Proceedings of the

12th International Workshop on Confluence

August 23-24, 2023

Obergurgl, Austria

Foreword

The 12th International Workshop on Confluence (IWC 2023) is held on August 23 and 24, 2023, in Obergurgl, Austria, co-located with the 19th International Workshop on Termination (WST 2023).

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 2023 continues this tradition. The present report comprises ten regular submissions and the abstracts of two invited talks, as well as descriptions of tools participating in the 12th Confluence Competition (CoCo 2023).

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 2023 provides a general overview as well as system descriptions of all competition entrants.

IWC 2023 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, and the organizers of CoCo, as well as the local organizers. Their hard work is very much appreciated.

Cyrille Chenavier and Sarah Winkler — Limoges and Bolzano, 28 July 2023

Organization

IWC Steering Committee

- Takahito Aoto, Niigata University, Japan
- Mauricio Ayala-Rincón, Brasilia University, Brasil

IWC Program Committee

- Patrick Bahr, IT University of Copenhagen, Denmark
- Cyrille Chenavier, Université de Limoges, France (co-chair)
- Benjamin Dupont, Université Grenoble Alpes, France
- Jörg Endrullis, Vrije Universiteit Amsterdam, Netherlands
- Claudia Faggian, Université de Paris, France
- Carsten Fuhs, Birkbeck University of London, Great Britain
- Raúl Gutiérrez, Universidad Politécnica de Madrid, Spain
- Dohan Kim, University of Innsbruck, Austria
- Sarah Winkler, Free University of Bozen-Bolzano, Italy (co-chair)

Additional Reviewers

- Aart Middeldorp, 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
- Kiraku Shintani, JAIST, Japan

Contents

Foreword ii

Organization iii

Abstracts of Invited Talks 1

Unravelings and Narrowing Trees Towards Confluence of Deterministic CTRSs
Naoki Nishida . . . 1

History and Future of the CeTA-Certifier for CoCo - Including a New Decision Procedure for Pattern Completeness
René Thiemann . . . 2

Workshop Contributions 3

Reducing Confluence of LCTRSs to Confluence of TRSs
Fabian Mitterwallner, Jonas Schöpf and Aart Middeldorp . . . 3

On Confluence Criteria for Non-terminating Abstract Rewriting Systems
Ievgen Ivanov . . . 9

Church–Rosser Modulo for Left-Linear TRSs Revisited
Johannes Niederhauser, Nao Hirokawa and Aart Middeldorp . . . 14

Residuation = Skolemised Confluence
Vincent van Oostrom . . . 20

Confluence of a Computational Lambda Calculus for Higher-Order Relational Queries
Claudio Sacerdoti Coen and Riccardo Treglia . . . 26

A New Format for Rewrite Systems
Takahito Aoto, Nao Hirokawa, Dohan Kim, Misaki Kojima, Aart Middeldorp, Fabian Mitterwallner, Naoki Nishida, Teppei Saito, Jonas Schöpf, Kiraku Shintani, René Thiemann and Akihisa Yamada . . . 32

The Z-property for left-linear term rewriting via convective context-sensitive completeness
Vincent van Oostrom . . . 38

Ground Canonical Rewrite Systems Revisited
Aart Middeldorp, Masahiko Sakai and Sarah Winkler . . . 44

Formalizing Confluence and Commutation Criteria Using Proof Terms
Christina Kohl and Aart Middeldorp . . . 49

A verified algorithm for deciding pattern completeness and related properties
René Thiemann . . . 55

Confluence Competition 61

Confluence Competition 2023
Raúl Gutiérrez, Aart Middeldorp, Naoki Nishida, Kiraku Shintani . . . 61

ConfCSR
Filip Stevanovic and Fabian Mitterwallner . . . 63

Toma 0.5: An Equational Theorem Prover
Teppei Saito and Nao Hirokawa . . . 64

Hakusan 0.8: A Confluence Tool
Kiraku Shintani and Nao Hirokawa . . . 65

CoLL 1.6.1: A Commutation Tool
Kiraku Shintani . . . 66

CO3 (Version 2.4)
Naoki Nishida, Misaki Kojima, and Ayuka Matsumi . . . 67

CSI 1.2.7
Fabian Mitterwallner and Aart Middeldorp . . . 68

FORT-h 2.0
Fabian Mitterwallner, Aart Middeldorp . . . 69

FORTify 2.0
Alexander Lochmann, Fabian Mitterwallner, Aart Middeldorp . . . 70

CONFident
Miguel Vítores, Raúl Gutiérrez, and Salvador Lucas . . . 71

infChecker
Raúl Gutiérrez, Salvador Lucas, Miguel Vítores 72
AGCP
Takahito Aoto 73
ACP
Takahito Aoto 74
nonreach 1.2
Florian Meßner 75
CeTA 2.45
René Thiemann, Christina Kohl, and Aart Middeldorp 76

Unravelings and Narrowing Trees Towards Confluence of Deterministic CTRSs

Naoki Nishida

Nagoya University, Japan
naoki.nishida@nagoya-u.jp

In this talk, we present two main techniques developed in CO3, a COnverter for proving COnfluence of COnditional term rewrite systems, for proving confluence and infeasibility problems of deterministic conditional term rewrite systems (DCTRSs, for short). One is unraveling transformations: An unraveling is a reduction-preserving transformation of a join or oriented CTRS into a term rewrite system (TRS, for short); for a syntactically deterministic CTRS, if the unraveled TRS of the CTRS is weakly left-linear and confluent, then the CTRS is confluent. The other is narrowing trees: A narrowing tree of finitely many oriented conditions represent the set of substitutions satisfying all the conditions w.r.t. constructor-based rewriting; for a conditional critical pair of a DCTRS, if the unraveled TRS of the DCTRS is right-linear and a narrowing tree of the conditional part represents the empty set, then the conditional critical pair is infeasible.

History and Future of the CeTA-Certifier for CoCo - Including a New Decision Procedure for Pattern Completeness

René Thiemann

University of Innsbruck, Austria
rene.thiemann@uibk.ac.at

This talk is split into two parts. In the first part we will provide some history of CeTA, the certifier that is used to validate the generated proofs of the annual confluence competition CoCo. In the second part we present some future plans of CeTA. Here, we present some initial steps that have been done to support ground confluence proofs. For instance, we developed a new and simple algorithm to decide pattern completeness for first-order TRSs. This property implies quasi-reducibility, a prerequisite if one wants to prove ground confluence by rewriting induction.

Reducing Confluence of LCTRSs to Confluence of TRSs*

Fabian Mitterwallner, Jonas Schöpf, and Aart Middeldorp

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

Abstract

We present a transformation from logically constrained term rewrite systems (LCTRSs) to plain term rewrite systems (TRSs) such that critical pairs of the latter correspond to constrained critical pairs of the former. This allows us to transfer confluence results for TRSs based on critical pair conditions to LCTRSs.

1 Introduction

Numerous techniques exist to (dis)prove confluence of plain TRSs. For LCTRSs much less is known. Kop and Nishida [1] established (weak) orthogonality as sufficient confluence criteria for LCTRSs. Joinability of critical pairs for terminating systems is implicit in [4]. Very recently, strong closedness for linear LCTRSs and (almost) parallel closedness for left-linear LCTRSs were established [2]. The proofs of these results were obtained by *replaying* existing proofs for TRSs in a constrained setting, involving a non-trivial effort. For more advanced confluence criteria, this is not feasible.

In this paper we present a simple transformation from LCTRSs to TRSs which allows us to relate results for the latter to the former. This transformation is presented in the next section and used in Section 3 to prove that (almost) development closed left-linear LCTRSs are confluent by *reusing* the corresponding result for TRSs obtained by van Oostrom [3].

We assume familiarity with the basic notions of term rewriting. In the remainder of this introductory section we recall a few key notions for LCTRSs. For more background information we refer to [1, 2, 4]. We assume a many-sorted signature $\mathcal{F} = \mathcal{F}_{\mathsf{te}} \cup \mathcal{F}_{\mathsf{th}}$. For every sort ι in $\mathcal{F}_{\mathsf{th}}$ we have a non-empty set $\mathcal{V}\mathsf{al}_\iota \subseteq \mathcal{F}_{\mathsf{th}}$ of value symbols, such that all $c \in \mathcal{V}\mathsf{al}_\iota$ are constants of sort ι. We demand $\mathcal{F}_{\mathsf{te}} \cap \mathcal{F}_{\mathsf{th}} \subseteq \mathcal{V}\mathsf{al}$ where $\mathcal{V}\mathsf{al} = \bigcup_\iota \mathcal{V}\mathsf{al}_\iota$. In the case of integers this results in an infinite signature with $\mathbb{Z} \subseteq \mathcal{V}\mathsf{al} \subseteq \mathcal{F}_{\mathsf{th}}$. A term in $\mathcal{T}(\mathcal{F}_{\mathsf{th}}, \mathcal{V})$ is called a *logical* term. Ground logical terms are mapped to values by an interpretation $\mathcal{J}$: $[\![f(t_1, \ldots, t_n)]\!] = f_{\mathcal{J}}([\![t_1]\!], \ldots, [\![t_n]\!])$. Logical terms of sort bool are called *constraints*. A constraint φ is *valid* if $[\![\varphi\gamma]\!] = \top$ for all substitutions γ such that $\gamma(x) \in \mathcal{V}\mathsf{al}$ for all $x \in \mathcal{V}\mathsf{ar}(\varphi)$. A *constrained rewrite rule* is a triple $\ell \to r\ [\varphi]$ where $\ell, r \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ are terms of the same sort such that $\mathsf{root}(\ell) \in \mathcal{F}_{\mathsf{te}} \setminus \mathcal{F}_{\mathsf{th}}$ and φ is a constraint. We denote the set $\mathcal{V}\mathsf{ar}(\varphi) \cup (\mathcal{V}\mathsf{ar}(r) \setminus \mathcal{V}\mathsf{ar}(\ell))$ of *logical* variables in $\ell \to r\ [\varphi]$ by $\mathcal{LV}\mathsf{ar}(\ell \to r\ [\varphi])$. We write $\mathcal{EV}\mathsf{ar}(\ell \to r\ [\varphi])$ for the set $\mathcal{V}\mathsf{ar}(r) \setminus (\mathcal{V}\mathsf{ar}(\ell) \cup \mathcal{V}\mathsf{ar}(\varphi))$. A set of constrained rewrite rules is called an LCTRS. A substitution σ is said to *respect* a rule $\ell \to r\ [\varphi]$, denoted by $\sigma \vDash \ell \to r\ [\varphi]$, if $\mathcal{D}\mathsf{om}(\sigma) = \mathcal{V}\mathsf{ar}(\ell) \cup \mathcal{V}\mathsf{ar}(r) \cup \mathcal{V}\mathsf{ar}(\varphi)$, $\sigma(x) \in \mathcal{V}\mathsf{al}$ for all $x \in \mathcal{LV}\mathsf{ar}(\ell \to r\ [\varphi])$, and $\varphi\sigma$ is valid. Moreover, a constraint φ is respected by σ, denoted by $\sigma \vDash \varphi$, if $\sigma(x) \in \mathcal{V}\mathsf{al}$ for all $x \in \mathcal{V}\mathsf{ar}(\varphi)$ and $\varphi\sigma$ is valid. We call $f(x_1, \ldots, x_n) \to y\ [y = f(x_1, \ldots, x_n)]$ with a fresh variable y and $f \in \mathcal{F}_{\mathsf{th}} \setminus \mathcal{V}\mathsf{al}$ a *calculation rule*. The set of all calculation rules induced by the signature $\mathcal{F}_{\mathsf{th}}$ of an LCTRS $\mathcal{R}$ is denoted by $\mathcal{R}_{\mathsf{ca}}$ and we abbreviate $\mathcal{R} \cup \mathcal{R}_{\mathsf{ca}}$ to $\mathcal{R}_{\mathsf{rc}}$. A rewrite step $s \to_{\mathcal{R}} t$ satisfies $s|_p = \ell\sigma$ and $t = s[r\sigma]_p$ for some position p, constrained rewrite rule $\ell \to r\ [\varphi]$ in $\mathcal{R}_{\mathsf{rc}}$, and substitution σ such that $\sigma \vDash \ell \to r\ [\varphi]$.

*This research is funded by the Austrian Science Fund (FWF) project I5943.

A *constrained term* is a pair $s\ [\varphi]$ consisting of a term s and a constraint φ. Two constrained terms $s\ [\varphi]$ and $t\ [\psi]$ are *equivalent*, denoted by $s\ [\varphi] \sim t\ [\psi]$, if for every substitution γ respecting φ there is some substitution δ respecting ψ such that $s\gamma = t\delta$, and vice versa. Let $s\ [\varphi]$ be a constrained term. If $s|_p = \ell\sigma$ for some constrained rewrite rule $\rho\colon \ell \to r\ [\psi] \in \mathcal{R}_{\mathsf{rc}}$, position p, and substitution σ such that $\sigma(x) \in \mathcal{V}\mathsf{al} \cup \mathcal{V}\mathsf{ar}(\varphi)$ for all $x \in \mathcal{LV}\mathsf{ar}(\rho)$, φ is satisfiable and $\varphi \Rightarrow \psi\sigma$ is valid then $s\ [\varphi] \to_{\mathcal{R}} s[r\sigma]_p\ [\varphi]$. The rewrite relation $\rightsquigarrow_{\mathcal{R}}$ on constrained terms is defined as $\sim \cdot \to_{\mathcal{R}} \cdot \sim$ and $s\ [\varphi] \rightsquigarrow_p t\ [\psi]$ indicates that the rewrite step in $\rightsquigarrow_{\mathcal{R}}$ takes place at position p. Similarly, we write $s\ [\varphi] \rightsquigarrow_{\geqslant p} t\ [\psi]$ if the position in the rewrite step is below position p. Note that in our definition of $\to_{\mathcal{R}}$ the constraint is not modified. This is different from [1, 2] where calculation steps $s[f(v_1, \ldots, v_n)]_p\ [\varphi] \to s[v]_p\ [\varphi \wedge v = f(v_1, \ldots, v_n)]$ modify the constraint. Our relation $\rightsquigarrow$ is equivalent to the relation $\sim \cdot (\to_{\mathsf{ru}} \cup \to_{\mathsf{ca}}) \cdot \sim$ in [1, 2] since the constraint can be expanded as exemplified below.

Example 1. Consider the constrained term $x+1\ [x > 3]$. Calculation steps as defined in [1, 2] permit $x + 1\ [x > 3] \to z\ [z = x + 1 \wedge x > 3]$. In our setting, an initial equivalence step is required to introduce the fresh variable z and the corresponding assignment needed to perform a calculation: $x + 1\ [x > 3] \sim x + 1\ [z = x + 1 \wedge x > 3] \to z\ [z = x + 1 \wedge x > 3]$.

Our treatment allows for a much simpler definition of parallel and multi-step rewriting since we do not have to merge different constraints.

2 Transformation

Our transformation is defined below.

Definition 2. Given an LCTRS $\mathcal{R}$, the TRS $\overline{\mathcal{R}}$ consists of the following rules: (1) $\ell\tau \to r\tau$ for all $\rho\colon \ell \to r\ [\varphi] \in \mathcal{R}$ with $\tau \vDash \rho$ and $\mathcal{D}\mathsf{om}(\tau) = \mathcal{LV}\mathsf{ar}(\rho)$, and (2) $f(v_1, \ldots, v_n) \to [\![f(v_1, \ldots, v_n)]\!]$ for all $f \in \mathcal{F}_{\mathsf{th}} \setminus \mathcal{V}\mathsf{al}$ and $v_1, \ldots, v_n \in \mathcal{V}\mathsf{al}$.

Note that $\overline{\mathcal{R}}$ typically consists of infinitely many rules.

Lemma 3. *The rewrite relations of $\mathcal{R}$ and $\overline{\mathcal{R}}$ are the same. Moreover $\xrightarrow{p}_{\mathcal{R}} = \xrightarrow{p}_{\overline{\mathcal{R}}}$ for all positions p.*

Proof. We first show $\xrightarrow{p}_{\mathcal{R}} \subseteq \xrightarrow{p}_{\overline{\mathcal{R}}}$. Assume $s \xrightarrow{p}_{\mathcal{R}} t$. So either $s = s[f(v_1, \ldots, v_n)]_p \to s[v]_p = t$ for some $f \in \mathcal{F}_{\mathsf{th}} \setminus \mathcal{V}\mathsf{al}$ and $v = [\![f(v_1, \ldots, v_n)]\!]$ or $s = s[\ell\sigma]_p \to s[r\sigma]_p = t$ for some $\rho\colon \ell \to r\ [\varphi] \in \mathcal{R}$ and $\sigma \vDash \rho$. In the first case $s \xrightarrow{p}_{\overline{\mathcal{R}}} t$ by the definition of $\overline{\mathcal{R}}$. In the second case we split σ into two substitutions $\tau = \{x \mapsto \sigma(x) \mid x \in \mathcal{LV}\mathsf{ar}(\rho)\}$ and $\delta = \{x \mapsto \sigma(x) \mid x \in \mathcal{V}\mathsf{ar}(\ell) \setminus \mathcal{LV}\mathsf{ar}(\rho)\}$. From $\sigma \vDash \rho$ we infer $\tau \vDash \rho$ and thus $\tau(x) \in \mathcal{V}\mathsf{al}$ for all $x \in \mathcal{LV}\mathsf{ar}(\rho)$. Hence $\sigma = \tau \cup \delta = \tau\delta$. We have $\ell\tau \to r\tau \in \overline{\mathcal{R}}$. Hence $s = s[\ell\tau\delta]_p \xrightarrow{p}_{\overline{\mathcal{R}}} s[r\tau\delta]_p = t$ as desired. To show the reverse inclusion $\xrightarrow{p}_{\overline{\mathcal{R}}} \subseteq \xrightarrow{p}_{\mathcal{R}}$ we assume $s \xrightarrow{p}_{\overline{\mathcal{R}}} t$. When the applied rule stems from a calculation rule in $\mathcal{R}$, we trivially have $s \xrightarrow{p}_{\mathcal{R}} t$. Otherwise $s = s[\ell\tau\delta]_p \xrightarrow{p}_{\overline{\mathcal{R}}} s[r\tau\delta]_p$ for some rule $\rho\colon \ell \to r\ [\varphi] \in \mathcal{R}$ with $\tau \vDash \rho$. Let $\sigma = \tau\delta$. Since $\tau(x) \in \mathcal{V}\mathsf{al}$ for all $x \in \mathcal{LV}\mathsf{ar}(\rho)$, we have $x\sigma = x\tau$ for all $x \in \mathcal{LV}\mathsf{ar}(\rho)$. Hence $\sigma \vDash \rho$ and thus $s = s[\ell\sigma]_p \xrightarrow{p}_{\mathcal{R}} s[r\sigma]_p = t$. □

Since $\to_{\mathcal{R}}$ and $\to_{\overline{\mathcal{R}}}$ coincide, we drop the subscript in the sequel. Rules of type (2) in Definition 2 can be viewed as type (1) for the rule $f(x_1, \ldots, x_n) \to y\ [y = f(x_1, \ldots, x_n)]$ in $\mathcal{R}_{\mathsf{ca}}$ by taking $\tau = \{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n, y \mapsto [\![f(v_1, \ldots, v_n)]\!]\}$. Hence we do not distinguish between the two cases and consider only rules of type (1) with $\rho \in \mathcal{R}_{\mathsf{rc}}$.

Definition 4. An *overlap* of an LCTRS $\mathcal{R}$ is a triple $\langle \rho_1, p, \rho_2 \rangle$ with rules $\rho_1 \colon \ell_1 \to r_1 \ [\varphi_1]$ and $\rho_2 \colon \ell_2 \to r_2 \ [\varphi_2]$, satisfying the following conditions: (1) ρ_1 and ρ_2 are variable-disjoint variants of rewrite rules in $\mathcal{R}_{\mathsf{rc}}$, (2) $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$, (3) ℓ_1 and $\ell_2|_p$ unify with mgu σ such that $\sigma(x) \in \mathcal{V}\mathsf{al} \cup \mathcal{V}$ for all $x \in \mathcal{LV}\mathsf{ar}(\rho_1) \cup \mathcal{LV}\mathsf{ar}(\rho_2)$, (4) $\varphi_1\sigma \wedge \varphi_2\sigma$ is satisfiable, and (5) if $p = \epsilon$ then ρ_1 and ρ_2 are not variants, or $\mathcal{V}\mathsf{ar}(r_1) \not\subseteq \mathcal{V}\mathsf{ar}(\ell_1)$. In this case we call $\ell_2\sigma[r_1\sigma]_p \approx r_2\sigma \ [\varphi_1\sigma \wedge \varphi_2\sigma \wedge \psi\sigma]$ a *constrained critical pair* obtained from the overlap $\langle \rho_1, p, \rho_2 \rangle$. Here $\psi = \bigwedge\{x = x \mid x \in \mathcal{EV}\mathsf{ar}(\rho_1) \cup \mathcal{EV}\mathsf{ar}(\rho_2)\}$. The set of all constrained critical pairs of $\mathcal{R}$ is denoted by $\mathsf{CCP}(\mathcal{R})$.

A key ingredient of our approach is to relate critical pairs of the transformed TRS to constrained critical pairs of the originating LCTRS.

Theorem 5. *For every critical pair $s \approx t$ of $\overline{\mathcal{R}}$ there exists a constrained critical pair $s' \approx t' \ [\varphi']$ of $\mathcal{R}$ and a substitution γ such that $s = s'\gamma$, $t = t'\gamma$ and $\gamma \vDash \varphi'$.*

Proof. Let $s \approx t$ be a critical pair of $\overline{\mathcal{R}}$, originating from the critical peak $\ell_2\mu\sigma[r_1\nu\sigma]_p \leftarrow \ell_2\mu\sigma = \ell_2\mu\sigma[\ell_1\nu\sigma]_p \to r_2\mu\sigma$ with variants $\rho_1 \colon \ell_1 \to r_1 \ [\varphi_1]$ and $\rho_2 \colon \ell_2 \to r_2 \ [\varphi_2]$ of rules in $\mathcal{R}_{\mathsf{rc}}$ without shared variables, $\mathcal{D}\mathsf{om}(\nu) = \mathcal{LV}\mathsf{ar}(\rho_1)$, $\mathcal{D}\mathsf{om}(\mu) = \mathcal{LV}\mathsf{ar}(\rho_2)$, $\nu \vDash \rho_1$, $\mu \vDash \rho_2$, $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2\mu)$, and σ is an mgu of $\ell_2\mu|_p$ and $\ell_1\nu$. Moreover, if $p = \epsilon$ then $\ell_1\nu \to r_1\nu$ and $\ell_2\mu \to r_2\mu$ are not variants. Define $\tau = \nu \uplus \mu$. Clearly, $\ell_1\tau = \ell_1\nu$, $r_1\tau = r_1\nu$, $\ell_2\tau = \ell_2\mu$, $r_2\tau = r_2\mu$, $\tau \vDash \rho_1$ and $\tau \vDash \rho_2$. Hence the given peak can be written as $\ell_2\tau\sigma[r_1\tau\sigma]_p \leftarrow \ell_2\tau\sigma = \ell_2\tau\sigma[\ell_1\tau\sigma]_p \to r_2\tau\sigma$ and $\tau \vDash \varphi$ where $\varphi = \varphi_1 \wedge \varphi_2 \wedge \bigwedge\{x = x \mid x \in \mathcal{EV}\mathsf{ar}(\rho_1) \cup \mathcal{EV}\mathsf{ar}(\rho_2)\}$. Since $\ell_2|_p\tau\sigma = \ell_1\tau\sigma$ there exists an mgu δ of $\ell_2|_p$ and ℓ_1, and a substitution γ such that $\delta\gamma = \tau\sigma$. Let $s' = \ell_2\delta[r_1\delta]_p$ and $t' = r_2\delta$. We claim that $\langle \rho_1, p, \rho_2 \rangle$ is an overlap of $\mathcal{R}$, resulting in the constrained critical pair $s' \approx t' \ [\varphi\delta]$. Condition (1) of Definition 4 is trivially satisfied. For condition (2) we need to show $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$. This follows from $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2\mu)$, $\mu(x) \in \mathcal{V}\mathsf{al}$ for every $x \in \mathcal{D}\mathsf{om}(\mu)$, and $\mathsf{root}(\ell_2\mu|_p) = \mathsf{root}(\ell_1\nu) \in \mathcal{F} \setminus \mathcal{V}\mathsf{al}$. For condition (3) it remains to show that $\delta(x) \in \mathcal{V}\mathsf{al} \cup \mathcal{V}$ for all $x \in \mathcal{LV}\mathsf{ar}(\rho_1) \cup \mathcal{LV}\mathsf{ar}(\rho_2)$. Suppose to the contrary that $\mathsf{root}(\delta(x)) \in \mathcal{F} \setminus \mathcal{V}\mathsf{al}$ for some $x \in \mathcal{LV}\mathsf{ar}(\rho_1) \cup \mathcal{LV}\mathsf{ar}(\rho_2)$. Then $\mathsf{root}(\delta(x)) = \mathsf{root}(\gamma(\delta(x))) = \mathsf{root}(\sigma(\tau(x))) \in \mathcal{F} \setminus \mathcal{V}\mathsf{al}$, which contradicts $\tau \vDash \varphi$. Condition 4 follows from the identity $\delta\gamma = \tau\sigma$ together with $\tau \vDash \varphi$ which imply $\delta\gamma \vDash \varphi$ and thus $\varphi\delta$ is satisfiable. Hence also $\varphi_1\delta \wedge \varphi_2\delta$ is satisfiable. It remains to show condition 5, so let $p = \epsilon$ and further assume that ρ_1 and ρ_2 are variants. So there exists a variable renaming π such that $\rho_1\pi = \rho_2$. In particular, $\ell_1\pi = \ell_2$ and $r_1\pi = r_2$. Let $x \in \mathcal{V}\mathsf{ar}(\ell_1)$. If $x \in \mathcal{LV}\mathsf{ar}(\rho_1) = \mathcal{D}\mathsf{om}(\nu)$ then $\tau(x) = \nu(x) \in \mathcal{V}\mathsf{al}$. Moreover, $\pi(x) \in \mathcal{LV}\mathsf{ar}(\rho_2) = \mathcal{D}\mathsf{om}(\mu)$ and thus $\tau(\pi(x)) = \mu(\pi(x)) \in \mathcal{V}\mathsf{al}$. Since $\ell_1\tau$ and $\ell_2\tau$ are unifiable, $\pi(\tau(x)) = \tau(x) = \tau(\pi(x))$. If $x \notin \mathcal{LV}\mathsf{ar}(\rho_1)$ then $\tau(x) = x$, $\pi(x) \notin \mathcal{LV}\mathsf{ar}(\rho_2)$ and similarly $\tau(\pi(x)) = \pi(x) = \pi(\tau(x))$. All in all, $\ell_1\tau\pi = \ell_1\pi\tau = \ell_2\tau$. Now, if $\mathcal{V}\mathsf{ar}(r_1) \subseteq \mathcal{V}\mathsf{ar}(\ell_1)$ then we obtain $r_1\tau\pi = r_1\pi\tau = r_2\tau$, contradicting the fact that $\ell_1\nu \to r_1\nu$ and $\ell_2\mu \to r_2\mu$ are not variants. We conclude that $s' \approx t' \ [\varphi\delta]$ is a constrained critical pair of $\mathcal{R}$. So we can take $\varphi' = \varphi\delta$. Clearly, $s = s'\gamma$ and $t = t'\gamma$. Moreover, $\gamma \vDash \varphi'$ since $\varphi'\gamma = \varphi\tau\sigma = \varphi\tau$ and $\tau \vDash \varphi$. □

The converse does not hold in general.

Example 6. Consider the LCTRS $\mathcal{R}$ consisting of the single rule $\mathsf{a} \to x \ [x = 0]$ where the variable x ranges over the integers. Since x appears on the right-hand side but not the left, we obtain a constrained critical pair $x \approx x' \ [x = 0 \wedge x' = 0]$. Since the constraint uniquely determines the values of x and x', the TRS $\overline{\mathcal{R}}$ consists of the single rule $\mathsf{a} \to 0$. Obviously $\overline{\mathcal{R}}$ has no critical pairs.

The above example also shows that orthogonality of $\overline{\mathcal{R}}$ does not imply orthogonality of $\mathcal{R}$. However, the counterexample relies somewhat on a technicality in condition (5) of Definition 4.

It only occurs when the two rules $\ell_1 \to r_1\ [\varphi_1]$ and $\ell_2 \to r_2\ [\varphi_2]$ involved in the critical pair overlap at the root and have instances $\ell_1\tau_1 \to r_1\tau_1$ and $\ell_2\tau_2 \to r_2\tau_2$ in $\overline{\mathcal{R}}$ which are variants of each other. By dealing with such cases separately we can prove the following theorem.

Theorem 7. *For every constrained critical pair $s \approx t\ [\varphi]$ of $\mathcal{R}$ and every substitution σ with $\sigma \vDash \varphi$, (1) $s\sigma = t\sigma$ or (2) there exist a critical pair $u \approx v$ of $\overline{\mathcal{R}}$ and a substitution δ such that $s\sigma = u\delta$ and $t\sigma = v\delta$.*

Proof. Let $s \approx t\ [\varphi]$ be a constrained critical pair of $\mathcal{R}$ originating from the critical peak $s = \ell_2\theta[r_1\theta]_p \leftarrow \ell_2\theta[\ell_1\theta]_p \to r_2\theta = t$ with variants $\rho_1\colon \ell_1 \to r_1\ [\varphi_1]$ and $\rho_2\colon \ell_2 \to r_2\ [\varphi_2]$ of rules in $\mathcal{R}_{\mathsf{rc}}$, and an mgu θ of $\ell_2|_p$ and ℓ_1 where $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$. Moreover $\theta(x) \in \mathcal{V}\mathsf{al} \cup \mathcal{V}$ for all $x \in \mathcal{LV}\mathsf{ar}(\rho_1) \cup \mathcal{LV}\mathsf{ar}(\rho_2)$, and $\varphi = \varphi_1\theta \wedge \varphi_2\theta \wedge \psi\theta$ with $\psi = \bigwedge\{x = x \mid x \in \mathcal{EV}\mathsf{ar}(\rho_1) \cup \mathcal{EV}\mathsf{ar}(\rho_2)\}$. Let σ be a substitution with $\sigma \vDash \varphi$. Hence $\theta\sigma \vDash \varphi_1 \wedge \varphi_2 \wedge \psi$ and further $\sigma(\theta(x)) \in \mathcal{V}\mathsf{al}$ for all $x \in \mathcal{LV}\mathsf{ar}(\rho_1) \cup \mathcal{LV}\mathsf{ar}(\rho_2)$. We split $\theta\sigma$ into substitutions τ_1, τ_2 and π as follows:

$$\tau_i(x) = \begin{cases} x\theta\sigma & \text{if } x \in \mathcal{LV}\mathsf{ar}(\rho_i) \\ x & \text{otherwise} \end{cases} \qquad \pi(x) = \begin{cases} x\theta\sigma & \text{if } x \in \mathcal{D}\mathsf{om}(\theta\sigma) \setminus (\mathcal{LV}\mathsf{ar}(\rho_1) \cup \mathcal{LV}\mathsf{ar}(\rho_2)) \\ x & \text{otherwise} \end{cases}$$

for $i \in \{1,2\}$. From $\theta\sigma \vDash \varphi_1 \wedge \varphi_2 \wedge \psi$ and $\mathcal{V}\mathsf{ar}(\varphi_i) \subseteq \mathcal{LV}\mathsf{ar}(\rho_i)$ we infer $\tau_i \vDash \varphi_i$ for $i \in \{1,2\}$. Since $\mathcal{D}\mathsf{om}(\tau_i) = \mathcal{LV}\mathsf{ar}(\rho_i)$, $\ell_i\tau_i \to r_i\tau_i \in \overline{\mathcal{R}}$ for $i \in \{1,2\}$. Furthermore, $\tau_i\pi = \tau_i \cup \pi$ for $i \in \{1,2\}$. Hence $\ell_2|_p\tau_2\pi = \ell_2|_p\theta\sigma = \ell_1\theta\sigma = \ell_1\tau_1\pi$, implying that $\ell_2|_p\tau_2$ and $\ell_1\tau_1$ are unifiable. Let γ be an mgu of these two terms. There exists a substitution δ such that $\gamma\delta = \pi$. Clearly $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2\tau_2)$. If $p \neq \epsilon$ or $\ell_1\tau_1 \to r_1\tau_1$ and $\ell_2\tau_2 \to r_2\tau_2$ are not variants, then $u \approx v$ with $u = \ell_2\tau_2\gamma[r_1\tau_1\gamma]_p$ and $v = r_2\tau_2\gamma$ is a critical pair of $\overline{\mathcal{R}}$. Moreover $t\sigma = r_2\theta\sigma = r_2\tau_2\pi = r_2\tau_2\gamma\delta = v\delta$, and similarly $s\sigma = u\delta$. Thus option (2) is satisfied. If $p = \epsilon$ and $\ell_1\tau_1 \to r_1\tau_1$ and $\ell_2\tau_2 \to r_2\tau_2$ are variants then $s\sigma = r_1\tau_1\gamma\delta = r_2\tau_2\gamma\delta = t\sigma$, fulfilling (1). □

A direct consequence is that weak orthogonality of $\overline{\mathcal{R}}$ implies weak orthogonality of $\mathcal{R}$.

3 Confluence

Using Theorem 5 we can easily transfer confluence criteria for TRSs to LCTRSs. Rather than reproving the confluence results reported in [1, 4, 2], we illustrate this by extending the result of van Oostrom [3] concerning (almost) development closed critical pairs from TRSs to LCTRSs. The following result from [4] plays an important role.

Lemma 8. *Suppose $s \approx t\ [\varphi] \mathrel{\tilde{\to}}_p u \approx v\ [\psi]$ with $\gamma \vDash \varphi$ and position p. If $p \geqslant 1$ then $s\gamma \to u\delta$ and $t\gamma = v\delta$ for some substitution δ with $\delta \vDash \psi$. If $p \geqslant 2$ then $s\gamma = u\delta$ and $t\gamma \to v\delta$ for some substitution δ with $\delta \vDash \psi$.* □

Definition 9. Let $\mathcal{R}$ be an LCTRS. The multi-step relation $\mathrel{-\!\!\circ\!\!\to}$ on terms is defined inductively as follows: (1) $x \mathrel{-\!\!\circ\!\!\to} x$ for all variables x, (2) $f(s_1, \ldots, s_n) \mathrel{-\!\!\circ\!\!\to} f(t_1, \ldots, t_n)$ if $s_i \mathrel{-\!\!\circ\!\!\to} t_i$ with $1 \leqslant i \leqslant n$, (3) $\ell\sigma \mathrel{-\!\!\circ\!\!\to} r\tau$ if $\ell \to r\ [\varphi] \in \mathcal{R}_{\mathsf{rc}}$, $\sigma \vDash \ell \to r\ [\varphi]$ and $\sigma \mathrel{-\!\!\circ\!\!\to} \tau$, where $\sigma \mathrel{-\!\!\circ\!\!\to} \tau$ denotes $\sigma(x) \mathrel{-\!\!\circ\!\!\to} \tau(x)$ for all variables $x \in \mathcal{D}\mathsf{om}(\sigma)$.

The next definition inductively defines multi-step rewriting on constrained terms.

Definition 10. Let $\mathcal{R}$ be an LCTRS. The multi-step relation $\mathrel{-\!\!\circ\!\!\to}$ on constrained terms is defined inductively as follows:

1. $x\ [\varphi] \mathrel{-\!\!\circ\!\!\to} x\ [\varphi]$ for all variables x,

2. $f(s_1, \ldots, s_n)\ [\varphi] \mathrel{-\!\!\circ\!\!\to} f(t_1, \ldots, t_n)\ [\varphi]$ if $s_i\ [\varphi] \mathrel{-\!\!\circ\!\!\to} t_i\ [\varphi]$ for $1 \leqslant i \leqslant n$,

3. $\ell\sigma\ [\varphi] \mathrel{-\!\!\circ\!\!\to} r\tau\ [\varphi]$ if $\rho\colon \ell \to r\ [\psi] \in \mathcal{R}_{\mathsf{rc}}$, $\sigma(x) \in \mathcal{V}\mathsf{al} \cup \mathcal{V}\mathsf{ar}(\varphi)$ for all $x \in \mathcal{LV}\mathsf{ar}(\rho)$, φ is satisfiable, $\varphi \Rightarrow \psi\sigma$ is valid, and $\sigma\ [\varphi] \mathrel{-\!\!\circ\!\!\to} \tau\ [\varphi]$.

Here $\sigma\ [\varphi] \mathrel{-\!\!\circ\!\!\to} \tau\ [\varphi]$ denotes $\sigma(x)\ [\varphi] \mathrel{-\!\!\circ\!\!\to} \tau(x)\ [\varphi]$ for all variables $x \in \mathcal{D}\mathsf{om}(\sigma)$. The multi-step rewrite relation $\mathrel{\widetilde{-\!\!\circ\!\!\to}}$ on constrained terms is then defined as $\sim \cdot \mathrel{-\!\!\circ\!\!\to} \cdot \sim$.

