I failed to *use* it.) It should be of interest to have a *single* generic proof that can be instantiated to both the full *development* and *superdevelopment* bullet maps.

```
_• : ∀ {Γ A} → Γ ⊢ A → Γ ⊢ A
(` x)• = ` x
(ƛ M)• = ƛ (M •)
((ƛ M) · N)• = M • [ N • ]
(M · N)• = (M •) ·(N •)
```

Figure 4: Full-development bullet map •

The Agda code for the four key properties below, taken from [15, Sections 3.3.1, 3.4][5], with the Z-property the conjunction of (upperbound) and (monotonic), fitting on one page (Figure 5) bears witness to the simplicity of the proof of confluence via the Z-property. These 65 loc constitute Part II of the formalisation, its core, the rest being generic / boilerplate code.

(extensive) $M \twoheadrightarrow_\beta M^\bullet$ ($\bullet$ gives an upperbound);

(upperbound) if $M \to_\beta N$ then $N \twoheadrightarrow_\beta M^\bullet$ ($\bullet$ gives an upperbound on all 1-step reducts); $\bullet$

(rhss) $(M^\bullet)^{\sigma^\bullet} \twoheadrightarrow_\beta (M^\sigma)^\bullet$ (applying $\bullet$ to whole exceeds applying it to parts (of the rhs)); and

(monotonic) if $M \to_\beta N$ then $M^\bullet \twoheadrightarrow_\beta N^\bullet$ ($\bullet$ is monotonic with respect to reduction).

Remark. Each property may be shown either by induction on the term M (and then distinguishing cases on steps (proofterms) $M \to_\beta N$ possible from M) or by induction on $M \to_\beta N$ (and then distinguishing cases on the possible shapes of M). This choice is largely immaterial. Here we found it convenient to prove the properties (extensive) and (rhhs) by induction on the term M, and the properties (upperbound) and (monotonic) by induction on $M \to_\beta N$.

Confluence That the Z-property entails confluence, holds abstractly [15, 8]. We arbitrarily chose to formalise the first proof of it in [15, Lemma 51], factoring confluence through the Strip Lemma [3, 20] strip, and that through an easy recombination of (extensive), (upperbound) and (rew-monotonic), the lifting of (monotonic) from steps to reductions. Combining the trivial formalisation of this final part with that of Parts I,II yields some 140 loc[6] confirming our claim.

Conclusion Upon sending comments on the module Confluence to the authors of [20] in the summer of 2021, Jeremy Siek challenged me to formalise the comments myself. Since for a long time I had wanted to learn some Agda and in October of that year I had some time on my hands, I then took up the challenge resulting in the current alternative formalisation of that module. It closely followed the pen-and-paper proof [15] and confirmed my comments (that the module Confluence could be simplified by basing it on the Z-property; its core is only 65 loc). Formalising was surprisingly smooth (taking a full week only; not knowing Agda), helped by that the code in [20] is well-explained and modularised, allowing for easy reuse and adaptation. Accordingly, the proofs of the key properties in Figure 5 are by direct inductions on terms and steps (proofterms), mostly making use of properties in the modules Untyped and Substitution.

Design. Having renaming *and* substitution *and* parallel substitution causes redundancy in [20].

Whether one can easily formalise a proof of confluence of the λ-calculus is often used as a litmus test for proof assistants so formalisations abound [11]. When starting the endeavour I was not aware of other formalisations of confluence via the Z-property in Agda. At IWC 2023, I learned from Riccardo Treglia that he had supervised the student project in 2019/2020 of Andrea Laretto on formalising the Church–Rosser theorem for the λ-calculus in Agda (not relying on [20]). I have tried to initiate a discussion, but that has not materialised yet. The authors of [20] still have to react to both the original comments and the subsequent formalisation, and the improvements suggested to the untyped λ-calculus and its confluence mostly still apply.

[5]The (rhss) property here is the one which was intended in [15, Lemma 44(Rhs)]. The property (Rhs) given there, that $M^{(\sigma^\bullet)}$ reduces to $(M^\sigma)^\bullet$, is correct in it being a trivial consequence of (extensive) and (rhss). However, (Rhs) did not capture the idea that the result ($(M^\bullet)^{(\sigma^\bullet)}$) of applying the bullet map to the *parts* (M and σ) of the right-hand side of the term rewrite rule (M^σ) reduces to the result ($(M^\sigma)^\bullet$) of applying the bullet map to right-hand side. Consequently, the property (Rhs) given there is too weak to be useful.

[6]Of which 50 loc show the λ-calculus in the module Untyped is a term rewrite system so belong in that module, leaving 90 loc for confluence itself in the module Untyped.

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991. doi:10.1017/S0956796800000186.

[2] T. Altenkirch and B. Reus. Monadic presentations of lambda terms using generalized inductive types. In *Computer Science Logic (CSL '99), Madrid, 1999*, volume 1683 of *Lecture Notes in Computer Science*, pages 453–468. Springer, 1999. doi:10.1007/3-540-48168-0_32.

[3] H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103. North-Holland, Amsterdam, 2nd revised edition, 1984.

[4] H.P. Barendregt, J. Bergstra, J.W. Klop, and H. Volken. Degrees, reductions and representability in the lambda calculus. Preprint 22, Utrecht University, Department of Mathematics, 1976. URL: https://dspace.library.uu.nl/handle/1874/15119.

[5] R.S. Bird and R.A. Paterson. de Bruijn notation as a nested datatype. *Journal of Functional Programming*, 9(1):77–91, 1999. doi:10.1017/S0956796899003366.

[6] H.J.S. Bruggink. *Equivalence of Reductions in Higher-Order Rewriting*. PhD thesis, Utrecht University, 2008. URL: https://dspace.library.uu.nl/handle/1874/27575.

[7] N.G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church–Rosser theorem. *Indagationes Mathematicae (Proceedings)*, 75(5):381–392, 1972. doi:10.1016/1385-7258(72)90034-0.

[8] F.L.C. de Moura and L.O. Rezende. A formalization of the (compositional) Z property. Conference on Intelligent Computer Mathematics, 2021. URL: http://flaviomoura.info/files/fmm21.pdf.

[9] P. Dehornoy. *Braids and Self-Distributivity*, volume 192 of *Progress in Mathematics*. Birkhäuser, 2000.

[10] P. Dehornoy and V. van Oostrom. Z; proving confluence by monotonic single-step upperbound functions. In *Logical Models of Reasoning and Computation (LMRC-08), Moscow*, 2008. URL: http://www.javakade.nl/research/talk/lmrc060508.pdf.

[11] B. Felgenhauer, J. Nagele, V. van Oostrom, and C. Sternagel. The Z property. *Archive of Formal Proofs*, June 2016. URL: https://www.isa-afp.org/entries/Rewriting_Z.shtml.

[12] D. Hendriks and V. van Oostrom. ʎ. In *Proceedings of CADE 19*, volume 2741 of *Lecture Notes in Artificial Intelligence*, pages 136–150. Springer, 2003. doi:10.1007/978-3-540-45085-6_11.

[13] G. Huet. Residual theory in λ-calculus: a formal development. *Journal of Functional Programming*, 4(3):371–394, July 1994. doi:10.1017/S0956796800001106.

[14] R. Loader. Notes on simply typed lambda calculus. ECS-LFCS- 98-381, Laboratory for Foundations of Computer Science, The University of Edinburgh, February 1998. URL: http://www.lfcs.inf.ed.ac.uk/reports/98/ECS-LFCS-98-381/.

[15] V. van Oostrom. Z; syntax-free developments. In *6th Conference on Formal Structures for Computation and Deduction (FSCD 2021), Buenos Aires*, volume 195 of *LIPIcs*, pages 24:1–24:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSCD.2021.24.

[16] V. van Oostrom, K.J. van de Looij, and M. Zwitserlood. Lambdascope. In *Workshop on Algebra and Logic on Programming Systems, Kyoto*, page 9 pp., April 2004. URL: http://www.javakade.nl/research/pdf/lambdascope.pdf.

[17] S. Schäfer, G. Smolka, and T. Tebbi. Completeness and decidability of de Bruijn substitution algebra in Coq. In *Proceedings of the 2015 Conference on Certified Programs and Proofs (CPP 2015), India*, pages 67–73. ACM, 2015. doi:10.1145/2676724.2693163.

[18] M. Takahashi. Parallel reductions in λ-calculus. *Information and Computation*, 118:120–127, April 1995. doi:10.1006/inco.1995.1057.

[19] Terese. *Term Rewriting Systems*. Cambridge University Press, 2003.

