IWC 2014

3rd International Workshop on Confluence

Proceedings

Editors: Takahito Aoto & Delia Kesner

July 13, 2014, Vienna, Austria

Preface

This report contains the proceedings of the *3rd International Workshop on Confluence (IWC 2014)*, which was held in Vienna on July 13, 2014. The workshop is affiliated with the Joint meeting of the 25th International Conference on Rewriting Techniques and Applications and the 12th International Conference on Typed Lambda Calculi and Applications (RTA-TLCA 2014), which is a part of the Federated Logic Conference (FLoC 2014) collocated with the Vienna Summer of Logic (VSL 2014). The 1st IWC took place in Nagoya (2012) and 2nd IWC in Eindhoven (2013).

Confluence provides a general notion of determinism and has been conceived as one of the central properties of rewriting. Confluence relates to many topics of rewriting (completion, modularity, termination, commutation, etc.) and had been investigated in many formalisms of rewriting such as first-order rewriting, lambda-calculi, higher-order rewriting, constrained rewriting, conditional rewriting, etc. Recently there is a renewed interest in confluence research, resulting in new techniques, tool supports, certification as well as new applications. The workshop promotes and stimulates research and collaboration on confluence and related properties. In addition to original contributions, the workshop solicited short versions of recently published articles and papers submitted elsewhere.

IWC 2014 received 7 submissions. Each submission was reviewed by 3 program committee members. After deliberations the program committee decided to accept all submissions, which are contained in this report. Apart from these contributed talks, the workshop had an invited talk by *Beniamino Accattoli* on *On the Formalization of Lambda-Calculus Confluence and Residuals*, and jointly with the 8th International Workshop on Computing with Terms and Graphs (TERMGRAPH 2014), an invited talk by *Samuel Mimram* on *An Introduction to Higher-Dimensional Rewriting Theory.* Their abstracts are also included in the report. Moreover, the 3rd Confluence Competition (CoCo 2014) was held during the workshop and the results are available at `http://coco.nue.riec.tohoku.ac.jp/2014/`.

Several persons helped to make IWC 2014 a success. We are grateful to the members of the program committee for their work. We also thank the members of VSL organizing committee for hosting IWC 2014 in VSL.

Sendai & Paris, June 2014 *Takahito Aoto & Delia Kesner*

Program Committee

Takahito Aoto	RIEC, Tohoku University	(co-chair)
Thibaut Balabonski	Inria Rocquencourt	
Eduardo Bonelli	Universidad Nacional de Quilmes	
Delia Kesner	University Paris - Diderot	(co-chair)
Naoki Nishida	Nagoya University	
Colin Riba	ENS Lyon	
Pierre-Yves Strub	IMDEA Software	
René Thiemann	University of Innsbruck	
Ashish Tiwari	SRI International	

Table of Contents

Author Index

On the Formalization of λ-Calculus Confluence and Residuals

Beniamino Accattoli

Università di Bologna, Italy

The confluence or Church-Rosser theorem is the first result in every course on the λ-calculus. Its standard proof follows Tait and Martin-Löf's technique, based on reducing confluence of β-reduction to the diamond property of its parallel closure. It is reasonably simple and yet non-trivial. Presumably because most proof assistants are built over some functional language, confluence for λ-calculus is the theorem with the highest number of formalized proofs [12, 15, 11, 16, 13, 14, 9, 4, 8, 5, 1].

On the one hand, such a formalization effort has helped to clarify the essence of the proof. On the other hand, the confluence property became a simple and yet significant benchmark for proof assistants, to test the faithfulness of the formalization to the informal, pen-and-paper reasoning on the λ-calculus. Humans can easily (but informally) work *up to equivalence or isomorphism*, while this kind of reasoning is a challenge for proof assistants. Formalizations about languages with binders, as the λ-calculus, have to face the difficulty of reasoning transparently modulo α-equivalence, *i.e.* renaming of bound variables. This issue is so relevant that in past years the *POPLmark challenge* [2]—consisting in the formalization of some theorems in the theory of λ-calculus—was proposed as a challenging benchmark for proof assistants.

In this talk I will introduce the proof assistant Abella [7, 6] and I will show a formalization of confluence that mimics *exactly* the informal reasoning. Then I will compare with formalizations in proof assistants based on different approaches to α-equivalence. Last, I will discuss some variations and refinements, including the similar proof based on finite developments (sometimes credited to Takahashi [17]), and the cube property for *residuals* [10, 3], a stronger form of diamond property for parallel reduction, having an elegant formalization based on a brilliant idea by Gérard Huet [9, 1].