Lemma 11. *If $s\ [\varphi] \mathrel{-\!\!\circ\!\!\to} t\ [\varphi]$ then $s\delta \mathrel{-\!\!\circ\!\!\to} t\delta$ for all substitutions $\delta \vDash \varphi$.*

Proof. We proceed by induction on $\mathrel{-\!\!\circ\!\!\to}$. In case 1 we have $x\ [\varphi] \mathrel{-\!\!\circ\!\!\to} x\ [\varphi]$, and $x\delta \mathrel{-\!\!\circ\!\!\to} x\delta$ holds trivially. In case 2 we have $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$ and $s_i\ [\varphi] \mathrel{-\!\!\circ\!\!\to} t_i\ [\varphi]$ for $1 \leqslant i \leqslant n$. From the induction hypothesis we obtain $s_i\delta \mathrel{-\!\!\circ\!\!\to} t_i\delta$ for all $1 \leqslant i \leqslant n$, which further implies $s\delta \mathrel{-\!\!\circ\!\!\to} t\delta$. In case 3 we have $s = \ell\sigma$ and $t = r\sigma$ for some rule $\rho\colon \ell \to r\ [\psi] \in \mathcal{R}_{\mathsf{rc}}$, $\sigma(x) \in \mathcal{V}\mathsf{al} \cup \mathcal{V}\mathsf{ar}(\varphi)$ for all $x \in \mathcal{LV}\mathsf{ar}(\rho)$, φ is satisfiable, $\varphi \Rightarrow \psi\sigma$ is valid, and $\sigma(x)\ [\varphi] \mathrel{-\!\!\circ\!\!\to} \tau(x)\ [\varphi]$ for all $x \in \mathcal{V}\mathsf{ar}(\varphi)$. From the induction hypothesis we obtain $\sigma(x)\delta \mathrel{-\!\!\circ\!\!\to} \tau(x)\delta$ for all $x \in \mathcal{V}\mathsf{ar}(\varphi)$. Moreover, since $\delta \vDash \varphi$ we have $\delta \vDash \psi\sigma$ and thus also $\sigma\delta \vDash \psi$. Therefore $s\delta = \ell\sigma\delta \mathrel{-\!\!\circ\!\!\to} r\tau\delta = t\delta$ as desired. □

Lemma 12. *If $s \approx t\ [\varphi] \mathrel{\widetilde{-\!\!\circ\!\!\to}}_{\geqslant 1} u \approx v\ [\psi]$ then for all substitutions $\sigma \vDash \varphi$ there exists a $\delta \vDash \psi$ such that $s\sigma \mathrel{-\!\!\circ\!\!\to} u\delta$ and $t\sigma = v\delta$.*

Proof. By unfolding the definition of $\mathrel{\widetilde{-\!\!\circ\!\!\to}}$ we obtain $s \approx t\ [\varphi] \sim s' \approx t'\ [\varphi'] \mathrel{-\!\!\circ\!\!\to}_{\geqslant 1} u' \approx v'\ [\varphi'] \sim u \approx v\ [\psi]$. Let σ be a substitution with $\sigma \vDash \varphi$. From the definition of $\sim$ we obtain a substitution τ such that $\tau \vDash \varphi'$, $s\sigma = s'\tau$ and $t\sigma = t'\tau$. As all contracted redexes in the multi-step $s' \approx t'\ [\varphi']$ are below the position 1, this corresponds to case 2 in Definition 10 with s' and t' being the first and second argument of $\approx$. Hence $s'\ [\varphi'] \mathrel{-\!\!\circ\!\!\to} u'\ [\varphi']$ and $t' = v'$. We therefore obtain $t'\tau = v'\tau$ and $s'\tau \mathrel{-\!\!\circ\!\!\to} u'\tau$ from Lemma 11. Now considering the equivalence $u' \approx v'\ [\varphi'] \sim u \approx v\ [\psi]$ together with $\tau \vDash \varphi'$ we obtain a substitution δ such that $\delta \vDash \psi$, $u'\tau = u\delta$ and $v'\tau = v\delta$. Putting this all together we have $s\sigma = s'\tau \mathrel{-\!\!\circ\!\!\to} u'\tau = u\delta$ and $t\sigma = t'\tau = v'\tau = v\delta$. □

Definition 13. A constrained critical pair $s \approx t\ [\varphi]$ is *development closed* if $s \approx t\ [\varphi] \mathrel{\widetilde{-\!\!\circ\!\!\to}}_{\geqslant 1} u \approx v\ [\psi]$ for some trivial $u \approx v\ [\psi]$. A constrained critical pair is *almost development closed* if it is an inner critical pair and development closed, or it is an overlay and $s \approx t\ [\varphi] \mathrel{\widetilde{-\!\!\circ\!\!\to}}_{\geqslant 1} \cdot \mathrel{\tilde{\to}}^*_{\geqslant 2} u \approx v\ [\psi]$ for some trivial $u \approx v\ [\psi]$. An LCTRS is called (almost) development closed if all its constrained critical pairs are (almost) development closed.

Lemma 14. *If a constrained critical pair $s \approx t\ [\varphi]$ is almost development closed then for all substitutions σ with $\sigma \vDash \varphi$ we have $s\sigma \mathrel{-\!\!\circ\!\!\to} \cdot {}^*\!\leftarrow t\sigma$.*

Proof. Let $s \approx t\ [\varphi]$ be an almost development closed constrained critical pair, and $\sigma \vDash \varphi$ some substitution. From Definition 13 we obtain $s \approx t\ [\varphi] \mathrel{-\!\!\circ\!\!\to}_{\geqslant 1} u' \approx v'\ [\psi'] \to^*_{\geqslant 2} u \approx v\ [\psi]$ where $u\tau = v\tau$ for all $\tau \vDash \psi$ for some constrained term $u' \approx v'\ [\psi']$. Looking at the first part of the sequence, $s \approx t\ [\varphi] \mathrel{-\!\!\circ\!\!\to}_{\geqslant 1} u' \approx v'\ [\psi']$ and $s\sigma \mathrel{-\!\!\circ\!\!\to} u'\delta$ where $v'\delta = t\sigma$ for some $\delta \vDash \psi'$ by Lemma 12. For the second part of the sequence $u' \approx v'\ [\psi'] \to^*_{\geqslant 2} u \approx v\ [\psi]$ we obtain $v'\delta \to^* v\gamma$, $u'\delta = u\gamma$ for some $\gamma \vDash \psi$, by repeated application of Lemma 8. Moreover we have $u\gamma = v\gamma$. Hence $s\sigma \mathrel{-\!\!\circ\!\!\to} u'\delta = u\gamma = v\gamma \mathrel{{}^*\!\leftarrow} v'\delta = t\sigma$. □

Theorem 15. *If an* LCTRS $\mathcal{R}$ *is almost development closed then so is* $\overline{\mathcal{R}}$.

Proof. Take any critical pair $s \approx t$ from $\overline{\mathcal{R}}$. From Theorem 5 we know that there exists a constrained critical pair $s' \approx t'$ $[\varphi]$ in $\mathcal{R}$ where $s'\sigma = s$ and $t'\sigma = t$ for some $\sigma \vDash \varphi$. Since the constrained critical pair must be almost development closed, Lemma 14 yields $s = s'\sigma \mathrel{-\!\!\circ\!\!\rightarrow} \cdot {}^*\!\leftarrow t'\sigma = t$ if it is an overlay and $s = s'\sigma \mathrel{-\!\!\circ\!\!\rightarrow} t'\sigma = t$ otherwise. This proves that $\overline{\mathcal{R}}$ is almost development closed. □

Interestingly, the converse does not hold, as seen in the following example.

Example 16. Consider the LCTRS $\mathcal{R}$ with the theory LIA and consisting of the rules:

$$\mathsf{f}(x) \to \mathsf{g}(x) \quad \mathsf{f}(x) \to \mathsf{h}(x)\ [1 \leqslant x \leqslant 2] \quad \mathsf{g}(x) \to \mathsf{h}(2)\ [x = 2z] \quad \mathsf{g}(x) \to \mathsf{h}(1)\ [x = 2z + 1]$$

The TRS $\overline{\mathcal{R}}$ consists of the rules

$$\begin{array}{lll} \mathsf{f}(x) \to \mathsf{g}(x) & \mathsf{f}(1) \to \mathsf{h}(1) & \mathsf{g}(n) \to \mathsf{h}(1) \quad \text{for all odd } n \in \mathbb{Z} \\ & \mathsf{f}(2) \to \mathsf{h}(2) & \mathsf{g}(n) \to \mathsf{h}(2) \quad \text{for all even } n \in \mathbb{Z} \end{array}$$

and has two (modulo symmetry) critical pairs $\mathsf{g}(1) \approx \mathsf{h}(1)$ and $\mathsf{g}(2) \approx \mathsf{h}(2)$. Since $\mathsf{g}(1) \mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{h}(1)$ and $\mathsf{g}(2) \mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{h}(2)$, $\overline{\mathcal{R}}$ is almost development closed. The constrained critical pair $\mathsf{g}(x) \approx \mathsf{h}(x)$ $[1 \leqslant x \leqslant 2]$ is not almost development closed, since it is a normal form with respect to the rewrite relation on constrained terms.

This also makes intuitive sense, since a rewrite step $s \approx t$ $[\varphi] \rightsquigarrow u \approx v$ $[\psi]$ implies that the same step can be taken on all instances $s\sigma \approx t\sigma$ where $\sigma \vDash \varphi$. However it may be the case, like in the above example, that different instances of the constrained critical pair require different steps to obtain a closing sequence, which cannot directly be modeled using rewriting on constraint terms.

Since left-linearity of $\overline{\mathcal{R}}$ is preserved and left-linear almost development closed TRSs are confluent [3] the following corollary is obtained via Theorem 15. In fact $\mathcal{R}$ only has to be linear in the variables $x \notin \mathcal{LV}\mathsf{ar}$, since that is sufficient for $\overline{\mathcal{R}}$ to be linear.

Corollary 17. *Left-linear almost development closed* LCTRS*s are confluent.* □

4 Conclusion

We presented a left-linearity preserving transformation from LCTRSs into TRSs such that critical pairs in the latter correspond to constrained critical pairs in the former. As a consequence, confluence results for TRSs based on restricted joinability conditions carry directly over to LCTRSs. This drastically simplifies correctness proofs (like the ones in [1, 2]) and makes the formalization of confluence proofs for LCTRSs in a proof assistant a realistic goal.

References

[1] Cynthia Kop and Naoki Nishida. Term rewriting with logical constraints. In *Proc. 9th FRoCoS*, volume 8152 of *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 CADE*, LNAI, 2023. To appear.

[3] Vincent van Oostrom. Developing developments. *Theoretical Computer Science*, 175(1):159–181, 1997. doi:10.1016/S0304-3975(96)00173-9.

[4] Sarah Winkler and Aart Middeldorp. Completion for logically constrained rewriting. In *Proc. 3rd FSCD*, volume 108 of *LIPIcs*, pages 30:1–30:18, 2018. doi:10.4230/LIPIcs.FSCD.2018.30.

On Confluence Criteria for Non-terminating Abstract Rewriting Systems

Ievgen Ivanov

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

Abstract

We give a summary of confluence criteria based on the generalized Newman's lemma recently proposed in [6]. The mentioned criteria are applicable to wide classes of not-necessarily-terminating abstract rewriting systems, in particular those that arise from reachability relations on states of discrete-continuous (hybrid) dynamical models.

1 Introduction

Textbooks and monographs related to term rewriting and/or its applications in computer science (e.g. [7, 1, 8]) usually include a chapter on basic theory of abstract rewriting systems (ARS). Besides other material, such a chapter typically includes a definition of the confluence property for ARS and Newman's lemma [10] as a confluence criterion for terminating ARS. Although in the literature one can find many other confluence conditions applicable to (special classes of) abstract rewriting systems (e.g. Hindley-Rosen lemma [3, 12], Rosen's request lemma [12], Huet's strong confluence lemma [4], De Bruijn's lemma [2], Van Oostrom's theorem [13, Theorem 3.7] and its corollaries concerning locally decreasing and decreasing Church-Rosser ARS), arguably, Newman's lemma is the most widely known one. Among its intrinsic qualities are simple formulation, reduction of global analysis to local analysis, and the lack of assumptions about the structure of elements and/or the reduction relation. However, some new potential application areas, like confluence analysis of ARS derived from reachability relations on states of nondeterminisic discrete-continuous (hybrid) models (in particular, those that arise from modeling of cyber-physical systems that combine computational and physical processes), are beyond the scope of Newman's lemma because of the termination assumption.

In [6] we proposed a generalization of Newman's lemma that aimed to preserve its positive qualities, but extend its scope to a wide class of not-necessarily-terminating ARS. The main result of [6] was formalized and machine-checked in Isabelle 2022 proof assistant using HOL logic in [5].

In this extended abstract we give a summary of the results of [6] and additional corollaries from them.

2 Preliminaries

2.1 Standard Notions and Well-Known Facts

An abstract rewriting system (ARS) is a pair $(A, \to)$, where A is a set and $\to \subseteq A \times A$ is a binary relation called reduction[1]. We will use the following notation:

[1] In some works an ARS is allowed to have an indexed family of reduction relations. In this work we restrict attention to rewriting systems with a single reduction relation.

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

Also note that in this work we assume that the axiom of choice holds.
An ARS $(A, \rightarrow)$ is

- confluent, if $\forall a, b, c \in A\ (a \rightarrow^* b \wedge a \rightarrow^* c \Rightarrow \exists d \in A\ (b \rightarrow^* d \wedge c \rightarrow^* d))$
- locally confluent, if $\forall a, b, c \in A\ (a \rightarrow b \wedge a \rightarrow c \Rightarrow \exists d \in A\ (b \rightarrow^* d \wedge c \rightarrow^* d))$
- (weakly) normalizing, if $\forall a \in A\ \exists\ b \in A\ (a \rightarrow^* b \wedge \neg(\exists c \in A\ b \rightarrow c))$
- strongly normalizing, or, alternatively, terminating, or Noetherian, if there exists no infinite reduction sequence $a_1 \rightarrow a_2 \rightarrow ...$ $(a_i \in A)$
- inductive, if for every (infinite) reduction sequence $a_1 \rightarrow a_2 \rightarrow ...$ (where $a_i \in A$ for all $i \geq 1$) there exists $a \in A$ such that $a_n \rightarrow^* a$ for all $n \geq 1$
- acyclic, if $\neg\exists a \in A\ (a \rightarrow^+ a)$.

Some well-known facts about these notions are [7, 1, 8, 4]:

1. Any terminating ARS is acyclic and inductive.
2. A terminating ARS is confluent if (and only if) it is locally confluent
(a variant of formulation of Newman's lemma).

2.2 Topology on ARS

For any ARS $(A, \rightarrow)$ the relation $\rightarrow^*$ can be considered as a preorder (a reflexive and transitive relation). This fact can be used to transfer some preorder-specific notions to ARS and define a topology on them as follows:

- a subset $C \subseteq A$ is a *chain* (in the preordered set $(A, \rightarrow^*)$), if
$$\forall a, b \in C\ (a \rightarrow^* b\ \vee\ b \rightarrow^* a)$$
- an element $a \in A$ is an *upper bound* of a subset $S \subseteq A$ (in the preordered set $(A, \rightarrow^*)$), if
$$\forall b \in S\ b \rightarrow^* a$$
- $a \in A$ is a *least upper bound* of a subset $S \subseteq A$ (in the preordered set $(A, \rightarrow^*)$), if
$$(\forall b \in S\ b \rightarrow^* a) \wedge \forall a' \in A\ ((\forall b \in S\ b \rightarrow^* a') \Rightarrow a \rightarrow^* a').$$
- a subset $S \subseteq A$ is *closed* [6] in the preordered set $(A, \rightarrow^*)$, if for each nonempty chain C in $(A, \rightarrow^*)$ and for each $a \in A$, if a is a least upper bound of C and $C \subseteq S$, then $a \in S$.
- a subset $S \subseteq A$ is *open* [6] in the preordered set $(A, \rightarrow^*)$, if $A \backslash S$ is closed in $(A, \rightarrow^*)$
- for a given $B \subseteq A$, a subset $S \subseteq B$ is *relatively open* [6] in B (in the preordered set $(A, \rightarrow^*)$), if S is open in the preordered set $(B, \rightarrow^* \cap (B \times B))$.

For any ARS $(A, \rightarrow)$ we will denote:
$$\mathcal{T}(A, \rightarrow) = \{S \subseteq A \mid S \text{ is open in the preordered set } (A, \rightarrow^*)\}.$$

2.3 Induction Principles for ARS and Related Facts

It is well-known (e.g. [8, paragraph 1.3.15]) that an ARS is terminating if and only if it has sound Noetherian induction principle: for every unary predicate $P : A \to \{True, False\}$, if

$$\forall a \in A\ ((\forall b \in A\ (a \to^+ b \Rightarrow P(b))) \Rightarrow P(a))$$

holds, then $\forall a \in A\ P(a)$.

We will say that an ARS $(A, \to)$

- has *sound open induction principle* [11, 6], if for every unary predicate $P : A \to \{True, False\}$ such that $\{a \in A \mid P(a)\} \in \mathcal{T}(A, \to)$ (i.e. the truth domain of P is open), if

  $$\forall a \in A\ ((\forall b \in A\ (a \to^+ b \Rightarrow P(b))) \Rightarrow P(a))$$

  holds, then $\forall a \in A\ P(a)$
- is *strictly inductive* [9, 6], if $(A, \to^*)$ is a strictly inductive preordered set, i.e. every nonempty chain in $(A, \to^*)$ has a least upper bound in $(A, \to^*)$
- is *openly normalizing*, if $(A, \to)$ is (weakly) normalizing and for each $a, a' \in A$ such that a' is a normal form of a (i.e. $a \to^* a' \wedge \neg(\exists a'' \in A\ a' \to a'')$), the set

  $$\{b \in A \mid a \to^* b \text{ and } a' \text{ is the only normal form of } b\ \}$$

  is relatively open in $\{b \in A \mid a \to^* b\}$ in the preordered set $(A, \to^*)$.

Proposition 1.

(1) An ARS has sound open induction principle if and only if it is acyclic and strictly inductive [6, Proposition 14] (a related fact is [11, Theorem 3.3]).

(2) Any strictly inductive ARS is inductive (but is not necessarily terminating).

(3) Any terminating ARS is acyclic, strictly inductive, and openly normalizing.

Example 1. *If $[0, 1]$ denotes the real unit interval and $\to$ denotes the standard strict order on real numbers restricted to $[0, 1]$ (i.e. $< \cap([0, 1] \times [0, 1])$), then the ARS $([0, 1], \to)$ has sound open induction principle, is strictly inductive, acyclic, openly normalizing, but is not terminating (e.g. $0.9 \to 0.99 \to 0.999 \to ...$ is an infinite reduction sequence).*

Example 2. *The ARS $(\{1\}, \to)$, where $\to = \{(1, 1)\}$, is strictly inductive, but is not acyclic. In fact, every ARS with finite set of elements is strictly inductive [6, Proposition 16].*

3 Confluence Conditions for Strictly Inductive ARS

Below we give a necessary and sufficient condition for confluence of a strictly inductive ARS (Theorem 1) and two more specialized confluence conditions (Lemma 1 and 2).

Firstly, let consider specialized conditions (they have simpler formulation).

Lemma 1. *An acyclic and strictly inductive ARS is confluent if and only if it is openly normalizing and locally confluent (a consequence of [6, Theorem 28]).*

From Proposition 1(1) and Lemma 1(1) it follows that

Lemma 2. *Let $(A, \to)$ be an ARS with sound open induction principle.*
Then $(A, \to)$ is confluent if and only if it is openly normalizing and locally confluent.

Remark 3.1. *The ordinary Newman's lemma implies that if $(A, \to)$ is an ARS with sound Noetherian induction principle, then $(A, \to)$ is confluent if and only if $(A, \to)$ is locally confluent. Thus Lemma 2 can be considered as an extension of Newman's lemma that replaces Noetherian induction with open induction [11]. The open normalization condition is not needed in the case of terminating ARS, because every terminating ARS is openly normalizing.*

Note that the local confluence condition is not so useful in confluence criteria when a reduction relation $\to$ is transitive (since in this case the notions of confluence and local confluence become almost trivially equivalent). To obtain useful confluence conditions in the case of a transitive reduction relation, and also to cover cases where an ARS is not acyclic, we introduce the following notions.

Let $(A, \to)$ be an ARS and $a, a' \in A$. Then

- a is *quasi-irreducible* [6], if $\forall b \in A\ (a \to^* b \Rightarrow b \to^* a)$
- a' is a *quasi-normal form* (QNF [6]) of a, if $a \to^* a'$ and a' is quasi-irreducible
- a, a' are *QNF-equivalent* [6], if $\{b \in A \mid b \text{ is a QNF of } a\} = \{b \in A \mid b \text{ is a QNF of } a'\}$.

An ARS $(A, \to)$ is

- *quasi-normalizing* [6], if each $a \in A$ has a quasi-normal form
- *openly quasi-normalizing* [6], if $(A, \to)$ is quasi-normalizing and for all $a, a' \in A$, if a' is a quasi-normal form of a, the set

  $$\{b \in A \mid a \to^* b \ \wedge\ b \text{ and } a' \text{ are QNF-equivalent }\}$$

  is relatively open in $\{b \in A \mid a \to^* b\}$ in the preordered set $(A, \to^*)$.
- *quasi-locally confluent* [6], if for each $a \in A$ there exists $S \subseteq \{a' \in A \mid a \to^+ a'\}$ such that the following two conditions hold (called in [6] the two-consistency and coinitiality condition respectively):

  $$\forall b, c \in S\ \exists d \in A\ (b \to^* d \wedge c \to^* d),$$

  $$\forall a' \in A\ (a \to^+ a' \Rightarrow (a' \to^* a) \vee (\exists b \in S\ b \to^* a' \wedge \neg(b \to^* a))).$$

Lemma 3. *Let $(A, \to)$ be an acyclic ARS. Then*

(1) $(A, \to)$ is quasi-normalizing if and only if $(A, \to)$ is (weakly) normalizing

(2) $(A, \to)$ is openly quasi-normalizing if and only if $(A, \to)$ is openly normalizing

(3) if $(A, \to)$ is locally confluent, then $(A, \to)$ is quasi-locally confluent [6, Proposition 27].

Theorem 1 ([6], Theorem 28)**.** *Let $(A, \to)$ be a strictly inductive ARS. Then $(A, \to)$ is confluent if and only if $(A, \to)$ is openly quasi-normalizing and quasi-locally confluent.*

Note that one cannot simply omit the strict inductivity assumption, or the open quasi-normalization, or quasi-local confluence condition from the statement of Theorem 1:

Proposition 2. *(1) There exists a strictly inductive and quasi-locally confluent ARS that is not confluent. An example of such an ARS is* $(A, \to)$*, where* $A = \{(x,t) \in \mathbb{R} \times \mathbb{R} \mid t \leq 0\}$ *and* $(x,t) \to (x',t')$ *if and only if* $t < t' \wedge x' - x \leq t' - t \wedge x \leq x'$ *(also see [6, Example 31]).*

(2) There exists a strictly inductive and openly quasi-normalizing ARS that is not confluent. An example of such an ARS is $(\{0,1,2\}, \to)$*, where* $\to = \{(0,1),(0,2)\}$*.*

(3) There exists an inductive, openly quasi-normalizing, and quasi-locally confluent ARS that is not confluent. An example of such an ARS is $(\mathbb{N}_0 \times \{0,1\}, \to)$*, where* $\mathbb{N}_0 = \{0,1,2,...\}$ *is the set of non-negative integers and* $(i,j) \to (i',j')$ *if and only if* $(1 \leq i < i' \wedge j = j') \vee (i \geq 1 \wedge i' = 0)$*.*

Note that example ARS explicitly mentioned in Proposition 2 are not confluent. An example of a strictly inductive, confluent ARS with more than one irreducible element that is in the scope of the confluence criterion given in Theorem 1 can be found in [6, Example 32]:

$([0,1] \times [0,1], \to)$, where $[0,1]$ is the real unit interval and $(a,b) \to (a',b')$ if and only if $(a < a' \vee (a = a' \wedge b' < b \wedge (a < 1 \vee b < 1))) \wedge (a - b \leq a' - b') \wedge (a = b \Rightarrow a' = b') \wedge (a \leq b \Rightarrow a' \leq b')$. Note that here $(1,0)$ and $(1,1)$ are irreducible.

Other similar examples can be constructed using reachability relations on states of nondeterministic discrete-continuous dynamical models, e.g. see a nondeterministic model of a bouncing ball in [6, Figures 4–6].

References

[1] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge university press, 1999.

[2] N.G. de Bruijn. A note on weak diamond properties. Memorandum 7808, Eindhoven Uninversity of Technology, 1978.

[3] J.R. Hindley. The Church–Rosser property and a result in combinatory logic. PhD thesis, University of Newcastle-upon-Tyne, 1964.

[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. Formalization of Generalized Newman's Lemma. https://doi.org/10.5281/zenodo.7855691, 2023. [Online].

[6] Ievgen Ivanov. Generalized Newman's lemma for discrete and continuous systems. In *8th International Conference on Formal Structures for Computation and Deduction (FSCD 2023)*, Leibniz International Proceedings in Informatics (LIPIcs), 2023. (Accepted for publication).

[7] Jan Willem Klop. *Term rewriting systems.* Centrum voor Wiskunde en Informatica, 1990.

[8] Philippe Malbos. Lectures on algebraic rewriting. http://hal.archives-ouvertes.fr/hal-02461874, 2019. [Online].

[9] George Markowsky. Chain-complete posets and directed sets with applications. *Algebra universalis*, 6(1):53–68, 1976.

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

[11] Jean-Claude Raoult. Proving open properties by induction. *Information processing letters*, 29(1):19–23, 1988.

[12] B.K. Rosen. Tree-manipulating systems and Church-Rosser theorems. *Journal of the ACM*, 20:160–187, 1973.

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

Church–Rosser Modulo for Left-Linear TRSs Revisited

Johannes Niederhauser[1], Nao Hirokawa[2], and Aart Middeldorp[1]

[1] Department of Computer Science, University of Innsbruck, Austria
[2] School of Information Science, JAIST, Japan

Abstract

It is known that ordinary critical pairs suffice to establish the Church–Rosser property modulo an equational theory $\mathcal{B}$ for a left-linear and $\mathcal{B}$-terminating TRS. We extend this result to prime critical pairs by introducing a new confluence criterion for ARSs.

1 Introduction

In this paper, we present a new characterization of the Church–Rosser property modulo an equational theory $\mathcal{B}$ for left-linear TRSs which are terminating modulo $\mathcal{B}$. This works for every variable-preserving (i.e., $\mathcal{V}\mathsf{ar}(\ell) = \mathcal{V}\mathsf{ar}(r)$ for all $\ell \approx r \in \mathcal{B}$) equational theory $\mathcal{B}$. The result is based on an observation due to Huet [3] but allows us to use prime critical pairs [4] instead of ordinary critical pairs. The proof of our new result is facilitated by *peak-and-cliff decreasingness*, an extension of peak decreasingness [2] which is a simple confluence criterion for ARSs designed to replace complicated proof orderings in the correctness proofs of completion procedures. Both the confluence criterion as well as the main result are crucial ingredients of a novel fairness condition for left-linear $\mathcal{B}$-completion presented in our recent paper [5]. For the special case of AC, we also present a novel counterexample which shows the necessity of AC termination as a precondition of the main theorem. To the best of our knowledge, this was not documented before.

2 Preliminaries

We assume that the reader is familiar with term rewriting but recapitulate the important definition of (prime) critical pairs. Let $\mathcal{R}$ be a TRS. An *overlap* is a triple $\langle \ell_1 \to r_1, p, \ell_2 \to r_2 \rangle$ satisfying the following properties:

- $\ell_1 \to r_1$ and $\ell_2 \to r_2$ are variants of rewrite rules of $\mathcal{R}$ without common variables,
- p is a non-variable position in ℓ_2,
- ℓ_1 and $\ell_2|_p$ are unifiable, and
- if $p = \epsilon$ then $\ell_1 \to r_1$ and $\ell_2 \to r_2$ are not variants.

Let σ be a most general unifier of ℓ_1 and $\ell_2|_p$. The corresponding *critical peak*

$$\ell_2\sigma[r_1\sigma]_p \xleftarrow{p} \ell_2\sigma \xrightarrow{\epsilon} r_2\sigma$$

represents the two ways in which $\ell_2\sigma$ can be rewritten and the equation $\ell_2\sigma[r_1\sigma]_p \approx r_2\sigma$ is its associated *critical pair*. The set of critical pairs of a TRS $\mathcal{R}$ is denoted by $\mathsf{CP}(\mathcal{R})$. A critical peak $t \xleftarrow{p} s \xrightarrow{\epsilon} u$ is *prime* if all proper subterms of $s|_p$ are in normal form. Critical pairs derived

from prime critical peaks are called prime. The set of all prime critical pairs of a TRS $\mathcal{R}$ is denoted by $\mathsf{PCP}(\mathcal{R})$. Throughout the paper, we use the following abbreviations:

$$\begin{aligned} \downarrow^{\sim}_{\mathcal{R}} &= \to^{*}_{\mathcal{R}} \cdot \sim_{\mathcal{B}} \cdot \overset{*}{\mathcal{R}}\!\leftarrow \\ \mathcal{B}^{\pm} &= \mathcal{B} \cup \{ r \approx \ell \mid \ell \approx r \in \mathcal{B} \} \\ \mathsf{CP}^{\pm}(\mathcal{R}_1, \mathcal{R}_2) &= \mathsf{CP}(\mathcal{R}_1, \mathcal{R}_2) \cup \mathsf{CP}(\mathcal{R}_2, \mathcal{R}_1) \end{aligned}$$

Here $\mathsf{CP}(\mathcal{R}_1, \mathcal{R}_2)$ denotes the set of critical pairs (t, u) that originate from critical peaks of the form $t \;{}^{p}_{\mathcal{R}_1}\!\leftarrow s \to^{\epsilon}_{\mathcal{R}_2} u$. The starting point of our work is the following result by Huet [3].

Lemma 1. *For left-linear* TRS*s* $\mathcal{R}$*, the inclusion* ${}_{\mathcal{R}}\!\leftarrow \cdot \leftrightarrow_{\mathcal{B}} \subseteq \downarrow^{\sim}_{\mathcal{R}} \cup \leftrightarrow_{\mathsf{CP}^{\pm}(\mathcal{R},\, \mathcal{B}^{\pm})}$ *holds.* □

Note that Lemma 1 allows us to use ordinary critical pairs instead of $\mathcal{B}$-critical pairs. In particular, equational unification modulo $\mathcal{B}$ can be replaced by syntactic unification which improves efficiency. Furthermore, the form of the joining sequence $(\downarrow^{\sim}_{\mathcal{R}})$ is advantageous as it uses the normal rewrite relation and just one $\mathcal{B}$-equality check in the end as opposed to rewrite steps modulo the theory $(\sim_{\mathcal{B}} \cdot \to_{\mathcal{R}} \cdot \sim_{\mathcal{B}})$. However, left-linearity is necessary in Lemma 1 as the following example illustrates.

Example 1. *Consider the* TRS $\mathcal{R}$ *consisting of the single rule* $\mathsf{f}(x, x) \to x$ *with* $+$ *as an additional AC function symbol. Consider the conversion*

$$x + y \;{}_{\mathcal{R}}\!\leftarrow\; \mathsf{f}(x + y, x + y) \;\sim_{\mathsf{AC}}\; \mathsf{f}(x + y, y + x)$$

There are no critical pairs in $\mathcal{R}$ *and between* $\mathcal{R}$ *and* $\mathsf{AC}^{\pm}$*, so* $\mathsf{CP}(\mathcal{R}) = \mathsf{CP}^{\pm}(\mathcal{R}, \mathsf{AC}^{\pm}) = \varnothing$*. Moreover,* $x + y \downarrow^{\sim}_{\mathcal{R}} \mathsf{f}(x + y, y + x)$ *does not hold because* $x + y$ *and* $\mathsf{f}(x + y, y + x)$ *are* $\mathcal{R}$*-normal forms which are not* AC *equivalent.*

3 Peak-and-Cliff Decreasingness

In the following, we assume that equivalence relations $\sim$ are defined as the reflexive and transitive closure of a symmetric relation $\mapsto$, so $\sim = \mapsto^{*}$. We refer to conversions of the form $\leftarrow \cdot \mapsto$ or $\mapsto \cdot \to$ as *local cliffs*. Furthermore, we assume that steps are labeled with labels from a set I, so let $\mathcal{A} = \langle A, \{\to_\alpha\}_{\alpha \in I} \rangle$ be an ARS and $\sim = (\bigcup \{ \mapsto_\alpha \mid \alpha \in I \})^{*}$ an equivalence relation on A.

Definition 1. *The* ARS $\mathcal{A}$ *is* peak-and-cliff decreasing *if there is a well-founded order* $>$ *on* I *such that for all* $\alpha, \beta \in I$ *the inclusions*

$$\alpha\!\leftarrow \cdot \to_\beta \;\subseteq\; \overset{*}{\underset{\vee\alpha\beta}{\Longleftrightarrow}} \qquad\qquad \alpha\!\leftarrow \cdot \mapsto_\beta \;\subseteq\; \overset{*}{\underset{\vee\alpha}{\Longleftrightarrow}} \cdot \overset{=}{\underset{\beta}{\leftarrow}}$$

hold. Here $\vee\alpha\beta$ *denotes the set* $\{\gamma \in I \mid \alpha > \gamma \text{ or } \beta > \gamma\}$ *and if* $J \subseteq I$ *then* $\to_J$ *denotes* $\bigcup \{ \to_\gamma \mid \gamma \in J \}$*. We abbreviate* $\vee\alpha\alpha$ *to* $\vee\alpha$*.*

In the remainder of this section, we show that peak-and-cliff decreasingness implies the Church–Rosser modulo $\sim$ property.

Lemma 2. *Every conversion modulo* $\sim$ *is a valley modulo* $\sim$ *or contains a local peak or cliff:*

$$\Leftrightarrow^{*} \;\subseteq\; \downarrow^{\sim} \cup \Leftrightarrow^{*} \cdot \leftarrow \cdot \to \cdot \Leftrightarrow^{*} \cup \Leftrightarrow^{*} \cdot \mapsto \cdot \to \cdot \Leftrightarrow^{*} \cup \Leftrightarrow^{*} \cdot \leftarrow \cdot \mapsto \cdot \Leftrightarrow^{*}$$

Proof. We define $\leftrightharpoons = \Leftrightarrow^* \cdot \leftarrow \cdot \rightarrow \cdot \Leftrightarrow^* \cup \Leftrightarrow^* \cdot \mapsto \cdot \rightarrow \cdot \Leftrightarrow^* \cup \Leftrightarrow^* \cdot \leftarrow \cdot \mapsto \cdot \Leftrightarrow^*$ in order to simplify the notation. Suppose $a \Leftrightarrow^n b$. We show that $a \downarrow^\sim b$ or $a \leftrightharpoons b$ by induction on n. If $n = 0$ then $a = b$ and therefore also $a \downarrow^\sim b$. If $n > 0$ then $a \Leftrightarrow c \Leftrightarrow^{n-1} b$ for some c. The induction hypothesis yields $c \downarrow^\sim b$ or $c \leftrightharpoons b$. In the latter case we are already done because $\Leftrightarrow \cdot \leftrightharpoons \subseteq \leftrightharpoons$. In the former case, we distinguish between three subcases: $a \rightarrow c$, $a \leftarrow c$ or $a \sim c$. If $a \rightarrow c$, we immediately obtain $a \downarrow^\sim c$. For the remaining two cases, note that there exists a k such that $c \rightarrow^k \cdot \sim \cdot {}^*\!\leftarrow b$. We continue with a case analysis on k:

- $k = 0$: If $a \leftarrow c$ we have $a \leftarrow c \sim c' {}^*\!\leftarrow b$ for some c'. Now either $c = c'$ and $a \downarrow^\sim b$ or $c \mapsto \cdot \sim c'$ and therefore $a \leftrightharpoons b$. If $a \sim c$ we have $a \downarrow^\sim b$ because $\sim$ is transitive.
- $k > 0$: If $a \leftarrow c$ then there exists a c' such that $a \leftarrow c \rightarrow c' \Leftrightarrow^* b$ and therefore $a \leftrightharpoons b$. Finally, if $a \sim c$ then $a \sim c \rightarrow c' \Leftrightarrow^* b$ for some c'. If $a = c$ then we obtain $a \downarrow^\sim b$ from the induction hypothesis as there is a conversion between a and b of length $n - 1$. Otherwise, $a \sim \cdot \mapsto c$ and therefore $a \leftrightharpoons b$. □

The proof of the following theorem is based on a well-founded order on multisets. We denote the multiset extension of an order $>$ by $>_{\mathsf{mul}}$. It is well-known that the multiset extension of a well-founded order is also well-founded.