[20] P. Wadler, W. Kokke, and J.G. Siek. *Programming Language Foundations in Agda*. 2024. http://plfa.inf.ed.ac.uk/20.07/, http://plfa.inf.ed.ac.uk/22.08/, http://plfa.inf.ed.ac.uk/.

```
extensive : ∀ {Γ A} → (M : Γ ⊢ A) → M —↠ M •
extensive (` _) = _ ∎
extensive (ƛ M) = abs-cong (extensive M)
extensive ((ƛ M) · N) = _ —→⟨ β ⟩ rew-rew (extensive M) (extensive N)
extensive (` _ · N) = appR-cong (extensive N)
extensive (L · M · N) = app-cong (extensive (L · M)) (extensive N)

upperbound : ∀ {Γ} → {M N : Γ ⊢ ★}
  → M —→ N
    ---------
  → N —↠ M •
upperbound {_} {ƛ _} (ζ ϕ) = abs-cong (upperbound ϕ)
upperbound {_} {(` _) · _} {_} (ξ₂ ϕ) = appR-cong (upperbound ϕ)
upperbound {_} {(ƛ _) · M} {((ƛ _) · M)} (ξ₁ (ζ ϕ)) = _ —→⟨ β ⟩ rew-rew (upperbound ϕ) (extensive M)
upperbound {_} {(ƛ L) · _} {.((ƛ L) · _)} (ξ₂ ϕ) = _ —→⟨ β ⟩ rew-rew (extensive L) (upperbound ϕ)
upperbound {_} {(ƛ L) · M} {.(subst (subst-zero M) L)} β = rew-rew (extensive L) (extensive M)
upperbound {_} {_ · _ · M} {.(_ · M)} (ξ₁ ϕ) = app-cong (upperbound ϕ) (extensive M)
upperbound {_} {K · L · _} {.(_ · _ · _)} (ξ₂ ϕ) = app-cong (extensive (K · L)) (upperbound ϕ)

rhss : ∀{Γ Δ} (M : Γ ⊢ ★) {σ τ : Subst Γ Δ} → ((x : Γ ∋ ★) → τ x ≡ σ x •)
    -----------------------------
  → subst τ (M •) —↠ (subst σ M)•
rhss (` x) eq rewrite (eq x) = _ ∎
rhss (ƛ M) eq = abs-cong (rhss M (exts-bullet eq))
rhss ((` x) · M) {σ} eq rewrite (eq x) = —↠-trans
  (appR-cong (rhss M eq)) (app-bullet (σ x) (subst σ M)) where
  {- auxiliary rhs/monotonicity lemma for application -}
    app-bullet : ∀{Γ} (L M : Γ ⊢ ★) → L • · M • —↠ (L · M)•
    app-bullet (` _) _ = _ ∎
    app-bullet (ƛ _) _ = (_ —→⟨ β ⟩ _ ∎)
    app-bullet (_ · _) _ = _ ∎
rhss ((ƛ L) · M) {τ = τ} eq rewrite (sym (subst-commute {N = L •} {M •} {τ})) =
  rew-rew (rhss L (exts-bullet eq)) (rhss M eq)
rhss (K · L · M) eq = app-cong (rhss (K · L) eq) (rhss M eq)

monotonic : ∀{Γ} → {M N : Γ ⊢ ★}
  → M —→ N
    ----------
  → M • —↠ N •
monotonic (ζ ϕ) = abs-cong (monotonic ϕ)
monotonic {_} {(` _) · _} {(` _) · _} (ξ₂ ϕ) = appR-cong (monotonic ϕ)
monotonic {Γ} {(ƛ M) · N} {.(subst (subst-zero N) M)} β = rhss M bullet-zero where
  {- bullet commutes with lifting terms to substitutions -}
  bullet-zero : (x : Γ , ★ ∋ ★) → subst-zero (N •) x ≡ subst-zero N x •
  bullet-zero Z = refl
  bullet-zero (S x) = refl
monotonic {_} {(ƛ _) · _} {(ƛ _) · _} (ξ₁ (ζ ϕ)) = rew-rew (monotonic ϕ) (_ ∎)
monotonic {_} {(ƛ M) · _} {.((ƛ M) · _)} (ξ₂ ϕ) = rew-rew (M • ∎) (monotonic ϕ)
monotonic {_} {_ · _ · _} {(ƛ _) · _} (ξ₁ ϕ) = —↠-trans (appL-cong (monotonic ϕ)) (_ —→⟨ β ⟩ _ ∎)
monotonic {_} {_ · _ · _} {(` _) · _} (ξ₁ ϕ) = appL-cong (monotonic ϕ)
monotonic {_} {_ · _ · _} {_ · _ · _} (ξ₁ ϕ) = appL-cong (monotonic ϕ)
monotonic {_} {_ · _ · _} {_ · _ · _} (ξ₂ ϕ) = appR-cong (monotonic ϕ)
```

Figure 5: Key properties (extensive,upperbound,rhss,monotonic) for confluence of λ-calculus by Z

Confluence Competition 2024

Rául Gutiérrez[1], Aart Middeldorp[2], Naoki Nishida[3],
Teppei Saito[4], and René Thiemann[2]

[1] Universidad Politécnica de Madrid, Madrid, Spain
[2] Department of Computer Science, University of Innsbruck, Austria
[3] Department of Computing and Software Systems, Nagoya University, Japan
[4] School of Information Science, JAIST, Japan

The next few pages in these proceedings contain the descriptions of the tools participating in the 13th Confluence Competition (CoCo 2024). CoCo is a yearly competition in which software tools attempt to automatically (dis)prove confluence and related properties of rewrite systems in a variety of formats. For a detailed description we refer to [2]. This year there were 14 tools (listed in order of registration) participating in 11 categories (listed in order of first appearance in CoCo):

	TRS	CTRS	GCR	UNR	UNC	NFP	COM	INF	SRS	CSR	LCTRS
crest											✓
Hakusan	✓								✓		
Moca								✓			
CO3		✓						✓			
CONFident	✓	✓							✓	✓	
infChecker								✓			
AGCP			✓								
ACP	✓	✓		✓	✓		✓		✓		
CeTA	⋆						⋆	⋆	⋆		
CSI	✓			✓	✓	✓			✓		
FORT-h	✓		✓	✓	✓	✓	✓				
FORTify	⋆		⋆	⋆	⋆	⋆	⋆				
CRaris											✓
NaTT								✓			

This year CoCo adopted the new ARI[1] format for input problems. In addition, there is new category (LCTRS) about confluence of logically constrained rewrite systems [1], with two new tools (crest and CRaris). Tools producing certifiable output in a specific category team up with a certifier and participate as combination in that category. The certifiers in CoCo 2024 are CeTA and FORTify. The latter teamed up with FORT-h. The former with ACP, CSI and Hakusan in the SRS and TRS categories, with ACP in the COM category, and with Moca in the INF category.

The winning (for combined YES/NO answers) tools[2] of CoCo 2023 participated as demonstration tools, to provide a benchmark to measure progress. The live run of CoCo 2024 on

[1]https://project-coco.uibk.ac.at/ARI/
[2]They are not listed in the table but see http://project-coco.uibk.ac.at/2023/results.php.

StarExec [3] can be viewed at `http://cocograph.uibk.ac.at/2024.html`. Further information about CoCo 2024, including a description of the categories and detailed results, can be obtained from `http://project-coco.uibk.ac.at/2024/`.

Acknowledgements The CoCo steering committee is grateful to Nao Hirokawa and Fabian Mitterwallner for their support.

References

[1] Cynthia Kop and Naoki Nishida. Term Rewriting with Logical Constraints. In *Proc. 9th International Symposium on Frontiers of Combining Systems*, volume 8152 of *LNCS (LNAI)*, pages 343–358, 2013. doi: 10.1007/978-3-642-40885-4_24.

[2] Aart Middeldorp, Julian Nagele, and Kiraku Shintani. CoCo 2019: Report on the Eighth Confluence Competition. *International Journal on Software Tools for Technology Transfer*, 2021. doi: 10.1007/s10009-021-00620-4.

[3] Aaron Stump, Geoff Sutcliffe, and Cesare Tinelli. StarExec: A Cross-Community Infrastructure for Logic Solving. In *Proc. 7th International Joint Conference on Automated Reasoning*, volume 8562 of *LNCS (LNAI)*, pages 367–373, 2014. doi: 10.1007/978-3-319-08587-6_28.