References

[1] Beniamino Accattoli. Proof pearl: Abella formalization of λ-calculus cube property. In *CPP*, pages 173–187, 2012.

[2] Brian E. Aydemir, Aaron Bohannon, Matthew Fairbairn, J. Nathan Foster, Benjamin C. Pierce, Peter Sewell, Dimitrios Vytiniotis, Geoffrey Washburn, Stephanie Weirich, and Steve Zdancewic. Mechanized Metatheory for the Masses: The POPLMark Challenge. In *TPHOLs*, pages 50–65, 2005.

[3] Gérard Berry and Jean-Jacques Lévy. Minimal and optimal computations of recursive programs. In *POPL*, pages 215–226, 1977.

[4] James Brotherston and René Vestergaard. A formalised first-order confluence proof for the λ-calculus using one-sorted variable names. *Inf. Comput.*, 183(2):212–244, 2003.

[5] Joshua Dunfield and Brigitte Pientka. Beluga: a framework for programming and reasoning with deductive systems (system description). In *IJCAR*, pages 15–21, 2010.

[6] Andrew Gacek. The Abella interactive theorem prover (system description). In *IJCAR*, pages 154–161, 2008.

[7] Andrew Gacek. *A framework for specifying, prototyping, and reasoning about computational systems.* PhD thesis, University of Minnesota, September 2009.

[8] Peter V. Homeier. A proof of the Church-Rosser theorem for the λ-calculus in higher order logic. In *TPHOLs'01: Supplemental Proceedings*, pages 207–222, 2001.

[9] Gérard Huet. Residual theory in λ-calculus: A formal development. *J. Funct. Program.*, 4(3):371–394, 1994.

[10] Jean-Jacques Lévy. Réductions correctes et optimales dans le lambda-calcul. Thése d'Etat, Univ. Paris VII, France, 1978.

[11] James McKinna and Robert Pollack. Pure type systems formalized. In *TLCA*, pages 289–305, 1993.

[12] Tobias Nipkow. More Church-Rosser proofs (in Isabelle/HOL). In *Journal of Automated Reasoning*, pages 733–747. Springer, 1996.

[13] Frank Pfenning. A proof of the Church-Rosser theorem and its representation in a logical framework. Technical Report CMU-CS-92-186, Carnegie Mellon University, 1992.

[14] Robert Pollack. Polishing up the Tait-Martin-Löf proof of the Church-Rosser theorem. 1995.

[15] Ole Rasmussen. The Church-Rosser theorem in Isabelle: a proof porting experiment. Technical Report 164, University of Cambridge, 1995.

[16] Natarajan Shankar. A mechanical proof of the Church-Rosser theorem. *J. ACM*, 35(3):475–522, 1988.

[17] Masako Takahashi. Parallel reductions in λ-calculus. *Inf. Comput.*, 118(1):120–127, 1995.

An Introduction to Higher-Dimensional Rewriting Theory

Samuel Mimram[1]

CEA, LIST / École Polytechnique
samuel.mimram@cea.fr

The general methodology of rewriting systems has been applied to various settings: strings, terms, graphs, etc. In this introductory talk, I will present *higher-dimensional rewriting systems* which provide a unifying framework for many of them. These were introduced by Street (as computads) and by Burroni (as polygraphs) with the following motivations. A string rewriting system can be seen as a particular *presentation* of a monoid, describing it by the means of generators and relations. Moreover, when the presentation is confluent and terminating, normal forms provide us with a notion of canonical representative for the elements of the monoid, allowing one to perform many computations on the monoid. The starting point of higher-dimensional rewriting systems is that they should generalize these fruitful tools from monoids to the much richer setting of n-categories.

I will present the nice inductive definition of those rewriting systems, in which an $(n+1)$-dimensional rewriting rule rewrites a rewriting path in an n-dimensional rewriting system, and show how usual rewriting formalisms can be recovered as particular low-dimensional cases. After that, I will explain how usual tools extend to this setting: in particular, I will show that a finite rewriting system can have an infinite number of critical pairs, and present generic ways of constructing termination orders. Finally, we will review some of the applications of those rewriting systems: they can namely be used in order to obtain coherence theorems for various categorical structures (such as MacLane's coherence theorem for monoidal categories), they also found applications in the study of algebraic structures up to homotopy through Koszul duality theory for operads.