Theorem 1. *If $\mathcal{A}$ is peak-and-cliff decreasing then $\mathcal{A}$ is Church–Rosser modulo $\sim$.*

Proof. With every conversion C we associate a multiset M_C consisting of labels of its rewrite and equivalence relation steps. Since $\mathcal{A}$ is peak-and-cliff decreasing, there is a well-founded order $>$ on I which allows us to replace conversions C of the forms ${}_\alpha\!\leftarrow \cdot \rightarrow_\beta$, ${}_\alpha\!\leftarrow \cdot \mapsto_\beta$ and $\mapsto_\beta \cdot \rightarrow_\alpha$ by conversions C' where $M_C >_{\mathsf{mul}} M_{C'}$. Hence, we prove that $\mathcal{A}$ is Church–Rosser modulo $\sim$, i.e., $\Leftrightarrow^* \subseteq \downarrow^\sim$, by well-founded induction on $>_{\mathsf{mul}}$. Consider a conversion $a \Leftrightarrow^* b$ which we call C. By Lemma 2 we either have $a \downarrow^\sim b$ (which includes the case that C is empty) or one of the following cases holds:

$$a \Leftrightarrow^* \cdot \leftarrow \cdot \rightarrow \cdot \Leftrightarrow^* b \qquad a \Leftrightarrow^* \cdot \leftarrow \cdot \mapsto \cdot \Leftrightarrow^* b \qquad a \Leftrightarrow^* \cdot \mapsto \cdot \rightarrow \cdot \Leftrightarrow^* b$$

If $a \downarrow^\sim b$ we are immediately done. In the remaining cases, we have a local peak or cliff with concrete labels α and β, so $M_C = \Gamma_1 \uplus \{\alpha, \beta\} \uplus \Gamma_2$. Since $\mathcal{A}$ is peak-and-cliff decreasing, there is a conversion C' with $M_{C'} = \Gamma_1 \uplus \Gamma \uplus \Gamma_3$ where $\{\alpha, \beta\} >_{\mathsf{mul}} \Gamma$. Hence, $M_C >_{\mathsf{mul}} M_{C'}$ and we finish the proof by applying the induction hypothesis. □

For the main result of this paper, a simpler version of peak-and-cliff decreasingness suffices.

Definition 2. *Let $\mathcal{A} = \langle A, \rightarrow \rangle$ be an* ARS *equipped with a $\sim$-compatible well-founded order $>$ on A and $\sim = \mapsto^*$ an equivalence relation on A. We write $b \xrightarrow{a} c$ ($b \overset{a}{\mapsto} c$) if $b \rightarrow c$ ($b \mapsto c$) and $b \sim a$. We say that $\mathcal{A}$ is* source decreasing modulo $\sim$ *if the inclusions*

$$\leftarrow a \rightarrow \subseteq \overset{*}{\underset{\vee a}{\Longleftrightarrow}} \qquad\qquad \leftarrow a \mapsto \subseteq \overset{*}{\underset{\vee a}{\Longleftrightarrow}} \cdot \overset{=}{\underset{a}{\leftarrow}}$$

hold for all $a \in A$. Here $\leftarrow a \rightarrow$ ($\leftarrow a \mapsto$) denotes the binary relation consisting of all pairs (b, c) such that $a \rightarrow b$ and $a \rightarrow c$ ($a \mapsto c$). Furthermore, $\overset{*}{\underset{\vee a}{\Longleftrightarrow}}$ denotes the binary relation consisting of all pairs of elements which are connected by a conversion where each step is labeled by an element smaller than a.

Corollary 1. *Every* ARS *which is source decreasing modulo $\sim$ is Church–Rosser modulo $\sim$.*

Proof. In the definition of peak decreasingness we set $I = A$. Note that this implies $\alpha = \beta$ for all local peaks and cliffs. Hence, the ARS is peak-and-cliff decreasing and we can conclude by an appeal to Theorem 1. □

4 Prime Critical Pairs

In the following, $\mathsf{PCP}^{\pm}(\mathcal{R}, \mathcal{B}^{\pm})$ denotes the restriction of $\mathsf{CP}^{\pm}(\mathcal{R}, \mathcal{B}^{\pm})$ to prime critical pairs but where irreducibility is always checked with respect to $\mathcal{R}$, i.e., the critical peaks $t \overset{p}{\mathcal{R}}\!\leftarrow s \leftrightarrow^{\epsilon}_{\mathcal{B}} u$ and $t' \leftrightarrow^{p}_{\mathcal{B}} s \rightarrow^{\epsilon}_{\mathcal{R}} u'$ are both prime if proper subterms of $s|_p$ are irreducible with respect to $\mathcal{R}$.

Example 2. *Consider the TRS $\mathcal{R}$ consisting of the rewrite rules*

$$\mathsf{f}(\mathsf{a} + x) \to x \qquad \mathsf{f}(x + \mathsf{a}) \to x \qquad \mathsf{f}(\mathsf{b} + x) \to x \qquad \mathsf{f}(x + \mathsf{b}) \to x \qquad \mathsf{a} \to \mathsf{b}$$

and let $\mathcal{B} = \{x + y \approx y + x\}$. The TRS *$\mathcal{R}$ admits six (modulo symmetry) critical peaks of the form $t \overset{p}{\mathcal{R}}\!\leftarrow s \rightarrow^{\epsilon}_{\mathcal{R}} u$:*

$$\underline{\mathsf{f}(\mathsf{a}+\mathsf{a})}\ \swarrow\ \searrow\ \mathsf{a}\ \ \mathsf{a} \qquad \underline{\mathsf{f}(\mathsf{a}+\mathsf{b})}\ \swarrow\ \searrow\ \mathsf{a}\ \ \mathsf{b} \qquad \underline{\mathsf{f}(\mathsf{b}+\mathsf{a})}\ \swarrow\ \searrow\ \mathsf{a}\ \ \mathsf{b} \qquad \underline{\mathsf{f}(\mathsf{b}+\mathsf{b})}\ \swarrow\ \searrow\ \mathsf{b}\ \ \mathsf{b}$$

$$\mathsf{f}(\underline{\mathsf{a}}+x)\ \swarrow\ \searrow\ \mathsf{f}(\mathsf{b}+x)\ \ x \qquad \mathsf{f}(x+\underline{\mathsf{a}})\ \swarrow\ \searrow\ \mathsf{f}(x+\mathsf{b})\ \ x$$

Here the positions p in s are indicated by underlining. The first three peaks are not prime due to the reducible proper subterm a in $s|_p$. So $\mathsf{PCP}(\mathcal{R}) = \{\mathsf{b} \approx \mathsf{b}, \mathsf{f}(\mathsf{b} + x) \approx x, \mathsf{f}(x + \mathsf{b}) \approx x\}$. Similarly, $\mathcal{R}$ and $\mathcal{B}$ admit four critical peaks of the forms $t \overset{p}{\mathcal{R}}\!\leftarrow s \leftrightarrow^{\epsilon}_{\mathcal{B}} u$ and $t \leftrightarrow^{p}_{\mathcal{B}} s \rightarrow^{\epsilon}_{\mathcal{R}} u$:

$$\mathsf{f}(\underline{\mathsf{a}+x})\ \swarrow\ \searrow\ \mathsf{f}(x+\mathsf{a})\ \ x \qquad \mathsf{f}(\underline{x+\mathsf{a}})\ \swarrow\ \searrow\ \mathsf{f}(\mathsf{a}+x)\ \ x \qquad \mathsf{f}(\underline{\mathsf{b}+x})\ \swarrow\ \searrow\ \mathsf{f}(x+\mathsf{b})\ \ x \qquad \mathsf{f}(\underline{x+\mathsf{b}})\ \swarrow\ \searrow\ \mathsf{f}(\mathsf{b}+x)\ \ x$$

Here the first two peaks are not prime and thus $\mathsf{PCP}^{\pm}(\mathcal{R}, \mathcal{B}^{\pm}) = \{\mathsf{f}(\mathsf{b} + x) \approx x, \mathsf{f}(x + \mathsf{b}) \approx x\}$.

Definition 3. *Given a* TRS *$\mathcal{R}$ and terms s, t and u, we write $t \bigtriangledown_s u$ if $s \rightarrow^{+}_{\mathcal{R}} t$, $s \rightarrow^{+}_{\mathcal{R}} u$, and $t \downarrow_{\mathcal{R}} u$ or $t \leftrightarrow_{\mathsf{PCP}(\mathcal{R})} u$. We write $t \bigtriangledown^{\sim}_s u$ if $s \rightarrow^{+}_{\mathcal{R}} t$, $s \sim u$ and $t \downarrow^{\sim}_{\mathcal{R}} u$ or $t \leftrightarrow_{\mathsf{PCP}^{\pm}(\mathcal{R},\mathcal{B}^{\pm})} u$. Furthermore, ${}^{\sim}_{s}\bigtriangledown = \{(u, t) \mid t \bigtriangledown^{\sim}_s u\}$.*

Note that the joinability of ordinary critical peaks is not affected by incorporating $\mathcal{B}$ into conversions. Hence, the following result is taken from [2, Lemma 2.15] and therefore stated without a proof.

Lemma 3. *Let $\mathcal{R}$ be a* TRS*. If $t \overset{p}{\underset{\mathcal{R}}{\leftarrow}} s \overset{\epsilon}{\underset{\mathcal{R}}{\rightarrow}} u$ is a critical peak then $t \bigtriangledown^{2}_s u$.* □

Lemma 4. *Let $\mathcal{R}$ be a* TRS*. The following two inclusions hold:*

1. *If $t \overset{p}{\underset{\mathcal{R}}{\leftarrow}} s \overset{\epsilon}{\underset{\mathcal{B}}{\leftrightarrow}} u$ is a critical peak then $t \bigtriangledown_s \cdot \bigtriangledown^{\sim}_s u$.*
2. *If $t \overset{p}{\underset{\mathcal{B}}{\leftrightarrow}} s \overset{\epsilon}{\underset{\mathcal{R}}{\rightarrow}} u$ is a critical peak then $t {}^{\sim}_{s}\bigtriangledown \cdot \bigtriangledown_s u$.*

Proof. We only prove (1) as the other case is symmetrical. If all proper subterms of $s|_p$ are in normal form with respect to $\rightarrow_{\mathcal{R}}$, $t \approx u \in \mathsf{PCP}(\mathcal{R}, \mathcal{B}^{\pm})$ which establishes $t \bigtriangledown^{\sim}_s u$. Since also $t \bigtriangledown_s t$, we obtain the desired result. Otherwise, there are a position $q > p$ and a term v such that $s \overset{q}{\underset{\mathcal{R}}{\rightarrow}} v$ and all proper subterms of $s|_q$ are in normal form with respect to $\rightarrow_{\mathcal{R}}$. Together with Lemma 1 we obtain either $v \downarrow^{\sim}_{\mathcal{R}} u$ or $v \leftrightarrow_{\mathsf{PCP}^{\pm}(\mathcal{R},\mathcal{B}^{\pm})} u$. In both cases $v \bigtriangledown^{\sim}_s u$ holds. A

similar case analysis applies to the local peak $t \xleftarrow[\mathcal{R}]{p} s \xrightarrow[\mathcal{R}]{q} v$: By the Critical Pair Lemma, either $t \downarrow_{\mathcal{R}} v$ or $t \leftrightarrow_{\mathsf{CP}(\mathcal{R})} v$. In the latter case

$$v|_p \xleftarrow[\mathcal{R}]{q\backslash p} s|_p \xrightarrow[\mathcal{R}]{\epsilon} t|_p$$

is an instance of a prime critical peak as $q > p$ and all proper subterms of $s|_q$ are in normal form with respect to $\to_{\mathcal{R}}$. Closure of rewriting under contexts and substitutions yields $t \leftrightarrow_{\mathsf{PCP}(\mathcal{R})} v$. Therefore, we have $t \bigtriangledown_s v$ in both cases, concluding the proof. □

The following lemma generalizes the previous results of this section to arbitrary local peaks and cliffs.

Lemma 5. *Let $\mathcal{R}$ be a left-linear* TRS*. The following two properties hold:*

1. *If $t \,{}_{\mathcal{R}}\!\leftarrow s \to_{\mathcal{R}} u$ then $t \bigtriangledown_s^2 u$.*
2. *If $t \,{}_{\mathcal{R}}\!\leftarrow s \leftrightarrow_{\mathcal{B}} u$ then $t \bigtriangledown_s \cdot \bigtriangledown_s^{\sim} u$.*

Proof. We only prove (2) as the proof of (1) (which depends on the Critical Pair Lemma) can be found in [2, Lemma 2.16]. Let $t \,{}_{\mathcal{R}}\!\leftarrow s \leftrightarrow_{\mathcal{B}} u$. From Lemma 1 we obtain $t \downarrow_{\mathcal{R}}^{\sim} u$ or $t \leftrightarrow_{\mathsf{CP}^{\pm}(\mathcal{R},\mathcal{B}^{\pm})} u$. In the former case we are done as $t \bigtriangledown_s u \bigtriangledown_s u$. For the latter case we further distinguish between the two subcases $t \to_{\mathsf{CP}(\mathcal{R},\mathcal{B}^{\pm})} u$ and $u \to_{\mathsf{CP}(\mathcal{B}^{\pm},\mathcal{R})} t$. Note that this list of subcases is exhaustive due to the direction of the local cliff. If $t \to_{\mathsf{CP}(\mathcal{R},\mathcal{B}^{\pm})} u$, $t \bigtriangledown_s \cdot \bigtriangledown_s^{\sim} u$ follows from Lemma 4(1) and closure of rewriting under contexts and substitutions. If $u \to_{\mathsf{CP}(\mathcal{B}^{\pm},\mathcal{R})} t$, $u \,{}_s^{\sim}\!\bigtriangledown \cdot \bigtriangledown_s t$ and therefore $t \bigtriangledown_s \cdot \bigtriangledown_s^{\sim} u$ follows from Lemma 4(2) as well as closure of rewriting under contexts and substitutions. □

Now, we are able to prove the main result of this section, a novel necessary and sufficient condition for the Church–Rosser property modulo an equational theory $\mathcal{B}$ which strengthens the original result from [3] to prime critical pairs.

Theorem 2. *A left-linear* TRS $\mathcal{R}$ *which is terminating modulo $\mathcal{B}$ is Church–Rosser modulo $\mathcal{B}$ if and only if $\mathsf{PCP}(\mathcal{R}) \cup \mathsf{PCP}^{\pm}(\mathcal{R},\mathcal{B}^{\pm}) \subseteq \downarrow_{\mathcal{R}}^{\sim}$.*

Proof. The "only if" direction is trivial. For a proof of the "if" direction, we show that $\mathcal{R}$ is source decreasing; the Church–Rosser property modulo $\mathcal{B}$ is then an immediate consequence of Corollary 1. From the termination of $\mathcal{R}$ modulo $\mathcal{B}$ we obtain the well-founded order $> = \to_{\mathcal{R}/\mathcal{B}}^{+}$.

Consider an arbitrary local peak $t \,{}_{\mathcal{R}}\!\leftarrow s \to_{\mathcal{R}} u$. Lemma 5(1) yields a term v such that $t \bigtriangledown_s v \bigtriangledown_s u$. Together with $\mathsf{PCP}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}}^{\sim}$ we obtain $t \downarrow_{\mathcal{R}}^{\sim} v \downarrow_{\mathcal{R}}^{\sim} u$. By definition, $s > t, v, u$ so the corresponding condition required by source decreasingness is fulfilled.

Now consider an arbitrary local cliff $t \,{}_{\mathcal{R}}\!\leftarrow s \leftrightarrow_{\mathcal{B}} u$. Lemma 5(2) yields a term v such that $t \bigtriangledown_s v \bigtriangledown_s^{\sim} u$. Together with $\mathsf{PCP}(\mathcal{R}) \cup \mathsf{PCP}^{\pm}(\mathcal{R},\mathcal{B}^{\pm}) \subseteq \downarrow_{\mathcal{R}}^{\sim}$ we obtain $t \downarrow_{\mathcal{R}}^{\sim} v \downarrow_{\mathcal{R}}^{\sim} u$. By definition, $s > t, v$ and $s \sim u$. The conversion between v and u is of the form $v \to_{\mathcal{R}}^{*} \cdot \sim \cdot \,{}_{\mathcal{R}}^{k}\!\leftarrow u$ for some k. If $k = 0$ then all steps between v and u are labeled with terms which are smaller than s. If $k > 0$ then there exists a $w < s$ such that $v \to_{\mathcal{R}}^{*} \cdot \sim \cdot \,{}_{\mathcal{R}}^{k-1}\!\leftarrow w \,{}_{\mathcal{R}}\!\leftarrow u$. In this case all steps of the conversion are labeled with terms which are smaller than s except for the rightmost step which we may label with s. Hence, the corresponding condition required by source decreasingness is fulfilled in all cases. □

Example 3 (continued from Example 2)**.** *One can verify the termination of $\mathcal{R}/\mathcal{B}$ and the inclusion $\mathsf{PCP}(\mathcal{R}) \cup \mathsf{PCP}^{\pm}(\mathcal{R},\mathcal{B}^{\pm}) \subseteq \downarrow_{\mathcal{R}}^{\sim}$. By Theorem 2 the Church–Rosser modulo property holds.*

Finally, we show that the previous result does not hold if we just demand termination of $\mathcal{R}$. The counterexample shows this for the concrete case of AC and is based on Example 4.1.8 from [1] which uses an ARS. Note that the usage of prime critical pairs instead of critical pairs has no effect.

Example 4. *Consider the* TRS $\mathcal{R}$ *consisting of the rules*

$$\begin{aligned}
(\mathsf{b}+\mathsf{a})+\mathsf{a} &\to \mathsf{a}+(\mathsf{a}+\mathsf{b}) & (\mathsf{a}+\mathsf{b})+\mathsf{a} &\to \mathsf{a}+(\mathsf{a}+\mathsf{b}) & (\mathsf{a}+\mathsf{a})+\mathsf{b} &\to \mathsf{a}+(\mathsf{a}+\mathsf{b})\\
\mathsf{a}+(\mathsf{a}+\mathsf{b}) &\to \mathsf{b}+(\mathsf{a}+\mathsf{a}) & \mathsf{b}+(\mathsf{a}+\mathsf{a}) &\to \mathsf{c}\\
\mathsf{a}+(\mathsf{a}+\mathsf{b}) &\to \mathsf{a}+(\mathsf{b}+\mathsf{a}) & \mathsf{a}+(\mathsf{b}+\mathsf{a}) &\to \mathsf{d}
\end{aligned}$$

where $+$ *is an* AC *function symbol. Clearly, the (prime) critical pairs of* $\mathcal{R}$ *are joinable modulo* AC *because* $\mathsf{b}+(\mathsf{a}+\mathsf{a}) \sim_{\mathsf{AC}} \mathsf{a}+(\mathsf{b}+\mathsf{a})$. *For* $\mathsf{PCP}^{\pm}(\mathcal{R},\mathsf{AC}^{\pm})$ *we only have to consider the rules which rewrite to* c *and* d *respectively since all other rules only involve* AC *equivalent terms. Modulo symmetry, these (prime) critical pairs are*

$$\begin{aligned}
\mathsf{c} &\approx \mathsf{b}+(\mathsf{a}+\mathsf{a}) & \mathsf{c} &\approx (\mathsf{a}+\mathsf{a})+\mathsf{b} & \mathsf{c} &\approx (\mathsf{b}+\mathsf{a})+\mathsf{a}\\
\mathsf{c}+x &\approx \mathsf{b}+((\mathsf{a}+\mathsf{a})+x) & x+\mathsf{c} &\approx (x+\mathsf{b})+(\mathsf{a}+\mathsf{a})\\
\mathsf{d} &\approx \mathsf{a}+(\mathsf{a}+\mathsf{b}) & \mathsf{d} &\approx (\mathsf{b}+\mathsf{a})+\mathsf{a} & \mathsf{d} &\approx (\mathsf{a}+\mathsf{b})+\mathsf{a}\\
\mathsf{d}+x &\approx \mathsf{a}+((\mathsf{b}+\mathsf{a})+x) & x+\mathsf{d} &\approx (x+\mathsf{a})+(\mathsf{b}+\mathsf{a})
\end{aligned}$$

which can be joined by adding the rules

$$\begin{aligned}
\mathsf{b}+((\mathsf{a}+\mathsf{a})+x) &\to (\mathsf{b}+(\mathsf{a}+\mathsf{a}))+x & (x+\mathsf{b})+(\mathsf{a}+\mathsf{a}) &\to x+(\mathsf{b}+(\mathsf{a}+\mathsf{a}))\\
\mathsf{a}+((\mathsf{b}+\mathsf{a})+x) &\to (\mathsf{a}+(\mathsf{b}+\mathsf{a}))+x & (x+\mathsf{a})+(\mathsf{b}+\mathsf{a}) &\to x+(\mathsf{a}+(\mathsf{b}+\mathsf{a}))
\end{aligned}$$

to $\mathcal{R}$. *The new (prime) critical pairs in* $\mathsf{PCP}(\mathcal{R}) \cup \mathsf{PCP}^{\pm}(\mathcal{R},\mathsf{AC}^{\pm})$ *are trivially joinable modulo* AC *as they are* AC *equivalent. To sum up,* $\mathsf{PCP}(\mathcal{R}) \cup \mathsf{PCP}^{\pm}(\mathcal{R},\mathsf{AC}^{\pm}) \subseteq \downarrow_{\mathcal{R}}^{\sim}$. *Termination of* $\mathcal{R}$ *can be checked by e.g.* TTT2, *but the loop*

$$\mathsf{a}+(\mathsf{a}+\mathsf{b}) \to_{\mathcal{R}} \mathsf{a}+(\mathsf{b}+\mathsf{a}) \sim_{\mathsf{AC}} \mathsf{a}+(\mathsf{a}+\mathsf{b})$$

shows that $\mathcal{R}$ *is not* AC *terminating. We have* $\mathsf{c} \Leftrightarrow^{*} \mathsf{d}$ *but not* $\mathsf{c} \downarrow_{\mathcal{R}}^{\sim} \mathsf{d}$ *as the terms are normal forms and not* AC *equivalent. Hence,* $\mathcal{R}$ *is not Church–Rosser modulo* AC.

References

[1] Jürgen Avenhaus. *Reduktionssysteme*. Springer Berlin Heidelberg, 1995. In German. doi:10.1007/978-3-642-79351-6.

[2] Nao Hirokawa, Aart Middeldorp, Christian Sternagel, and Sarah Winkler. Abstract completion, formalized. *Logical Methods in Computer Science*, 15(3):19:1–19:42, 2019. doi:10.23638/LMCS-15(3:19)2019.

[3] Gérard 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.

[4] Deepak Kapur, David R. Musser, and Paliath Narendran. Only prime superpositions need be considered in the Knuth–Bendix completion procedure. *Journal of Symbolic Computation*, 6(1):19–36, 1988. doi:10.1016/S0747-7171(88)80019-1.

[5] Johannes Niederhauser, Nao Hirokawa, and Aart Middeldorp. Left-linear completion with AC axioms. In Brigitte Pientka and Cesare Tinelli, editors, *Proc. 29th International Conference on Automated Deduction*, volume 14132 of *Lecture Notes in Computer Science*, 2023. To appear.

Residuation = Skolemised Confluence

Vincent van Oostrom

oostrom@javakade.nl

Abstract

We express local confluence and the diamond property by means of residuation on peaks of steps. We extend residuation to peaks of reductions by means of tiling, and to 3-peaks of faces by means of bricklaying, and investigate some ramifications of our approach.

Skolemising local confluence into residuation. Recall [18, 1] a rewrite system $\to$ is locally confluent (has the diamond property) if for every local *peak* [3] ϕ, ψ of co-initial steps there exist reductions (steps) ψ', ϕ' constituting a *confluence* $C(\phi, \psi, \psi', \phi')$, i.e. reductions such that ϕ, ψ are co-initial, ψ', ϕ' are co-final, and ϕ, ψ' and ψ, ϕ' both compose. From the statement we obtain by introducing two skolem-functions $\setminus$ and $/$ for ψ' respectively ϕ' (binary as they depend on ϕ, ψ), the (equisatisfiable) statement that $C(\phi, \psi, \phi \setminus \psi, \phi / \psi)$ for every local peak ϕ, ψ; see Fig. 1. We will refer to such skolem-functions from peaks to reductions as *residuations*.

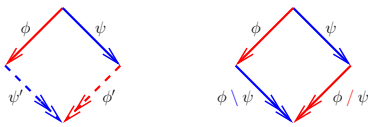


Figure 1: Local confluence (left) and its skolemisation (right)

Exploiting C is symmetric, a single skolem-function $|$ (notation of [12, Sect. 8–12]) will do:

Lemma 1. *$\to$ is locally confluent (has the diamond property) iff there is a* single *residuation $|$ to reductions (steps) such that $C(\phi, \psi, \psi \mid \phi, \phi \mid \psi)$, for all local peaks of steps ϕ, ψ.*

Stated differently, we may assume $\setminus$ is the *converse* of $/$, i.e. $\phi \setminus \psi = \psi / \phi$ for local peaks ϕ, ψ, hence that $C(\phi, \psi, \phi \setminus \psi, \phi / \psi) = C(\phi, \psi, \psi / \phi, \psi \setminus \phi)$.

Residuation by tiling. The aim of both introducing local confluence and the diamond property in [12] was to provide a way to establish *global* confluence by means of repeated *tiling* [3, 12, 10, 17] with *local* confluences. Rephrased in terms of residuation, the aim was to construct residuation for the rewrite system $\twoheadrightarrow$ of *reductions* from the residuation for *steps* $\to$. Tiling can be described by means of a rewrite system $\Rightarrow$ on *conversions* [3, 12, 18]. Tiling rules transform peaks into *valleys* [3]. Formally, to do so we associate to each given confluence $C(\phi, \psi, \chi, \omega)$ a rule $\phi^{-1} \cdot \psi \Rightarrow \chi \cdot \omega^{-1}$, where $\cdot$ and $^{-1}$ denote *composition* and *converse*. Applying such a rule $\ell \Rightarrow r$ to a conversion ς of shape $\zeta \cdot \ell \cdot \xi$ yields $\zeta \cdot r \cdot \xi$. If there is *at least one* local confluence for every local peak ϕ, ψ then $\Rightarrow$-normal forms are valleys, and if there is *at most one* then $\Rightarrow$ has *random descent* [12, 14] meaning that if there exists a $\Rightarrow$-reduction to normal form, then all (maximal) $\Rightarrow$-reductions end in that normal form and all such $\Rightarrow$-reductions have the same length. Thus, since skolemising local confluence or the diamond property yields *exactly one* local confluence $C(\phi, \psi, \phi \setminus \psi, \phi / \psi)$ for every local peak ϕ, ψ, normal forms for the corresponding rules $\phi^{-1} \cdot \psi \Rightarrow (\phi \setminus \psi) \cdot (\phi / \psi)^{-1}$ are valleys and

unique (if they exist); cf. [19, Lem. 2].[1] Accordingly, we may extend $\backslash, /$ on peaks of steps to partial functions (with the same notations) on peaks ϕ, ψ of reductions by setting $\phi \setminus \psi := \chi$ and $\phi / \psi := \omega$ if $\chi \cdot \omega^{-1}$ is the $\Rightarrow$-normal form of $\phi^{-1} \cdot \psi$, and both $\phi \setminus \psi$ and ϕ / ψ to undefined otherwise. This preserves $\backslash$ and $/$ being each other's converse since $^{-1}$ is an involution:

Proposition 2. *$\backslash$ is the converse of $/$ on $\to$ iff the same holds for their extension to $\twoheadrightarrow$.*

Thus, if local confluence or the diamond property is expressed by means of a *single* residuation $|$ on peaks of steps, then so is (partially) its extension to peaks of reductions.

Partiality of extending residuation by tiling. The proviso in the definition of residuation by tiling, turning the extensions $\backslash, /$ into *partial* functions only, is needed as tiling need not terminate; an injudicious choice of residuations for steps may lead to their extension to reductions being partial, even if the rewrite system is confluent.[2] Still, a judicious choice then always *is* available (though non-computably so) due to completeness of *decreasing diagrams* [13, Prop. 2.3.28]:

Theorem 3. *For any countable confluent rewrite system there are residuations on peaks of steps that extend by tiling to residuations on peaks of reductions.*

By inspection of the proof of [13, Prop. 2.3.28], we see the constructed residuations to be each other's converse, so that a locally confluent countable rewrite system is confluent iff there exists a *single* residuation on peaks of steps that extends to a total residuation on peaks of reductions.

Although it is undecidable whether a locally confluent rewrite system $\to$ is confluent, cf. [6], various conditions sufficient for tiling to terminate are known [15]. For instance, if $\to$ is terminating then its local confluence entails termination of tiling by Newman's Lemma [12, Thm. 3], and if $\to$ has the diamond property then tiling is terminating by [12, Thm. 1].

Residuation for reductions by recursion. The following (left, right) unit and composition laws of *residual systems* [18, Sect. 8.7][12, 2, 9] are seen to hold *by tiling*; see Fig. 2:

$$\begin{array}{rcl} \phi / \varepsilon & = & \phi \\ \phi \setminus \varepsilon & = & \varepsilon \\ \phi / (\psi \cdot \chi) & \simeq & (\phi / \psi) / \chi \\ \phi \setminus (\psi \cdot \chi) & \simeq & (\phi \setminus \psi) \cdot ((\phi / \psi) \setminus \chi) \end{array} \qquad \begin{array}{rcl} \varepsilon \setminus \phi & = & \phi \\ \varepsilon / \phi & = & \varepsilon \\ (\phi \cdot \psi) \setminus \chi & \simeq & \psi \setminus (\phi \setminus \chi) \\ (\phi \cdot \psi) / \chi & \simeq & (\phi / \chi) \cdot (\psi / (\phi \setminus \chi)) \end{array}$$

where ϕ, ψ, χ range over reductions and where $\simeq$ is Kleene-equality expressing that either both sides denote and are equal, or that neither side denotes. The laws justify *defining* extend-

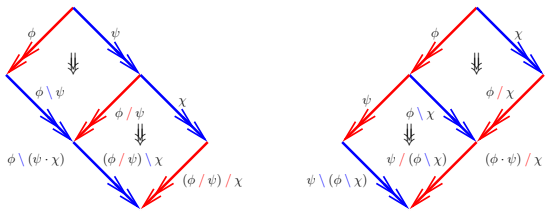


Figure 2: Composition laws for residuation

[1] Unlike [18, 4, 19] we do not assume that $\phi \setminus \phi = \varepsilon$ or $\phi / \phi = \varepsilon$ for a local peak ϕ, ϕ of identical steps.
[2] Seen e.g. by varying on Kleene's [18, Fig. 1.2] example of a locally confluent but not confluent system [12].

ing residuation from steps to reductions by *recursion*, having as base cases peaks where both reductions are steps or one of them is empty, and as recursive clauses:

$$\begin{array}{lcl}(\phi\cdot\psi)\,/\,(\chi\cdot\omega) & := & ((\phi\,/\,\chi)\cdot(\psi\,/\,(\phi\setminus\chi)))\,/\,\omega\\ (\phi\cdot\psi)\setminus(\chi\cdot\omega) & := & \psi\setminus((\phi\setminus\chi)\cdot((\phi\,/\,\chi)\setminus\omega))\end{array}$$

where ϕ,χ range over steps and ψ,ω over non-empty reductions, which turn into the following single recursive clause[3] in case of a single residuation $|$, i.e. if $\setminus$ is the converse of $/$:

$$(\phi\cdot\psi)\,|\,(\chi\cdot\omega) \quad := \quad ((\phi\,|\,\chi)\cdot(\psi\,|\,(\chi\,|\,\phi)))\,|\,\omega$$

It is justified by the above laws governing the interaction between residuation and composition: $(\phi\cdot\psi)\,|\,(\chi\cdot\omega)\simeq((\phi\cdot\psi)\,|\,\chi)\,|\,\omega\simeq((\phi\,|\,\chi)\cdot(\psi\,|\,(\chi\,|\,\phi)))\,|\,\omega$. Vice versa, since the laws give rise to a (this) tiling strategy, the recursive definition is the least (when representing functions as sets of pairs, ordering them by subset) extension of $|$ satisfying them; cf. [4, II Lem. 4.32].

From vertical to horizontal tiling. We show any diagram tiled top–down by confluences can be obtained by tiling left–right, now associating to a confluence $C(\phi,\psi,\chi,\omega)$ both a *vertical* and *horizontal* tiling rule, $\phi^{-1}\cdot\psi\Rightarrow_v\chi\cdot\omega^{-1}$ respectively $\phi\cdot\chi\Rightarrow_h\psi\cdot\omega$. The idea is then to

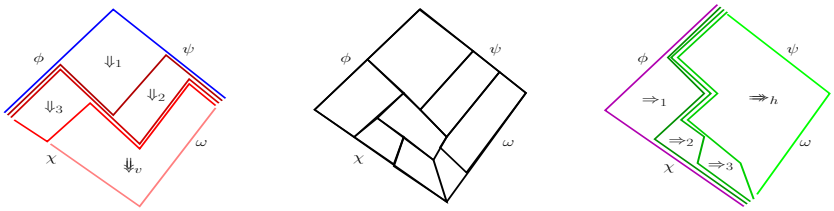


Figure 3: From vertical tiling (left) via tiled diagram (middle) to horizontal tiling (right)

reuse the same tiles, but from left to right instead of from top to bottom; Fig. 3.

Theorem 4. *If* $\phi^{-1}\cdot\psi\Rightarrow\!\!\!\!\Rightarrow_v\chi\cdot\omega^{-1}$ *then* $\phi\cdot\chi\Rightarrow\!\!\!\!\Rightarrow_h\psi\cdot\omega$*, for reductions* ϕ,ψ,χ,ω*.*

As Fig. 3 suggests, the proof can easily be adapted to show the numbers of vertical and horizontal tiles indeed to be the same, and to the case of *commutations* $C(\phi,\psi,\chi,\omega)$ where ϕ,ω are reductions of one rewrite system $\to$ and ψ,χ of another $\rightsquigarrow$; that conversely $\phi\cdot\chi\Rightarrow\!\!\!\!\Rightarrow_h\psi\cdot\omega$ entails $\phi^{-1}\cdot\psi\Rightarrow\!\!\!\!\Rightarrow_v\chi\cdot\omega^{-1}$ is then obtained by *symmetry* [16], for $\leftarrow$ and $\rightsquigarrow$. The theorem is simpler to state and prove, and holds for *global* confluences and commutations, compared to just for *local* confluences as in [4, II Sect. 4.2, Lem. 4.24, Prop. 4.34].

3-confluence. Call a rewrite system $\to$ (*locally*) 3-*confluent* if every 3-*peak* ϕ,ψ,χ of co-initial reductions (steps) can be completed by 9 reductions into a (*local*) *brick* as in Fig. 4 left. (If local 3-confluence holds using *steps*, $\to$ has the *cube* property.) If local confluence is given by a residuation $|$ that extends to a total residuation on reductions, local 3-confluence may fail due to failure of the *cube* law [11] $(\varsigma\,|\,\zeta)\,|\,(\xi\,|\,\zeta)=(\varsigma\,|\,\xi)\,|\,(\zeta\,|\,\xi)$. This failure is well-known (since at least the 90s) for systems such as positive braids, TRSs and the $\lambda\beta$-calculus; cf. [16]. (Even if the diamond property holds, the cube property may fail; cf. gadget qp2 in Fig. 5.) We

[3] In Haskell: `resred (i:u) (j:v) = resred ((resstp i j)++(resred u (resstp j i))) v`, where `resstp` is the given residuation on steps and `resred` its extension to reductions, represented as lists of steps.

give ways to extend local 3-confluence to 3-confluence by *bricklaying*. *Decreasing diagrams* [13] (DD) is one way, when defining a brick to be 3-*decreasing* if its 6 *faces* are decreasing. This is shown by induction, measuring a 3-peak by the multiset sum of the *lexicographic maximum* measures [13] of the 3 reductions in it, with the decrease of measure from the 3-peak $\phi, \psi \cdot \bar{\psi}, \chi$ to $\phi', \bar{\psi}, \chi'$ visualised on the right in Fig. 4, and the induction step(s) on the left.

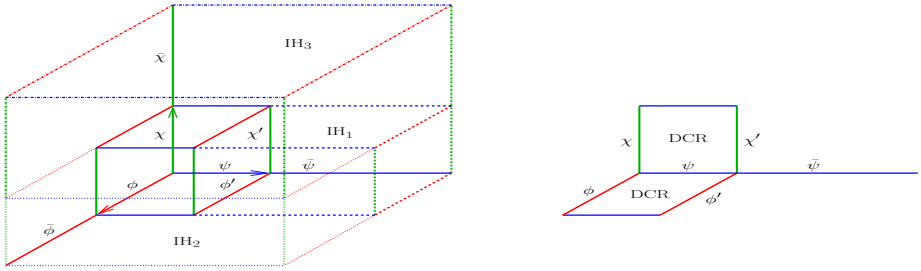


Figure 4: Induction (left) and decrease in measure (right) of 3-decreasingness by bricklaying

Theorem 5 (3-DD)**.** *A locally* 3*-decreasing rewrite system is* 3*-decreasing; cf. [13, Thm. 2.3.20].*

As corollaries we obtain that systems that are terminating and locally 3-confluent, or that have the cube property, are 3-confluent. Using that DD is complete for countable confluent systems, cf. [13, Prop. 2.3.28], we even have that *any* such system is 3-confluent, if we are free to *choose* residuation to 'go to' the *least common reduct* on a chosen spanning forest [5, 8].