CoCo 2024 Participant: crest 0.8*

Jonas Schöpf and Aart Middeldorp

Department of Computer Science, University of Innsbruck, Innsbruck, Austria
{jonas.schoepf,aart.middeldorp}@uibk.ac.at

The Constrained REwriting Software Tool (crest, for short) is a tool for automatically proving (non-)confluence of logically constrained rewrite systems (LCTRSs). LCTRSs are an extension of term rewrite systems (TRSs) where rules are equipped with constraints. These must be satisfied in order for a rule to be applied. Analysis methods for plain TRSs cannot be applied in this setting, hence special techniques, adapted for LCTRSs, are used.

The development of crest started in the beginning of 2023 as part of the ARI project[1] and initial experiments were presented at CADE-29 in 2023 [3]. More information and executables of crest can be found at

http://cl-informatik.uibk.ac.at/software/crest/

Currently, crest supports (non-)confluence [2, 3, 4] and limited (non-)termination analysis [2, 1]. The confluence methods mostly rely on closure properties of (parallel) critical pairs. In particular confluence analysis via (weak) orthogonality [2], strong closedness [3], (almost) parallel closedness [3], (almost) development closedness [4], closure of parallel critical pairs [4] or joinable critical pairs of terminating LCTRSs [5] is implemented. Moreover, non-confluence is shown by finding a witness for two different normal forms originating from the same peak.

Input problems in the new ARI format are accepted.[2] However, crest extends the format slightly, e.g. by allowing theory symbols, which are only available in the SMT-LIB logic of quantifier-free bit-vector arithmetic.[3] Also most variable sort annotations, which are mandatory in the LCTRS format, can be omitted due to the implemented type inference algorithm. Reasoning about the theory part of an LCTRS is utilized via the SMT solver Z3.[4]

Our tool crest participates in the LCTRS category of CoCo 2024.

References

[1] Cynthia Kop. Termination of LCTRSs. *CoRR*, abs/1601.03206, 2016. doi: https://doi.org/10.48550/ARXIV.1601.03206.

[2] Cynthia Kop and Naoki Nishida. Term rewriting with logical constraints. In Pascal Fontaine, Christophe Ringeissen, and Renate A. Schmidt, editors, *Proc. 9th International Symposium on Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Artificial Intelligence*, pages 343–358, 2013. doi: https://doi.org/10.1007/978-3-642-40885-4_24.

[3] Jonas Schöpf and Aart Middeldorp. Confluence criteria for logically constrained rewrite systems. In Brigitte Pientka and Cesare Tinelli, editors, *Proc. 29th International Conference on Automated Deduction*, volume 14132 of *Lecture Notes in Artificial Intelligence*, pages 474–490, 2023. doi: https://doi.org/10.1007/978-3-031-38499-8_27.

*This research is funded by the Austrian Science Fund (FWF) project I 5943-N.

[1] https://ari-informatik.uibk.ac.at/

[2] https://project-coco.uibk.ac.at/ARI/lctrs.php

[3] https://smt-lib.org/logics-all.shtml#QF_BV

[4] https://www.microsoft.com/en-us/research/project/z3-3/

[4] Jonas Schöpf, Fabian Mitterwallner, and Aart Middeldorp. Confluence of logically constrained rewrite systems revisited. *CoRR*, abs/2402.13552, 2024. doi: https://doi.org/10.48550/ARXIV.2402.13552.

[5] Sarah Winkler and Aart Middeldorp. Completion for logically constrained rewriting. In Hélène Kirchner, editor, *Proc. 3rd International Conference on Formal Structures for Computation and Deduction*, volume 108 of *Leibniz International Proceedings in Informatics*, pages 30:1–30:18, 2018. doi: https://doi.org/10.4230/LIPIcs.FSCD.2018.30.

Hakusan 0.11: A Confluence Tool

Fuyuki Kawano, Nao Hirokawa, and Kiraku Shintani

JAIST, Japan
`{f-kawano,hirokawa}@jaist.ac.jp`, `s.kiraku@gmail.com`

Hakusan (`https://www.jaist.ac.jp/project/saigawa/`) is a confluence tool for left-linear term rewrite systems (TRSs). It analyzes confluence by successive application of *rule removal* criteria [6, 4] based on rule labeling [5, 8] and critical pair systems [3]. Confluence proofs of Hakusan are now verifiable by CeTA [7], see [2].

With a small example we illustrate confluence analysis in our tool. Let $\mathcal{R}$ be a TRS and $\mathcal{S} \subseteq \mathcal{R}$ a subsystem of $\mathcal{R}$. We write $\mathsf{PCPS}(\mathcal{R},\mathcal{S})$ for the *parallel critical pair system* given by $\{s \to t, s \to u \mid t \mathrel{{}_{\mathcal{R}}\!\!\leftarrow\!\!\!\!\!\!\parallel} s \xrightarrow{\epsilon}_{\mathcal{R}} u$ is a parallel critical peak but not $t \leftrightarrow^*_{\mathcal{S}} u\}$. Moreover, we write $\mathcal{R}{\restriction_{\mathcal{S}}}$ for the TRS $\{\ell \to r \in \mathcal{R} \mid \mathcal{F}\mathsf{un}(\ell) \subseteq \mathcal{F}\mathsf{un}(\mathcal{S})\}$.

Theorem 1 ([4])**.** *Let $\mathcal{R}$ be a left-linear TRS and $\mathcal{S} \subseteq \mathcal{R}$. If every parallel critical pair of $\mathcal{R}$ is joinable, $\mathsf{PCPS}(\mathcal{R},\mathcal{S})/\mathcal{R}$ is terminating, and $\mathcal{R}{\restriction_{\mathcal{S}}} \subseteq \to^*_{\mathcal{S}}$ then $\mathcal{R}$ and $\mathcal{S}$ are equi-confluent.*

Example 1. *Consider the left-linear TRS $\mathcal{R}$:*

$$1\colon\ \mathsf{s}(\mathsf{p}(x)) \to \mathsf{p}(\mathsf{s}(x)) \qquad 2\colon\ \mathsf{p}(\mathsf{s}(x)) \to x \qquad 3\colon\ \infty \to \mathsf{s}(\infty)$$

We show the confluence of $\mathcal{R}$ by using the rule removal criteria based on parallel critical pair system and rule labeling.

(i) *The TRS $\mathcal{R}$ admits two parallel critical peaks and the corresponding critical pairs join:*

$$\begin{tikzcd}
& \mathsf{s}(\mathsf{p}(\mathsf{s}(x))) \arrow[dl] \arrow[dr, "\epsilon"] & \\
\mathsf{s}(x) & & \mathsf{p}(\mathsf{s}(\mathsf{s}(x))) \arrow[ll, dashed]
\end{tikzcd}
\qquad
\begin{tikzcd}
& \mathsf{p}(\mathsf{s}(\mathsf{p}(x))) \arrow[dl] \arrow[dr, "\epsilon"] & \\
\mathsf{p}(\mathsf{p}(\mathsf{s}(x))) \arrow[rr, dashed] & & \mathsf{p}(x)
\end{tikzcd}$$

Let $\mathcal{S} = \{3\}$. The parallel critical pair system $\mathsf{PCPS}(\mathcal{R},\mathcal{S})$ consists of the four rules:

$$\begin{array}{ll}
\mathsf{s}(\mathsf{p}(\mathsf{s}(x))) \to \mathsf{s}(x) & \mathsf{p}(\mathsf{s}(\mathsf{p}(x))) \to \mathsf{p}(\mathsf{p}(\mathsf{s}(x))) \\
\mathsf{s}(\mathsf{p}(\mathsf{s}(x))) \to \mathsf{p}(\mathsf{s}(\mathsf{s}(x))) & \mathsf{p}(\mathsf{s}(\mathsf{p}(x))) \to \mathsf{p}(x)
\end{array}$$

By taking the linear polynomial interpretation $\mathcal{A}$ on $\mathbb{N}$ with

$$\mathsf{s}_{\mathcal{A}}(n) = 2n \qquad \mathsf{p}_{\mathcal{A}}(n) = n + 1 \qquad \infty_{\mathcal{A}} = 0$$

the inclusions $\mathsf{PCPS}(\mathcal{R},\mathcal{S}) \subseteq >_{\mathcal{A}}$ and $\mathcal{R} \subseteq \geqslant_{\mathcal{A}}$ hold. Thus, $\mathsf{PCPS}(\mathcal{R},\mathcal{S})/\mathcal{R}$ is terminating. As $\mathcal{F}\mathsf{un}(\mathcal{S}) = \{\mathsf{s}, \infty\}$ implies $\mathcal{R}{\restriction_{\mathcal{S}}} = \{3\} = \mathcal{S}$, we obtain $\mathcal{R}{\restriction_{\mathcal{S}}} \subseteq \to^*_{\mathcal{S}}$. Therefore, by Theorem 1 the TRSs $\mathcal{R}$ and $\mathcal{S}$ are equi-confluent.