Confluence of linear rewriting and homology of algebras

Yves Guiraud[1], Eric Hoffbeck[2], and Philippe Malbos[3]

[1] INRIA, Laboratoire Preuves, Programmes et Systèmes, Université Paris 7
yves.guiraud@pps.univ-paris-diderot.fr
[2] Université Paris 13, Sorbonne Paris Cité, LAGA, CNRS UMR 7539
hoffbeck@math.univ-paris13.fr
[3] Université de Lyon, Institut Camille Jordan, Université Claude Bernard Lyon 1, Villeurbanne
malbos@math.univ-lyon1.fr

Abstract

We introduce the notion of higher dimensional linear rewriting systems for presentations of algebras, generalizing the notion of non-commutative Gröbner bases. We show how to use this notion to compute homological invariants of associative algebras, using the confluence properties of presentations of these algebras. Our method constitutes a new application of confluence in algebra.

1 Introduction

Several methods of computing homological invariants of associative algebras are based on non-commutative Gröbner bases of the ideal of relations of the algebra, [1, 3]. They consist in computing free resolutions of modules over the algebra, generated by some iterated overlaps of the leading terms of the Gröbner basis. In particular, these methods can be applied to homogeneous algebras that arise in many contexts, for instance representation theory, non-commutative geometry and mathematical physics. One of the fundamental properties in the homological description of these algebras is the Koszul property, introduced by Priddy [8] and generalized by Berger [2]. There exist methods to prove Koszulity based on Gröbner bases.

We present an extension of the categorical framework of higher-dimensional rewriting to the linear setting in order to describe Koszulity in terms of confluence. We introduce the notion of a linear polygraph encoding a presentation of an algebra by a rewriting system. Linear polygraphs do not require a monomial order, allowing more possibilities of termination orders than those associated with Gröbner bases. Therefore, we improve the known methods using Gröbner bases to prove Koszulity. Finally, the higher-dimensional rewriting allows us to refine methods to prove Koszulity using the homotopy reduction on convergent rewriting systems developed in [6].

In this note, we consider the case of algebras. A more general case is developed in [4].

2 Linear rewriting

2.1 Linear 2-polygraph

We fix a base field $\mathbb{K}$. A *linear* 2*-polygraph* Λ consists in a data (Σ_1, Λ_2) made of
- a set Σ_1, that we will suppose finite, say $\Sigma_1 = \{x_1, \ldots, x_k\}$,
- a vector space Λ_2 equipped with two linear maps s and t (source and target) from Λ_2 to the free algebra on Σ_1, denoted by Σ_1^ℓ, which is the free vector space on the set Σ_1^* of *monomials* in the variables $x_1, \ldots, x_k$. The length of the monomials induces a *weight grading* on Σ_1^ℓ. Elements in Σ_1^ℓ are called 1-cells and elements in Λ_2 are called 2-cells.

Any 1-cell f in Σ_1^ℓ can be written uniquely as a linear sum $f = \lambda_1 m_1 + \ldots + \lambda_p m_p$, where, for any $1 \leq i \leq p$, $\lambda_i \in \mathbb{K} \setminus \{0\}$ and m_i is a monomial 1-cell. A 2-polygraph Λ is said to be *monic* if the vector space Λ_2 has a basis Σ_2 such that any 2-cell in Σ_2 has a monomial source. For $N \geq 2$, a *monic N-homogeneous* linear 2-polygraph is a linear 2-polygraph Λ, such that any 2-cell in Σ_2 has the form $\alpha : m \Rightarrow \lambda_1 m_1 + \ldots + \lambda_p m_p$ with m and the m_i's are in weight N.

Example 2.1. Consider the algebra $\mathbf{A}$ with generators x, y, z and the relation $x^3+y^3+z^3 = xyz$ in weight 3. The monic 3-homogeneous linear 2-polygraph Λ defined by $\Sigma_1 = \{x, y, z\}$ and $\Sigma_2 = \{xyz \Rightarrow x^3 + y^3 + z^3\}$ presents the algebra $\mathbf{A}$.

In this note, Λ denotes a monic linear 2-polygraph and $\mathbf{A}$ denotes the algebra presented by Λ.

2.2 Rewriting properties of linear 2-polygraphs

The rewriting paths of a string rewriting system form a structure of a 2-category, in which the 1-cells are the strings, the 2-cells are the rewriting paths and the compositions correspond to the sequential and parallel compositions of rewriting paths, as described in [6]. In [4], we show that for linear 2-polygraphs, the 2-category induced by the rewriting paths is linearly enriched. We denote by Λ_2^ℓ the free monoid enriched in 2-vector spaces generated by Λ. Its 1-cells are the 1-cells in Σ_1^ℓ and its 2-cells are linear combinations of all possible parallel and sequential compositions of generating 2-cells in Λ_2.