Lemma 6. *A countable confluent rewrite system is locally* 3*-decreasing for some residuation.*

Bricklaying. Analogously to how tiling a peak of reductions with local confluences turns it into a valley, *bricklaying* with local bricks is a way to turn a 3-*peak* into a 3-*valley*. Also analogously: an injudicious choice of local bricks may lead to non-termination of bricklaying (even if the system is 3-confluent). We define bricklaying in an attempt to make sense of that. Whereas (2D) tiling has (1D) *conversions* as intermediate stages, we introduce (2D) *beds* as intermediate stages of (3D) bricklaying. As a first approximation to beds we use graphs having (red, blue, green) *coloured* edges to model steps in 3 dimensions, which we then embed. (Hypermaps as in [7] could be a suitable alternative; we leave this to future research.)

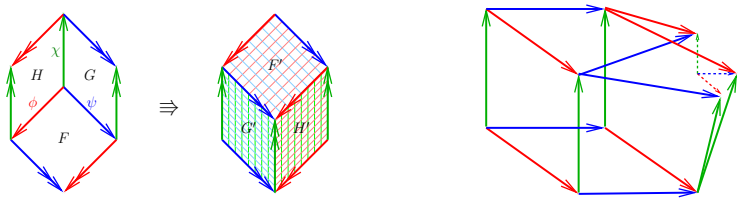


Figure 5: Bricklaying (left) and gadget qp2 (right; dashed arrows indicate recursive nature)

Definition 7. A *bed*-dag for a rewrite system $\to$ is a finite connected dag having nodes labelled by objects of $\to$ and edges having a unique colour and labelled by steps of $\to$ such that the label of the source (target) of an edge is the source (target) of its label, satisfying: (i) the dag is the union of its tiles, where a *tile* is a red–blue (blue–green, green–red) *tetragonal* cycle

$\leftarrow^+ \cdot \leftarrow \cdot \rightarrow \cdot \rightarrow^+$ such that interior nodes of its red (blue) valley path don't have blue (red) in-edges, (ii) a node has at most 1 in-edge and at most 1 out-edge of a given colour; (iii) there are no paths having edges of all 3 colours; (iv) every $\rightarrow\cdot\rightarrow$ (consecutive edges of distinct colours) belongs to some tile; and (v) the source of the edges of a local peak $\leftarrow \cdot \rightarrow$ (edges of distinct colours) not belonging to a tile, has an in-edge of the 3rd colour (here green).

A bed-dag is a *bed* if nodes are elements of $\mathbb{R}^3$ and a red (blue, green) edge is a pair of nodes such that only its 1st (2nd, 3rd) coordinates differ, with the target greater than the source.

The *boundary* of a dag is its subgraph of edges belonging to exactly one tile. A *bricklaying* step $\Rrightarrow$ replaces a bed-dag L by another R, fresh except for having the same *hexagonal* cycle $\leftarrow^+ \cdot \rightarrow^+ \cdot \leftarrow^+ \cdot \rightarrow^+ \cdot \leftarrow^+ \cdot \rightarrow^+$ as boundary, where L is the union of 3 tiles pairwise sharing only one peak edge, a *redex*, and R is without nodes of out-degree 3, a 3-*valley*; see Fig. 5 left.

We designed bed-dags such that they are preserved by $\Rrightarrow$, and beds such that for *any* node of out-degree 3, its 3 peaks lie in orthogonal planes, giving a 3-peak having the valleys as boundary.

Definition 8 (bricklaying)**.** Assuming local 3-confluence for a (total) residuation $\mid$ on steps, proceed in III phases: (I) We reify[4] the empty reduction as the *empty* step ε, update residuation by setting $\phi \mid \psi$ to the *step* ε if it was the empty *reduction*, and set $\varepsilon \mid \phi := \varepsilon$ and $\phi \mid \varepsilon := \phi$; (II) We represent a 3-peak as a *flat*bed, a bed that need not satisfy condition (v), in the way illustrated in Fig. 6, and associate to each tiling step $\phi^{-1} \cdot \psi \Rightarrow (\psi \mid \phi) \cdot (\phi \mid \psi)^{-1}$ a *flat* bricklaying step replacing a subgraph with boundary $\phi \cdot \varepsilon \cdot \phi^{-1} \cdot \psi \cdot \varepsilon^{-1} \cdot \psi^{-1}$ by a tile with boundary $(\psi \mid \phi)^{-1} \cdot \varepsilon \cdot \phi^{-1} \cdot \psi \cdot \varepsilon^{-1} \cdot (\phi \mid \psi)$, *scaling* it to make it fit in the bed. For reductions ϕ, ψ then $\phi^{-1} \cdot \psi \twoheadrightarrow (\psi \mid \phi) \cdot (\phi \mid \psi)$ iff flat bricklaying terminates for the 3-peak ϕ, ψ, ε (where ε is the empty reduction, not the empty step) in a bed, with inside it a 3-*valley* with tetragonal boundary $(\psi \mid \phi)^{-1} \cdot \phi^{-1} \cdot \psi \cdot (\phi \mid \psi)$. This allows to produce 3-valleys for each of the 6 faces F, G, H, F', G', H' of a bricklaying step $\Rrightarrow$ for 3-peak of steps ϕ, ψ, χ as in Fig. 5; (III) A lhs L for local 3-peak ϕ, ψ, χ is the bed obtained by pasting F, G, H along theses 3 shared steps.[5] The rhs R is the 3-valley obtained by pasting the red–blue F', blue–green G' and green–red H' 3-valleys along their 'shared' reductions; *caveat*: in general these are *not* shared due to having reified empty steps, as illustrated for the reduction 'shared' between F' and H' on the right in Fig. 6. We overcome this by a process we dub *buttering*, *refining* both 3-valleys by inserting εs in tiles, and *rescaling* the opposite reductions (with a knock-on effect through the bed).[6]

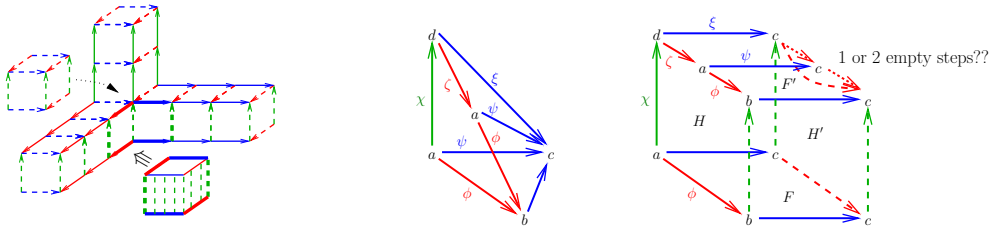


Figure 6: Embedding a conversion as a flatbed using tiles with empty (dashed) steps (left), and reifying empty reductions in a local 3-confluence may break it (right)

Just like tiling, bricklaying has random descent if $\Rrightarrow$ is deterministic (if the three faces of a 3-peak determine the rhs), hence the bricklaying *strategy* implicit in the proof of Thm. 5 (Fig. 4

[4] An idea due to Klop to keep confluence diagrams *rectangular* in drawings. It was given a *description but not a definition* in [10]. Our *buttering* keeps 3-confluence diagrams rectangular in drawings; cf. Fig. 6 right.

[5] *Caveat*: we get different Ls for different local confluences *with empty steps* for the 3 local peaks of steps.

[6] This *is* possible; in the example we insert 1 ε in H'; as the opposite side is 1 (ε-)step no rescaling is needed.

left) being terminating in the decreasing case, then entails termination of bricklaying, yielding a residuation satisfying the cube law *per construction*; residuals can be read off from the bed.

Conclusion We have identified residuation as skolemised confluence, and studied computing residuation via tiling of local peaks in 2D yielding valleys, and via bricklaying of local bricks in 3D yielding 3-valleys, giving confluence respectively 3-confluence (such that Lévy's cube law holds). We introduced 3-decreasingness as a way to show 3-confluence by means of local bricks.

References

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

[2] H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, 1984.

[3] A. Church and J.B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39:472–482, 1936. https://doi.org/10.1090/S0002-9947-1936-1501858-0.

[4] P. Dehornoy and alii. *Foundations of Garside Theory*. European Mathematical Society, 2015.

[5] J. Endrullis, J.W. Klop, and R. Overbeek. Decreasing diagrams with two labels are complete for confluence of countable systems. In *3rd FSCD*, volume 108 of *LIPIcs*, pages 14:1–14:15, 2018. https://doi.org/10.4230/LIPIcs.FSCD.2018.14.

[6] A. Geser, A. Middeldorp, E. Ohlebusch, and H. Zantema. Relative undecidability in term rewriting: II. the confluence hierarchy. *Information and Computation*, 178(1):132–148, 2002. https://doi.org/10.1006/inco.2002.3150.

[7] G. Gonthier. Formal proof – the four-color theorem. *Notices of the American Mathematical Society*, 55(11):1382–1394, 2008. https://www.ams.org/journals/notices/200811/tx081101382p.pdf.

[8] N. Hirokawa, J. Nagele, V. van Oostrom, and M. Oyamaguchi. Confluence by critical pair analysis revisited. In *CADE 27*, volume 11716 of *LNCS*, pages 319–336. Springer, 2019. https://doi.org/10.1007/978-3-030-29436-6_19.

[9] G. Huet and J.-J. Lévy. Computations in orthogonal rewriting systems, Part I + II. In *Computational Logic – Essays in Honor of Alan Robinson*, pages 395–443. MIT Press, 1991.

[10] J.W. Klop. *Combinatory Reduction Systems*. PhD thesis, Rijksuniversiteit Utrecht, 1980.

[11] J.-J. Lévy. *Réductions correctes et optimales dans le λ-calcul*. Thèse de doctorat d'état, Université Paris VII, 1978.

[12] M.H.A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43:223–243, 1942. https://doi.org/10.2307/1968867.

[13] V. van Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, Vrije Universiteit Amsterdam, 1994. https://research.vu.nl/en/publications/confluence-for-abstract-and-higher-order-rewriting.

[14] V. van Oostrom. Random descent. In *RTA*, volume 4533 of *LNCS*, pages 314–328. Springer, 2007. https://doi.org/10.1007/978-3-540-73449-9_24.

[15] V. van Oostrom. Confluence by decreasing diagrams; converted. In *RTA*, volume 5117 of *LNCS*, pages 306–320. Springer, 2008. https://doi.org/10.1007/978-3-540-70590-1_21.

[16] V. van Oostrom. Some symmetries of commutation diamonds. In *9th IWC*, pages 1–7, 2020. http://iwc2020.cic.unb.br/iwc2020_proceedings.pdf.

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

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

[19] H. Zantema and V. van Oostrom. The paint pot problem and common multiples in monoids. *Applicable Algebra in Engineering, Communication and Computing*, page 23, 2023. https://doi.org/10.1007/s00200-023-00613-6.

Confluence of a Computational Lambda Calculus for Higher-Order Relational Queries

Claudio Sacerdoti Coen and Riccardo Treglia

Università di Bologna, Bologna, Italy
claudio.sacerdoticoen@unibo.it
riccardo.treglia@unibo.it

Abstract

We study the operational semantics of an untyped computational lambda-calculus whose normal forms represent queries on databases. The calculus extends the computational core with additional operations and rewriting rules whose effect is to turn the monadic type of computations into a multiset monad that capture tables. Moreover, we introduce comonadic constructs and additional rewriting rules to be able to form tables of tables. Proving confluence becomes tricky: we succeed exploiting decreasing diagrams.

1 Introduction to the Calculus: Syntax and Reduction Relation

The second author et al. have introduced and studied in [dLT20, FGdLT22] the computational core $\lambda_{\copyright}$, a λ-calculus inspired by Moggi's computational one [Mog89, Mog91]. The calculus differentiates between values and computations, the latter obtained via return/bind constructs for a generic monad. The strong operational semantics is obtained simply orienting the monadic laws, and confluence was proved among other properties.

In this work, we extend $\lambda_{\copyright}$ with specific additional operations and rewriting rules over computations that turn the generic monad into a multiset monad: the 0-ary operation $\emptyset$ represents the empty multiset, $\uplus$ the union of multisets, and the monadic return, denoted by $[V]$, is now interpreted as forming a singleton. The rewriting rules partially capture the algebraicity of the operations in the sense of Plotking and Power [PP02, PP03] by letting the operators commute with those rewriting contexts that are built from bind operators, only. Because in $\lambda_{\copyright}$, contrary to Moggi's computational λ-calculus, values and computations are rigidly split, the extension obtained so far does not allow formation of multisets of multisets, because multisets are not values. To overcome the issue, we add two more co-monadic constructs to reflect computations into values, following ideas by [Fil94]. These constructs are the thunk/force constructs of Levy's call-by-push-value [Lev99]; however, our calculus is strong, i.e. it allows reduction inside values as well. Finally, we introduce an equational theory over computations to capture associativity and commutativity of $\uplus$ and idempotency of $\emptyset$: this is the minimal theory that turns the calculus into a confluent one.

The exact choice of rewriting and equational rules that we pick seems rather arbitrary at first: the empty set is not the neutral element of $\uplus$ and the monadic operations are not forced to be completely algebraic (e.g. $\uplus$ does not commute with contexts that include thunks or force). It is to the (untyped) NRCλ calculus [RC20] as $\lambda_{\copyright}$ is to the (untyped) λ-calculus, and indeed we are introducing it with the intent of studying semantic properties of the NRCλ-calculus via intersection types, trying to scale what the second author already did for $\lambda_{\copyright}$. The NRCλ calculus is an example of nested, higher-order relational calculus that provides a principled foundation for integrating database queries into programming languages. In NRCλ, a database

table is represented by the multiset of its rows, where each row is just a value (NRCλ has tuples). The main properties of the calculus are that it is confluent and strongly normalizing and, moreover, some normal forms can be directly interpreted as SQL queries (those such that the types of the free variables and of the result are just tables of base types and not tables of tables). In particular, the set of rewriting and equational rules that our calculus inherits from the NRCλ-calculus is the minimal set that grants the previous properties.

Because of the important application to database, from now on we call our extension of the $\lambda_{\copyright}$ calculus the λ_{SQL} calculus.

Contributions The first contribution of the work is the design of the λ_{SQL} calculus, which goes beyond the mere effort to fit the NRCλ into a well-assessed monadic frame. Indeed, this can be considered as an experiment of extending $\lambda_{\copyright}$ with algebraic operators (other cases are [dT21, AKR23]), but here it immediately highlights, for example, the need to introduce other kind of constructs, such as the comonadic unit, that could be added to $\lambda_{\copyright}$ independently of the algebraic operators.

The second contribution is the proof of a fundamental property of the calculus: confluence. The proof is labour-intensive because the rewriting rules associated to algebraicity of the operators turn them into control operators: each operator can capture its context and then erase or duplicate it, and many critical pairs arise. Moreover, there is also the issue of the interplay between the equational theory and the rewriting theory. Technically, we make strong use of von Oostrom's decreasing diagram technique [vO94], the most difficult point of which is to find the order relation between the labels of the calculus reduction rule. This will be done by considering orthogonal and nested closures of certain reduction rules, inspired by the work in [ADJL17], postponing in a final step the commutation with respect to the union operator.

Long-term perspectives Our long-term goal is to extract from the confuence proof based on decreasing diagrams an order over reduction rules to design a well-behaved normalizing strategy. We will then define an appropriate intersection type system based on tight multi-types [AGK20] to capture quantitatively the set of terminating queries according to that strategy, the length of their reduction and the size of the normal forms, i.e. the size of the computed SQL queries. Ultimately we want to capture even more quantitative information over the queries itself.

Syntax and Reduction The syntax of the untyped computational SQL λ-calculus, shortly λ_{SQL}, and its reduction relation are reported below:

Definition 1.1 (Term syntax)**.**

$$\begin{array}{rrcl} \mathit{Val}: & V, W & ::= & x \mid \lambda x.M \mid \langle\!\langle M \rangle\!\rangle \\ \mathit{Com}: & M, N & ::= & [V] \mid M \star V \mid M \uplus M \mid \emptyset \mid\, !V \end{array}$$

Like in $\lambda_{\copyright}$, terms are of either sorts *Val* and *Com*, representing values and computations, respectively. Variables x, abstractions $\lambda x.M$ — where x is bound in M — and the constructors $[V]$ and $M \star V$, written `return` V and M `>>=` V in Haskell-like syntax, respectively, form the syntax of $\lambda_{\copyright}$, which is agnostic on the interpretation of computations. In λ_{SQL}, instead, computations are meant to be understood as tables, i.e. multisets of values, and therefore $[V]$ is interpreted as the singleton whose only element is V and $\star$ as the bind operator of the list monad. The binary and 0-ary operators $\uplus$ and $\emptyset$ are additionally used to construct tables. The pair of constructs $\langle\!\langle \cdot \rangle\!\rangle$ and ! are used to reflect computations into labels, allowing to form tables of (reflected) tables. Note that $\langle\!\langle \cdot \rangle\!\rangle$ can be understood as the unit of a comonad. Terms are identified up to renaming of bound variables so that the capture avoiding substitution $M\{V/x\}$ is always well defined; $FV(M)$ denotes the set of free variables in M. Finally, like in $\lambda_{\copyright}$, application among computations can be encoded by $MN \equiv M \star (\lambda z.\ N \star z)$, where z is fresh.

Wrapping up, the syntax can be condensed in the motto:

$$\lambda_{\text{SQL}} \approx \lambda_{\copyright} + \text{operations over tables} + \text{monadic reification/reflection}$$

with the latter extension being orthogonal to the second one.

We are now in place to introduce the λ_{SQL} **reduction relation**, later closed under contexts:

Definition 1.2 (Reduction)**.** *The reduction relation is the union of the following binary relations over Com:*

$$\begin{array}{rrll}
\beta_c) & [V] \star \lambda x.M & \mapsto_{\beta_c} & M\{V/x\} \\
\sigma) & (L \star \lambda x.M) \star \lambda y.N & \mapsto_{\sigma} & L \star \lambda x.(M \star \lambda y.N) \qquad \textit{for } x \notin \mathsf{fv}(N) \\
\uplus_1) & (M \uplus N) \star \lambda x.P & \mapsto_{\uplus_1} & (M \star \lambda x.P) \uplus (N \star \lambda x.P) \\
\uplus_2) & M \star \lambda x.(N \uplus P) & \mapsto_{\uplus_2} & (M \star \lambda x.N) \uplus (M \star \lambda x.P) \\
\emptyset_1) & \emptyset \star \lambda x.M & \mapsto_{\emptyset_1} & \emptyset \\
\emptyset_2) & M \star \lambda x.\emptyset & \mapsto_{\emptyset_2} & \emptyset \\
!) & !\langle\!\langle M \rangle\!\rangle & \mapsto_{!} & M
\end{array}$$

The first two rules, taken from $\lambda_{\copyright}$, are oriented monadic equations. The next four rules capture algebraicity of the $\uplus$ operator, but only w.r.t. contexts made of $\star$ only (e.g. there is no rule $(M \uplus N) \uplus P \mapsto (M \uplus P) \uplus (N \uplus P)$ because that would be unsound for tables). The latter rule is the usual rule for the thunk/force redex in call-by-push-value.

The *reduction* $\rightarrow_{\lambda_{\text{SQL}}}$ (when it is clear from the context we omit the subscript) is the contextual closure of λ_{SQL} under *computational contexts*, where such contexts are mutually defined with values contexts as follows:

$$\begin{array}{lll}
\mathsf{V} ::= \langle\cdot_{Val}\rangle \mid \lambda x.\mathsf{C} \mid \langle\!\langle \mathsf{C} \rangle\!\rangle & & \text{Value Contexts} \\
\mathsf{C} ::= \langle\cdot_{Com}\rangle \mid [\mathsf{V}] \mid \mathsf{C} \star V \mid M \star \mathsf{V} \mid \mathsf{C} \uplus M \mid M \uplus \mathsf{C} \mid !\mathsf{V} & & \text{Computation Contexts}
\end{array}$$

Notice that the hole of each kind of context has to be filled in with a proper kind of term.

We equip the calculus with a sound, but not complete, equational theory for multisets, taken from [RC20].

Definition 1.3 (Equational theory E)**.**

$$\begin{array}{rrcl}
Comm) & M \uplus N & = & N \uplus M \\
Empty) & \emptyset \uplus \emptyset & = & \emptyset
\end{array}$$

2 Route to Confluence

We modularize the proof of confluence by first showing that the equational part can be postponed.

Getting rid of the equational theory. A classic tool to modularize a proof of confluence is Hindley-Rosen lemma, stating that the union of confluent reductions is itself confluent if they all commute with each other. Let us first define what commutation between a reduction relation and an equational theory means, and then state that result properly.

Definition 2.1. *Given a reduction relation $\rightarrow$ and an equational theory $=_E$, we say that $\rightarrow$ commutes over $=_E$ if for all M, N, L such that $M =_E N \rightarrow L$, there exists P such that $M \rightarrow P =_E L$.*

Lemma 2.2 (Hindley-Rosen)**.** *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be relations on the set A. If $\mathcal{R}_1$ and $\mathcal{R}_2$ are confluent and commute with each other, then $\mathcal{R}_1 \cup \mathcal{R}_2$ is confluent.*

We will exploit that to focus just on the reduction relation while proving confluence.

Lemma 2.3. *$=_E$ commutes with $\to$.*

Hence, by Lemma 2.3 one needs just the confluence of $\to$ to assert the confluence of $\to$ modulo E.

Decreasing diagram. Now that is possible to omit the equational theory induced by Definition 1.3, we need to prove the commutation of all the reduction rules, and in this intent we use decreasing diagrams by van Oostrom [vO94, vO08]. This is a powerful and general tool to establish commutation properties, which reduces the problem of showing commutation to a local test; in exchange of localization, the diagrams need to be decreasing with respect to some labelling.

Definition 2.4 (Decreasing, [vO94])**.** *An rewriting relation $\mathcal{R}$ is* locally decreasing *if there exist a presentation $(R, \{\to_i\}_{i\in I})$ of $\mathcal{R}$ and a well-founded strict order $>$ on I such that:*

$$\underset{i}{\leftarrow} \cdot \underset{j}{\rightarrow} \subseteq \overset{*}{\underset{\vee i}{\longleftrightarrow}} \cdot \overset{=}{\underset{j}{\longrightarrow}} \cdot \overset{*}{\underset{\vee\{ij\}}{\longleftrightarrow}} \cdot \overset{=}{\underset{i}{\longleftarrow}} \cdot \overset{*}{\underset{\vee j}{\longleftrightarrow}},$$

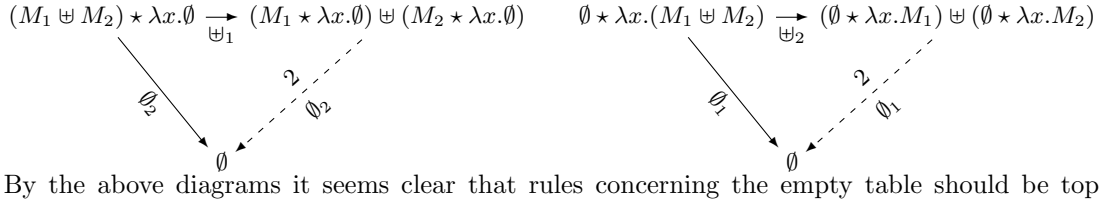
where $\vee\bar{I} = \{i \in I \mid \exists k \in \bar{I}.\ k > i\}$, $\vee i$ abbreviates $\vee\{i\}$, and $\overset{*}{\to}$ (resp. $\overset{*}{\longleftrightarrow}$) and $\overset{=}{\to}$ (resp. $\overset{=}{\longleftrightarrow}$) are the transitive and reflexive closures of the relation $\to$ (resp. $\leftrightarrow$).

Let us give a hint the above definition. The property of decreasiness is stated for a relations, seen as a family of labelled binary relations. Such labels are equipped with a well-founded, strict, order such that every *peak* can be rejoined in a particular way, regulated by that specific order on labels.

The following theorem, due to van Oostrom, states that decreasiness implies confluence.

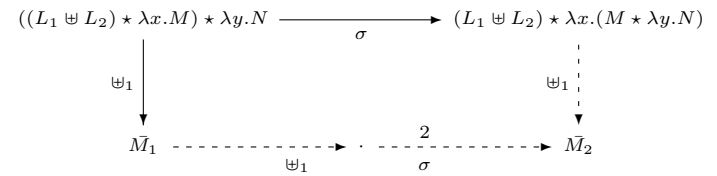
Theorem 2.5 (van Oostrom [vO94, vO08])**.** *Every locally decreasing rewriting relation $\mathcal{R}$ is confluent.*

Which order? Now the point is to find a proper labelling and a strict order on that labelling that satisfies the property of decreasiness. Let's start with harmless reductions, involving rules of union and empty.



By the above diagrams it seems clear that rules concerning the empty table should be top elements of the labelling we are searching for.

When it comes to comparing $\uplus_1$ vs. σ, the situation is a bit tricker because $\uplus_1$ only quasi-commutes over σ. The following diagrams shows that $\uplus_1$ must be made greater than σ.



where $\bar{M_1} = ((L_1 \star \lambda x.M) \uplus (L_2 \star \lambda x.M)) \star \lambda y.N$, $\bar{M_2} = (L_1 \star \lambda x.(M \star \lambda y.N)) \uplus (L_2 \star \lambda x.(M \star \lambda y.N))$

The case for β_c vs $\uplus_2$, however, shows the need for a non-trivial approach, since depending in which context the rules are applied, we need either $\beta_c > \uplus_2$ or $\beta_c < \uplus_2$. Indeed,

$$\begin{array}{ccc} [V] \star \lambda x.(N \uplus P) & \xrightarrow{\beta_c} & (N \uplus P)\{V/x\} \\ \downarrow{\scriptstyle \uplus_2} & & \| \\ ([V] \star \lambda x.N) \uplus ([V] \star \lambda x.P) & \xrightarrow[2]{\beta_c} & N\{V/x\} \uplus P\{V/x\} \end{array}$$

... but ...

$V_1 = \lambda x.(M \star \lambda y.(N_1 \uplus N_2))$
$V_2 = \lambda x.((M \star \lambda y.N_1) \uplus (M \star \lambda y.N_2))$

$$\begin{array}{ccc} [V_1] \star \lambda z.([z] \star z) & \xrightarrow{\uplus_2} & [V_2] \star \lambda z.([z] \star z) \\ \downarrow{\scriptstyle \beta_c} & & \vdots{\scriptstyle \beta_c} \\ [V_1] \star V_1 & \xrightarrow[2]{\uplus_2} & [V_2] \star V_2 \end{array}$$

Generalized version of unions and Multi-reduction. Before stating the main result we have to introduce two new notion of reduction that will lead to a right labelling order to prove the decreasiness. The first one is a *generalized* version of rules $\uplus_1$ and $\uplus_2$, taking into account not just union symbols in the scope of the rule one by one, but all together.

Definition 2.6 (Generalized union step)**.** *Let us define as* generalized $\uplus_1$ *and* $\uplus_2$ *steps, notation* $\boldsymbol{Gen}\uplus_1$ *and* $\boldsymbol{Gen}\uplus_2$*, as follows*

$$\begin{array}{lll} \boldsymbol{Gen}\uplus_1) & (\ldots(M_1 \uplus M_2) \uplus \ldots \uplus M_n) \star \lambda x.N & \mapsto_{\boldsymbol{Gen}\uplus_1} \quad (M \star \lambda x.N) \uplus (M_2 \star \lambda x.N) \uplus \ldots \uplus (M_n \star \lambda x.N) \\ \boldsymbol{Gen}\uplus_2) & M \star \lambda x.(\ldots(N_1 \uplus N_2) \uplus \ldots \uplus N_n) & \mapsto_{\boldsymbol{Gen}\uplus_2} \quad (M \star \lambda x.N_1) \uplus (M \star \lambda x.N_2) \uplus \ldots \uplus (M \star \lambda x.N_n) \end{array}$$

Second, the confluence proof we are going to sketch avoids the issue with β_c vs. $\uplus_2$ reported above by considering *multiple reductions*. Roughly speaking, this means that we consider a labelling that comprehends reduction rules that can perform simultaneously in many 'part' of the term, called formally *positions*. For a fair formalization of these basic notions of rewriting theory, please see, *e.g.*, [BN98].

A **parallel rewrite step** is a sequence of reductions at a set P of **parallel** positions, ensuring that the result does not depend upon a particular sequentialization of P. Given a reduction step γ we define its parallel version as $\mathbf{Par}\gamma$.

We are now ready to state our main result:

Theorem 2.7 (Confluence)**.** λ_{SQL} *is confluent.*

Proof sketch.

1. All reduction rules strongly commute with !: proved by tedious inspection of all cases.
2. Under the following order for parallel rewriting steps, all remaining rules are decreasing as well: also proved by tedious inspection of all cases.

$$\mathbf{Par}\beta_c > \mathbf{Par}\sigma > \mathbf{ParGen}\uplus_2 > \mathbf{ParGen}\uplus_2 > \emptyset_1 > \emptyset_2$$

   The diagrams for the cases $\mathbf{Par}\uplus_1$ vs $\mathbf{Par}\uplus_2$ and $\mathbf{Par}\uplus_2$ vs $\emptyset_1$ only hold up to E. E.g., $\emptyset \; {}_{\emptyset_1}\!\leftarrow \emptyset \star \lambda x.M \uplus N \rightarrow_{\uplus_2} \rightarrow^2_{\emptyset_1} \emptyset \uplus \emptyset$.
3. Confluence is obtained combining the previous points with Lemma 2.3 and Theorem 2.5, following [ADJL17].

□

References

[ADJL17] Ali Assaf, Gilles Dowek, Jean-Pierre Jouannaud, and Jiaxiang Liu. Untyped Confluence In Dependent Type Theories. working paper or preprint, April 2017.

[AGK20] Beniamino Accattoli, Stéphane Graham-Lengrand, and Delia Kesner. Tight typings and split bounds, fully developed. *J. Funct. Program.*, 30:e14, 2020.

[AKR23] Sandra Alves, Delia Kesner, and Miguel Ramos. Quantitative global memory. arXiv:2303.08940, 2023.

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998.

[dLT20] U. de' Liguoro and R. Treglia. The untyped computational λ-calculus and its intersection type discipline. *Theor. Comput. Sci.*, 846:141–159, 2020.

[dT21] Ugo de'Liguoro and Riccardo Treglia. Intersection types for a λ-calculus with global store. In Niccolò Veltri, Nick Benton, and Silvia Ghilezan, editors, *PPDP 2021: 23rd International Symposium on Principles and Practice of Declarative Programming, Tallinn, Estonia, September 6-8, 2021*, pages 5:1–5:11. ACM, 2021.

[FGdLT22] Claudia Faggian, Giulio Guerrieri, Ugo de' Liguoro, and Riccardo Treglia. On reduction and normalization in the computational core. *Mathematical Structures in Computer Science*, 32(7):934–981, 2022.

[Fil94] A. Filinski. Representing monads. In *Conference Record of POPL'94: 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 446–457. ACM Press, 1994.

[Lev99] Paul Blain Levy. Call-by-push-value: A subsuming paradigm. In *Typed Lambda Calculi and Applications, 4th International Conference (TLCA'99)*, volume 1581 of *Lecture Notes in Computer Science*, pages 228–242, 1999.

[Mog89] E. Moggi. Computational lambda-calculus and monads. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89)*, pages 14–23. IEEE Computer Society, 1989.

[Mog91] E. Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, 1991.

[PP02] G. D. Plotkin and J. Power. Notions of computation determine monads. In *FOSSACS 2002*, volume 2303 of *Lecture Notes in Computer Science*, pages 342–356. Springer, 2002.

[PP03] G. D. Plotkin and J. Power. Algebraic operations and generic effects. *Appl. Categorical Struct.*, 11(1):69–94, 2003.

[RC20] Wilmer Ricciotti and James Cheney. Strongly normalizing higher-order relational queries. In *5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France (Virtual Conference)*, pages 28:1–28:22, 2020.

[Ter03] Terese. *Term rewriting systems*, volume 55 of *Cambridge tracts in theoretical computer science.* Cambridge University Press, 2003.

[vO94] Vincent van Oostrom. Confluence by decreasing diagrams. *Theor. Comput. Sci.*, 126(2):259–280, 1994.

[vO08] Vincent van Oostrom. Confluence by decreasing diagrams converted. In *Rewriting Techniques and Applications, 19th International Conference, RTA 2008,*, volume 5117 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2008.

A New Format for Rewrite Systems*

Takahito Aoto[1], Nao Hirokawa[2], Dohan Kim[3], Misaki Kojima[4], Aart Middeldorp[3], Fabian Mitterwallner[3], Naoki Nishida[4], Teppei Saito[2], Jonas Schöpf[3], Kiraku Shintani[2], René Thiemann[3], and Akihisa Yamada[5]

[1] Faculty of Engineering, Niigata University, Japan
[2] School of Information Science, JAIST, Japan
[3] Department of Computer Science, University of Innsbruck, Austria
[4] Department of Computing and Software Systems, Nagoya University, Japan
[5] National Institute of Advanced Industrial Science and Technology, Japan

Abstract

We propose a new format for a variety of rewrite systems, to replace the current COPS format used in the Confluence Competition. We include a proposal for logically constrained rewrite system, to prepare for a future competition category.

1 Introduction

The Confluence Competition (CoCo) [5][1] is an annual competition in which software tools try to solve confluence-related problems for a variety of rewrite formalisms. Problems in CoCo are selected from COPS [3],[2] an online database for confluence and related properties in term rewriting.

In the *basic* COPS format the rewrite rules of the problem at hand are specified and the variables are declared, but the function symbols are left implicit. This design decision goes back to the Termination and Complexity Competition (termCOMP) [1][3] and is based on the assumption that the property that needs to be (dis)proved is not affected by additional function symbols. This is true for most CoCo and termCOMP categories, but not for all. For instance, ground-confluence is well-known not to be preserved under signature extension. Since the corresponding GCR category employs the *many-sorted* COPS format, in which every function symbol is specified using a sort declaration, this causes no problem. Other properties that are known not to be closed under signature extension include NFP and UNR [4]. Also, termination under outermost strategies (corresponding to the termCOMP category TRS Outermost) is not preserved under signature extension [2]. For the corresponding categories in CoCo, besides the basic format, problems may be specified in the *extended* COPS format, which allows the declaration of additional function symbols.

We propose a change in the COPS format in which function symbols that can be used to construct terms must be listed, assuming that we have infinitely many variables (of any type) at our disposal, which is the usual convention in the term rewriting literature. In this paper we propose new formats for term rewrite systems (TRSs), conditional term rewrite systems (CTRSs), context-sensitive term rewrite systems (CSTRSs), a combination of the latter two (CSCTRSs), many-sorted term rewrite systems (MSTRSs), and logically constrained term rewrite systems (LCTRSs). For the latter, no previous format is known. We also describe how

*This research was funded by the Austrian Science Fund (FWF) project I5943 and JSPS-FWF JPJSBP120222001.

[1] http://project-coco.uibk.ac.at/
[2] https://cops.uibk.ac.at/
[3] http://termination-portal.org/wiki/Termination_Competition

multiple TRSs are represented. The new format specifies rewrite systems only, thus strategies like innermost and outermost are not part of the format. We take the stance that these should be part of competition categories, to avoid unnecessary duplication in databases.

2 Format

The following code represents the TRS $\{\mathsf{F}(x,x) \to \mathsf{A}, \mathsf{G}(x) \to \mathsf{F}(x,\mathsf{G}(x)), \mathsf{C} \to \mathsf{G}(\mathsf{C})\}$ in our new syntax:

```
; @author Takahito Aoto
; @author Junichi Yoshida
; @author Yoshihito Toyama
; @doi 10.1145/322217.322230
; p. 813, attributed to Barendregt

(format TRS)
(fun F 2)
(fun A 0)
(fun G 1)
(fun C 0)
(rule (F x x) A)
(rule (G x) (F x (G x)))
(rule C (G C))
```

The new format adopts a Lisp/Scheme-like syntax. All specifications are written in the form of S-expressions. A semicolon (`;`) indicates a line comment. The text between a semicolon and the end of the line is regarded as a comment. A line starting with `; @` indicates meta-information like information about authors and references.