(ii) *As $\mathcal{S}$ admits no parallel critical peaks, the rule removal criterion based on rule labeling [4] proves the equi-confluence of $\mathcal{S}$ and $\varnothing$.*

(iii) *The empty TRS $\varnothing$ is trivially confluent.*

Hence the original TRS $\mathcal{R}$ is confluent.

As a final remark, our tool employs the SMT solver Z3 [1] for automating the compositional confluence criteria and the reduction method.

References

[1] L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *Proceedings of 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 of *LNCS*, pages 337–340, 2008. The website of Z3 is: https://github.com/Z3Prover/z3. doi:10.1007/978-3-540-78800-3_24.

[2] N. Hirokawa, D. Kim, K. Shintani, and R. Thiemann. Certification of confluence- and commutation-proofs via parallel critical pairs. In *Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 147–161, 2024. doi:10.1145/3636501.3636949.

[3] N. Hirokawa and A. Middeldorp. Decreasing diagrams and relative termination. *Journal of Automated Reasoning*, 47:481–501, 2011. doi:10.1007/s10817-011-9238-x.

[4] N. Hirokawa and K. Shintani. Rule removal for confluence. In *Proceedings of 13th International Workshop on Confluence*, 2024. In this proceedings.

[5] V. van Oostrom. Confluence by decreasing diagrams, converted. In *Proceedings of 19th International Conference on Rewriting Techniques and Applications*, volume 5117 of *LNCS*, pages 306–320, 2008. doi:10.1007/978-3-540-70590-1_21.

[6] K. Shintani and N. Hirokawa. Compositional confluence criteria. *Logical Methods in Computer Science*, 20:6:1–6:28, 2024. doi:10.46298/lmcs-20(1:6)2024.

[7] R. Thiemann and C. Sternagel. Certification of termination proofs using CeTA. In *Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics*, volume 5674 of *LNCS*, pages 452–468, 2009.

[8] H. Zankl, B. Felgenhauer, and A. Middeldorp. Labelings for decreasing diagrams. *Journal of Automated Reasoning*, 54(2):101–133, 2015. doi:10.1007/s10817-014-9316-y.

Moca 0.3: A First-Order Theorem Prover for Horn Clauses

Yusuke Oi, Nao Hirokawa, and Teppei Saito

JAIST, Japan

Overview Moca is a fully automatic first-order theorem prover for Horn clauses. The tool, written in Haskell, is freely available from:

http://www.jaist.ac.jp/project/maxcomp/

The usage is: `moca.sh <file>`. Given a satisfiability problem in the TPTP CNF format [5], the tool outputs `Satisfiable` or `Unsatisfiable` if its satisfiability or unsatisfiability is proved, respectively, and `Maybe` otherwise. Given an infeasibility problem in the ARI format, the tool outputs `YES` if its infeasibility is proved, and `MAYBE` otherwise.

New feature This version of Moca can generate *certifiable* infeasibility proofs in the CPF 3 format. Using this feature, Moca forms a coalition with the certifier CeTA to give new certified proofs to a number of problems in the infeasibility category.

Techniques Moca implements *maximal ordered completion* [7] together with *approximation* techniques [4], the generalized *split-if* encoding [1, 4] (akin to unraveling [3, 2]), and inlining for conditional rewrite rules [6]. With a small example we illustrate how Moca uses them to solve problems. Consider the infeasibility problem of the conversion $x - x \leftrightarrow^* \mathsf{s}(x)$ for the TRS:

$$x - \mathsf{0} \to x \qquad \mathsf{0} - x \to \mathsf{0} \qquad \mathsf{s}(x) - \mathsf{s}(y) \to x - y$$

The problem can be regarded as the satisfiability problem of the Horn clauses:

$$x - \mathsf{0} \approx x \qquad \mathsf{0} - x \approx \mathsf{0} \qquad \mathsf{s}(x) - \mathsf{s}(y) \approx x - y \qquad x - x \not\approx \mathsf{s}(x)$$

By applying the split-if encoding the problem reduces to the word problem of deciding $\mathsf{T} \not\approx_{\mathcal{E}} \mathsf{F}$ for the equational system $\mathcal{E}$:

$$x - \mathsf{0} \approx x \qquad \mathsf{0} - x \approx \mathsf{0} \qquad \mathsf{s}(x) - \mathsf{s}(y) \approx x - y \qquad \mathsf{f}(\mathsf{s}(x), x) \approx \mathsf{F} \qquad \mathsf{f}(x - x, x) \approx \mathsf{T}$$

In order to solve it our tool attempts to construct a ground-complete presentation of $\mathcal{E}$ by using maximal ordered completion. However, the attempt is doomed to fail as the completion diverges. Moca overcomes the divergence by approximating the last equation to the more general equation $\mathsf{f}(x - x, y) \approx \mathsf{T}$. This results in the following equational system:

$$x - \mathsf{0} \approx x \qquad \mathsf{0} - x \approx \mathsf{0} \qquad \mathsf{s}(x) - \mathsf{s}(y) \approx x - y \qquad \mathsf{f}(\mathsf{s}(x), x) \approx \mathsf{F} \qquad \mathsf{f}(x - x, y) \approx \mathsf{T}$$

Now maximal ordered completion builds up the finite ground-complete presentation $\mathcal{R}$ of the approximated equational system:

$$\begin{array}{ccc} x - \mathsf{0} \to x & \mathsf{0} - x \to \mathsf{0} & \mathsf{s}(x) - \mathsf{s}(y) \to x - y \\ \mathsf{f}(\mathsf{0}, y) \to \mathsf{T} & \mathsf{f}(\mathsf{s}(x), x) \to \mathsf{F} & \mathsf{f}(x - x, y) \to \mathsf{T} \end{array}$$

Since $\mathsf{T}{\downarrow_{\mathcal{R}}} \neq \mathsf{F}{\downarrow_{\mathcal{R}}}$ holds, infeasibility of the conversion $x - x \leftrightarrow^* \mathsf{s}(x)$ is concluded.

References

[1] K. Claessen and N. Smallbone. Efficient Encodings of First-Order Horn Formulas in Equational Logic. In *Proc. 9th IJCAR*, LNCS 10900, pages 388–404, 2018.

[2] K. Gmeiner, N. Nishida and B. Gramlich. Proving Confluence of Conditional Term Rewriting Systems via Unravelings In *Proc. 2nd IWC*, pages 35–41, 2013.

[3] M. Marchiori. Unravelings and Ultra-Properties. In *Proc. 5th ALP*, LNCS 1139, pages 107–21, 1996.

[4] Y. Oi. Refutation by Completion and Approximations. Master's thesis, JAIST, 2019.

[5] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure: From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.

[6] C. Sternagel and T. Sternagel. Certifying Confluence of Quasi-Decreasing Strongly Deterministic Conditional Term Rewrite Systems. In *Proc. 26th CADE*, LNCS 10395, pp. 413–431, 2017.

[7] S. Winkler and G. Moser. MædMax: A Maximal Ordered Completion Tool. In *Proc. 9th IJCAR*, LNCS 10900, pages 472–480, 2018.

CO3 (Version 2.5)

Naoki Nishida and Misaki Kojima

Nagoya University, Nagoya, Japan
`nishida@i.nagoya-u.ac.jp` `k-misaki@nagoya-u.jp`

CO3, a **co**nverter for proving **co**nfluence of **co**nditional TRSs,[1] tries to prove confluence of conditional term rewrite systems (CTRSs, for short) by using a transformational approach (cf. [6]). The tool first transforms a given weakly-left-linear (WLL, for short) 3-DCTRS into an unconditional term rewrite system (TRS, for short) by using $\mathbb{U}_{conf}$ [2], a variant of the *unraveling* $\mathbb{U}$ [8], and then verifies confluence of the transformed TRS by using the following theorem: A 3-DCTRS $\mathcal{R}$ is confluent if $\mathcal{R}$ is WLL and $\mathbb{U}_{conf}(\mathcal{R})$ is confluent [1, 2]. The tool is very efficient because of very simple and lightweight functions to verify properties such as confluence and termination of TRSs.

Since version 2.0, a *narrowing-tree*-based approach [7, 3] to prove infeasibility of a condition w.r.t. a CTRS has been implemented [4]. The approach is applicable to *syntactically deterministic* CTRSs that are operationally terminating and *ultra-right-linear* w.r.t. the *optimized* unraveling. To prove infeasibility of a condition c, the tool first proves confluence, and then linearizes c if failed to prove confluence; then, the tool computes and simplifies a narrowing tree for c, and examines the emptiness of the narrowing tree. Since version 2.2, CO3 accepts both *join* and *semi-equational* CTRSs, and transforms them into equivalent DCTRSs to prove confluence or infeasibility [5].

This version has a new disproof criterion for confluence of CTRSs. A CTRS is trivially non-confluent if there exist a (possibly unconditional) critical pair $\langle s,t\rangle \Leftarrow c$ of $\mathcal{R}$ and a substitution θ such that θ satisfies c and $s\theta, t\theta$ are different constructor terms of $\mathcal{R}$. To find such a substitution θ, we implemented the following sufficient condition for $\langle s,t\rangle \Leftarrow s_1 \approx t_1, \ldots, s_k \approx t_k$: There exists some $i \in \{1,\ldots,k\}$ such that $s_i\theta \to^{=}_{\mathcal{R}'} t_i\theta$, $s_i\theta, t_i\theta$ are constructor terms of $\mathcal{R}$, and for all $j \in \{1,\ldots,i-1,i+1,\ldots,n\}$, $s_j\theta \to_{\varepsilon,\mathcal{R}'} t_j\theta$, where $\mathcal{R}' = \{\ell \to r \mid \ell \to r \Leftarrow c \in \mathcal{R},\ c = \epsilon\}$. This criterion works for, e.g., `251.ari` and `319.ari` in ARI-COPS[2] (`262.trs` and `330.trs`, resp., in COPS[3]).

Example 1. *Consider the 3-DCTRS* `251.trs`*:*

$$\mathcal{R}_{251} = \left\{ \begin{array}{r} \mathsf{plus}(0,y) \to y \\ \mathsf{plus}(\mathsf{s}(x),y) \to \mathsf{plus}(x,\mathsf{s}(y)) \\ \mathsf{f}(x,y) \to z \Leftarrow \mathsf{plus}(x,y) \twoheadrightarrow \mathsf{plus}(z,z') \end{array} \right\}$$

The (conditional) critical pairs of $\mathcal{R}_{251}$ *are*

$$\langle z,w\rangle \Leftarrow \mathsf{plus}(x,y) \twoheadrightarrow \mathsf{plus}(z,z'),\ \mathsf{plus}(x,y) \twoheadrightarrow \mathsf{plus}(w,w')$$

The first condition $\mathsf{plus}(x,y) \twoheadrightarrow \mathsf{plus}(z,z')$ *is satisfied by the substitution* $\theta_1 = \{z \mapsto x, z' \mapsto y\}$, *which does not instantiate the second condition* $\mathsf{plus}(x,y) \twoheadrightarrow \mathsf{plus}(w,w')$*. The second condition is satisfied by the second rule of* $\mathcal{R}_{251}$ *by means of the substitution* $\theta_2 = \{x \mapsto \mathsf{s}(x'), w \mapsto x', w' \mapsto y\}$*:* $\mathsf{plus}(x,y)\theta_2 \to_{\mathcal{R}_{251}} \mathsf{plus}(w,w')\theta_2$*. The composed substitution* $\theta = \{z \mapsto \mathsf{s}(x'), z' \mapsto y, x \mapsto \mathsf{s}(x'), w \mapsto x', w' \mapsto y\}$ *is a constructor substitution satisfying the conditional part of the critical pair. For the composed substitution* θ*,* $z\theta$ *and* $w\theta$ *are not joinable because they are different constructor terms. Therefore,* θ *is a witness disproving confluence of* $\mathcal{R}_{251}$*.*

[1] `http://www.trs.css.i.nagoya-u.ac.jp/co3/`
[2] `https://ari-cops.uibk.ac.at/ARI/`
[3] `https://ari-cops.uibk.ac.at/COPS/`

References

[1] K. Gmeiner, B. Gramlich, and F. Schernhammer. On soundness conditions for unraveling deterministic conditional rewrite systems. In A. Tiwari, editor, *Proceedings of the 23rd International Conference on Rewriting Techniques and Applications*, volume 15 of *Leibniz International Proceedings in Informatics*, pages 193–208, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012.

[2] K. Gmeiner, N. Nishida, and B. Gramlich. Proving confluence of conditional term rewriting systems via unravelings. In N. Hirokawa and V. van Oostrom, editors, *Proceedings of the 2nd International Workshop on Confluence*, pages 35–39, 2013.

[3] Y. Maeda, N. Nishida, M. Sakai, and T. Kobayashi. Extending narrowing trees to basic narrowing in term rewriting. IEICE Technical Report SS2018-39, Vol. 118, No. 385, pp. 73–78, the Institute of Electronics, Information and Communication Engineers, 2019 (in Japanese).

[4] N. Nishida. CO3 (Version 2.1). In M. Ayala-Rinćon and S. Mimram, editors, *Proceedings of the 9th International Workshop on Confluence*, page 67, 2020.

[5] N. Nishida. CO3 (Version 2.2). In S. Mimram and C. Rocha, editors, *Proceedings of the 10th International Workshop on Confluence*, page 61, 2021.

[6] N. Nishida, T. Kuroda, and K. Gmeiner. CO3 (Version 1.3). In B. Accattoli and A. Tiwari, editors, *Proceedings of the 5th International Workshop on Confluence*, page 74, 2016.

[7] N. Nishida and Y. Maeda. Narrowing trees for syntactically deterministic conditional term rewriting systems. In H. Kirchner, editor, *Proceedings of the 3rd International Conference on Formal Structures for Computation and Deduction*, volume 108 of *Leibniz International Proceedings in Informatics*, pages 26:1–26:20, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

[8] E. Ohlebusch. Termination of logic programs: Transformational methods revisited. *Applicable Algebra in Engineering, Communication and Computing*, 12(1/2):73–116, 2001.

CONFident at the 2024 Confluence Competition[*]

Miguel Vítores[2], Raúl Gutiérrez[1], and Salvador Lucas[2]

[1] Universidad Politécnica de Madrid, Madrid, Spain
r.gutierrez@upm.es
[2] VRAIN, Universitat Politècnica de València, Valencia, Spain
mvitvic@posgrado.upv.es
slucas@dsic.upv.es

1 Overview

CONFident 2.0 is a tool which is able to prove confluence of TRSs, CS-TRSs, CTRSs and CS-CTRSs. The tool is available here:

http://zenon.dsic.upv.es/confident/.

It is written in Haskell implementing the Confluence Framework:

- we consider two types of problems: *confluence problems* and *joinability problems*. Confluence problems encapsulate the different variants of rewrite systems. Joinability problems encapsulate any possible type of critical pair generated by rewrite systems.
- processors are partial functions that are applied to problems. Our processors encapsulate techniques for simplification, modular decomposition, problem transformation and direct confluence/joinability checks.

We implement these processors using the logical approach presented in [1, 3, 7] and mechanizing them by external tools like MU-TERM [3], infChecker [1], AGES [2], Prover9 and Mace4 [9] and Barcelogic[1].

Although CONFident 2.0 is the same as last year's, there were results submitted to conferences or journals before this publication and accepted afterwards. Latest description of the tool can be found now in [4]. Furthermore, a newly documented technique based on the notion of V-orthogonality is presented in [6]. V-orthogonal Generalized Term Rewriting Systems are confluent. For V-orthogonality, the usual left-linearity requirement for rules is relaxed. However, besides the absence of proper conditional critical pairs, the infeasibility of the conditional variable pairs introduced by conditional rules is also required.

References

[1] R. Gutiérrez and S. Lucas. Automatically Proving and Disproving Feasibility Conditions. In N. Peltier and V. Sofronie-Stokkermans, editor, *Proc. of IJCAR'2020*, LNCS 12167:416–435. Springer, 2020.

[2] R. Gutiérrez and S. Lucas. Automatic Generation of Logical Models with AGES. In *CADE 2019: Automated Deduction - CADE 27*, LNCS 11716:287:299. Springer, 2019.

[*]Partially supported by grants PID2021-122830OB-C42 and PID2021-122830OB-C44 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe" and by the grant CIPROM/2022/6 funded by Generalitat Valenciana.

[1]https://barcelogic.com/

[3] R. Gutiérrez and S. Lucas. MU-TERM: Verify Termination Properties Automatically (System Description). In N. Peltier and V. Sofronie-Stokkermans, editor, *Proc. of IJCAR'2020*, LNCS 12167:436–447. Springer, 2020.