All formats have an optional *meta-info* header and a *content* part. The *meta-info* part consists of line comments following a specific format, which are used to embed metadata in a file. They resemble key-value pairs, where the key is some arbitrary string not containing white space and the value is a string not containing newlines. In a meta-info line the key is written using a leading `@` symbol, separated from the value by a single space. For example in the line

```
; @author John Smith
```

the key `author` is followed by the value `John Smith`. Such metadata is used to attribute the problem to one or more authors or contributors, or cite the literature where the system first appeared. It is also easily extensible for future uses, for example to indicate that a specific property (confluence, termination, ...) is satisfied. The *content* part of the format represents the rewrite system, and depends on the type of system represented. The following (extended) BNF indicates the lexical and parsing rules for the common syntax of the new format:

$$
\begin{aligned}
\mathit{identifier} &::= [\texttt{\^{}␣\textbackslash t\textbackslash r\textbackslash n();:}]+ & \mathit{term} &::= \mathit{identifier} \mid \texttt{(}\ \mathit{identifier}\ \mathit{term}{+}\ \texttt{)}\\
\mathit{space} &::= [\texttt{␣\textbackslash t\textbackslash r\textbackslash n}] & \mathit{comment} &::= \texttt{;}\ [\texttt{␣\textbackslash t}]{*}\ (\epsilon \mid [\texttt{\^{}@\textbackslash n}]\ [\texttt{\^{}\textbackslash n}]{*})\ \texttt{\textbackslash n}\\
\mathit{number} &::= [\texttt{0} - \texttt{9}]+ & \mathit{meta\text{-}info} &::= \texttt{;}\ \texttt{␣}\ \texttt{@}\ [\texttt{\^{}\textbackslash n}]{*}\ \texttt{\textbackslash n}\\
\mathit{file} &::= \mathit{meta\text{-}info}{*}\ \mathit{content}\\
\mathit{content} &::= \mathit{TRS} \mid \mathit{CTRS} \mid \mathit{MSTRS} \mid \mathit{LCTRS} \mid \mathit{CSTRS} \mid \mathit{CSCTRS}
\end{aligned}
$$

Note that *space* and *comment* are ignored in the parsing rules. As illustrated by the introductory

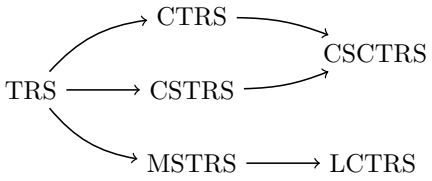


Figure 1: The formats ordered by inclusion.

example, function applications $f(t_1, \ldots, t_n)$, are written as `(`$f\ t_1\ \cdots\ t_n$`)`, while variables and constants are written without parentheses.

The relation between the formats is shown in Figure 1, where an arrow between formats means the target of the arrow is an extension of the source of the arrow.

2.1 TRS Format

The syntax of the TRS format is specified as follows:

$$\begin{aligned} \mathit{TRS} &::= \texttt{(format TRS)}\ \mathit{fun}{*}\ \mathit{rule}{*} \\ \mathit{fun} &::= \texttt{(fun}\ \mathit{identifier}\ \mathit{number}\ \texttt{)} \\ \mathit{rule} &::= \texttt{(rule}\ \mathit{term}\ \mathit{term}\ \texttt{)} \end{aligned}$$

Function symbols are declared by `(fun` f n`)` together with their arities. Undeclared identifiers are regarded as variables. The format does not exclude ill-formed TRSs such as $\{x \to \mathtt{f}(y)\}$. Validation of specifications is beyond the scope of our proposal.

2.2 CTRS Format

The format for CTRSs reuses the parsing rules for *fun* and *term*, while the remaining rules look as follows:

$$\begin{aligned} \mathit{CTRS} &::= \texttt{(format CTRS}\ \mathit{cond\text{-}type}\ \texttt{)}\ \mathit{fun}{*}\ \mathit{rule}{*} \\ \mathit{cond\text{-}type} &::= \texttt{oriented} \mid \texttt{join} \mid \texttt{semi-equational} \\ \mathit{rule} &::= \texttt{(rule}\ \mathit{term}\ \mathit{term}\ \mathit{cond}{*}\ \texttt{)} \\ \mathit{cond} &::= \texttt{(=}\ \mathit{term}\ \mathit{term}\ \texttt{)} \end{aligned}$$

For instance, the one-rule oriented CTRS $\{\mathtt{f}(x) \to \mathtt{f}(y) \Leftarrow \mathtt{g}(x) \approx z, \mathtt{g}(z) \approx \mathtt{h}(y)\}$ is specified as follows:

```
(format CTRS oriented)
(fun f 1)
(fun g 1)
(fun h 1)
(rule (f x) (f y) (= (g x) z) (= (g z) (h y)))
```

2.3 CSTRS format

To specify the replacement map, needed for context-sensitive rewriting, the *fun* parsing rule is extended with an optional argument:

$$\begin{aligned} \mathit{CSTRS} &::= \texttt{(}\ \texttt{format CSTRS}\ \texttt{)}\ \mathit{fun}{*}\ \mathit{rule}{*} \\ \mathit{fun} &::= \texttt{(}\ \texttt{fun}\ \mathit{identifier}\ \mathit{number}\ (\texttt{:replacement-map (}\ \mathit{number}{*}\ \texttt{)})?\ \texttt{)} \end{aligned}$$

and all other grammar rules are the same as in the TRS format. For example,

```
(fun f 3 :replacement-map (1 3))
```

indicates that the first and third argument positions of `f` are active. Declarations `(fun f` n`)` are equivalent to `(fun` f n `:replacement-map (`$1\ \cdots\ n$`))`, so the full replacement map is default.

2.4 CSCTRS Format

To also allow conditional rules in context-sensitive TRSs the format below is the straightforward extension of the CTRS and CSTRS formats:

$$\begin{aligned} \mathit{CSCTRS} &::= \texttt{(}\ \texttt{format CSCTRS}\ \mathit{cond\text{-}type}\ \texttt{)}\ \mathit{fun}{*}\ \mathit{rule}{*} \\ \mathit{cond\text{-}type} &::= \texttt{oriented} \mid \texttt{join} \mid \texttt{semi-equational} \\ \mathit{fun} &::= \texttt{(}\ \texttt{fun}\ \mathit{identifier}\ \mathit{number}\ \texttt{:replacement-map (}\ \mathit{number}{*}\ \texttt{)}\ \texttt{)} \\ \mathit{rule} &::= \texttt{(}\ \texttt{rule}\ \mathit{term}\ \mathit{term}\ \mathit{cond}{*}\ \texttt{)} \\ \mathit{cond} &::= \texttt{(}\ \texttt{=}\ \mathit{term}\ \mathit{term}\ \texttt{)} \end{aligned}$$

2.5 MSTRS Format

The format for many-sorted TRSs adapts the TRS format as follows:

$$\begin{aligned} \mathit{MSTRS} &::= \texttt{(}\ \texttt{format MSTRS}\ \texttt{)}\ \mathit{sort}{*}\ \mathit{fun}{*}\ \mathit{rule}{*} \\ \mathit{sort} &::= \texttt{(}\ \texttt{sort}\ \mathit{identifier}\ \texttt{)} \\ \mathit{fun} &::= \texttt{(}\ \texttt{fun}\ \mathit{identifier}\ \mathit{type}\ \texttt{)} \\ \mathit{type} &::= \mathit{identifier} \mid \texttt{(}\ \texttt{->}\ \mathit{identifier}{+}\ \mathit{identifier}\ \texttt{)} \\ \mathit{rule} &::= \texttt{(}\ \texttt{rule}\ \mathit{term}\ \mathit{term}\ \texttt{)} \end{aligned}$$

For example, the $\{\mathsf{N}, \mathsf{L}\}$-sorted signature $\{\mathsf{nil} : \mathsf{N},\ \mathsf{cons} : \mathsf{N} \times \mathsf{L} \to \mathsf{L}\}$ is represented as follows:

```
(sort N)
(sort L)
(fun nil L)
(fun cons (-> N L L))
```

We adopt arrow notation `->` for function types, anticipating that a consistent notation will be used in a new format for higher-order rewrite systems. Function declarations such as `(fun cons 2)` are invalid in the MSTRS format. The sorts of variables used in many-sorted rewrite rules must be inferred from the rules separately. So the following specification is valid in the format.

```
(format MSTRS)
(sort N)
(sort L)
(fun f (-> N N))
(fun g (-> L L))
(rule (f x) x)
(rule (g x) x)
```

Here `x` in the first rule has sort `N` whereas `x` in the second rules has sort `L`.

2.6 LCTRS Format

The format for logically constrained TRSs extends that of many-sorted TRSs:

$$
\begin{aligned}
\mathit{LCTRS} &::= \texttt{(}\ \texttt{format LCTRS}\ \texttt{)}\ \mathit{smt\text{-}theory}{*}\ \mathit{smt\text{-}def}{*}\ \mathit{sort}{*}\ \mathit{fun}{*}\ \mathit{rule}{*}\\
\mathit{smt\text{-}theory} &::= \texttt{(}\ \texttt{theory}\ \mathit{theory}\ \texttt{)}\\
\mathit{smt\text{-}def} &::= \texttt{(}\ \texttt{define-fun}\ \ldots\ \texttt{)}\\
\mathit{rule} &::= \texttt{(}\ \texttt{rule}\ \mathit{term}\ \mathit{term}\ (\texttt{:guard}\ \mathit{formula})?\ (\texttt{:var}\ \mathit{vars})?\ \texttt{)}\\
\mathit{vars} &::= \texttt{(}\ \mathit{var}{*}\ \texttt{)}\\
\mathit{var} &::= \texttt{(}\ \mathit{identifier}\ \mathit{identifier}\ \texttt{)}
\end{aligned}
$$

The *smt-theory* declaration specifies the theory symbols and types of the LCTRS theory part. This theory should be available in an off-the-shelf SMT solver. Sensible examples can be found on the SMT-LIB webpage.[4] Furthermore, *smt-def* must adhere to the `define-fun` command of SMT-LIB, and *formula* must be an SMT-LIB formula of the corresponding theory. Since SMT-LIB also adopts a syntax based on S-expressions, the lexical and parsing rules remain consistent. The optional sort declaration (`:var` *vars*) is mandatory in the case that sorts of variables cannot be inferred. For example, assuming the theory `Ints`, the variables `x` and `y` in the rule `(rule a b :guard (not (= x y)))` can be of sort Bool or Int. An example of an LCTRS in the new format is given below:

```
(format LCTRS)
(theory Ints)
(define-fun isEven ((x Int)) Bool (= (mod x 2) 0))
(sort NList)
(fun build (-> Int NList NList))     (fun nats (-> NList))
(fun cons  (-> Int NList NList))     (fun nil  (-> NList))
(rule nats (build 0 nil))
(rule (build n xs) (build (+ n 1) (cons n xs))
  :guard (and (isEven n) (>= n 0)))
(rule (build n xs) (build (+ n 1) xs)
  :guard (and (not (isEven n)) (>= n 0)) :var ((n Int) (xs NList)))
```

2.7 Multiple Rewrite Systems

Properties like relative termination and commutation rely on *two* rewrite systems over a common signature. To represent multiple rewrite systems, we use `:number` n in the format specification, where $n > 1$ specifies the number of rewrite systems. Rewrite rules have an optional

[4] https://smtlib.cs.uiowa.edu/theories.shtml

argument `:index` i where $1 \leqslant i \leqslant n$ specifies that the rule belongs to the i-th rewrite system. If the argument is not given, then $i = 1$. We give an example:

```
(format TRS :number 2)
(fun f 1)
(fun h 2)
(fun a 0)
(fun b 0)
(rule a (f b))
(rule (f a) b :index 1)
(rule (h a a) b :index 2)
(rule (f b) b :index 2)
```

This example specifies $\{\mathsf{a} \to \mathsf{f}(\mathsf{b}), \mathsf{f}(\mathsf{a}) \to \mathsf{b}\}$ as the first and $\{\mathsf{h}(\mathsf{a},\mathsf{a}) \to \mathsf{b}, \mathsf{f}(\mathsf{b}) \to \mathsf{b}\}$ as the second TRS over the common signature $\{\mathsf{a} : 0, \mathsf{b} : 0, \mathsf{f} : 1, \mathsf{h} : 2\}$.

3 Future Work

We are planning to adopt the new format in CoCo from 2024. Before that, COPS will be renewed. To support a smooth transition for tool developers, we will offer a tool[5] that converts problems in the old COPS formats to problems in the new syntax. Using the conversion tool, tools that do not support the new format can participate in CoCo with no effort.

COPS is not the only database for rewrite systems. The Termination Problem Database (TPDB) also hosts a large collection of rewrite systems. Having a unified database for rewrite systems with a uniform syntax would be beneficial to the rewriting community. Designing a new format for higher-order rewrite systems is future work.

Acknowledgements

We thank the anonymous reviewers for helpful comments.

References

[1] Jürgen Giesl, Albert Rubio, Christian Sternagel, Johannes Waldmann, and Akihisa Yamada. The termination and complexity competition. In Tomáš Vojnar and Lijun Zhang, editors, *Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 11429 of *Lecture Notes in Computer Science*, pages 156–166, 2019.

[2] Bernhard Gramlich and Klaus Györgyfalvay. On modularity of termination properties of rewriting under strategies. In *Proceedings of the 12th International Workshop on Termination*, pages 59–63, 2012.

[3] Nao Hirokawa, Julian Nagele, and Aart Middeldorp. Cops and CoCoWeb: Infrastructure for confluence tools. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Proceedings of the 9th International Joint Conference on Automated Reasoning*, volume 10900 of *Lecture Notes in Artificial Intelligence*, pages 346–353, 2018.

[4] Aart Middeldorp. *Modular Properties of Term Rewriting Systems*. PhD thesis, Vrije Universiteit, Amsterdam, 1990.

[5] Aart Middeldorp, Julian Nagele, and Kiraku Shintani. CoCo 2019: Report on the Eighth Confluence Competition. *International Journal on Software Tools for Technology Transfer*, 23(6):905–916, 2021.

[5] A preliminary version of the tool is available at https://ari-informatik.uibk.ac.at/tools/conversion/

The Z-property for left-linear term rewriting via convective context-sensitive completeness

Vincent van Oostrom

oostrom@javakade.nl

Abstract

We present a method to derive the Z-property, hence confluence, of a first-order term rewrite system $\mathcal{T}$ from completeness of an associated context-sensitive term rewrite system $\mathcal{T},\mu$ with replacement map μ. We generalise earlier such results by only requiring left-linearity of $\mathcal{T}$ and that $\mathcal{T}$-critical peaks be $\mathcal{T},\mu$-critical peaks. We introduce convective replacement maps as a generalisation of the canonical maps known from the literature.

Background This note concerns a method to *transfer* confluence of a terminating context-sensitive term rewrite system (CSR) $\mathcal{T},\mu$ to its underlying term rewrite system (TRS) $\mathcal{T}$. The direct inspiration was [3] in its contemplation of *cofinal* strategies [8], which raised the obvious question whether the Z-property could play a rôle in the theory developed there (by Hirokawa based on earlier work of Gramlich and Lucas), as it is known that the Z-property gives rise to a (hyper-)cofinal *bullet* strategy [7] and entails confluence. We answer that question in the affirmative by providing two assumptions allowing to establish the Z-property [7] for a TRS and its *layered bullet map* $\bullet$, introduced here, that inside–out and layer-wise $\mathcal{T},\mu$-normalises a term, where the notion of layer is afforded by the replacement map μ of the CSR.

Preliminaries. For first-order term rewriting we base ourselves on [8], for context-sensitive term rewriting on [1], and for the Z-property on [7]. We will only recapitulate some key notions relevant to the developments here, referring the reader to the literature for more.

Context-sensitive term rewrite systems are term rewrite systems where contracting a redex is restricted by a so-called *replacement map* mapping each function symbol in the signature to its set of *active* argument positions. The notion of being active extends compositionally to an occurrence of one term in another, via the latter occurring only in active arguments of the function symbols occurring on its path from the root in the former. Given a replacement map, context-sensitive rewriting only allows to contract active occurrences of redexes. Formally, for μ a replacement map, a *μ-redex* is a redex at an active occurrence.

Given a context-sensitive term rewrite system (CSR) $\mathcal{T},\mu$, with $\mathcal{T}$ a term rewrite system (TRS) and a replacement map μ, we use $\to$ to denote the rewrite system induced by $\mathcal{T}$, and $\hookrightarrow$ to denote the rewrite system induced by $\mathcal{T},\mu$, contracting μ-redexes only. We will exploit that, despite appearances, whether or not the *occurrence*[1] $\langle t \mid C[\,]\rangle$ of one term t in another $s = C[t]$ is active, does not depend on the (whole) *context* $C[\,]$, but only on the function symbols occurring on its *access path*, the path from the root to the hole of the context.

The main technique. We are interested in transferring confluence of $\hookrightarrow$ to that of $\to$. To that end, we will work throughout under the following two assumptions.

(i) $\mathcal{T}$ critical peaks are $\mathcal{T},\mu$ critical peaks.

(ii) $\mathcal{T},\mu$ is a left-linear and complete (confluent and terminating) CSR.

[1] See [8, Sect. 2.1.1]. Below we will make do with specifying occurrences via paths in terms.

Remark 1. (1) Without assumption **(i)** one can't expect to *transfer* confluence from $\hookrightarrow$ to $\to$ because context-sensitive rewriting in $\mathcal{T}, \mu$ may then miss out on critical peaks of $\mathcal{T}$. For instance, consider the TRS $\mathcal{T}$ with rules $a \to b$ and $f(\overline{a}) \to c$ where we used (as we will do below) overlining[2] to indicate that the argument of f is frozen, i.e. that $\mu(f) := \emptyset$. Then $\hookrightarrow$ is confluent, which may be shown by checking that the only $\hookrightarrow$-reducible terms are a and $f(\overline{a})$, and those are deterministic. In particular, we do *not* have $f(\overline{a}) \hookrightarrow f(\overline{b})$ since a is frozen in $f(\overline{a})$, see [1, 6]. However, $\to$ is not confluent due to the non-joinable critical peak $f(\overline{b}) \leftarrow f(\overline{a}) \hookrightarrow c$. (2) Neither assumption **(i)** nor assumption **(ii)** is necessary. That assumption **(i)** is not, may be shown by adjoining $c \to f(\overline{b})$ to $\mathcal{T}$. That preserves confluence of $\hookrightarrow$, which may be transferred to confluence of $\to$ using that the source of $f(\overline{a}) \to f(\overline{b})$ is $\hookrightarrow$-reducible to its target: $f(\overline{a}) \hookrightarrow c \hookrightarrow f(\overline{b})$, showing that the problematic critical peak is *redundant*, cf. [4].

To *maximise* the chance that the context-sensitive rewrite system $\hookrightarrow$ is terminating, i.e. to maximise applicability of assumption **(ii)**, it is best to *minimise* the number of active arguments or, stated differently, to *maximise* the number of frozen arguments [1]. That is, letting μ map each function symbol to the empty set $\emptyset$ would be best, but that may not be possible as assumption **(i)** forces for every rule $\ell \to r$ that for every position p in ℓ such that $\ell|_p$ unifies with some left-hand side of a rule, p be active / not frozen. This motivates:

Definition 2 (convective)**.** A replacement map μ is *convective* if $\mu^{cnv} \subseteq \mu$, i.e. if μ is not more restrictive than μ^{cnv}, where μ^{cnv} is the most restrictive replacement map such that for every rule $\ell \to r$, for every position p in ℓ such that $\ell|_p$ unifies with some left-hand side of a rule (i.e. an overlap), $i \in \mu^{cnv}(\ell(q))$ for any $qi \preceq p$ (i.e. q is the position of a function symbol on the path from the root to the overlap position p and i is its argument for which this holds.

Convectivity guarantees that if two left-hand sides occurring in a term have overlap the one is active iff the other is, but nothing more. In particular, in a critical peak the inner redex occurrence is active since the outer occurrence, at the root, is.

Example 3 (convective running example)**.** *Consider the CSR (suggested to us by Nao Hirokawa) having rules and replacement map μ^{cnv}:*

$$\begin{array}{rlllrll}
\mathsf{nats} & \to_1 & \mathsf{from}(\overline{0}) & & \mathsf{tl}(\overline{x} : \overline{y}) & \to_4 & y \\
\mathsf{inc}(\overline{x} : \overline{y}) & \to_2 & \overline{\mathsf{s}(\overline{x})} : \overline{\mathsf{inc}(y)} & & \mathsf{from}(\overline{x}) & \to_5 & \overline{x} : \overline{\mathsf{from}(\mathsf{s}(\overline{x}))} \\
\mathsf{hd}(\overline{x} : \overline{y}) & \to_3 & x & & \mathsf{inc}(\mathsf{tl}(\mathsf{from}(\overline{x}))) & \to_6 & \mathsf{tl}(\mathsf{inc}(\mathsf{from}(\overline{x})))
\end{array}$$

The only critical peak is between the fifth and sixth rules, for which convectivity entails we must at least have $1 \in \mu(\mathsf{inc}), \mu(\mathsf{tl})$. These two constraints give the convective replacement map μ^{cnv}. For this CSR $\mathcal{T}, \mu$ context-sensitive rewriting $\hookrightarrow$ trivially is terminating (checked by tools), whereas ordinary term rewriting for $\mathcal{T}$ is non-terminating (still, confluence is checked by tools).

Remark 4. In the literature so-called *canonical* replacement maps, for which only the variables may occur frozen in the left-hand sides of rewrite rules, play an important rôle. Formally, μ is *canonical* if $\mu^{can} \subseteq \mu$, i.e. if μ is not more *restrictive* than μ^{can}, where μ^{can} is defined by $i \in \mu^{can}(f)$ if for some position p and some rule $\ell \to r$, we have $\ell(p) = f$ and $\ell(pi)$ is a function symbol. Following-up on the preliminaries, the intuitive difference between canonical and convective replacement maps is that a canonical replacement map requires *all* (non-variable) positions in the redex-pattern to be active, whereas a convective replacement map requires this only of the positions on an *access path* to where the redex-patern may be overlapped by another.

[2] Our overlining notation suggests that the overlined argument position is *cut off* from its context, i.e. *frozen*.

Example 5. *In Ex. 3 canonicity requires we also have* $1 \in \mu^{can}(\mathsf{hd})$ *due to the (first) argument belonging to the pattern of the left-hand side of the third rule, illustrating* $\mu^{cnv} \subset \mu^{can}$ *here.*

The idea of our terminology *convective* is to view a term as a fluid, and the paths from the root of a left-hand side to the roots of overlapping left-hand sides as representing flows within the fluid, with the flow enabling *activation* of the latter. A term is in $\hookrightarrow$-normal form iff there's no *flow* from the root of the term to any redex-pattern. It then makes some intuitive sense to speak of its layer at depth 0 as being *solid*. Formally, the *depth* of an occurrence is the number of frozen argument positions it is in on the path to the root, inducing a natural stratification of terms into *layers* of symbols, subterms, and redexes occurring at a given depth.

Lemma 6. *If* $t \to s$ *then* $t^\bullet \twoheadrightarrow s^\bullet$,[3] *where* $\bullet$ *maps a term to its* $\hookrightarrow$*-normal form, unique by* ***(ii)****.*

Proof of Lem. 6. We claim $t \mathrel{-\!\!\!\!\twoheadrightarrow} s$ entails[4] $t^\bullet \twoheadrightarrow \hat{s} \twoheadleftarrow s$ for some $\hat{s}$. From the claim we conclude using $\hat{s} \twoheadrightarrow s^\bullet$ by assumption **(ii)** and $\hookrightarrow \subseteq \to$. We prove the claim by induction on t w.r.t. $\hookleftarrow$ well-founded (cf. [8, Def. A.1.5(vii)]) by assumption **(ii)**, and by distinguishing cases on $t \mathrel{-\!\!\!\!\twoheadrightarrow} s$:

If $t \mathrel{-\!\!\!\!\twoheadrightarrow} s$ decomposes as $t \hookrightarrow t' \mathrel{-\!\!\!\!\twoheadrightarrow} s$, we conclude by the IH for $t' \mathrel{-\!\!\!\!\twoheadrightarrow} s$ and $t^\bullet = t'^\bullet$.

Otherwise $t \mathrel{-\!\!\!\!\twoheadrightarrow} s$ only contracts non-μ-redexes, occurring at depths at least 1 in t. By assumption **(i)** those cannot have overlap with any redex-pattern at depth 0 in t, as that would give rise to a critical peak of $\mathcal{T}$ that is not a critical peak of $\mathcal{T}, \mu$.

If $t = t^\bullet$ we may trivially set $\hat{s} := s$. Otherwise, for some t' there is a step $t \hookrightarrow t'$ orthogonal to $t \mathrel{-\!\!\!\!\twoheadrightarrow} s$, hence by the assumed left-linearity of $\mathcal{T}$ the steps commute. Because $t \hookrightarrow t'$ is not below (any redex-pattern in) $t \mathrel{-\!\!\!\!\twoheadrightarrow} s$, the residual of the former after the latter is again a (single) $\hookrightarrow$-step, inducing a diagram of shape $t \hookrightarrow t' \mathrel{-\!\!\!\!\twoheadrightarrow} s' \hookleftarrow s$. By the IH for $t' \mathrel{-\!\!\!\!\twoheadrightarrow} s'$ and assumption **(ii)** we conclude to $t^\bullet = t'^\bullet \twoheadrightarrow \hat{s} \twoheadleftarrow s' \hookleftarrow s$ for some $\hat{s}$, as desired. □

Assumption **(ii)** ensures $\hookrightarrow$ has the Z-property[5] for *bullet* map $\bullet$ by [7, Lem. 11]. That bullet map is *extensive* for $\hookrightarrow$, i.e. $t \hookrightarrow\!\!\!\!\twoheadrightarrow t^\bullet$ [7, Definition 4]. We show $\to$ has the Z-property under assumptions **(i)** and **(ii)** for some bullet map $\circledcirc$ based on $\bullet$. To define $\circledcirc$ we use that any term can be uniquely decomposed into its *active* layer at depth 0 w.r.t. μ (called *maximal replacing context MRC*$^\mu$ in [5]) and its *frozen* arguments at depth 1. Accordingly, we write $C\langle\vec{t}\rangle$ to denote such a unique decomposition, where C is the active layer and $\vec{t}$ the vector of frozen arguments.

Definition 7. The *layering* $\circledcirc$ (of $\bullet$) is inductively defined by $C\langle\vec{t}\rangle^\circledcirc := C\langle\vec{t}^\circledcirc\rangle^\bullet$.

Lemma 8. $C[\vec{t}^\circledcirc] \twoheadrightarrow C[\vec{t}]^\circledcirc$.

Proof. By induction and cases on C. The base cases $C = \Box$ and $C = x$ being trivial, suppose C has shape $f(\vec{C})$ and decompose $\vec{t}$ accordingly. We conclude to $C[\vec{t}^\circledcirc] = f(\overrightarrow{C[\vec{t}^\circledcirc]}) \twoheadrightarrow f(\overrightarrow{C[\vec{t}]^\circledcirc}) \twoheadrightarrow f(\overrightarrow{C[\vec{t}]})^\circledcirc = C[\vec{t}]^\circledcirc$ by, respectively, the decomposition of $C[\vec{t}]$, the induction hypothesis for $\vec{C}$ and closure under contexts of $\to$, the claim that $g(\vec{s}^\circledcirc) \twoheadrightarrow g(\vec{s})^\circledcirc$ for all g and $\vec{s}$, and by definition of the decomposition again.

To prove the claim, first observe that $g(\vec{s}^\circledcirc) \twoheadrightarrow g(\vec{s}^\circledcirc)^\bullet$ by extensivity of $\bullet$ and $\hookrightarrow \subseteq \to$. Therefore, to conclude it suffices to show $g(\vec{s}^\circledcirc)^\bullet = g(\vec{s})^\circledcirc$. To that end, let $g(\vec{s})$ uniquely decompose as $g(\overrightarrow{D[\vec{u}]})$ with for $i \in \mu(g)$, $D_i\langle\vec{u_i}\rangle$ the unique decomposition of s_i, and for $i \notin \mu(g)$,

[3] We employ Klop's convention, cf. [8], to use an arrow with a *double* arrowhead to denote the *reflexive–transitive* closure of the rewrite relation denoted by the arrow with a *single* arrowhead.

[4] We employ Huet's convention, cf. [8], to use an arrow adorned with two vertical strokes to denote *parallel* reduction, allowing to perform steps with respect to the unadorned reduction at a number of *parallel* positions.

[5] A rewrite system $\hookrightarrow$ has the *Z-property* [7] for a map $\bullet$ on its objects, if $a \hookrightarrow b$ entails $b \hookrightarrow\!\!\!\!\twoheadrightarrow a^\bullet \hookrightarrow\!\!\!\!\twoheadrightarrow b^\bullet$.

$D_i = \Box$ and $\vec{u_i} = s_i$. Hence $g(\vec{s})^{\circledcirc} = g(\overrightarrow{D[\vec{u}^{\circledcirc}]})^{\bullet}$ per construction of the decomposition and by definition of $\circledcirc$. To conclude to $g(\vec{s}^{\circledcirc})^{\bullet} = g(\vec{s})^{\circledcirc} = g(\overrightarrow{D[\vec{u}^{\circledcirc}]})^{\bullet}$ it then suffices to show that $g(\vec{s}^{\circledcirc})$ and $g(\overrightarrow{D[\vec{u}^{\circledcirc}]})$ are $\hookrightarrow$-convertible since $\hookrightarrow$ is complete by assumption **(ii)**. Convertibility follows from that for each active argument $i \in \mu(g)$ we have that s_i uniquely decomposes as $D_i\langle\vec{u_i}\rangle$ so that $s_i^{\circledcirc} = D_i\langle\vec{u_i}^{\circledcirc}\rangle^{\bullet}$ hence $s_i^{\circledcirc}$ and $D_i\langle\vec{u_i}^{\circledcirc}\rangle$ are $\hookrightarrow$-convertible and by i being active this extends to the respective ith arguments of $g(\vec{s}^{\circledcirc})$ and $g(\overrightarrow{D[\vec{u}^{\circledcirc}]})$, and from that for each frozen argument $i \notin \mu(g)$ we have by definition of D_i and $\vec{u_i}$ that $s_i^{\circledcirc} = D_i[\vec{u_i}^{\circledcirc}]$. □

Theorem 9. *$\to$ has the Z-property for $\circledcirc$.*

Proof. We have to show that if $\phi : t \to s$ is a TRS step, then there are reductions $s \twoheadrightarrow t^{\circledcirc}$ and $t^{\circledcirc} \twoheadrightarrow s^{\circledcirc}$, giving rise to the Z in [7, Figures 1 and 5]. This we prove by induction on the decomposition $C\langle\vec{t}\rangle$ of the source t of ϕ and by cases on whether or not ϕ is a μ-step.

- if $t \hookrightarrow s$, then by definition of $\circledcirc$ and extensivity of $\circledcirc$, there is a reduction $t \twoheadrightarrow t^{\circledcirc}$ that decomposes into a reduction $\gamma : C\langle\vec{t}\rangle \twoheadrightarrow C\langle\vec{t}^{\circledcirc}\rangle$ with steps at depth at least 1, followed by a reduction $\delta : C\langle\vec{t}^{\circledcirc}\rangle \hookrightarrow\!\!\!\!\twoheadrightarrow C\langle\vec{t}^{\circledcirc}\rangle^{\bullet} = t^{\circledcirc}$ with steps at depth 0. Since ϕ is a step at depth 0, assumption **(i)** yields it and its residuals (after any prefix of γ) are orthogonal to (the corresponding suffix of) γ, giving rise by standard residual theory [8, Chapter 8] for the left-linear TRS $\mathcal{T}$, to a valley completing the peak between ϕ and γ that comprises a step $\phi/\gamma : C\langle\vec{t}^{\circledcirc}\rangle \hookrightarrow u$ and reduction $\gamma/\phi : s \twoheadrightarrow u$ for some term u.

  To conclude to $s \twoheadrightarrow t^{\circledcirc}$ we compose $\gamma/\phi : s \twoheadrightarrow u$ with the $\hookrightarrow$-reduction (lifted to a $\to$-reduction using $\hookrightarrow \subseteq \to$) of its target u to $\hookrightarrow$-normal form, which is $t^{\circledcirc}$ since $t^{\circledcirc} = C\langle\vec{t}^{\circledcirc}\rangle^{\bullet} = u^{\bullet}$ by definition respectively ϕ/γ and completeness of $\hookrightarrow$.

  To conclude to $t^{\circledcirc} \twoheadrightarrow s^{\circledcirc}$, we claim that u has shape $E[\vec{u}^{\circledcirc}]$ and s has shape $E[\vec{u}]$ for some context E and vector of terms $\vec{u}$. Then, composing $\phi/\gamma : C\langle\vec{t}^{\circledcirc}\rangle \hookrightarrow u$ with $u = E[\vec{u}^{\circledcirc}] \twoheadrightarrow E[\vec{u}]^{\circledcirc} = s^{\circledcirc}$ obtained by Lem. 8, yields $C\langle\vec{t}^{\circledcirc}\rangle \twoheadrightarrow s^{\circledcirc}$. From this we conclude to $t^{\circledcirc} = C\langle\vec{t}^{\circledcirc}\rangle^{\bullet} \twoheadrightarrow (s^{\circledcirc})^{\bullet} = s^{\circledcirc}$ by Lem. 6 and idempotence of $\bullet$.

  It remains to prove the claim that u has shape $E[\vec{u}^{\circledcirc}]$ and s has shape $E[\vec{u}]$ for some context E and vector of terms $\vec{u}$. The idea is that both C and ℓ are preserved under non-μ-steps, so their *join* is so too, and we set E be the result of contracting ℓ in the join. Formally, we construct E as follows. Let $\varsigma := \mathbf{let}\ X = C[\vec{x}]\ \mathbf{in}\ X(\vec{t})$ be the *cluster* [4] corresponding to the occurrence of the context C in t, and let ζ be the cluster of shape $\mathbf{let}\ Y = \ell\ \mathbf{in}\ \ldots$ corresponding to the occurrence in t of the left-hand side ℓ of the rule $\ell \to r$ contracted in the step $\phi : t \hookrightarrow s$. Their join $\xi := \varsigma \sqcup \zeta$ has shape $\mathbf{let}\ Z = D[\vec{z}]\ \mathbf{in}\ Z(\vec{u})$ for some context D and terms $\vec{u}$, by ς being a root cluster of ς having overlap with ζ.

  Per construction of ξ and by left-linearity of $\mathcal{T}$ there is a step ψ from $D[\vec{z}]$ contracting the occurrence of ℓ such that ϕ is a substitution instance of ψ.[6] We define E from the target of ψ writing it uniquely as $E[\vec{w}]$ for $\vec{w}$ comprising the replicated variables of $\vec{z}$, so that $\psi : D[\vec{z}] \hookrightarrow E[\vec{w}]$. We define $\vec{u}$ from the target s of $\phi : t \hookrightarrow s$, noting s can be written as the unique substitution instance $E[\vec{w}]^{\upsilon} = E[\vec{u}]$ of the target $E[\vec{w}]$ of ψ, for substitution υ mapping z_i to u_i such that $\phi = \psi^{\upsilon}$. Per construction, $t = D[\vec{z}]^{\upsilon}$ and $s = E[\vec{w}]^{\upsilon} = E[\vec{u}]$.

  Finally, we must show that $u = E[\vec{u}^{\circledcirc}]$. To that end, note that any $\hookrightarrow$-step ϕ' of shape ψ^{σ} for term substitution σ, is orthogonal to any non-μ-step χ having the same source,

[6] D could be described as being obtained by unifying the occurrence of the left-hand side ℓ with the context C (both linear and renamed apart). E is then the result of contracting the ℓ-redex in D. We avoided such an account here since D and E are not simply contexts, but linear terms; the names of the holes in E do matter.

as (the redex-pattern of) χ can neither have overlap with ς by χ being non-μ, nor have overlap with ζ by assumption **(i)** using that ψ is at depth 0 and χ at depth at least 1, so χ cannot have overlap with their join $\varsigma \sqcup \zeta$ either. Thus, χ is of shape $D[\vec{z}]^\tau$ for some step-substitution[7] τ, and $\chi/\phi' = E[\vec{w}]^\tau$ and $\phi'/\chi = \psi^{\tau'}$ with τ' the step-substitution such that $\tau'(z_i)$ is the target of $\tau(z_i)$, for all i.

By induction on the length of γ, we obtain from the above that the reduction $\gamma : t = C\langle\vec{t}\rangle \twoheadrightarrow C\langle\vec{t}^{\bullet}\rangle$, comprises only steps that are substitution instances of $D[\vec{z}]$ so that $C\langle\vec{t}^{\bullet}\rangle$ is as well. In particular note that each reduction from t_i to $t_i^\bullet$ does not change its top part (if any) overlapping the occurrence of ℓ, so is the same as that top part where all its arguments have been reduced to $\bullet$-normal form. That is, $C\langle\vec{t}^{\bullet}\rangle$ has shape $D[\vec{z}]^{\upsilon^\bullet}$. By the above, u then has shape $E[\vec{w}]^{\upsilon^\bullet} = E[\vec{u}^{\bullet}]$ as common target of ϕ/γ and γ/ϕ.