[4] R. Gutiérrez. and S. Lucas Proving Confluence in the Confluence Framework with CONFident. *Fundamenta Informaticae*, 193, to appear 2024.

[5] R. Gutiérrez, M. Vítores and S. Lucas. Confluence Framework: Proving Confluence with CONFident. In A. Villanueva, editor, *Proc. of LOPSTR'2022*, LNCS 13474:24–43. Springer, 2022.

[6] S. Lucas. Orthogonality of Generalized Term Rewriting Systems. *Proc. of IWC'2024*, to appear, 2024.

[7] S. Lucas. Proving semantic properties as first-order satisfiability. *Artificial Intelligence* 277, paper 103174, 24 pages, 2019.

[8] S. Lucas and R. Gutiérrez. Use of Logical Models for Proving Infeasibility in Term Rewriting. *Information Processing Letters*, 136:90–95, 2018.

[9] W. McCune. Prover9 and Mace4. [online]. Available at https://www.cs.unm.edu/~mccune/mace4/.

infChecker at the 2024 Confluence Competition[*]

Raúl Gutiérrez[1], Salvador Lucas[2], and Miguel Vítores[2]

[1] Universidad Politécnica de Madrid, Madrid, Spain
r.gutierrez@upm.es
[2] VRAIN, Universitat Politècnica de València, Valencia, Spain
slucas@dsic.upv.es
mvitvic@posgrado.upv.es

1 Overview

infChecker is a tool for checking *(in)feasibility* of sequences of rewrite and relations with respect to *first-order theories*, called goals [3]. infChecker participates in the INF category at the Confluence Competition but it is also used as as a external tool in CONFident, which participates in several categories in the Competition. In 2024, we participate with the same version as in 2023.

The tool is available here:

http://zenon.dsic.upv.es/infChecker/.

It is written in Haskell implementing the Feasibility Framework:

- we consider *f-problems* that are form by a theory and and goal. In the competition, goals only contain reachability conditions.
- processors are partial functions that are applied to problems. Our processors encapsulate techniques for simplification, splitting, satisfiability and provability.

Some processors are mechanized using external tools like AGES [2], Prover9 and Mace4 [4]. Latest description of the tool can be found in [1].

References

[1] R. Gutiérrez and S. Lucas. Automatically Proving and Disproving Feasibility Conditions. In N. Peltier and V. Sofronie-Stokkermans, editor, *Proc. of IJCAR'2020*, LNCS 12167:416–435. Springer, 2020.

[2] R. Gutiérrez and S. Lucas. Automatic Generation of Logical Models with AGES. In *CADE 2019: Automated Deduction - CADE 27*, LNCS 11716:287:299. Springer, 2019.

[3] S. Lucas and R. Gutiérrez. Use of Logical Models for Proving Infeasibility in Term Rewriting. *Information Processing Letters*, 136:90–95, 2018.

[4] W. McCune. Prover9 and Mace4. [online]. Available at https://www.cs.unm.edu/~mccune/mace4/.

[*]Partially supported by grants PID2021-122830OB-C42 and PID2021-122830OB-C44 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe" and by the grant CIPROM/2022/6 funded by Generalitat Valenciana.

AGCP: System Description for CoCo 2024

Takahito Aoto

Institute of Science and Technology, Niigata University
aoto@ie.niigata-u.ac.jp

AGCP (Automated Groud Confluence Prover) [1] is a tool for proving ground confluence of many-sorted term rewriting systems. In this year, no new ground (non-)confluence criterion has been incorporated from the one submitted for CoCo 2023. Below we provide a brief overview of AGCP.

AGCP is written in Standard ML of New Jersey (SML/NJ). AGCP proves ground confluence of many-sorted term rewriting systems based on two ingredients. One ingredient is to divide the ground confluence problem of a many-sorted term rewriting system $\mathcal{R}$ into that of $\mathcal{S} \subseteq \mathcal{R}$ and the inductive validity problem of equations $u \approx v$ w.r.t. $\mathcal{S}$ for each $u \to r \in \mathcal{R} \setminus \mathcal{S}$. Here, an equation $u \approx v$ is inductively valid w.r.t. $\mathcal{S}$ if all its ground instances $u\sigma \approx v\sigma$ is valid w.r.t. $\mathcal{S}$, i.e. $u\sigma \overset{*}{\leftrightarrow}_{\mathcal{S}} v\sigma$. Another ingredient is to prove ground confluence of a many-sorted term rewriting system via the *bounded ground convertibility* of the critical pairs. Here, an equation $u \approx v$ is said to be bounded ground convertibile w.r.t. a quasi-order $\succsim$ if $u\theta_g \xleftrightarrow[\succsim]{*}_{\mathcal{R}} v\theta_g$ for any its ground instance $u\sigma_g \approx v\sigma_g$, where $x \xleftrightarrow[\succsim]{*} y$ iff there exists $x = x_0 \leftrightarrow \cdots \leftrightarrow x_n = y$ such that $x \succsim x_i$ or $y \succsim x_i$ for every x_i.

Rewriting induction [3] is a well-known method for proving inductive validity of many-sorted term rewriting systems. In [1], an extension of rewriting induction to prove bounded ground convertibility of the equations has been reported. Namely, for a reduction quasi-order $\succsim$ and a quasi-reducible many-sorted term rewriting system $\mathcal{R}$ such that $\mathcal{R} \subseteq \succ$, the extension proves bounded ground convertibility of the input equations w.r.t. $\succsim$. The extension not only allows to deal with non-orientable equations but also with many-sorted TRSs having non-free constructors. Several methods that add wider flexibility to the this approach are given in [2]: when suitable rules are not presented in the input system, additional rewrite rules are constructed that supplement or replace existing rules in order to obtain a set of rules that is adequate for applying rewriting induction; and an extension of the system of [2] is used if if the input system contains non-orientable constructor rules. AGCP uses these extension of the rewriting induction to prove not only inductive validity of equations but also the bounded ground convertibility of the critical pairs. Finally, some methods to deal with disproving ground confluence are added as reported in [2].

References

[1] T. Aoto and Y. Toyama. Ground confluence prover based on rewriting induction. In *Proc. of 1st FSCD*, volume 52 of *LIPIcs*, pages 33:1–33:12. Schloss Dagstuhl, 2016.

[2] T. Aoto, Y. Toyama and Y. Kimura. Improving Rewriting Induction Approach for Proving Ground Confluence. In *Proc. of 2nd FSCD*, volume 84 of *LIPIcs*, pages 7:1–7:18. Schloss Dagstuhl, 2017.

[3] U.S. Reddy. Term rewriting induction. In *Proc. of CADE-10*, volume 449 of *LNAI*, pages 162–177. Springer-Verlag, 1990.

ACP: System Description for CoCo 2024

Takahito Aoto

Institute of Science and Technology, Niigata University
`aoto@ie.niigata-u.ac.jp`

ACP (Automated Confluence Prover) is a tool for proving confluence and some related properties of (conditional) term rewriting systems. In this year, we have revised methods for (dis)proving the UNR property employed in our prover, whose deails are described in [4]. Also, some revision has been done for generating some proofs in CPF format. Below we provide a brief overview of ACP.

A primary functionality of ACP is proving confluence (CR) of term rewriting systems (TRSs). ACP integrates multiple direct criteria for guaranteeing confluence of TRSs. It also incorporates divide–and–conquer criteria by which confluence or non-confluence of TRSs can be inferred from those of their components. Several methods for disproving confluence are also employed. For some criteria, it supports generation of proofs in CPF format that can be certified by certifiers. The internal structure of the prover is kept simple and is mostly inherited from the version 0.11a, which has been described in [3]. It also deal with confluence of oriented conditional term rewriting systems. Besides confluence, ACP supports proving the UNC property (unique normal form property w.r.t. conversion) and the commutation property of term rewriting systems. The ingredients of the former property have been appeared in [2, 5]. Our (dis)proofs of commutation are based on a development closed criterion [6] and a simple search for counter examples.

ACP is written in Standard ML of New Jersey (SML/NJ) and the source code is also available from [1]. It uses a SAT prover such as MiniSAT and an SMT prover YICES as external provers. It internally contains an automated (relative) termination prover for TRSs but external (relative) termination provers can be substituted optionally. Users can specify criteria to be used so that each criterion or any combination of them can be tested. Several levels of verbosity are available for the output so that users can investigate details of the employed approximations for each criterion or can get only the final result of prover's attempt.

References

[1] ACP (Automated Confluence Prover). `http://www.nue.ie.niigata-u.ac.jp/tools/acp/`.

[2] T. Aoto and Y. Toyama. Automated proofs of unique normal forms w.r.t. conversion for term rewriting systems. In *Proc. of 12th FroCoS*, volume 11715 of *LNAI*, pages 330–347. Springer-Verlag, 2019.

[3] T. Aoto, J. Yoshida, and Y. Toyama. Proving confluence of term rewriting system automatically. In *Proc. of 20th RTA*, volume 5595 of *LNCS*, pages 93–102. Springer-Verlag, 2009.

[4] T. Aoto. Proving uniqueness of normal forms w.r.t. reduction of term rewriting systems. submitted, 2024.

[5] M. Yamaguchi and T. Aoto, A fast decision procedure for uniqueness of normal forms w.r.t. conversion of shallow term rewriting systems. In *Proc. of 5th FSCD*, volume 167 of *LIPIcs*, pages 9:1–9:23. Schloss Dagstuhl, 2020.

[6] J. Yoshida, T. Aoto, and Y. Toyama. Automating confluence check of term rewriting systems. *Computer Software*, 26(2):76–92, 2009.

CoCo 2024 Participant: CeTA 3.1

Dohan Kim[1], Christina Kirk[1], Teppei Saito[2], and René Thiemann[1]

[1] University of Innsbruck, Austria
[2] Japan Advanced Institute of Science and Technology, Japan

The tool CeTA [1] is a certifier for, among other properties, (non-)confluence of term rewrite systems with and without conditions. Its soundness is proven as part of the formal proof library IsaFoR, the Isabelle Formalization of Rewriting. Below, we present the relevant changes from last year's version (2.45) to this year's version (3.1). For a complete reference of supported techniques we refer to the certification problem format (CPF) and the IsaFoR/CeTA website:

http://cl-informatik.uibk.ac.at/isafor/

In CeTA 3.1 several new term orderings have been added. These can be used in non-joinability proofs as discrimination pairs, or in infeasibility proofs as co-rewrite pairs. This includes CoWPO and WPO, also in combination with non-strongly normalizing orders such as polynomial interpretations with negative coefficients [2]. Moreover, the monotonic semantic path order and the generalized WPO have become available [3].

For non-joinability proofs [4] for terms s and t we further support the technique of showing that both s and t only reach a finite set of terms, and that these sets are disjoint.

One of the big differences is the switch to CPF 3, which has the following benefits:

- Joining sequences for various confluence and commutation criteria now have a uniform format. One of the possibilities is to provide just the intermediate terms, without having to explicitly specify the positions or the applied rules of each rewrite step. Moreover, whenever the join allows arbitrary many steps $s \to^* t$, then the intermediate terms can be connected by parallel rewrite steps.

- Rules and terms can be indexed, a technique to reduce the size of certificates. For instance in a modular decomposition it suffices to state that the system is split into rules (A), (B), (E) and rules (C), (D), without having to fully spell out these five rules again.

- Several redundant CPF 2 elements have been removed. For instance, the longish

  ```
  <polynomial><sum>
    <polynomial><variable>1</variable></polynomial>
    <polynomial><coefficient><integer>2</integer></coefficient></polynomial>
  </sum></polynomial>
  ```

  in CPF 2 is just `<sum><variable>1</variable><integer>2</integer></sum>` in CPF 3 in order to specify the polynomial $x_1 + 2$.

References

[1] René Thiemann and Christian Sternagel. Certification of Termination Proofs using CeTA. In *Proc. 22nd International Conference on Theorem Proving in Higher Order Logics*, volume 5674 of *Lecture Notes in Computer Science*, pages 452–468, 2009. doi: 10.1007/978-3-642-03359-9_31.

[2] Akihisa Yamada. Term Orderings for Non-reachability of (Conditional) Rewriting. In *Proc. 11th International Joint Conference, IJCAR 2022*, volume 13385 of *Lecture Notes in Computer Science*, pages 248–267, 2022. doi: 10.1007/978-3-031-10769-6_15.

[3] Teppei Saito and Nao Hirokawa. Weighted Path Orders Are Semantic Path Orders. In *Proc. 14th International Symposium on Frontiers of Combining Systems, Frocos 2023*, volume 14279 of *Lecture Notes in Computer Science*, pages 63–80, 2023. doi: 10.1007/978-3-031-43369-6_4.

[4] Takahito Aoto. Disproving Confluence of Term Rewriting Systems by Interpretation and Ordering. In *Proc. 9th International Symposium on Frontiers of Combining Systems, Frocos 2013*, volume 8152 of *Lecture Notes in Computer Science*, pages 311–326, 2013. doi: 10.1007/978-3-642-40885-4_22.

CoCo 2024 Participant: CSI 1.2.7

Fabian Mitterwallner and Aart Middeldorp

Department of Computer Science, University of Innsbruck, Austria
fabian.mitterwallner@uibk.ac.at, aart.middeldorp@uibk.ac.at

CSI is an automatic tool for (dis)proving confluence and related properties of first-order term rewrite systems (TRSs). It has been in development since 2010. Its name is derived from the Confluence of the rivers Sill and Inn in Innsbruck. The tool is available from

http://cl-informatik.uibk.ac.at/software/csi

under a LGPLv3 license. A detailed description of CSI can be found in [3]. Some of the implemented techniques are described in [1,2,4]. CSI can also produce certificates for confluence results, which are checked by CeTA.

CSI participates in the following CoCo 2024 categories: NFP, SRS, TRS, UNC and UNR. Additionally, it participates together with CeTA in the SRS and TRS categories, providing certified YES/NO answers. In 2023 CSI won the NFP, TRS and SRS categories and came second in the UNC and UNR categories.

CSI uses the conversion tool[1] to transform ARI problems into COPS problems.

References

[1] Bertram Felgenhauer. Confluence for Term Rewriting: Theory and Automation. PhD thesis, University of Innsbruck, 2015.

[2] Julian Nagele. Mechanizing Confluence: Automated and Certified Analysis of First- and Higher-Order Rewrite Systems. PhD thesis, University of Innsbruck, 2017.

[3] Julian Nagele, Bertram Felgenhauer, and Aart Middeldorp. CSI: New Evidence – A Progress Report. In *Proc. 26th International Conference on Automated Deduction*, volume 10395 of *Lecture Notes in Artificial Intelligence*, pages 385–397, 2017. doi: 10.1007/978-3-319-63046-5_24.

[4] Harald Zankl. Challenges in Automation of Rewriting. Habilitation thesis, University of Innsbruck, 2014.

[1] https://project-coco.uibk.ac.at/ARI/#conversion

CoCo 2024 Participant: FORT-h 2.1

Fabian Mitterwallner and Aart Middeldorp

Department of Computer Science, University of Innsbruck, Austria
fabian.mitterwallner@uibk.ac.at, aart.middeldorp@uibk.ac.at

The first-order theory of rewriting is a decidable theory for finite left-linear right-ground rewrite systems. The decision procedure goes back to Dauchet and Tison [1]. FORT-h is a reimplementation of the tool FORT [4], but is based on a new variant of the decision procedure, described in [2], for the larger class of linear variable-separated rewrite systems. This variant supports a more expressive theory and is based on anchored ground tree transducers. More importantly, it can produce certificates for the YES/NO answers. These certificates can then be verified by FORTify, an independent Haskell program that is code-generated from the formalization of the decision procedure in the proof assistant Isabelle/HOL.

A command-line version of FORT-h can be downloaded from

http://fortissimo.uibk.ac.at/fort(ify)/

FORT-h participates in the following CoCo 2024 categories: COM, GCR, NFP, TRS, UNC, and UNR. In all six categories it additionally participates together with FORTify [3] to produce certified YES/NO answers. In 2023 FORT-h had the most YES answers in the UNR category. Moreover, it won the RELIABILITY[1] category.

FORT-h 2.1 accepts input problems in ARI[2] format.

References

[1] Max Dauchet and Sophie Tison. The Theory of Ground Rewrite Systems is Decidable. In *Proc. 5th IEEE Symposium on Logic in Computer Science*, pages 242–248, 1990. doi: 10.1109/LICS.1990.113750.

[2] Aart Middeldorp, Alexander Lochmann, and Fabian Mitterwallner. First-Order Theory of Rewriting for Linear Variable-Separated Rewrite Systems: Automation, Formalization, Certification. *Journal of Automated Reasoning*, 67(14):1–76, 2023. doi: 10.1007/s10817-023-09661-7.

[3] Fabian Mitterwallner and Aart Middeldorp. CoCo 2024 Participant: FORTify 2.0. In *Proc. 13th International Workshop on Confluence*, 2024. This volume.