- if $t \to s$ is not a μ-step then $s = C\langle\vec{s}\rangle$ with $t_i \to s_i$ for some i and $t_j = s_j$ for all $j \neq i$. Then the Z-property holds for $\vec{s}$, i.e. $\vec{s} \twoheadrightarrow \vec{t}^{\bullet} \twoheadrightarrow \vec{s}^{\bullet}$ since by the IH $s_i \twoheadrightarrow t_i^{\bullet} \twoheadrightarrow s_i^{\bullet}$, and $s_j \twoheadrightarrow t_j^{\bullet} = s_j^{\bullet}$ for all $j \neq i$ by extensivity of $\bullet$. We conclude to $s = C\langle\vec{s}\rangle \twoheadrightarrow C\langle\vec{t}^{\bullet}\rangle \twoheadrightarrow C\langle\vec{t}^{\bullet}\rangle^\bullet = t^{\bullet} \twoheadrightarrow C\langle\vec{s}^{\bullet}\rangle^\bullet = s^{\bullet}$, using that the Z-property holds for $\vec{s}$ by the IH and closure of $\to$ under contexts for the first reduction, extensivity of $\bullet$ and $\hookrightarrow \subseteq \to$ for the second, and Z for $\vec{s}$ and closure under contexts and preservation of $\twoheadrightarrow$ by $\bullet$ for the third. □

Corollary 10. *Under assumptions **(i)** and **(ii)**, $\to$ is confluent and the* bullet *strategy $\to\!\!\!\!\bullet\!\!\!\!\to$, iterating the bullet map $\bullet$ on objects [7], is a hyper-cofinal strategy.*[8]

By Thm. 9 and [7, Lem. 51 & Thm. 50]. Thus $\to\!\!\!\!\bullet\!\!\!\!\to$ is (hyper-)normalising [8], and the layered bullet function $\bullet$ induces an *effective* (if $\hookrightarrow$ is) confluence construction and cofinal strategy.

A concrete criterion Our approach to confluence of a term rewrite system (via the Z-property) has confluence of context-sensitive rewriting $\hookrightarrow$ as an assumption; in fact local confluence suffices given termination is also assumed. The following is a known sufficient condition for local confluence of context-sensitive rewriting $\hookrightarrow$; see e.g. [6] (also for other conditions).

(iii) $\mathcal{T}, \mu$ is 0-*preserving* if, whenever a variable occurs at depth 0 in the left-hand side of a rule, then all its occurrences in the right-hand side are at depth 0 as well.

Lemma 11 (Thm. 30 of [6])**.** *If $\mathcal{T}, \mu$ is a left-linear CSR satisfying assumptions **(i)** and **(iii)** with $\hookrightarrow$-joinable critical peaks, then context-sensitive rewriting $\hookrightarrow$ is locally confluent.*

Since convectivity entails assumption **(i)**, and $\hookrightarrow$-joinability of critical peaks and 0-preservingness entail confluence of $\hookrightarrow$ for left-linear CSRs by Lem. 11, combining this with termination of $\mathcal{T}$ all assumptions of Thm. 9 are satisfied:

Corollary 12. *If $\mathcal{T}, \mu$ is a left-linear 0-preserving CSR such that μ is convective, critical peaks are $\hookrightarrow$-joinable, and context-sensitive rewriting $\hookrightarrow$ is terminating, then the TRS $\mathcal{T}$, i.e. the rewrite system $\to$, has the Z-property for the layered bullet function $\bullet$.*

This generalises [1, Thm. 2], the main result of that paper, both by *relaxing* two of its assumptions, canonicity to convectivity and level-decreasingness to 0-preservingness, and by *strengthening* its conclusion from confluence to the Z-property.

[7] A substitution τ such that for all i, $\tau(z_i)$ either is a single step or a term.

[8] A $\to$-strategy is *hyper-cofinal* [8, 7] if for any $a \twoheadrightarrow b$, starting from a always eventually performing a $\to\!\!\!\!\bullet\!\!\!\!\to$-step after a number of $\to$-steps will yield an object c that *exceeds* b in the sense that $b \twoheadrightarrow c$.

Example 13 (application to running example)**.** *The CSR of Ex. 3 is left-linear (by inspection of the left-hand sides; no repeated variables),* 0*-preserving (vacuously so, since there are no variables at depth* 0 *in left-hand sides; all occur in overlined subterms), has a convective replacement map (μ^{cnv} is the most restrictive such), and is terminating as was observed.*

The (only) critical peak is between its fifth and sixth rules and is $\hookrightarrow$-joinable as shown by (the following) two legs of its confluence diagram: $\mathsf{inc}(\mathsf{tl}(\mathsf{from}(\overline{x}))) \hookrightarrow \mathsf{tl}(\mathsf{inc}(\mathsf{from}(\overline{x}))) \hookrightarrow \mathsf{tl}(\mathsf{inc}(\overline{x} : \overline{\mathsf{from}(\mathsf{s}(\overline{x}))})) \hookrightarrow \mathsf{tl}(\overline{\mathsf{s}(\overline{x})} : \overline{\mathsf{inc}(\mathsf{from}(\mathsf{s}(\overline{x})))}) \hookrightarrow \mathsf{inc}(\mathsf{from}(\mathsf{s}(\overline{x})))$ *and* $\mathsf{inc}(\mathsf{tl}(\mathsf{from}(\overline{x}))) \hookrightarrow \mathsf{inc}(\mathsf{tl}(\overline{x}{:}\overline{\mathsf{from}(\mathsf{s}(\overline{x}))})) \hookrightarrow \mathsf{inc}(\mathsf{from}(\mathsf{s}(\overline{x})))$.[9] *Corollary 12 yields $\to$ has the Z-property, is confluent, and $\twoheadrightarrow$ is an effective cofinal $\twoheadrightarrow$-strategy.*

Remark 14. The methods of [1] do not apply to yield the result of Ex. 13. Their methods require level-decreasingness of the rules and the fifth added rule is not for the canonical replacement map μ^{can} employed by them: the level of x in the lhs is then 1 whereas in the rhs it occurs not only with level 1 but also with level 3. The only way to regain level-decreasingness is to make both the second argument of : and the argument of s active, but that would violate termination of $\hookrightarrow$ (the fifth rule becomes spiralling), one of the other assumptions of [1, Thm. 2].

Conclusion Based on a notion of convectivity introduced here, and by relaxing the assumptions of [1, Thm. 2], we *partially* settled [1, Open Problem 1] by Cor. 12. In the long note http://www.javakade.nl/research/pdf/z-csr.pdf we also positively settled [1, Open Problem 2]. Though that note provides several examples other than Ex. 13 illustrating our method, implementing it on top of an extant tool would open up the database of confluence problems for easy experimentation. See [2] for more on that w.r.t. the tool CONFident.

Acknowledgments We thank Nao Hirokawa & Salvador Lucas for inspiration & feedback.

References

[1] B. Gramlich and S. Lucas. Generalizing Newman's lemma for left-linear rewrite systems. In 17*th RTA*, volume 4098 of *LNCS*, pages 66–80. Springer, 2006. https://doi.org/10.1007/11805618_6.

[2] R. Gutiérrez, S. Lucas, and M. Vítores. Proving confluence in the confluence framework with CONFident, 2023. https://doi.org/10.48550/arXiv.2306.16330.

[3] N. Hirokawa. Seven confluence criteria for solving COPS #20. In C. Rocha and S. Winkler, editors, 11*th IWC*, 2022. Invited talk. http://cl-informatik.uibk.ac.at/iwc/2022/.

[4] N. Hirokawa, J. Nagele, V. van Oostrom, and M. Oyamaguchi. Confluence by critical pair analysis revisited. In *CADE 27*, volume 11716 of *LNCS*, pages 319–336. Springer, 2019. https://doi.org/10.1007/978-3-030-29436-6_19.

[5] S. Lucas. Derivational complexity and context-sensitive rewriting. *Journal of Automated Reasoning*, 65(8):1191–1229, 2021. https://doi.org/10.1007/s10817-021-09603-1.

[6] S. Lucas, M. Vítores, and R. Gutiérrez. Proving and disproving confluence of context-sensitive rewriting. *Journal of Logical and Algebraic Methods in Programming*, 126:100749, 2022. https://doi.org/10.1016/j.jlamp.2022.100749.

[7] V. van Oostrom. Z; syntax-free developments. In *6th FSCD 2021*, volume 195 of *LIPIcs*, pages 24:1–24:22. Leibniz-Zentrum für Informatik, 2021. https://doi.org/10.4230/LIPIcs.FSCD.2021.24.

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

[9] Indeed, as pointed out by Salvador Lucas (personal communication 1-6-2023) termination and local confluence of this CSR are established automatically by CONFident (http://zenon.dsic.upv.es/confident/).

Ground Canonical Rewrite Systems Revisited

Aart Middeldorp[1], Masahiko Sakai[2], and Sarah Winkler[3]

[1] University of Innsbruck, Innsbruck, Austria, aart.middeldorp@uibk.ac.at
[2] Nagoya University, Nagoya, Japan, sakai@i.nagoya-u.ac.jp
[3] Free University of Bozen-Bolzano, Bolzano, Italy, winkler@inf.unibz.it

Abstract

Systems of ground equations can always be transformed into equivalent canonical rewrite systems. Moreover, only finitely many distinct canonical rewrite systems exist for a given system of ground equations, a result proved by Snyder using congruence closure. Snyder also introduced a simple transformation to convert one canonical presentation into another one. In this paper we prove that this transformation is sound and complete, using standard rewrite techniques. We show that the transformation fails in the AC case.

1 Introduction

Congruence closure is an efficient technique to solve the word problem for systems of ground equations. Completion is a well-known technique for transforming systems of equations into equivalent canonical rewrite systems. It takes a reduction order as input and if it succeeds, the word problem is decidable by computing and comparing the unique normal forms of the two terms involved. For systems of ground equations, completion can be tamed such that it always terminates. Snyder [5] proved that the number of distinct canonical rewrite systems representing a given set of ground equations is at most 2^k where k is the number of equations. Furthermore, each of these canonical rewrite systems has the same number of rewrite rules. Finally, given one canonical rewrite system, all others can be obtained by a simple transformation.

This transformation is the topic of the paper. Using traditional rewrite techniques, which we briefly recall in Section 2, we prove in Section 3 that the transformation preserves canonicity and that it is complete in the sense that all canonical presentations of a given system of ground equations can be obtained from any of them by a finite number of transformation steps. In Section 4 we consider the extension of Snyder's transformation in the presence of associative and commutative operators. We conclude with some open questions.

2 Preliminaries

We assume familiarity with term rewriting but recall some important concepts and results in this preliminary section. An equational system (ES for short) is a set of equations between terms over a common signature. Throughout this paper we will consider finite ground TRSs. Every ground ES is also a TRS, and vice versa. A TRS is right-reduced if the right-hand sides of its rewrite rules are normal forms. It is left-reduced if every left-hand side of a rewrite rule is a normal form with respect to the other rules. A *reduced* TRS is both left-reduced and right-reduced. A TRS that is confluent, terminating and reduced is called *canonical*. Given a TRS $\mathcal{R}$, we denote the set of left-hand (right-hand) sides of its rules by $\mathsf{LHS}(\mathcal{R})$ ($\mathsf{RHS}(\mathcal{R})$). The set of its normal forms is denoted by $\mathsf{NF}(\mathcal{R})$.

Two TRSs $\mathcal{R}$ and $\mathcal{S}$ are *(conversion) equivalent* if $\leftrightarrow^*_{\mathcal{R}} = \leftrightarrow^*_{\mathcal{S}}$ and *normalization equivalent* if $\to^!_{\mathcal{R}} = \to^!_{\mathcal{S}}$. Here $t \to^!_{\mathcal{R}} u$ if both $t \to^*_{\mathcal{R}} u$ and $u \in \mathsf{NF}(\mathcal{R})$. A simple sufficient condition for normalization equivalence of canonical TRSs is stated in the following lemma, which is a special case of [1, Lemma 4.4(2)].

Lemma 1. *If $\mathcal{R}$ and $\mathcal{S}$ are canonical* TRS*s such that $\mathsf{NF}(\mathcal{S}) \subseteq \mathsf{NF}(\mathcal{R})$ and $\to_{\mathcal{S}} \subseteq \leftrightarrow^*_{\mathcal{R}}$ then $\mathcal{R}$ and $\mathcal{S}$ are normalization equivalent.* □

We conclude this preliminary section with two basic results that will be used in the sequel. We refer to [1] for modern (formalized) proofs of these results. The first one is a variation of a result by Métivier [3] and the second one is due to Snyder [5, Theorem 2.14].

Theorem 2.

1. *Normalization equivalent reduced* TRS*s are unique up to literal similarity.*
2. *Reduced ground* TRS*s are canonical.*

For ground TRSs, literal similarity amounts to equality, so normalization equivalent reduced ground TRSs are unique and equivalent canonical ground TRSs compatible with the same reduction order are identical.

3 Snyder's Transformation

Snyder [5, Theorem 3.18] showed how any set of ground equations $\mathcal{E}$ can be transformed into a special structure-sharing dag G from which an abstract relation T is extracted which represents a canonical ground TRS $\mathcal{R}$ equivalent to $\mathcal{E}$. The procedure runs in $O(n \log n)$ time, where n is the size of $\mathcal{E}$, though it may require more than $O(n \log n)$ time to actually obtain the TRS $\mathcal{R}$ from the dag. The method is complete in the sense that any canonical ground TRS $\mathcal{R}$ equivalent to $\mathcal{E}$ can be obtained in this way [5, Theorem 4.6]. This completeness result relies on the existence of a compatible well-founded order with the subterm property that is total on ground terms [5]. Furthermore, if $\mathcal{E}$ consists of k equations then there are at most 2^k equivalent canonical ground TRSs [5, Theorem 4.7]. This final result relies on the fact that any equivalent canonical TRS has at most k rewrite rules (which follows from ground completion, as remarked in [5, p. 424]).

Snyder mentions at the end of Section 4 that a single transformation is sufficient to generate all other equivalent canonical ground TRS from any given canonical TRS. This transformation is defined as follows:

$$\mathcal{R} \uplus \{\ell \to r\} \implies \mathcal{R}' \cup \{r \to \ell\} \qquad (\star)$$

Here $\mathcal{R} \uplus \{\ell \to r\}$ is a canonical ground TRS such that r is not a proper subterm of ℓ, and $\mathcal{R}'$ is the ground TRS obtained from $\mathcal{R}$ by replacing every occurrence of r in the rewrite rules by ℓ. No proofs are provided, cf. [5, footnote 15].

Example 3. The collection of ground equations

$$\begin{array}{rrr} \mathsf{f}(\mathsf{a}) \approx \mathsf{g}(\mathsf{b},\mathsf{b}) & \mathsf{f}(\mathsf{f}(\mathsf{a})) \approx \mathsf{a} & \mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{a}))) \approx \mathsf{a} \\ \mathsf{g}(\mathsf{b},\mathsf{h}(\mathsf{a})) \approx \mathsf{g}(\mathsf{b},\mathsf{b}) & \mathsf{h}(\mathsf{a}) \approx \mathsf{b} & \mathsf{i}(\mathsf{f}(\mathsf{a})) \approx \mathsf{c} \end{array}$$

admits six different canonical TRSs, which are connected by $(\star)$ as depicted in Figure 1, where the labels of arrows indicate the rule that was flipped.

Below we present detailed proofs concerning the transformation $(\star)$. We write $t\{r \mapsto \ell\}$ for the term obtained from t after replacing all subterms r by ℓ. The canonicity of $\mathcal{R}' \cup \{r \to \ell\}$ is relatively easy to prove.

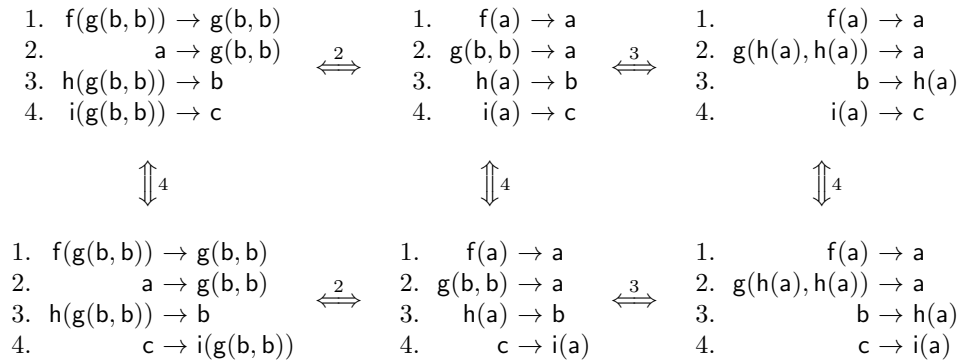


Figure 1: Six equivalent canononical TRSs.

Lemma 4. *The* TRS $\mathcal{R}' \cup \{r \to \ell\}$ *is canonical.*

Proof. We first show that $\mathcal{R}' \cup \{r \to \ell\}$ is right-reduced. Suppose to the contrary that a right-hand side t of a rule in $\mathcal{R}' \cup \{r \to \ell\}$ is reducible with a rule $u \to v \in \mathcal{R}' \cup \{r \to \ell\}$. Since we deal with ground TRSs, $t \trianglerighteq u$. Since right-hand sides of rules in $\mathcal{R}' \cup \{r \to \ell\}$ do not contain occurrences of r, $u \to v \in \mathcal{R}'$. Let $u' \to v'$ be the rule in $\mathcal{R}$ such that $u = u'\{r \mapsto \ell\}$ and $v = v'\{r \mapsto \ell\}$. Similarly, let t' be the right-hand side of a rule in $\mathcal{R}$ such that $t = t'\{r \mapsto \ell\}$. From $t \trianglerighteq u$ we infer $t' \trianglerighteq u'$, contradicting the fact that $\mathcal{R} \cup \{\ell \to r\}$ is right-reduced. Next we show that $\mathcal{R}' \cup \{r \to \ell\}$ is left-reduced. Let t be the left-hand side of a rewrite rule in $\mathcal{R}' \cup \{r \to \ell\}$. We distinguish two cases.

- Suppose $t = r \notin \mathsf{NF}(\mathcal{R}')$. So $r \trianglerighteq v$ for some left-hand side v of a rewrite rule in $\mathcal{R}'$. Since ℓ is not a proper subterm of r by assumption and $\ell = r$ is excluded by the right-reducedness of $\mathcal{R} \cup \{\ell \to r\}$, v contains no occurrences of ℓ. It follows that v is the left-hand side of a rewrite rule in $\mathcal{R}$, contradicting the right-reducedness of $\mathcal{R} \cup \{\ell \to r\}$.

- Suppose $t \to u \in \mathcal{R}'$ with $t \notin \mathsf{NF}((\mathcal{R}' \cup \{r \to \ell\}) \setminus \{t \to u\})$. Since t does not contain any occurrences of r, we have $t \notin \mathsf{NF}(\mathcal{R}' \setminus \{t \to u\})$. Let $v \to w$ be a rewrite rule in $\mathcal{R}' \setminus \{t \to u\}$ such that $t \trianglerighteq v$. Let t' and v' be the left-hand sides of rules in $\mathcal{R}$ such that $t = t'\{r \mapsto \ell\}$ and $v = v'\{r \mapsto \ell\}$. We have $t' \trianglerighteq v'$, contradicting left-reducedness of $\mathcal{R}$.

The proof is concluded by the canonicity of reduced ground TRSs, cf. Theorem 2 (2). □

We next show that every canonical presentation $\mathcal{S}$ of an ES $\mathcal{E}$ can be obtained from another canonical presentation $\mathcal{R}$ by a sequence of ($\star$) transformations.

Theorem 5. *Transformation ($\star$) is complete.*

Proof. Let $\mathcal{E}$ be an ES and $\mathcal{R}$ a canonical presentation of $\mathcal{E}$. For an arbitrary canonical representation $\mathcal{S}$ of $\mathcal{E}$, we prove by induction on $|\mathsf{RHS}(\mathcal{S}) \setminus \mathsf{NF}(\mathcal{R})|$ that $\mathcal{R}$ can be transformed into $\mathcal{S}$ by a sequence of ($\star$) transformations. First, note the following:

$$\mathsf{NF}(\mathcal{R}) \cap \mathsf{LHS}(\mathcal{S}) = \varnothing \implies \mathcal{R} = \mathcal{S} \qquad (\dagger)$$

This can be shown as follows. From the assumption we obtain $\mathsf{NF}(\mathcal{R}) \subseteq \mathsf{NF}(\mathcal{S})$ as $\mathsf{NF}(\mathcal{R})$ is closed under subterms. Conversion equivalence implies $\to_{\mathcal{R}} \subseteq \leftrightarrow^*_{\mathcal{S}}$ and $\mathcal{R}$ is terminating. Hence $\mathcal{R}$ and $\mathcal{S}$ are normalization equivalent by Lemma 1 and equal by Theorem 2 (1).

In the base case $\mathsf{RHS}(\mathcal{S}) \subseteq \mathsf{NF}(\mathcal{R})$. By (†), if $\mathcal{S} \neq \mathcal{R}$, there is some $u \to v \in \mathcal{S}$ such that $u \in \mathsf{NF}(\mathcal{R})$. However, $\mathsf{RHS}(\mathcal{S}) \subseteq \mathsf{NF}(\mathcal{R})$ implies $v \in \mathsf{NF}(\mathcal{R})$, which contradicts conversion equivalence as u and v are different convertible normal forms in $\mathcal{R}$.

In the induction step, we can assume by (†) that $\mathcal{S}$ contains a rule $u \to v$ such that $u \in \mathsf{NF}(\mathcal{R})$. Conversion equivalence and completeness of $\mathcal{R}$ imply $v \to^+_{\mathcal{R}} u$. In particular, v is not a proper subterm of u and hence we can apply transformation ($\star$) to $u \to v$ in $\mathcal{S}$. Let $\mathcal{S}'$ be the resulting TRS, which is canonical and contains $v \to u$. We compare $\mathsf{RHS}(\mathcal{S}) \setminus \mathsf{NF}(\mathcal{R})$ and $\mathsf{RHS}(\mathcal{S}') \setminus \mathsf{NF}(\mathcal{R})$. The former contains v as $u \to v \in \mathcal{S}$ and $v \notin \mathsf{NF}(\mathcal{R})$. This rule is replaced by $v \to u$ in $\mathcal{S}'$, which does not contribute to $\mathsf{RHS}(\mathcal{S}') \setminus \mathsf{NF}(\mathcal{R})$ as $u \in \mathsf{NF}(\mathcal{R})$. In addition, ($\star$) may replace right-hand sides $r[v] \in \mathsf{RHS}(\mathcal{S})$ by $r[u] \in \mathsf{RHS}(\mathcal{S}')$, but for all such terms $r[v]$, we have $r[v] \in \mathsf{RHS}(\mathcal{S}) \setminus \mathsf{NF}(\mathcal{R})$. Independent of whether or not $r[u] \in \mathsf{RHS}(\mathcal{S}') \setminus \mathsf{NF}(\mathcal{R})$ for some of the modified right-hand sides, we have $|\mathsf{RHS}(\mathcal{S}) \setminus \mathsf{NF}(\mathcal{R})| > |\mathsf{RHS}(\mathcal{S}') \setminus \mathsf{NF}(\mathcal{R})|$. By the induction hypothesis a sequence of ($\star$) transformations can turn $\mathcal{S}'$ into $\mathcal{R}$. □

4 Ground AC-Canonical Rewrite Systems

Marché [2, Theorem 3.1] proved that any AC canonical ground TRS for a finite set of ground equations with AC operators must be finite. The interesting proof relies on Dickson's Lemma. In the same paper, Marché presents a version of ground completion for the AC setting and a strategy that ensures termination [2, Theorem 4.3]. Unlike ground completion, in the AC setting critical pairs involving rules with the same AC symbol at the root of left-hand side need to be deduced. Unlike for AC completion, AC unification is not needed for computing these critical pairs. Ground AC completion relies on an AC simplification order which is AC total on ground terms. The existence of such an order was first shown by Narendran and Rusinowitch [4].

The next example shows that the transformation ($\star$) is unsuitable in an AC setting.

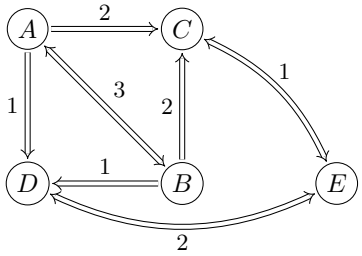
Example 6. The ground equations

$$\mathsf{f}(\mathsf{a},\mathsf{b}) \approx \mathsf{d} \qquad\qquad \mathsf{f}(\mathsf{b},\mathsf{c}) \approx \mathsf{e}$$

with AC symbol f admit five different AC canonical TRSs:

A	$\mathsf{f}(\mathsf{a},\mathsf{b}) \xrightarrow{1} \mathsf{d}$	$\mathsf{f}(\mathsf{b},\mathsf{c}) \xrightarrow{2} \mathsf{e}$	$\mathsf{f}(\mathsf{a},\mathsf{e}) \xrightarrow{3} \mathsf{f}(\mathsf{c},\mathsf{d})$
B	$\mathsf{f}(\mathsf{a},\mathsf{b}) \xrightarrow{1} \mathsf{d}$	$\mathsf{f}(\mathsf{b},\mathsf{c}) \xrightarrow{2} \mathsf{e}$	$\mathsf{f}(\mathsf{a},\mathsf{e}) \xleftarrow{3} \mathsf{f}(\mathsf{c},\mathsf{d})$
C	$\mathsf{f}(\mathsf{a},\mathsf{b}) \xrightarrow{1} \mathsf{d}$	$\mathsf{f}(\mathsf{b},\mathsf{c}) \xleftarrow{2} \mathsf{e}$	
D	$\mathsf{f}(\mathsf{a},\mathsf{b}) \xleftarrow{1} \mathsf{d}$	$\mathsf{f}(\mathsf{b},\mathsf{c}) \xrightarrow{2} \mathsf{e}$	
E	$\mathsf{f}(\mathsf{a},\mathsf{b}) \xleftarrow{1} \mathsf{d}$	$\mathsf{f}(\mathsf{b},\mathsf{c}) \xleftarrow{2} \mathsf{e}$	

If we apply ($\star$) to A by reversing rule 1 then rule 3 is first modified to $\mathsf{f}(\mathsf{a},\mathsf{e}) \approx \mathsf{f}(\mathsf{a},\mathsf{b},\mathsf{c})$ and subsequently deleted due to rule 2, resulting in D. (This cannot happen in the non-AC case.) Applying the AC version of ($\star$) systematically yields the following diagram:



Selecting rule 1 in D or rule 2 in C results in the TRS

$$F \qquad \mathsf{f}(\mathsf{a},\mathsf{b}) \xrightarrow{1} \mathsf{d} \qquad \mathsf{f}(\mathsf{b},\mathsf{c}) \xrightarrow{2} \mathsf{e}$$

which is not AC confluent. From F we obtain A and B by orienting the single *AC critical pair*.

The underlying problem is that reduced ground TRSs need not be AC confluent, necessitating the computation of AC critical pairs. In general, however, after applying ($\star$) and resolving AC critical pairs, new critical pairs may arise. Worse, reversing a rule might violate termination.

Example 7. The TRS $\mathcal{R}$ consisting of the two rules

$$\mathsf{f}(\mathsf{b},\mathsf{c}) \to \mathsf{f}(\mathsf{a},\mathsf{b}) \qquad \mathsf{f}(\mathsf{c},\mathsf{d}) \to \mathsf{f}(\mathsf{d},\mathsf{a})$$

with AC symbol f is AC canonical. Reversing the first rule results in the TRS $\mathcal{R}'$:

$$\mathsf{f}(\mathsf{a},\mathsf{b}) \to \mathsf{f}(\mathsf{b},\mathsf{c}) \qquad \mathsf{f}(\mathsf{c},\mathsf{d}) \to \mathsf{f}(\mathsf{d},\mathsf{a})$$

Transformation ($\star$) requires no further changes. However, AC termination is violated:

$$\mathsf{f}(\mathsf{a},\mathsf{b},\mathsf{d}) \to_{\mathcal{R}'/\mathsf{AC}} \mathsf{f}(\mathsf{b},\mathsf{c},\mathsf{d}) \to_{\mathcal{R}'/\mathsf{AC}} \mathsf{f}(\mathsf{a},\mathsf{b},\mathsf{d})$$

Despite the fact that the first AC completion procedures have been presented thirty years ago, many questions about even the simpler setting of ground AC completion are still open, for instance: Is the number of AC canonical presentations of an ES finite? If yes, how many such presentations are there? Can every equivalent AC canonical ground TRS be generated by modifying the AC termination order?

Acknowledgments. Part of this work was performed when the first author was employed at the Future Value Creation Research Center of Nagoya University, Japan.

References

[1] Nao Hirokawa, Aart Middeldorp, Christian Sternagel, and Sarah Winkler. Abstract completion, formalized. *Logical Methods in Computer Science*, 15(3):19:1–19:42, 2019.

[2] Claude Marché. On ground AC-completion. In *Proceedings of the 4th International Conference on Rewriting Techniques and Applications*, volume 488 of *Lecture Notes in Computer Science*, pages 411–422, 1991.

[3] Yves Métivier. About the rewriting systems produced by the Knuth–Bendix completion algorithm. *Information Processing Letters*, 16(1):31–34, 1983.

[4] Paliath Narendran and Michaël Rusinowitch. Any ground associative-commutative theory has a finite canonical system. In *Proceedings of the 4th International Conference on Rewriting Techniques and Applications*, volume 488 of *Lecture Notes in Computer Science*, pages 423–434, 1991.

[5] Wayne Snyder. A fast algorithm for generating reduced ground rewriting systems from a set of ground equations. *Journal of Symbolic Computation*, 15(4):415–450, 1993.

Formalizing Confluence and Commutation Criteria Using Proof Terms*

Christina Kohl and Aart Middeldorp

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

Abstract

We present recent advancements concerning formalizations of state-of-the-art confluence and commutation criteria. In particular we describe formalizations of several extensions of van Oostrom's development-closedness criterion in the proof assistant Isabelle/HOL. A key component for the formalized proofs is the concept of proof terms representing multi-steps.

1 Introduction

Recently we presented the first formalized proof of van Oostrom's development-closedness criterion [4]. Since then, we were able to extend this result in several ways, which we describe here. In Section 2 we first give some basic definitions and recap proof terms representing multi-steps in term rewriting—a concept which proved to be very valuable for formalizing critical pair criteria based on multi-steps (also known as development steps). In Section 3 we present our formalization of *almost* development closed critical pairs for commutation of two term rewrite systems.[1] This is an extension of the result described in [4] in two ways: First, it weakens the joinability requirement for critical pairs which are overlays,[2] second, it uses the critical pairs between two left-linear TRSs to determine whether they commute. In Section 4 we describe our most recent extension, namely a formalized proof of the results in [3].

2 Preliminaries

We assume familiarity with the basics of term rewriting, as can be found in [1], and only recap some important definitions here. A relation $\to$ is confluent if

$$ {}^*\!\leftarrow \cdot \to^* \subseteq \to^* \cdot {}^*\!\leftarrow $$

Two relations $\to_1$ and $\to_2$ (locally) commute if

$$ {}^*_1\!\leftarrow \cdot \to_2^* \subseteq \to_2^* \cdot {}^*_1\!\leftarrow \qquad ({}_1\!\leftarrow \cdot \to_2 \subseteq \to_2^* \cdot {}^*_1\!\leftarrow) $$

We say that $\to_1$ and $\to_2$ *strongly commute* if

$$ {}_1\!\leftarrow \cdot \to_2 \subseteq \to_2^{=} \cdot {}^*_1\!\leftarrow $$

Strong commutation of $\to_1$ and $\to_2$ implies commutation of $\to_1$ and $\to_2$. If $\to$ commutes with itself then it is confluent. The *multi-step* relation $\mathrel{-\!\!\!\to\!\!\!\!\!\circ}_{\mathcal{R}}$ is inductively defined on terms as follows:

*This research is funded by the Austrian Science Fund (FWF) project I5943.

[1] A more detailed description of this formalization effort will appear in [5].

[2] An idea first described by Toyama for parallel closed critical pairs in [6] and adapted by van Oostrom for development closed critical pairs [7].

- $x \mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}} x$ for all variables x,
- $f(s_1, \ldots, s_n) \mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}} f(t_1, \ldots, t_n)$ if $s_i \mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}} t_i$ for all $1 \leqslant i \leqslant n$, and
- $\ell\sigma \mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}} r\tau$ if $\ell \to r \in \mathcal{R}$ and $\sigma(x) \mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}} \tau(x)$ for all $x \in \mathcal{V}\mathsf{ar}(\ell)$.

A critical *overlap* $(\ell_1 \to r_1, p, \ell_2 \to r_2)_\sigma$ of two TRSs $\mathcal{R}$ and $\mathcal{S}$ consists of variants $\ell_1 \to r_1$ and $\ell_2 \to r_2$ of rewrite rules in $\mathcal{R}$ and $\mathcal{S}$ without common variables, a position $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$, and a most general unifier σ of ℓ_1 and $\ell_2|_p$. From a critical overlap $(\ell_1 \to r_1, p, \ell_2 \to r_2)_\sigma$ we obtain a critical peak $\ell_2\sigma[r_1\sigma]_p \mathrel{\mathcal{R}_1\!\leftarrow} \ell_2\sigma[\ell_1\sigma]_p = \ell_2\sigma \mathrel{\to_{\mathcal{R}_2}} r_2\sigma$ and the corresponding *critical pair* $\ell_2\sigma[r_1\sigma]_p \mathrel{\mathcal{R}\!\leftarrow\!\rtimes\!\to\!\mathcal{S}} r_2\sigma$. When $p = \epsilon$ we call $r_1\sigma \mathrel{\mathcal{R}\!\leftarrow\!\rtimes\!\to\!\mathcal{S}} r_2\sigma$ an *overlay*. TRSs $\mathcal{R}$ and $\mathcal{S}$ are *development closed* if $s \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} t$ for all critical pairs $s \mathrel{\mathcal{R}\!\leftarrow\!\rtimes\!\to\!\mathcal{S}} t$ and $s \mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}} t$ for all critical pairs $s \mathrel{\mathcal{S}\!\leftarrow\!\rtimes\!\to\!\mathcal{R}} t$.

Proof terms are built from function symbols, variables, and rule symbols. We use Greek letters for rule symbols. If α is a rule symbol then $\mathsf{lhs}(\alpha)$ ($\mathsf{rhs}(\alpha)$) denotes the left-hand (right-hand) side of the rewrite rule denoted by α. Furthermore $\mathsf{var}(\alpha)$ denotes the list $(x_1, \ldots, x_n)$ of variables appearing in α in some fixed order. The length of this list is the arity of α. The list $\mathsf{vpos}(\alpha) = (p_1, \ldots, p_n)$ denotes the corresponding variable positions in $\mathsf{lhs}(\alpha)$ such that $\mathsf{lhs}(\alpha)|_{p_i} = x_i$. Given a rule symbol α with $\mathsf{var}(\alpha) = (x_1, \ldots, x_n)$ and terms $t_1, \ldots, t_n$, we write $\langle t_1, \ldots, t_n \rangle_\alpha$ for the substitution $\{ x_i \mapsto t_i \mid 1 \leqslant i \leqslant n \}$. Given a proof term A, its source $\mathsf{src}(A)$ and target $\mathsf{tgt}(A)$ are computed by the following equations:

$$\begin{aligned}
\mathsf{src}(x) &= \mathsf{tgt}(x) = x \\
\mathsf{src}(f(A_1, \ldots, A_n)) &= f(\mathsf{src}(A_1), \ldots, \mathsf{src}(A_n)) \\
\mathsf{src}(\alpha(A_1, \ldots, A_n)) &= \mathsf{lhs}(\alpha)\langle \mathsf{src}(A_1), \ldots, \mathsf{src}(A_n) \rangle_\alpha \\
\mathsf{tgt}(f(A_1, \ldots, A_n)) &= f(\mathsf{tgt}(A_1), \ldots, \mathsf{tgt}(A_n)) \\
\mathsf{tgt}(\alpha(A_1, \ldots, A_n)) &= \mathsf{rhs}(\alpha)\langle \mathsf{tgt}(A_1), \ldots, \mathsf{tgt}(A_n) \rangle_\alpha
\end{aligned}$$

Proof terms A and B are said to be *co-initial* if they have the same source. The proof term A over TRS $\mathcal{R}$ is a witness of the multi-step $\mathsf{src}(A) \mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}} \mathsf{tgt}(A)$. For every multi-step there exists a proof term witnessing it. For co-initial proof terms A and B the *residual* $A \,/\, B$ is a proof term witnessing the remainder of A after contracting the redexes of B. This can be formally defined as a partial operation with several useful properties which are exploited in the proofs below. The amount of overlap between two co-initial proof terms A and B is denoted by $\blacktriangle(A, B)$ and is measured as the number of function symbols in $\mathsf{src}(A) = \mathsf{src}(B)$ that are part of a redex in both A and B. If a redex in A at position $p \in \mathsf{src}(A)$ overlaps with a redex in B at position $q \in \mathsf{src}(B)$ then we call the pair (p, q) an *overlap* between A and B. The formal definitions, as well as useful lemmata about the aforementioned operations, can be found in [4].

The formalized results of the next sections are integrated into the library IsaFoR.[3] The contributions described in this paper are located in the file `Development_Closed.thy`.

3 Almost Development Closed Critical Pairs

In [4] we described the formalized proof of van Oostrom's development-closedness criterion [7].

Theorem 1. *Left-linear development closed* TRS*s are confluent.*

[3] http://cl-informatik.uibk.ac.at/isafor

When introducing his criterion in [7] van Oostrom already gave an extension where the joining condition on overlays is weakened. This extension is modeled after Toyama's almost parallel closed critical pairs [6]. Just like Toyama's result, *almost development closed* critical pairs can be lifted to the commutation setting, resulting in the following theorem.

Theorem 2. *Let $\mathcal{R}$ and $\mathcal{S}$ be two left-linear* TRS*s. If $s \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} \cdot \mathrel{{}^{*}_{\mathcal{R}}\!\!\leftarrow} t$ for all critical pairs $s \mathrel{{}_{\mathcal{R}}\!\leftarrow\!\rtimes\!\rightarrow_{\mathcal{S}}} t$, and $s \mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}} t$ for all critical pairs $s \mathrel{{}_{\mathcal{S}}\!\leftarrow\!\rtimes\!\rightarrow_{\mathcal{R}}} t$ which are not overlays, then $\mathcal{R}$ and $\mathcal{S}$ commute.*

In [7] it is suggested to adapt the measure ($\blacktriangle$) used in the proof of Theorem 1 in order to obtain a proof of the extended result. This turned out to be problematic as we describe in [5]. For the formalized proof we instead add another case distinction in the step case of the proof used for Theorem 1.

Formalized proof. We show strong commutation of $\mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}}$ and $\mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}}$, which implies commutation of $\mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}}$ and $\mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}}$ and hence commutation of $\to_{\mathcal{R}}$ and $\to_{\mathcal{S}}$. Assume $t \mathrel{{}_{\mathcal{R}}\!\leftarrow\!\circ} s \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} u$ and let A be a proof term representing $s \mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}} t$ and let B be a proof term representing $s \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} u$. We show $t \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} v \mathrel{{}^{*}_{\mathcal{R}}\!\!\leftarrow} u$ for some term v by well-founded induction on $\blacktriangle(A, B)$.

- In the base case $\blacktriangle(A, B) = 0$ which implies that $A \,/\, B$ is a proof term over $\mathcal{R}$ and $B \,/\, A$ a proof term over $\mathcal{S}$ such that $\mathsf{tgt}(A \,/\, B) = \mathsf{tgt}(B \,/\, A)$.
- In the induction step we assume $\blacktriangle(A, B) > 0$. To apply the induction hypothesis we need to obtain proof terms A' over $\mathcal{R}$ and B' over $\mathcal{S}$ such that $\blacktriangle(A', B') < \blacktriangle(A, B)$. We select an innermost overlap (p, q) and let α and β be the corresponding rule symbols in A and B. Moreover, let $\mathsf{vpos}(\alpha) = (p_1, \ldots, p_n), \mathsf{var}(\alpha) = (x_1, \ldots, x_n), \mathsf{vpos}(\beta) = (q_1, \ldots, q_m)$ and $\mathsf{var}(\beta) = (y_1, \ldots, y_m)$, where we assume $\{x_1, \ldots, x_n\} \cap \{y_1, \ldots, y_m\} = \varnothing$ without loss of generality. We define proof terms $\Delta_1 = s[\alpha(s|_{pp_1}, \ldots, s|_{pp_n})]_p$ and $\Delta_2 = s[\beta(s|_{qq_1}, \ldots, s|_{qq_m})]_q$. Then Δ_1 represents a single step $s \to t'$ and the residual $A \,/\, \Delta_1$ witnesses $t' \mathrel{-\!\!\circ\!\!\to} t$ for some term t'. Likewise Δ_2 represents a step $s \to u'$ and $B \,/\, \Delta_2$ witnesses $u' \mathrel{-\!\!\circ\!\!\to} u$ for some term u'. We distinguish three cases.

  1. For $q < p$ and $q' = p \backslash q$ we define the substitution

  $$\begin{aligned}\tau \;=\; &\{x_i \mapsto \mathsf{lhs}(\beta)|_{q'p_i} \mid 1 \leqslant i \leqslant n \text{ and } q'p_i \in \mathcal{P}\mathsf{os}(\mathsf{lhs}(\beta))\} \\ \cup\; &\{y_j \mapsto \mathsf{lhs}(\alpha)|_{q_j \backslash q} \mid 1 \leqslant j \leqslant m \text{ and } q_j \backslash q \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\mathsf{lhs}(\alpha))\}\end{aligned}$$

  which yields the critical peak [4, Lemma 7.2]

  $$\mathsf{lhs}(\beta)[\mathsf{rhs}(\alpha)\tau]_{q'} \;\mathrel{\mathcal{R}\!\leftarrow}\; \mathsf{lhs}(\beta)[\mathsf{lhs}(\alpha)\tau]_{q'} \;=\; \mathsf{lhs}(\beta)\tau \;\to_{\mathcal{S}}\; \mathsf{rhs}(\beta)\tau$$

  We define the position $q_\beta \in \mathcal{P}\mathsf{os}(B)$ such that $B = B[\beta(B_1, \ldots, B_m)]_{q_\beta}$ and $\mathsf{src}(B)[\;]_q = \mathsf{src}(B[\;]_{q_\beta})$. By the almost development closedness assumption there exists a multi-step $\mathsf{lhs}(\beta)[\mathsf{rhs}(\alpha)\tau]_{q'} \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} \mathsf{rhs}(\beta)\tau$. Let D' be a proof term representing this multi-step. We define the substitution

  $$\rho \;=\; \{y_j \mapsto B_j \mid 1 \leqslant j \leqslant m\} \cup \{x_i \mapsto \mathsf{lhs}(\beta)\langle B_1, \ldots, B_m\rangle_\beta|_{q'p_i} \mid 1 \leqslant i \leqslant n\}$$

  and show that the proof term $B' = B[D'\rho]_{q_\beta}$ witnesses a multi-step $t' \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} u$. Finally, we show $\blacktriangle(A', B') < \blacktriangle(A, B)$ for $A' = A \,/\, \Delta_1$ [4, Lemma 7.8].

  2. If $p < q$ a symmetric construction yields a proof term A' witnessing $u' \mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}} t$ such that $\blacktriangle(A', B') < \blacktriangle(A, B)$ for $B' = B \,/\, \Delta_2$.

3. If $p = q$ we can apply the same construction as in the first case, but the almost development closedness assumption yields a term v', a proof term D' witnessing $\mathsf{rhs}(\alpha)\tau \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} v'$, and a rewrite sequence $\mathsf{rhs}(\beta)\tau \to^*_{\mathcal{R}} v'$. Then $B' = B[D'\rho]_{q_\beta}$ witnesses a multi-step $t' \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} w$ for some term w. Like before, $\blacktriangle(A', B') < \blacktriangle(A, B)$ for $A' = A \,/\, \Delta_1$. Moreover, $u \to^*_{\mathcal{R}} w$ since $u = \mathsf{tgt}(B[\mathsf{rhs}(\beta)\tau\rho]_{q_\beta})$ and $w = \mathsf{tgt}(B[D'\rho]_{q_\beta}) = \mathsf{tgt}(B[v'\rho]_{q_\beta})$.

The previous items allow us to apply the induction hypothesis to obtain a term v such that $t \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} v \mathrel{{}^*_{\mathcal{R}}\!\!\leftarrow} u$, which completes the proof. □

4 Commutation via Relative Termination

We formalized another extension of Theorem 1 for commutation due to Hirokawa and Middeldorp [3]. It is based on local commutation together with relative termination of the *critical peak steps* between two TRSs $\mathcal{R}$ and $\mathcal{S}$. In this section, when we speak of a critical peak $t \mathrel{{}_{\mathcal{R}}\!\!\overset{p}{\leftarrow}} s \mathrel{\overset{q}{\to}_{\mathcal{S}}} u$ we either mean the critical peak $t \mathrel{{}_{\mathcal{R}}\!\!\overset{p}{\leftarrow}} s \mathrel{\overset{\epsilon}{\to}_{\mathcal{S}}} u$ or the critical peak $u \mathrel{{}_{\mathcal{S}}\!\!\overset{q}{\leftarrow}} s \mathrel{\overset{\epsilon}{\to}_{\mathcal{R}}} t$.

Definition 3. Let $t \mathrel{{}_{\mathcal{R}}\!\!\overset{p}{\leftarrow}} s \mathrel{\overset{q}{\to}_{\mathcal{S}}} u$ be a critical peak. It is $(\mathcal{R}, \mathcal{S})$-*closed* if $u \mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}} t$ whenever $p = \epsilon$ and $t \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} u$ whenever $q = \epsilon$. The set of all non-closed critical peak steps of $\mathcal{S}$ for $\mathcal{R}$ is defined as $\mathsf{CPS}_{\mathcal{R}}(\mathcal{S}) = \{\, s \to u \mid t \mathrel{{}_{\mathcal{R}}\!\!\overset{p}{\leftarrow}} s \mathrel{\overset{q}{\to}_{\mathcal{S}}} u$ is a critical peak which is not $(\mathcal{R}, \mathcal{S})$-closed$\,\}$.

Theorem 4 ([3, Theorem 4.3])**.** *Left-linear locally commuting* TRS*s* $\mathcal{R}$ *and* $\mathcal{S}$ *commute if* $\mathsf{CPS}_{\mathcal{S}}(\mathcal{R}) \cup \mathsf{CPS}_{\mathcal{R}}(\mathcal{S})$ *is relatively terminating with respect to* $\mathcal{R} \cup \mathcal{S}$*.*

Recall that a TRS $\mathcal{R}$ is relatively terminating with respect to a TRS $\mathcal{S}$ if $\mathcal{R}/\mathcal{S}$ is terminating. Here $\mathcal{R}/\mathcal{S}$ denotes the relation $\to^*_{\mathcal{S}} \cdot \to_{\mathcal{R}} \cdot \to^*_{\mathcal{S}}$. The following key lemma is needed in addition to results from the previous sections in order to prove Theorem 4. The formalized proof closely follows the paper proof in [3] and is very similar to the proof of Theorem 2.

Lemma 5. *Let* $\mathcal{R}$ *and* $\mathcal{S}$ *be left-linear* TRS*s. If* $t \mathrel{{}_{\mathcal{R}}\!\!\leftarrow\!\!\circ} s \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} u$ *then*

(a) $t \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} \cdot \mathrel{{}_{\mathcal{R}}\!\!\leftarrow\!\!\circ} u$*, or*

(b) $t \mathrel{{}_{\mathcal{R}}\!\!\leftarrow\!\!\circ} \cdot \mathrel{{}_{\mathsf{CPS}_{\mathcal{S}}(\mathcal{R})}\!\!\leftarrow} s' \mathrel{\to_{\mathsf{CPS}_{\mathcal{R}}(\mathcal{S})}} \cdot \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} u$ *and* $s \to^*_{\mathcal{R}\cup\mathcal{S}} s'$ *for some* s'*.*

Formalized proof. Assume $t \mathrel{{}_{\mathcal{R}}\!\!\leftarrow\!\!\circ} s \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} u$ and let A be a proof term representing $s \mathrel{-\!\!\circ\!\!\to_{\mathcal{R}}} t$ and let B be a proof term representing $s \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} u$. Like in the proof of Theorem 2 we proceed by induction on $\blacktriangle(A, B)$.

- If $\blacktriangle(A, B) = 0$ case (a) holds (by taking the residuals $B \,/\, A$ and $A \,/\, B$).
- If $\blacktriangle(A, B) > 0$. We select an innermost overlap (p, q), assume without loss of generality that $q \leqslant p$ and let $q' = p \backslash q$. Then we define Δ_1, Δ_2, and τ as in the proof of Theorem 2. Hence, we obtain a critical peak $t' \mathrel{{}_{\mathcal{R}}\!\!\leftarrow\!\!\circ} s' \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} u'$ where $t' = \mathsf{lhs}(\beta)[\mathsf{rhs}(\alpha)\tau]_{q'}$, $s' = \mathsf{lhs}(\beta)[\mathsf{lhs}(\alpha)\tau]_{q'} = \mathsf{lhs}(\beta)\tau$, and $u' = \mathsf{rhs}(\beta)\tau$. We distinguish two cases.

  1. If the peak $t' \mathrel{{}_{\mathcal{R}}\!\!\leftarrow\!\!\circ} s' \mathrel{-\!\!\circ\!\!\to_{\mathcal{S}}} u'$ is not $(\mathcal{R}, \mathcal{S})$-closed then $s' \to t' \in \mathsf{CPS}_S(R)$ and $s' \to u' \in \mathsf{CPS}_R(S)$. We define the substitution

$$\sigma = \{x_i \mapsto s|_{pp_i} \mid 1 \leqslant i \leqslant n\} \cup \{y_j \mapsto s|_{qq_j} \mid 1 \leqslant j \leqslant m\}$$

and show $\mathsf{tgt}(\Delta_1) = s[t'\sigma]_{q'}$ and $\mathsf{tgt}(\Delta_2) = s[u'\sigma]_{q'}$. Then $A \,/\, \Delta_1$ witnesses a multi-step $s[t'\sigma]_{q'} \mathrel{-\!\!\circ\!\!\to}_{\mathcal{R}} t$ and $B \,/\, \Delta_2$ witnesses a multi-step $s[u'\sigma]_{q'} \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S}} u$. Hence

$$t \; \mathcal{R}^{\leftarrow\!\circ} \cdot {}_{\mathsf{CPS}_{\mathcal{S}}(\mathcal{R})}{\leftarrow} \; s \; {\to}_{\mathsf{CPS}_{\mathcal{R}}(\mathcal{S})} \cdot \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S}} u$$

i.e., case (b) holds.

2. If the peak $t' \; \mathcal{R}^{\leftarrow\!\circ} \; s' \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S}} u'$ is $(\mathcal{R},\mathcal{S})$-closed then there exists a proof term D' witnessing $t' \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S}} u'$. Hence, we can apply the same constructions as in the proof of Theorem 2 to obtain a proof term $B[D'\rho]_{q_\beta}$ such that $\mathsf{src}(B[D'\rho]_{q_\beta}) = \mathsf{src}(A \,/\, \Delta_1) = \mathsf{tgt}(\Delta_1)$, $\mathsf{tgt}(B[D'\rho]_{q_\beta}) = \mathsf{tgt}(B) = u$, and $\blacktriangle(A \,/\, \Delta_1, B[D'\rho]_{q_\beta}) < \blacktriangle(A,B)$. The induction hypothesis now yields the following two cases:

   (a) $t \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S}} \cdot \, \mathcal{R}^{\leftarrow\!\circ} \, u$, or

   (b) $t \; \mathcal{R}^{\leftarrow\!\circ} \cdot {}_{\mathsf{CPS}_{\mathcal{S}}(\mathcal{R})}{\leftarrow} \; s'' \; {\to}_{\mathsf{CPS}_{\mathcal{R}}(\mathcal{S})} \cdot \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S}} u$ and $\mathsf{tgt}(\Delta_1) \to^*_{\mathcal{R}\cup\mathcal{S}} s''$ for some s''.

   In the first case we are immediately done. In the second case it remains to show $s \to^*_{\mathcal{R}\cup\mathcal{S}} s''$. This is straightforward since Δ_1 witnesses a rewrite step $s \to_{\mathcal{R}} \mathsf{tgt}(\Delta_1)$ and we have the rewrite sequence $\mathsf{tgt}(\Delta_1) \to^*_{\mathcal{R}\cup\mathcal{S}} s''$. □

Formalized proof of Theorem 4. Assume $\mathcal{R}$ and $\mathcal{S}$ are left-linear and locally commuting. Moreover, assume $(\mathsf{CPS}_{\mathcal{S}}(\mathcal{R}) \cup \mathsf{CPS}_{\mathcal{R}}(\mathcal{S}))/(\mathcal{R} \cup \mathcal{S})$ is terminating. We show that the TRSs $\mathcal{R}$ and $\mathcal{S}$ are decreasing with respect to the conversion version of van Oostrom's decreasing diagrams technique [8, Theorem 3]. In Isabelle we use Felgenhauer's formalization [2]. We first define the labeled multi-step relations $t \mathrel{-\!\!\circ\!\!\to}_{\mathcal{R},s} u$ if and only if $s \to^*_{\mathcal{R}\cup\mathcal{S}} t \mathrel{-\!\!\circ\!\!\to}_{\mathcal{R}} u$ and, similarly, $t \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S},s} u$ if and only if $s \to^*_{\mathcal{R}\cup\mathcal{S}} t \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S}} u$. The relation $>$ is defined as

$$\to^+_{(\mathsf{CPS}_{\mathcal{S}}(\mathcal{R})\cup\mathsf{CPS}_{\mathcal{R}}(\mathcal{S}))/(\mathcal{R}\cup\mathcal{S})}$$

We show that $(\{\mathrel{-\!\!\circ\!\!\to}_{\mathcal{R},s}\}_{s\in\mathcal{T}}, \{\mathrel{-\!\!\circ\!\!\to}_{\mathcal{S},s}\}_{s\in\mathcal{T}})$ is decreasing with respect to $>$. Note that $>$ is well-founded by assumption and transitive by definition. It remains to show that every local peak $t \; \mathcal{R},s_1{\leftarrow\!\circ} \; s \mathrel{-\!\!\circ\!\!\to}_{\mathcal{R},s_2} u$ can be completed into a decreasing diagram with conversions as illustrated in Figure 1. To this end, we apply Lemma 5 to the peak $t \; \mathcal{R},s_1{\leftarrow\!\circ} \; s \mathrel{-\!\!\circ\!\!\to}_{\mathcal{R},s_2} u$. We need to consider the two cases of Lemma 5:

(a) First assume there exists a term v such that $t \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S}} v \; \mathcal{R}^{\leftarrow\!\circ} u$. Since $s_2 \to^*_{\mathcal{R}\cup\mathcal{S}} t \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S}} v$ we have $t \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S},s_2} v$. Similarly we have $u \mathrel{-\!\!\circ\!\!\to}_{\mathcal{R},s_1} v$. Hence, by taking empty sequences for all conversions in Figure 1, we can complete the peak.

(b) Next assume there exist terms s', t', u' such that $t \; \mathcal{R}^{\leftarrow\!\circ} \; t' \; {}_{\mathsf{CPS}_{\mathcal{S}}(\mathcal{R})}{\leftarrow} \; s' \; {\to}_{\mathsf{CPS}_{\mathcal{R}}(\mathcal{S})} \; u' \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S}} u$ and $s \to^*_{\mathcal{R}\cup\mathcal{S}} s'$. Since $\mathcal{R}$ and $\mathcal{S}$ are locally commuting, we obtain a term v such that $t' \to^*_{\mathcal{S}} v \; {}^*_{\mathcal{R}}{\leftarrow} u'$. We show that the peak can be completed by a conversion

$$t \overset{*}{\underset{\vee s_1 s_2}{\longleftrightarrow}} t' \overset{*}{\underset{\vee s_1 s_2}{\longleftrightarrow}} v \overset{*}{\underset{\vee s_1 s_2}{\longleftrightarrow}} u' \overset{*}{\underset{\vee s_1 s_2}{\longleftrightarrow}} u$$

with steps in $\mathcal{R}\cup\mathcal{S}$. From $t' \mathrel{-\!\!\circ\!\!\to}_{\mathcal{R}} t$ we obtain $t' \mathrel{-\!\!\circ\!\!\to}_{\mathcal{R},t'} t$. Moreover, $t' < s_1$ since

$$s_1 \to^*_{\mathcal{R}\cup\mathcal{S}} s \to^*_{\mathcal{R}\cup\mathcal{S}} s' \mathrel{-\!\!\circ\!\!\to}_{\mathsf{CPS}_S(R)} t'$$

Similarly, $u' \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S},u'} u$ and $u' < s_2$. From $t' \to^*_{\mathcal{S}} v$ we obtain a rewrite sequence

$$t' \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S},t'} \cdots \mathrel{-\!\!\circ\!\!\to}_{\mathcal{S},t'} v$$

So each step can be labeled with t' for which we already showed $t' < s_1$. Similarly, $u' \mathrel{-\!\!\circ\!\!\to}_{\mathcal{R},u'} \cdots \mathrel{-\!\!\circ\!\!\to}_{\mathcal{R},u'} v$ is obtained. □

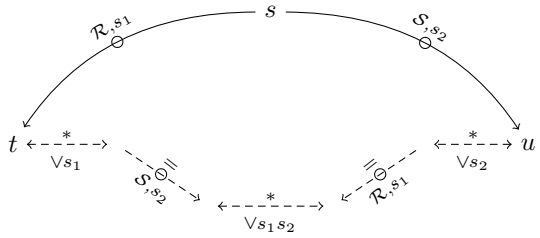


Figure 1: Decreasingness of $(\{ \multimap\!\!\to_{\mathcal{R},s} \}_{s \in \mathcal{T}}, \{ \multimap\!\!\to_{\mathcal{S},s} \}_{s \in \mathcal{T}})$.

Formalizing the results of this section turned out to be surprisingly straightforward. In order to implement $\mathsf{CPS}_{\mathcal{R}}(\mathcal{S})$, we had to add a definition of critical peaks for two different TRSs (the existing definition in IsaFoR only takes a single TRS as argument). All other relevant definitions and results were already present in IsaFoR or the Archive of Formal Proofs. In total the formalization only required a little more than 500 (new) lines of Isabelle code.

Theorem 4 subsumes Theorem 1 and its commutation version as $\mathsf{CPS}_{\mathcal{S}}(\mathcal{R}) \cup \mathsf{CPS}_{\mathcal{R}}(\mathcal{S}) = \varnothing$ and $\mathcal{R}$ and $\mathcal{S}$ are locally commuting for all left-linear TRSs $\mathcal{R}$ and $\mathcal{S}$ which are development closed. In [3] it is stated without proof that Theorem 4 can be strengthened by implementing the same weakening for overlays as in Theorem 2, hence also subsuming Theorem 2. It is however unclear how case (a) of the proof above works for this extension. The problem is that we might get a sequence $u \to_{\mathcal{R}}^* v$, instead of a multi-step, for which we would need to show that its labels are below s_1 or s_2.

References

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

[2] Bertram Felgenhauer. Decreasing diagrams II. *Archive of Formal Proofs*, August 2015. https://isa-afp.org/entries/Decreasing-Diagrams-II.html, Formal proof development.

[3] Nao Hirokawa and Aart Middeldorp. Commutation via relative termination. In *Proc. 2th IWC*, pages 29–33, 2013.

[4] Christina Kohl and Aart Middeldorp. A formalization of the development closedness criterion for left-linear term rewrite systems. In *Proc. 12th International Conference on Certified Programs and Proofs*, pages 197–210, 2023. doi:10.1145/3573105.3575667.

[5] Christina Kohl and Aart Middeldorp. Formalizing almost development closed critical pairs. In *Proc. 14th International Conference on Interactive Theorem Proving*, pages 38:1–38:8, 2023. doi:10.4230/LIPIcs.ITP.2023.38.

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

[7] Vincent van Oostrom. Developing developments. *Theoretical Computer Science*, 175(1):159–181, 1997. doi:10.1016/S0304-3975(96)00173-9.

[8] Vincent van Oostrom. Confluence by decreasing diagrams – Converted. In *Proc. 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.

A verified algorithm for deciding pattern completeness and related properties*

René Thiemann

Universität Innsbruck, Austria
rene.thiemann@uibk.ac.at

Abstract

Pattern completeness is the property that the left-hand sides of a functional program cover all cases w.r.t. pattern matching. In the context of term rewriting a related notion is quasi-reducibility, a prerequisite if one wants to perform ground confluence proofs by rewriting induction.

In order to certify such confluence proofs, we develop an algorithm that decides pattern completeness and that can be used to ensure quasi-reducibility. One of the advantages of the algorithm is its simple structure: it is similar to that of a regular matching algorithm, and it avoids the enumeration of all terms up to a given depth (the latter is required in an existing decision procedure for quasi-reducibility.) Despite having a simple structure, termination and soundness proofs for the algorithm are not immediate. However, these properties have been verified in Isabelle/HOL.

1 Introduction

Consider programs written in a declarative style such as functional programs or term rewrite systems, where evaluation is defined by pattern matching. In several applications it is important to know that evaluation of a given program cannot get stuck, i.e., the programs should be sufficiently complete. For instance in Isabelle/HOL [7], a function definition must be sufficiently complete since HOL is a logic of total functions. And methods that are based on rewriting induction [1, 8] require similar completeness results, e.g., for proving ground confluence.

In both applications the evaluation mechanism can be described as a set of rules $\ell \to r$ where evaluation replaces instances of left-hand sides (lhss) $\ell\sigma$ by instances of right-hand sides $r\sigma$. Let L be the set of lhss of some set of rules. We consider programs where lhss are first-order terms over some finite signature $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ where $\mathcal{C}$ are constructor symbols and $\mathcal{D}$ are defined symbols. Hence, input values to a function are represented by constructor ground terms, denoted by $\mathcal{T}(\mathcal{C})$. We further assume a typed setting where we consider first-order monomorphic types, i.e, every function symbol of arity n has a type $\tau_1 \times \cdots \times \tau_n \to \tau_0$, where each type τ_i is just a name and τ_0 is called the target type. Also variables are typed and we write $\mathcal{V}_\tau$ for the set of variables of type τ. We define $\mathcal{T}(\mathcal{C})_\tau$ as the set of constructor ground terms that have type τ, and we assume $\mathcal{T}(\mathcal{C})_\tau \neq \emptyset$ for all types τ.

We can now define a first notion to describe that a program cannot get stuck.

Definition 1 (Pattern Completeness of Programs)**.** *A program with lhss L is pattern complete, if for all terms $f(t_1, \ldots, t_n)$, with $f : \tau_1 \times \cdots \times \tau_n \to \tau_0 \in \mathcal{D}$ and $(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{C})_{\tau_1} \times \cdots \times \mathcal{T}(\mathcal{C})_{\tau_n}$, there is some $\ell \in L$ such that $t = f(t_1, \ldots, t_n)$ is matched by ℓ.*

*This research was supported by the Austrian Science Fund (FWF) project I 5943.

Example 2. *Let* $\mathcal{C}_{\mathbb{N}} = \{\mathsf{true} : \mathbb{B}, \mathsf{false} : \mathbb{B}, \mathsf{0} : \mathbb{N}, \mathsf{s} : \mathbb{N} \to \mathbb{N}\}$ *be the set of constructors to represent the Booleans and natural numbers in Peano notation. We consider a program* $\mathcal{R}_{\mathbb{N}}$ *that defines a function to compute whether a natural number is even, i.e.,* $\mathcal{D} = \{\mathsf{even} : \mathbb{N} \to \mathbb{B}\}$.

$$\mathsf{even}(\mathsf{0}) \to \mathsf{true} \qquad \mathsf{even}(\mathsf{s}(\mathsf{0})) \to \mathsf{false} \qquad \mathsf{even}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{even}(x) \tag{1}$$

This program is pattern complete, since no matter which number n *we provide as argument, one of the lhss will match the term* $\mathsf{even}(n)$*; this fact can easily be seen by a case-analysis on whether* n *represents 0, 1, or some larger number.*

Note the importance of types: without them $\mathsf{even}(\mathsf{s}(\mathsf{true}))$ *would contradict completeness.*

An alternative notion to pattern completeness is quasi-reducibility [5] where the difference is that the matching can happen for an arbitrary subterm.

Definition 3 (Quasi-Reducibility of Programs)**.** *A program with lhss* L *is quasi-reducible, if all terms* $f(t_1, \ldots, t_n)$*, with* $f : \tau_1 \times \cdots \times \tau_n \to \tau_0 \in \mathcal{D}$ *and* $(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{C})_{\tau_1} \times \cdots \times \mathcal{T}(\mathcal{C})_{\tau_n}$*, there is some* $\ell \in L$ *such that a subterm of* $t = f(t_1, \ldots, t_n)$ *is matched by* ℓ*.*

Clearly, pattern completeness implies quasi-reducibility, and if the root symbols of all lhss in a program are within $\mathcal{D}$, then the two notions coincide. The following example illustrates the difference between the two notions.

Example 4. *Consider* $\mathcal{C}_{\mathbb{Z}} = \{\mathsf{true} : \mathbb{B}, \mathsf{false} : \mathbb{B}, \mathsf{0} : \mathbb{Z}, \mathsf{s} : \mathbb{Z} \to \mathbb{Z}, \mathsf{p} : \mathbb{Z} \to \mathbb{Z}\}$ *to represent the Booleans and integers in Peano notation, e.g.,* $\mathsf{p}(\mathsf{0})$ *represents -1. Now we consider a program* $\mathcal{R}_{\mathbb{Z}}$ *that defines a function to compute whether an integer number is even, i.e.,* $\mathcal{D} = \{\mathsf{even} : \mathbb{Z} \to \mathbb{B}\}$*. It consists of all rules of* $\mathcal{R}_{\mathbb{N}}$ *and the following additional rules.*

$$\mathsf{even}(\mathsf{p}(\mathsf{0})) \to \mathsf{false} \qquad \mathsf{even}(\mathsf{p}(\mathsf{p}(x))) \to \mathsf{even}(x) \tag{2}$$

$$\mathsf{s}(\mathsf{p}(x)) \to x \qquad \mathsf{p}(\mathsf{s}(x)) \to x \tag{3}$$

This program is quasi-reducible since every term $\mathsf{even}(n)$ *with* $n \in \mathcal{T}(\mathcal{C}_{\mathbb{Z}})_{\mathbb{Z}}$ *has a subterm that is matched by some lhs: If* n *contains both* s *and* p *then one of the rules (3) is applicable. Otherwise* n *is of the form* $\mathsf{s}^i(\mathsf{0})$ *or* $\mathsf{p}^i(\mathsf{0})$ *and then rules (1) or (2) will be applicable.*

But the program is not pattern complete since $\mathsf{even}(\mathsf{s}(\mathsf{p}(\mathsf{0})))$ *is not matched by any lhs.*

Kapur et al. proved decidability of quasi-reducibility [5]. They show that one may replace the quantification over all constructor ground terms $t_1, \ldots, t_n$ in Definition 3 by a bounded quantification where the depth of the terms t_i is restricted by d, a number that can computed from L; overall their decision procedure requires to enumerate exponentially many terms whenever $\mathcal{C}$ contains a symbol of arity 2 or larger. We are aware of two other algorithms to deduce quasi-reducibility in more complex settings, e.g., where rules may be constrained by arithmetic constraints such as "this rule is only applicable if $x > 0$" [4, 6], but both algorithms do not properly generalize the result of Kapur et al. since they are restricted to linear lhss. Bouhoula and Jacquemard [3] also designed an algorithm in a more complex setting with conditions and constraints, and a back-end that is based on constrained tree automata techniques. Since their soundness result is restricted to ground confluent systems, their algorithm is not applicable in our use-case, since we want to verify ground confluence proofs on methods that rely upon quasi-reducibility. Finally, Bouhoula developed an algorithm to verify ground confluence and sufficient completeness at the same time [2], where we are not sure whether it can also be used to just ensure completeness, e.g., for non-ground confluent systems.

In this paper we will provide a simple algorithm to decide pattern completeness. It avoids to always enumerate all terms up to given depth, it is not restricted to linear lhss, and it does

not require tree automata algorithms. It can also be used as a sufficient criterion for quasi-reducibility, and as a decision procedure for those programs where all lhss have a defined symbol as root.

2 Pattern Completeness – The Linear Case

Before we design the new decision procedure for pattern completeness we first reformulate and generalize this notion. A slightly more general notion than pattern completeness is already provided by Aoto and Toyama [1]. They define the concept of a *cover*, where L *covers* a term t if for all σ_{cg} there is some $\ell \in L$ that matches $t\sigma_{cg}$ (here, σ_{cg} represents some arbitrary constructor ground substitution where all variables are replaced by constructor ground terms). Hence, pattern completeness can be formulated as the question whether $f(x_1, \ldots, x_n)$ is covered by L for all defined symbols f where the x_i's are distinct variables.

We generalize the notion of a cover further into a pattern problem.

Definition 5 (Matching Problem and Pattern Problem)**.** *A matching problem is a finite set $mp = \{(t_1, \ell_1), \ldots, (t_n, \ell_n)\}$ that contains arbitrary pairs of terms. A pattern problem is a finite set $pp = \{mp_1, \ldots, mp_k\}$ of matching problems.*

A matching problem mp is solvable w.r.t. some constructor ground substitution σ_{cg} if there is some substitution γ such that $t_i\sigma_{cg} = \ell_i\gamma$ for all $(t_i, \ell_i) \in mp$. A pattern problem pp is solvable if for each constructor ground substitution σ_{cg} there is some $mp \in pp$ such that mp is solvable w.r.t. σ_{cg}. A set of pattern problems P is solvable if each $pp \in P$ is solvable.

We further introduce a special matching problem $\bot_{mp}$ that represents an unsolvable matching problem. Similarly, we define $\top_{pp}$ as a new pattern problem that is always solvable. Finally, $\bot_P$ represents a new unsolvable set of pattern problems.

Hence, the question of whether L covers t can be encoded in the pattern problem $\{\{(t, \ell)\} \mid \ell \in L\}$. Similarly, Aoto and Toyama's notion of strong quasi-reducibility [1] can also be encoded as a pattern problem: $\bigcup_{t \in \{x_1, \ldots, x_n, f(x_1, \ldots, x_n)\}} \{\{(t, \ell)\} \mid \ell \in L\}$ expresses that one tries to find a match at the root ($t = f(x_1, \ldots, x_n)$) or a match for a direct subterm ($t = x_i$). Finally, the question of whether a program with lhss L and defined symbols $\mathcal{D}$ is pattern complete w.r.t. Definition 1 is expressible as solvability of the set of pattern problems $\{\{\{(f(x_1, \ldots, x_{n_f}), \ell)\} \mid \ell \in L\} \mid f \in \mathcal{D}\}$ where n_f is the arity of f and the variables $x_1, \ldots, x_{n_f}$ are distinct.

The following inference rules describe a decision procedure to determine solvability of *linear* pattern problems. A matching problem $\{(t_1, \ell_1), \ldots, (t_n, \ell_n)\}$ is linear if each ℓ_i is linear and the variables of ℓ_i and ℓ_j are disjoint for $i \neq j$. A pattern problem is linear if all its matching problems are linear.

Definition 6 (Inference Rules for Linear Pattern Problems)**.** *We define $\to$ as a set of simplification rules for matching problems.*

$$\{(f(t_1, \ldots, t_n), f(\ell_1, \ldots, \ell_n)\} \uplus mp \to \{(t_1, \ell_1), \ldots, (t_n, \ell_n)\} \cup mp \qquad \text{(decompose)}$$

$$\{(f(\ldots), g(\ldots)\} \uplus mp \to \bot_{mp} \qquad \text{if } f \neq g \qquad \text{(clash)}$$

$$\{(t, x)\} \uplus mp \to mp \qquad \text{(match)}$$

On top of this we define simplification rules $\Rightarrow$ for pattern problems.

$$\{mp\} \uplus pp \Rightarrow \{mp'\} \cup pp \qquad \text{if } mp \to mp' \qquad \text{(simp-mp)}$$

$$\{\bot_{mp}\} \uplus pp \Rightarrow pp \qquad \text{(remove-mp)}$$

$$\{\emptyset\} \uplus pp \Rightarrow \top_{pp} \qquad \text{(success)}$$

Finally we provide rules $\Rrightarrow$ for modifying sets of pattern problems.

$$\{pp\} \uplus P \Rrightarrow \{pp'\} \cup P \qquad \text{if } pp \Rightarrow pp' \qquad \text{(simp-pp)}$$
$$\{\emptyset\} \uplus P \Rrightarrow \bot_P \qquad \text{(failure)}$$
$$\{\top_{pp}\} \uplus P \Rrightarrow P \qquad \text{(remove-pp)}$$
$$\{pp\} \uplus P \Rrightarrow \{pp\sigma_{x,c} \mid c \in \mathcal{C}_\tau\} \cup P \qquad \text{if } mp \in pp,\ (x, f(\ldots)) \in mp, \text{ and } x \in \mathcal{V}_\tau \qquad \text{(instantiate)}$$

Here, $\mathcal{C}_\tau$ is the set of constructors with target type τ and $\sigma_{x,c}$ is a substitution which just replaces x by $c(x_1, \ldots, x_n)$ where n is the arity of c and $x_1, \ldots, x_n$ are fresh and distinct variables. The pattern problem $pp\sigma_{x,c}$ is obtained from pp by changing every pair (t, ℓ) in every matching problem of pp to $(t\sigma_{x,c}, \ell)$.