[4] Franziska Rapp and Aart Middeldorp. FORT 2.0. In *Proc. 9th International Joint Conference on Automated Reasoning*, volume 10900 of *LNCS (LNAI)*, pages 81–88, 2018. doi: 10.1007/978-3-319-94205-6_6.

[1] https://project-coco.uibk.ac.at/2023/results.php#certification
[2] https://project-coco.uibk.ac.at/ARI/

CoCo 2024 Participant: FORTify 2.1

Fabian Mitterwallner and Aart Middeldorp

Department of Computer Science, University of Innsbruck, Austria
`fabian.mitterwallner@uibk.ac.at, aart.middeldorp@uibk.ac.at`

The first-order theory of rewriting is a decidable theory for linear variable-separated rewrite systems. The decision procedure goes back to Dauchet and Tison [1]. In this theory confluence-related properties on ground terms are easily expressible. An extension of the theory to multiple rewrite systems, as well as the decision procedure, has been formalized in Isabelle/HOL [2–4]. The code generation facilities of Isabelle then give rise to the certifier FORTify which checks certificate constructed by FORT-h [6]. FORTify takes as input an answer (YES/NO), a formula, a list of TRSs, and a certificate proving that the formula holds (does not hold) for the given TRSs. It then checks the integrity and validity of the certificate. A command-line version of the tool can be downloaded from

`https://fortissimo.uibk.ac.at/fort(ify)/`

We refer to the recent article [5] for a detailed description of FORTify.

This year FORTify participates, together with FORT-h, in the following CoCo 2024 categories: COM, GCR, NFP, TRS, UNC, and UNR.

References

[1] Max Dauchet and Sophie Tison. The Theory of Ground Rewrite Systems is Decidable. In *Proc. 5th IEEE Symposium on Logic in Computer Science*, pages 242–248, 1990. doi: 10.1109/LICS.1990.113750.

[2] Alexander Lochmann. Reducing Rewrite Properties to Properties on Ground Terms, 2022. `https://isa-afp.org/entries/Rewrite_Properties_Reduction.html`, Formal proof development.

[3] Alexander Lochmann and Bertram Felgenhauer. First-Order Theory of Rewriting. *Archive of Formal Proofs*, 2022. `https://isa-afp.org/entries/FO_Theory_Rewriting.html`, Formal proof development.

[4] Alexander Lochmann, Bertram Felgenhauer, Christian Sternagel, René Thiemann, and Thomas Sternagel. Regular Tree Relations. *Archive of Formal Proofs*, 2021. `https://www.isa-afp.org/entries/Regular_Tree_Relations.html`, Formal proof development.

[5] Aart Middeldorp, Alexander Lochmann, and Fabian Mitterwallner. First-Order Theory of Rewriting for Linear Variable-Separated Rewrite Systems: Automation, Formalization, Certification. *Journal of Automated Reasoning*, 67(14):1–76, 2023. doi: 10.1007/s10817-023-09661-7.

[6] Fabian Mitterwallner and Aart Middeldorp. CoCo 2024 Participant: FORT-h 2.1. In *Proc. 13th International Workshop on Confluence*, 2024. This volume.

CRaris: CR checker for LCTRSs in ARI Style

Naoki Nishida and Misaki Kojima

Nagoya University, Nagoya, Japan
nishida@i.nagoya-u.ac.jp k-misaki@nagoya-u.jp

CRaris, a **CR** checker for LCTRSs in **ARI** style,[1] is a tool to prove confluence of *logically constrained term rewrite systems* (LCTRSs, for short) [4] written in ARI format [1].[2] The tool is based on Crisys2, **c**onstrained **r**ewriting **i**nduction **sys**tem (version **2**),[3] and receives LCTRSs written in ARI format only to prove confluence, while crisys2 has many functions to e.g., solve *all-path reachability problems* [2]. To prove confluence of LCTRSs, the tool uses the following criteria:

- weak orthogonality [4], and
- termination and joinability of critical pairs [6].

To prove termination, the tool uses the DP framework for LCTRSs [3] without any interpretation method, together with a criterion for LCTRSs with bitvector arithmetics [5].

The *critical pairs* of two constrained rewrite rules $\rho_1 : \ell_1 \to r_1\ [\varphi_1]$ and $\rho_2 : \ell_2 \to r_2\ [\varphi_2]$ with distinct variables (i.e., $\mathcal{V}ar(\ell_1, r_1, \varphi_1) \cap \mathcal{V}ar(\ell_2, r_2, \varphi_2) = \emptyset$) are all tuples $\langle s, t, \phi \rangle$ such that a non-variable subterm $\ell_1|_p$ of ℓ_1 at a position p is unifiable with ℓ_2, "$p \neq \varepsilon$, $\rho_1 \neq \rho_2$ up to variable renaming, or $\mathcal{V}ar(r_1) \subseteq \mathcal{V}ar(\ell_1)$", the most general unifier γ of $\ell_1|_p$ and ℓ_2 respects variables of both ρ_1 and ρ_2, i.e., $\gamma(x)$ is either a value or a variable for all variables x in $\mathcal{V}ar(\varphi_1, \varphi_2) \cup (\mathcal{V}ar(r_1, r_2) \setminus \mathcal{V}ar(\ell_1, \ell_2))$, $(\varphi_1 \wedge \varphi_2)\gamma$ is satisfiable, $s = r_1\gamma$, $t = (\ell_1[r_2]_p)\gamma$, and $\phi = (\varphi_1 \wedge \varphi_2)\gamma$. The set of critical pairs of an LCTRS $\mathcal{R}$ is denoted by $CP(\mathcal{R})$, which includes all critical pairs of two rules in $\mathcal{R} \cup \mathcal{R}_{calc}$. A critical pair $\langle s, t, \phi \rangle$ is called *trivial* if $s\ [\phi] \sim t\ [\phi]$. An LCTRS $\mathcal{R}$ is called *weakly orthogonal* if $\mathcal{R}$ is left-linear and all critical pairs of $\mathcal{R}$ are trivial.

Theorem 1 ([4])**.** *A weakly orthogonal LCTRS is confluent.*

A critical pair $\langle s, t, \phi \rangle$ is called *joinable* if $(\langle s, t \rangle\ [\phi]) \to^*_{\mathcal{R}} (\langle s', t' \rangle\ [\phi'])$ and $s'\ [\phi'] \sim t'\ [\phi']$.

Theorem 2 ([6])**.** *A terminating LCTRS is confluent if all its critical pairs are joinable.*

Instead of joinability of critical pairs, the tool uses a much simpler sufficient condition.

Proposition 3. *A critical pair $\langle s, s, \phi \rangle$ is trivial and thus joinable.*

We use "$s = t$" as a sufficient condition for triviality of $\langle s, s, \phi \rangle$.

References

[1] T. Aoto, N. Hirokawa, D. Kim, M. Kojima, A. Middeldorp, F. Mitterwallner, N. Nishida, T. Saito, J. Schöpf, K. Shintani, R. Thiemann, and A. Yamada. A new format for rewrite systems. In C. Chenavier and S. Winkler, editors, *Proceedings of the 12th International Workshop on Confluence*, pages 32–37, 2023.

[1] http://www.trs.css.i.nagoya-u.ac.jp/craris/
[2] https://project-coco.uibk.ac.at/ARI/lctrs.php
[3] https://www.trs.cm.is.nagoya-u.ac.jp/crisys/

[2] M. Kojima and N. Nishida. Reducing non-occurrence of specified runtime errors to all-path reachability problems of constrained rewriting. *Journal of Logical and Algebraic Methods in Programming*, 135:1–19, 2023.

[3] C. Kop. Termination of LCTRSs. In *Proceedings of the 13th International Workshop on Termination*, pages 1–5, 2013.

[4] C. Kop and N. Nishida. Term rewriting with logical constraints. In P. Fontaine, C. Ringeissen, and R. A. Schmidt, editors, *Proceedings of the 9th International Symposium on Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Artificial Intelligence*, pages 343–358, Springer, 2013.

[5] A. Matsumi, N. Nishida, M. Kojima, and D. Shin. On singleton self-loop removal for termination of LCTRSs with bit-vector arithmetic. In A. Yamada, editor, *Proceedings of the 19th International Workshop on Termination*, pages 1–6, 2023.

[6] J. Schöpf and A. Middeldorp. Confluence criteria for logically constrained rewrite systems. In B. Pientka and C. Tinelli, editors, *Proceedings of the 29th International Conference on Automated Deduction*, volume 14132 of *Lecture Notes in Computer Science*, pages 474–490, Springer, 2023.