Clearly, (decompose) and (clash) correspond to a standard matching algorithm. Similarly, (match) is standard for matching with linear lhss, but will cause problems in the non-linear case. Nearly all of the other rules mainly correspond to the universal and existential quantification that is done in the definition of solvability. The only exception is (instantiate). Here a matching algorithm would detect a failure since a variable x is never matched by a non-variable term $f(\ldots)$. However, since the x in our setting just represents an arbitrary constructor ground term, we need to make a case analysis on the outermost constructor. This is done by replacing $x \in \mathcal{V}_\tau$ by all possible constructor ground terms of shape $c(x_1, \ldots, x_n)$ for all $c \in \mathcal{C}_\tau$.

The following theorem states that $\Rrightarrow$ can be used to decide linear pattern problems. Here, $\Rrightarrow^!$ is defined as reduction to normal form, i.e., $P \Rrightarrow^! P'$ iff $P \Rrightarrow^* P' \wedge \neg\exists P''.\ P' \Rrightarrow P''$.

Theorem 7 (Decision Procedure for Solvability of Linear Pattern Problems)**.**

- *$\Rrightarrow$ is terminating.*
- *Whenever $P \Rrightarrow^! P'$ then $P' \in \{\emptyset, \bot_P\}$.*
- *Whenever P is linear and $P \Rrightarrow P'$ then P' is linear, and P is solvable iff P' is solvable.*
- *Whenever P is linear then P is solvable iff $P \Rrightarrow^! \emptyset$.*

So, solvability of linear pattern problems is decidable. Regarding the complexity, one can prove an exponential upper bound on the number of $\Rrightarrow$-steps. However, there might be room for improvement: for all examples we considered so far, there also is a strategy such that only polynomially many $\Rrightarrow$-steps are required, e.g., by changing the order of variables on which (instantiate) is applied.

Example 8. *The algorithm validates that $\mathcal{R}_\mathbb{N}$ in Example 2 is pattern complete. In the execution of the algorithm we interpret sets as multisets.*

$$\begin{aligned}
P = \{\ & \{\{(\mathsf{even}(y), \mathsf{even}(0))\}, \{(\mathsf{even}(y), \mathsf{even}(\mathsf{s}(0)))\}, \{(\mathsf{even}(y), \mathsf{even}(\mathsf{s}(\mathsf{s}(x))))\}\}\ \} \\
\Rrightarrow^3\ & \{\ \{\{(y, 0)\}, \{(y, \mathsf{s}(0))\}, \{(y, \mathsf{s}(\mathsf{s}(x)))\}\}\ \} \\
\Rrightarrow\ & \{\ \{\{(0, 0)\}, \{(0, \mathsf{s}(0))\}, \{(0, \mathsf{s}(\mathsf{s}(x)))\}\},\ \{\{(\mathsf{s}(z), 0)\}, \{(\mathsf{s}(z), \mathsf{s}(0))\}, \{(\mathsf{s}(z), \mathsf{s}(\mathsf{s}(x)))\}\}\ \} \\
\Rrightarrow^6\ & \{\ \{\emptyset, \bot_{mp}, \bot_{mp}\},\ \{\bot_{mp}, \{(z, 0)\}, \{(z, \mathsf{s}(x))\}\}\ \} \\
\Rrightarrow^3\ & \{\ \{\{(z, 0)\}, \{(z, \mathsf{s}(x))\}\}\ \} \\
\Rrightarrow\ & \{\ \{\{(0, 0)\}, \{(0, \mathsf{s}(x))\}\},\ \{\{(\mathsf{s}(y), 0)\}, \{(\mathsf{s}(y), \mathsf{s}(x))\}\}\ \} \\
\Rrightarrow^5\ & \{\ \{\emptyset, \bot_{mp}\},\ \{\bot_{mp}, \emptyset\}\ \} \\
\Rrightarrow^4\ & \emptyset
\end{aligned}$$

3 Pattern Completeness – The General Case

For achieving soundness for the non-linear case we have to modify the (match) rule.

Definition 9 (Match Rule for the General Case)**.**

$$\{(t,x)\} \uplus mp \to mp \qquad \textit{if for all } (t',\ell) \in mp,\ x \textit{ does not occur } \ell \qquad \text{(match')}$$

In the linear case the additional occurrence check in rule (match') is always satisfied, so we can still simulate the algorithm for the linear case with this modified rule. Soundness and termination of $\Rrightarrow$ still are satisfied, even for non-linear inputs.

However, after the switch from rule (match) to (match') it can happen that $\Rrightarrow$ gets stuck, e.g., if there is a matching problem $\{(t,x),(t',x)\}$ for $t \neq t'$. To treat these cases we have to add further simplification rules. In order to do so, we need to distinguish between finite and infinite types τ, i.e., whether the set $\mathcal{T}(\mathcal{C})_\tau$ is finite or infinite. To illustrate the problem, consider a program with three left-hand sides: $f(x,x,y)$, $f(x,y,x)$, and $f(y,x,x)$. If x is a variable of a finite type that just allows two different values, such as the Booleans, then these left-hand sides cover all cases. If the type has infinitely many values, such as lists, then the left-hand sides do not suffice, indicating an unsolvable problem. So, we must be able to instantiate in the finite-type case. However, we cannot allow an instantiation in the infinite-type case, since otherwise the resulting inference rules would no longer be terminating.

As final preparation for the new inference rules we define (the only two) reasons on why two terms differ. We say that terms $t \neq t'$ clash if $t|_p = f(\dots) \neq g(\dots) = t'|_p$ with $f \neq g$ for some shared position p of t and t'. The terms $t \neq t'$ differ in variable y if $t|_p \neq t'|_p$ and $y \in \{t|_p, t'|_p\}$ for some shared position p.

Definition 10 (Inference Rules for General Pattern Problems)**.** *We take all rules of the linear algorithm with the following modifications.*

- *Rule (match) is replaced by (match').*
- *We add the following three rules to avoid to get stuck.*

$$\{(t,x),(t',x)\} \uplus mp \to \bot_{mp} \qquad \textit{if } t \textit{ and } t' \textit{ clash} \qquad \text{(clash')}$$

$$\{pp\} \uplus P \Rrightarrow \{pp\sigma_{x,c} \mid c \in \mathcal{C}_\tau\} \cup P \qquad \text{(instantiate')}$$
$$\textit{if } mp \in pp,\ \{(t,y),(t',y)\} \subseteq mp,\ t \textit{ and } t' \textit{ differ in variable } x \in \mathcal{V}_\tau, \textit{ and } \tau \textit{ is finite}$$

$$\{pp\} \uplus P \Rrightarrow \bot_P \qquad \textit{if for each } mp \in pp \textit{ there are } \{(t,y),(t',y)\} \subseteq mp \qquad \text{(failure')}$$
$$\textit{such that } t \textit{ and } t' \textit{ differ in variable } x \in \mathcal{V}_\tau \textit{ and } \tau \textit{ is infinite}$$

Indeed, with these modifications, $\Rrightarrow$ cannot get stuck even for non-linear inputs.

We first remark that there is a different flavour of problems with non-linear matching problems of the form $\{(t,x),(t',x)\}$. Clashing of t and t' can always be resolved locally. If there is a difference of a finite-type variable, this can also be handled immediately by (instantiate'). However, differences of infinite-type variables can only be applied via (failure') if indeed all matching problems show such a difference. Note that it is unsound to turn (failure') into a local rule for matching problems, i.e., if we would make (failure') similar to (clash').

Overall, we arrive at a similar theorem to the linear case, though its proof is much more evolved. It has been proven in Isabelle (2700 lines), based on IsaFoR[1] and on a library on sorted terms by Akihisa Yamada.

[1] http://cl-informatik.uibk.ac.at/isafor/

Theorem 11 (Decision Procedure for Solvability of Pattern Problems)**.**

- $\Rrightarrow$ *is terminating.*
- *Whenever* $P \Rrightarrow^! P'$ *then* $P' \in \{\emptyset, \bot_P\}$.
- *Whenever* $P \Rrightarrow P'$ *then* P *is solvable iff* P' *is solvable.*
- P *is solvable iff* $P \Rrightarrow^! \emptyset$.

The formalization in Isabelle also contains a verified list-based implementation of the abstract inference rules. It fixes a strategy where first $\Rightarrow$-steps are applied exhaustively. Rules (instantiate) and in particular (instantiate') get applied as late as possible. However, this implementation is not fully working, as it expects a function to compute whether a given type τ is infinite or not, and we did not yet verify a suitable algorithm for this subtask.

Note that it is possible to extend $\Rrightarrow$ in a way that it provides a witness constructor ground substitution in case a pattern problem is not solvable. To this end one has to store the substitutions that have been applied via the two rules for instantiation; and in case rule (failure') has been used, a final constructor-ground substitution can be generated by following the construction in the soundness proof of that rule.

It remains open whether a similar syntax directed decision procedure for quasi-reducibility can be designed, i.e., without an explicit enumeration of terms.

Acknowledgements We thank the anonymous reviewers for their helpful remarks and for their references to related work.

References

[1] Takahito Aoto and Yoshihito Toyama. Ground confluence prover based on rewriting induction. In *Proc. FSCD 2016*, volume 52 of *LIPIcs*, pages 33:1–33:12, 2016.

[2] Adel Bouhoula. Simultaneous checking of completeness and ground confluence for algebraic specifications. *ACM Trans. Comput. Log.*, 10(3):20:1–20:33, 2009.

[3] Adel Bouhoula and Florent Jacquemard. Sufficient completeness verification for conditional and constrained TRS. *J. Appl. Log.*, 10(1):127–143, 2012.

[4] Stephan Falke and Deepak Kapur. Rewriting induction + linear arithmetic = decision procedure. In *Proc. IJCAR 2012*, volume 7364 of *LNCS*, pages 241–255, 2012.

[5] Deepak Kapur, Paliath Narendran, and Hantao Zhang. On sufficient-completeness and related properties of term rewriting systems. *Acta Informatica*, 24(4):395–415, 1987.

[6] Cynthia Kop. Quasi-reductivity of logically constrained term rewriting systems. *CoRR*, abs/1702.02397, 2017.

[7] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[8] Uday S. Reddy. Term rewriting induction. In *Proc. CADE 1990*, volume 449 of *LNCS*, pages 162–177, 1990.

Confluence Competition 2023

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

[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 12th Confluence Competition (CoCo 2023). 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 [1]. This year there were 15 tools (listed in order of registration) participating in 10 categories (listed in order of first appearance in CoCo):

	TRS	CTRS	GCR	UNR	UNC	NFP	COM	INF	SRS	CSR
ConfCSR										✓
Toma								✓		
Hakusan	✓								✓	
CoLL							✓			
CO3		✓						✓		
NaTT								✓		
CSI	✓			✓	✓	✓			✓	
FORT-h	✓		✓	✓	✓	✓	✓			
FORTify	⋆		⋆	⋆	⋆	⋆	⋆			
CONFident	✓	✓							✓	✓
infChecker								✓		
AGCP			✓							
ACP	✓	✓		✓	✓		✓		✓	
nonreach								✓		
CeTA	⋆						⋆	⋆	⋆	

New this year is that the certified categories have been disabled; tools producing certifiable output in a specific category team up with a certifier and participate as combination in that category. The certifiers in CoCo 2023 are CeTA and FORTify. The latter teams 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 nonreach in the INF category.

The winning (for combined YES/NO answers) tools[1] of CoCo 2022 participated as demonstration tools, to provide a benchmark to measure progress. The live run of CoCo 2023 on StarExec [2] can be viewed at http://cocograph.uibk.ac.at/2023.html. Further information about CoCo 2023, including a description of the categories and detailed results, can be obtained from

http://project-coco.uibk.ac.at/2023/

[1] They are not listed in the table but see http://project-coco.uibk.ac.at/2022/results.php.

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

References

[1] 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.

[2] 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 2023 Participant: ConfCSR

Filip Stevanovic and Fabian Mitterwallner

Department of Computer Science, University of Innsbruck, Austria
fil.stevanovic@gmail.com, fabian.mitterwallner@uibk.ac.at

ConfCSR is a tool for automatically (dis)proving confluence of context-sensitive rewrite systems (CSTRSs). It was developed as part of a bachelor project at the University of Innsbruck. ConfCSR implements techniques described in [1], namely orthogonality and (non-)joinability of extended μ-critical pairs. The latter also requires proving termination of the CSTRS, for which an external termination tool such as AProVE2 [2] is used.

The procedure for determining joinability of μ-critical pairs searches for common reducts. However, the extended μ-critical pairs also contain so called LH_μ-critical pairs, which are obtained from rules where some variables appear in both active and frozen positions. They take the shape $\ell[x']_p \approx r \Leftarrow x \to x'$, where $\ell \to r$ is a rule of the CSTRS R and x is a variable at an active position p in ℓ which also appears in a frozen position. Such a critical pair is called joinable if $\ell[x']_p\sigma \to \cdot \leftarrow r\sigma$ for all substitutions where $x\sigma \to x'\sigma$. Therefore checking joinability cannot be done via a straight forward search. Instead ConfCSR replaces occurrences of x' and x by fresh constants d and c respectively, and checks if $\ell[x']_p\{x' \mapsto d, x \mapsto c\}$ and $r\{x \mapsto c\}$ are joinable with respect to the CSTRS $R \cup \{c \to d\}$. This criterion is sufficient to check joinability but is not complete.

The source code is freely available at

https://github.com/F200907/ConfCSR

ConfCSR participates in the CSR category of CoCo 2023.

References

[1] 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. doi: 10.1016/j.jlamp.2022.100749.

[2] Jürgen Giesl, Cornelius Aschermann, Marc Brockschmidt, Fabian Emmes, Florian Frohn, Carsten Fuhs, Jera Hensel, Carsten Otto, Martin Plücker, Peter Schneider-Kamp, Thomas Ströder, Stephanie Swiderski, and René Thiemann. Analyzing Program Termination and Complexity Automatically with AProVE. *Journal of Automated Reasoning*, 58(1):3–31, 2017. doi: 10.1007/s10817-016-9388-y.

Toma 0.5: An Equational Theorem Prover

Teppei Saito and Nao Hirokawa

JAIST, Japan

Toma is an automatic theorem prover for first-order equational systems, freely available at https://www.jaist.ac.jp/project/maxcomp/. The typical usage is: `toma --inf <file>`, where `<file>` is an infeasibility problem in the CoCo format [5]. The tool outputs `YES` if infeasibility of the problem is shown, and `MAYBE` otherwise. It also accepts the TPTP CNF format [6].

Toma proves infeasibility as follows: By using the *split-if* encoding [2] a given infeasibility problem is transformed into a word problem of form $\mathcal{E} \vdash \mathsf{T} \not\approx \mathsf{F}$ whose validity entails infeasibility of the original problem. The word problem is solved by a new variant of maximal (ordered) completion [7, 3]:

1. Given an equational system $\mathcal{E}_1$, we construct a lexicographic path order $\succ_{\mathsf{lpo}}$ that maximizes reducibility of the ordered rewrite system $(\mathcal{E}_1, \succ_{\mathsf{lpo}})$ [7].
2. Using the order, we run ordered completion [1] on $\mathcal{E}_1$. Here we do not employ the deduce rule (critical pair generation). Such a run eventually ends with an inter-reduced version $(\mathcal{E}_2, \succ_{\mathsf{lpo}})$ of $(\mathcal{E}_1, \succ_{\mathsf{lpo}})$.
3. The tool checks ground-completeness of the ordered rewrite system $(\mathcal{E}_2, \succ_{\mathsf{lpo}})$ by Martin and Nipkow's method [4].
   (a) If $(\mathcal{E}_2, \succ_{\mathsf{lpo}})$ is ground-complete but T and F are not joinable, the tool outputs `YES` and terminates.
   (b) If T and F are joinable in $(\mathcal{E}_2, \succ_{\mathsf{lpo}})$, the tool outputs `MAYBE` and terminates.
   (c) Otherwise, there exists at least one equation that is valid in $\mathcal{E}_2$ but not ground-joinable in $(\mathcal{E}_2, \succ_{\mathsf{lpo}})$. Let $\mathcal{E}_3$ be a set of such equations. Setting $\mathcal{E}_1 := \mathcal{E}_2 \cup \mathcal{E}_3$, the tool goes back to the first step.

Compared to the last year's version, the performance has been improved.

References

[1] L. Bachmair, N. Dershowitz and D. A. Plaisted. Completion without Failure. *Resolution of Equations in Algebraic Structures vol. 2: Rewriting* , pp. 1–30, Academic Press, 1989.

[2] K. Claessen and N. Smallbone. Efficient Encodings of First-Order Horn Formulas in Equational Logic. *Proc. 9th IJCAR*, LNCS 10900, pp. 388–404, 2018.

[3] N. Hirokawa. Completion and Reduction Orders. *Proc. 6th FSCD*, LIPIcs, vol. 195, pp. 2:1-2:9, 2021.

[4] U. Martin and T. Nipkow. Ordered Rewriting and Confluence. *Proc. 10th CADE*, LNCS 499, pp. 366–380, 1990.

[5] A. Middeldorp, J. Nagele, and K. Shintani. Confluence Competition 2019. *Proc. 25th TACAS*, LNCS 11429, pp. 25–40, 2019.

[6] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure: From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning, vol. 59*, no. 4, pp. 483–502, 2017.

[7] S. Winkler and G. Moser. Mædmax: A Maximal Ordered Completion Tool. *Proc. 9th IJCAR*, LNCS 10900, pp. 472–480, 2018.

Hakusan 0.8: A Confluence Tool

Kiraku Shintani and Nao Hirokawa

JAIST, Japan
s1820017@jaist.ac.jp, hirokawa@jaist.ac.jp

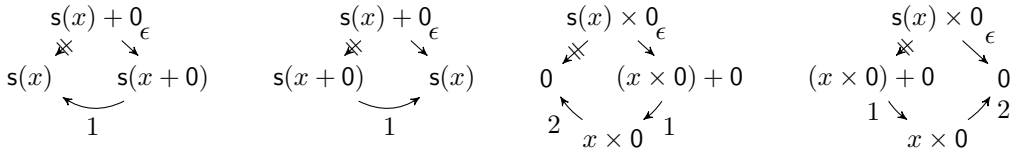
Hakusan (http://www.jaist.ac.jp/project/saigawa/) is a confluence tool for left-linear term rewrite systems (TRSs). It analyzes confluence by using the two *compositional* confluence criteria [2, Theorems 31 and 38] that originate from rule labeling and critical pair systems. This version supports two new features. One is certificate outputs for rule labeling [2, Theorem 28] which are verifiable by CeTA [3], and the other is the following *reduction method* for confluence problems (see the extended version of [2]). Let $\mathcal{R}\restriction_{\mathcal{C}} = \{\ell \to r \in \mathcal{R} \mid \mathcal{F}\mathsf{un}(\ell) \subseteq \mathcal{F}\mathsf{un}(\mathcal{C})\}$.

Theorem 1. *Let $\mathcal{C}$ be a subsystem of a left-linear TRS $\mathcal{R}$. Suppose ${}_{\mathcal{R}}{\leftarrowtail\!\!\rtimes}\xrightarrow{\epsilon}_{\mathcal{R}} \subseteq \leftrightarrow^*_{\mathcal{C}}$ and $\mathcal{R}\restriction_{\mathcal{C}} \subseteq \to^*_{\mathcal{C}}$. The TRS $\mathcal{R}$ is confluent if and only if $\mathcal{C}$ is confluent.*

To demonstrate the reduction method, we show the confluence of the left-linear TRS $\mathcal{R}$:

$$\begin{array}{lll} 1\colon\ x + 0 \to x & 3\colon\ \ 0 + y \to y & 5\colon\ \mathsf{s}(x) + y \to \mathsf{s}(x + y) \\ 2\colon\ x \times 0 \to 0 & 4\colon\ \mathsf{s}(x) \times 0 \to 0 & 6\colon\ \mathsf{s}(x) \times y \to (x \times y) + y \end{array}$$

There are four non-trivial parallel critical pairs and they admit the following diagrams:



(i) Let $\mathcal{C} = \{1, 2, 3\}$. We have ${}_{\mathcal{R}}{\leftarrowtail\!\!\rtimes}\xrightarrow{\epsilon}_{\mathcal{R}} \subseteq \leftrightarrow^*_{\mathcal{C}}$. As $\mathcal{F}\mathsf{un}(\mathcal{C}) = \{0, +, \times\}$, the inclusion $\mathcal{R}\restriction_{\mathcal{C}} = \{1, 2, 3\} \subseteq \to^*_{\mathcal{C}}$ holds. According to Theorem 1, the confluence problem of $\mathcal{R}$ is reduced to that of $\mathcal{C}$.

(ii) Since $\mathcal{C}$ only admits a trivial parallel critical pair, it is closed by the empty system $\varnothing$. Moreover, the inclusion $\mathcal{C}\restriction_{\varnothing} = \varnothing \subseteq \to^*_{\varnothing}$ holds. Hence, by Theorem 1 confluence of $\mathcal{C}$ is reduced to that of the empty system $\varnothing$.

(iii) Since the empty system $\varnothing$ is trivially confluent, we conclude that $\mathcal{R}$ is confluent.

As a final remark, our tool employs the SMT solver Z3 [1] and the termination tool NaTT [4] for automating the compositional confluence criteria and the reduction method.

References

[1] L. de Moura and N. Bjørner. Z3: An Efficient SMT Solver. In *Proc. 12th TACAS*, volume 4963 of LNCS, pages 337–340, 2008.

[2] K. Shintani and N. Hirokawa. Compositional Confluence Criteria. In *Proc. 7th FSCD*, volume 228 of *LIPIcs*, pages 28:1–28:19, 2022. The extended version, submitted to a journal, is available at: doi: 10.48550/arXiv.2303.03906

[3] R. Thiemann and C. Sternagel. Certification of Termination Proofs using CeTA. In *Proc. 22nd TPHOLs*, volume 5674 of LNCS, pages 452–468, 2009.

[4] A. Yamada and K. Kusakari and T. Sakabe. Nagoya Termination Tool. In *Proc. 25th RTA*, volume 8560 of *LNCS*, pages 446–475, 2014.

CoLL 1.6.1: A Commutation Tool

Kiraku Shintani

JAIST, Japan
s1820017@jaist.ac.jp

CoLL (version 1.6.1) is a tool for automatically proving **co**mmutation of **l**eft-**l**inear term rewrite systems (TRSs). The tool, written in OCaml, is freely available at:

http://www.jaist.ac.jp/project/saigawa/coll/

The typical usage is: `coll <file>`. Here the input file is written in the commutation problem format [4]. The tool outputs `YES` if commutation of the input TRSs is proved, `NO` if non-commutation is shown, and `MAYBE` if the tool does not reach any conclusion.

In this tool commutation of left-linear TRSs is shown by *Hindley's Commutation Theorem*:

Theorem 1 ([2, 7])**.** *ARSs* $\mathcal{A} = \langle A, \{\to_\alpha\}_{\alpha\in I}\rangle$ *and* $\mathcal{B} = \langle A, \{\to_\beta\}_{\beta\in J}\rangle$ *commute if* $\to_\alpha$ *and* $\to_\beta$ *commute for all* $\alpha \in I$ *and* $\beta \in J$*.*

Here indexes are interpreted as subsystems of the input TRSs. For every pair of subsystems CoLL proves its commutation property, employing the following criteria: simultaneous closedness [5], parallel closedness [9], parallel upside closedness and outside closedness [6], rule labeling with weight function [10, 1], and Church–Rosser modulo A/C [3]. A detailed description of CoLL can be found in [8].

References

[1] T. Aoto. Automated confluence proof by decreasing diagrams based on rule-labelling. In *Proc. 21st RTA*, volume 6 of *LIPIcs*, pages 7–16, 2010.

[2] J. R. Hindley. *The Church-Rosser Property and a Result in Combinatory Logic*. PhD thesis, University of Newcastle-upon-Tyne, 1964.

[3] 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.

[4] A. Middeldorp, J. Nagele, and K. Shintani. CoCo 2019: Report on the eighth confluence competition. *International Journal on Software Tools for Technology Transfer*, 23(6):905–916, 2021.

[5] S. Okui. Simultaneous critical pairs and Church–Rosser property. In *Proc. 9th RTA*, volume 1379 of *LNCS*, pages 2–16, 1998.

[6] M. Oyamaguchi and Y. Ohta. On the open problems concerning Church–Rosser of left-linear term rewriting systems. *IEICE Transactions on Information and Systems*, 87(2):290–298, 2004.

[7] B. K. Rosen. Tree-manipulating systems and Church-Rosser theorems. *Journal of the ACM*, 20:160–187, 1973.

[8] K. Shintani and N. Hirokawa. CoLL: A confluence tool for left-linear term rewrite systems. In *Proc. 25th CADE*, volume 9195 of *LNAI*, pages 127–136, 2015.

[9] Y. Toyama. On the Church–Rosser property of term rewriting systems. NTT ECL Technical Report, No.17672, NTT, 1981.

[10] V. van Oostrom. Confluence by decreasing diagrams converted. In A. Voronkov, editor, *Proc. 19th RTA*, volume 5117 of *LNCS*, pages 306–320, 2008.

CO3 (Version 2.4)

Naoki Nishida, Misaki Kojima, and Ayuka Matsumi

Nagoya University, Nagoya, Japan
{nishida@, k-misaki@trs.css., matsumi@trs.css.}i.nagoya-u.ac.jp

CO3, a converter for proving confluence of conditional TRSs,[1] tries to prove confluence of conditional term rewrite systems (CTRSs, for short) by using a transformational approach (cf. [7]). 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 $\mathbb{U}_{conf}$ [3], a variant of the *unraveling* $\mathbb{U}$ [9], 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 [2, 3]. 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 [8, 4] to prove infeasibility of a condition w.r.t. a CTRS has been implemented [5]. 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 prove 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 [6].

This version has an improvement on the removal of valid conditions: For a conditional rule $\ell \to r \Leftarrow c, s \twoheadrightarrow t, c' \in \mathcal{R}$, if there exist an unconditional rule $\ell' \to r' \in \mathcal{R}$ and a substitution θ such that $\ell'\theta = s$ and $r'\theta = t$, the condition $s \twoheadrightarrow t$ is dropped from the conditional part, replacing the rule by $\ell \to r \Leftarrow c, c'$. In addition, we slightly strengthen the function to disprove confluence: In proving strong irreducibility of a term t, if a subterm u of t is unifiable with the left-hand side of a rule $\ell \to r \Leftarrow c$ by means of an mgu θ, then we check infeasibility of $c\theta$; if $c\theta$ is infeaible, then the rule is considered to be inapplicable to u.

References

[1] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *J. Autom. Reason.*, 37(3):155–203, 2006.

[2] K. Gmeiner, B. Gramlich, and F. Schernhammer. On soundness conditions for unraveling deterministic conditional rewrite systems. In *Proc. RTA 2012*, vol. 15 of *LIPIcs*, pp. 193–208, 2012.

[3] K. Gmeiner, N. Nishida, and B. Gramlich. Proving confluence of conditional term rewriting systems via unravelings. In *Proc. IWC 2013*, pp. 35–39, 2013.

[4] Y. Maeda, N. Nishida, M. Sakai, and T. Kobayashi. Extending narrowing trees to basic narrowing in term rewriting. IEICE Tech. Rep. SS2018-39, Vol. 118, No. 385, pp. 73–78, 2019, in Japanese.

[5] N. Nishida. CO3 (Version 2.1). In *Proc. IWC 2020*, page 67, 2020.

[6] N. Nishida. CO3 (Version 2.2). In *Proc. IWC 2021*, page 61, 2021.

[7] N. Nishida, T. Kuroda, and K. Gmeiner. CO3 (Version 1.3). In *Proc. IWC 2016*, p. 74, 2016.

[8] N. Nishida and Y. Maeda. Narrowing trees for syntactically deterministic conditional term rewriting systems. In *Proc. FSCD 2018*, vol. 108 of *LIPIcs*, pp. 26:1–26:20, 2018.

[9] E. Ohlebusch. Termination of logic programs: Transformational methods revisited. *Appl. Algebra Eng. Commun. Comput.*, 12(1/2):73–116, 2001.

[1] http://www.trs.css.i.nagoya-u.ac.jp/co3/

CoCo 2023 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 [4]. Some of the implemented techniques are described in [1,3,7]. CSI can also produce certificates for confluence results, which are checked by CeTA. Compared to last year's version, CSI 1.2.7 can now produce certificates containing proofs based on *almost* development-closed critical pairs, which is a sufficient condition for confluence of left-linear TRSs [5]. This extends the previous certification of development-closed critical pairs. These certificates can be checked by the latest version of CeTA [6], due to the formalization and certification efforts by Christina Kohl, parts of which are described in [2].

CSI participates in the following CoCo 2023 categories: NFP, SRS, TRS, UNC, and UNR. Additionally, it participates together with CeTA in the TRS category providing certified YES/NO answers.

References

[1] Bertram Felgenhauer. Confluence for Term Rewriting: Theory and Automation. PhD thesis, University of Innsbruck, 2015.

[2] Christina Kohl and Aart Middeldorp. Formalizing almost development closed critical pairs. In *Proc. 14th International Conference on Interactive Theorem Proving*, pages 38:1–38:8, 2023. doi: 10.4230/LIPIcs.ITP.2023.38.

[3] Julian Nagele. Mechanizing Confluence: Automated and Certified Analysis of First- and Higher-Order Rewrite Systems. PhD thesis, University of Innsbruck, 2017.

[4] 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.

[5] Vincent van Oostrom. Developing Developments. *Theoretical Computer Science*, 175(1):159–181, 1997. doi: 10.1016/S0304-3975(96)00173-9.

[6] René Thiemann, Christina Kohl, and Dohan Kim. CoCo 2023 Participant: CeTA 2.45. In *Proc. 12th International Workshop on Confluence*, 2023. This volume.

[7] Harald Zankl. Challenges in Automation of Rewriting. Habilitation thesis, University of Innsbruck, 2014.

CoCo 2023 Participant: FORT-h 2.0*

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 2.0 can be downloaded from

http://fortissimo.uibk.ac.at/fort(ify)/

FORT-h participates in the following CoCo 2023 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.

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 2023 Participant: FORTify 2.0. In *Proc. 12th International Workshop on Confluence*, 2023. 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.

*Supported by FWF (Austrian Science Fund) project P30301.

CoCo 2023 Participant: FORTify 2.0

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 2023 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 2023 Participant: FORT-h 2.0. In *Proc. 12th International Workshop on Confluence*, 2023. This volume.

CONFident at the 2023 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

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, 5] and mechanizing them by external tools like MU-TERM [3], infChecker [1], AGES [2], Prover9 and Mace4 [7] and Barcelogic[1]. Latest description of the tool can be found in [4].

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] 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, 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.

[5] S. Lucas. Proving semantic properties as first-order satisfiability. *Artificial Intelligence* 277, paper 103174, 24 pages, 2019.

[6] S. Lucas and R. Gutiérrez. Use of Logical Models for Proving Infeasibility in Term Rewriting. *Information Processing Letters*, 136:90–95, 2018.

[7] 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.

[1] https://barcelogic.com/

infChecker at the 2023 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

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.

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 2023

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. 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 $u\theta_g \underset{\succsim}{\overset{*}{\longleftrightarrow}}_{\mathcal{R}} v\theta_g$ for any its ground instance $u\sigma_g \approx v\sigma_g$, where $x \underset{\succsim}{\overset{*}{\longleftrightarrow}} y$ 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].

No new ground (non-)confluence criterion has been incorporated from the one submitted for CoCo 2022.

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 2023

Takahito Aoto

Institute of Science and Technology, Niigata University
aoto@ie.niigata-u.ac.jp

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. In this year, we newly participates to the UNR category; some methods for (dis)proving the UNR property employed in our prover are described in [4]. We have also added the facility for generating some proofs in CPF format in commutativity (dis)proving.

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] K. Okamoto, A Method for Automatically Verifying Unique Normal Form Property w.r.t. Reduction. Bachlor's thesis, Niigata University, 2020.

[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 2023 Participant: nonreach 1.2

Florian Meßner

Independent Researcher, Innsbruck, Austria florian.messner@outlook.com

The tool nonreach is an automated, efficient tool to check infeasibility with respect to oriented conditional term rewrite systems (CTRSs). The Haskell source code can be obtained from a public *git* repository hosted on *bitbucket*:

https://bitbucket.org/fmessner/nonreach

Given a CTRS (or a TRS) and one or more infeasibility problems, nonreach uses a combination of *decomposition*, based on narrowing (with some heuristics) and proving root-nonreachability [4], and *fast checks*, based on etcap [5] and the *inductive symbol transition graph* [4].

These methods are applied alternately until nonreach either obtains infeasibility (by simplifying the tree to False), finds a satisfying substitution, or reaches a user-defined threshold of iterations (and concludes MAYBE).

For more details regarding the implementation and usage of nonreach, I refer to the tool demonstration paper published in TACAS 2019 [1] and my master thesis [2].

I previously participated with nonreach in the INF categories of CoCo 2019 and CoCo 2020 where it earned the second and third place respectively. Additionally, in 2020 the participant ConCon [3] used nonreach as an external tool and earned the second place in the INF category, as well as the first place in both the CTRS and CPF-CTRS categories.

Compared to the version participating in CoCo 2020, the new version nonreach 1.2 is mainly a refactoring release. However, the new competition rules of CoCo 2023 allow me to showcase the certified results of nonreach by running it together with CeTA [6].

References

[1] Florian Meßner and Christian Sternagel. nonreach - A tool for nonreachability analysis. In *Proc. 25th TACAS*, pages 337–343, 2019. doi:10.1007/978-3-030-17462-0_19.

[2] Florian Meßner. Automated Conditional Nonreachability Master Thesis, University of Innsbruck, 2020 https://diglib.uibk.ac.at/ulbtirolhs/content/titleinfo/5489915.

[3] Christian Sternagel. CoCo 2020 Participant: ConCon 1.10 In *Proc. 9th IWC*, page 65, 2020. http://iwc2020.cic.unb.br/iwc2020_proceedings.pdf.

[4] Christian Sternagel and Akihisa Yamada. Reachability analysis for termination and confluence of rewriting. In *Proc. 25th TACAS*, pages 262–278, 2019. doi:10.1007/978-3-030-17462-0_15.

[5] 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 *LNCS*, pages 452–468. Springer, 2009. doi:10.1007/978-3-642-03359-9_31.

[6] Christina Kohl, René Thiemann, and Aart Middeldorp CoCo 2022 Participant: CeTA 2.42 In *Proc. 11th IWC*, page 62, 2022. http://cl-informatik.uibk.ac.at/iwc/2022/proceedings.pdf.

CoCo 2023 Participant: CeTA 2.45

René Thiemann, Christina Kohl, and Dohan Kim

Department of Computer Science, University of Innsbruck, Austria

The tool CeTA [4] 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 (2.42) to this year's version (2.45). 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/

The development closedness criterion for confluence of left-linear TRSs has now been extended to *almost* development closed critical pairs, allowing a weaker joining condition for overlays [5, Corollary 28]. This result has also been extended for showing commutation of left-linear TRSs. Hence, CeTA now fully supports the results described in [5] for the first-order case. The Isabelle formalization for these extensions is described in [2].

Confluence criteria along with commutation criteria using *parallel critical pairs* [1] have been fully formalized and added to CeTA. In addition to the parallel critical pair condition, CeTA also supports rule labeling with parallel critical pairs for confluence and commutation [6]. Furthermore, compositional confluence criteria as discussed in [3] have been fully formalized and added to CeTA. Note that these confluence criteria subsume many well-known existing confluence criteria as corollaries.

Finally, the following changes have also been added to the current version of CeTA: (i) infeasibility proofs are supported as top-level proof obligations, (ii) decision procedure for (innermost-)right ground termination, (iii) improved bound for solving linear integer arithmetic constraints, (iv) improved efficiency of RPO implementation from cubic to quadratic (for fixed signature), and (v) improved efficiency of WPO implementation from exponential to polynomial.

References

[1] Bernhard Gramlich. Confluence without Termination via Parallel Critical Pairs. In *Proc. 21st International Colloquium on Trees in Algebra and Programming*, volume 1059 of Lecture Notes in Computer Science, pages 211–225, 1996. doi: 10.1007/3-540-61064-2_39.

[2] Christina Kohl and Aart Middeldorp. Formalizing Almost Development Closed Critical Pairs. In *Proc. 14th International Conference on Interactive Theorem Proving*, LIPIcs, volume 268, pages 38:1–38:8, 2023. doi: 10.4230/LIPIcs.ITP.2023.38.

[3] Kiraku Shintani and Nao Hirokawa. Compositional Confluence Criteria. In *Proc. 7th International Conference on Formal Structures for Computation and Deduction*, LIPIcs, volume 228, pages 28:1–28:19, 2022. doi: 10.4230/LIPIcs.FSCD.2022.28.

[4] 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.

[5] Vincent van Oostrom. Developing Developments. *Theoretical Computer Science*, 175(1):159–181, 1997. doi: 10.1016/S0304-3975(96)00173-9.

[6] Harald Zankl and Bertram Felgenhauer and Aart Middeldorp. Labelings for Decreasing Diagrams. In Journal of Automated Reasoning, 54(2):101–133, 2015. doi: 10.1007/s10817-014-9316-y.