The notion of a rewriting step induced by a linear 2-polygraph Λ needs to be defined with attention owing to the invertibility of 2-cells. Indeed, given a rule $\varphi : m \Rightarrow h$ in Λ, there are 2-cells $-\varphi : -m \Rightarrow -h$ and $-\varphi+(m+h) : h \Rightarrow m$ in Λ_2^ℓ. Thus we cannot have termination if we consider all 2-cells of Λ_2^ℓ as rewriting sequences. We define a rewriting step as the application of one rule on one monomial of a free linear combination of monomials. A *rewriting step* is a 2-cell in Λ_2^ℓ with the shape $\alpha = \lambda m_1 \varphi m_2 + g$:

$$\begin{tikzcd} \lambda\Big(\bullet \arrow[r, "m_1"] & \bullet \arrow[r, bend left, "m"] \arrow[r, bend right, "h"'] \arrow[r, phantom, "\Downarrow\varphi"] & \bullet \arrow[r, "m_2"] & \bullet \Big) + \bullet \arrow[r, "g"] & \bullet \end{tikzcd}$$

where $\lambda \in \mathbb{K} \setminus \{0\}$, m_1, m_2 are monomial 1-cells in $\Sigma_1^\ell \setminus \{0\}$, $\varphi : m \Rightarrow h$ is a monic rule and g a 1-cell in Σ_1^ℓ such that the monomial $m_1 m m_2$ does not appear in the basis decomposition of g.

A rewriting step α from f to f' is denoted by $\alpha : f \Rightarrow^p f'$. The relation $\Rightarrow^p$ is called the *reduction relation* induced by Λ. A *rewriting sequence* of Λ is a finite or an infinite sequence $f_1 \Rightarrow^p f_2 \Rightarrow^p f_3 \Rightarrow^p \cdots \Rightarrow^p f_n \Rightarrow^p \cdots$ of rewriting steps. We say that Λ *terminates* when it has no infinite rewriting sequence. If there is a non-empty rewriting sequence from f to g, we say that f *rewrites* into g and we denote $f \Rightarrow^* g$.

We denote by Λ_2^+ (resp. Λ_2^{+f}) the set of (resp. finite) rewriting sequences of Λ, also called *positive 2-cells* of the linear 2-polygraph Λ. We denote $f \Leftrightarrow^* g$ when there exists a finite zigzag of rewriting steps between f and g.

We denote by $I(\Lambda)$ the two-sided ideal of the algebra $\mathbf{A}$ generated by $\{\ m - h \mid m \Rightarrow h \in \Lambda_2^\ell\ \}$. Given 1-cells f and f' in Λ_1^ℓ, there is a 2-cell $f \Rightarrow f'$ in Λ_2^ℓ if and only if $f - f' \in I(\Lambda)$.

A 1-cell f of Λ_1^ℓ is *irreducible* when there is no rewriting step for Λ with source f. A *normal form* of f is an irreducible 1-cell g such that f rewrites into g. A 1-cell in Λ_1^ℓ is *reducible* if it is not irreducible. We denote by $\mathrm{ir}(\Lambda)$ (resp. $\mathrm{ir}_{\mathrm{m}}(\Lambda)$) the set of (resp. monomials) irreducible 1-cells for Λ. The rules being monic, the set $\mathrm{ir}(\Lambda)$ forms a vector space generated by $\mathrm{ir}_{\mathrm{m}}(\Lambda)$.

When Λ terminates, the vector space Λ_1^ℓ has the following decomposition $\Lambda_1^\ell = \mathrm{ir}(\Lambda) + I(\Lambda)$.

2.3 Confluence of 2-linear polygraphs

A *branching of* Λ is a pair (α, β) of 2-cells of Λ_2^+ with a common source. A branching (α, β) is *local* when α and β are rewriting steps. Local branchings belong to one of the following families:

aspherical branchings

$$\begin{tikzcd} f \arrow[r, Rightarrow, bend left, "\alpha", "p"'] \arrow[r, Rightarrow, bend right, "\alpha"', "p"] & g \end{tikzcd}$$

Peiffer branchings

$$\begin{tikzcd} & f'g \\ fg \arrow[ru, Rightarrow, "\alpha g\quad p"] \arrow[rd, Rightarrow, "f\beta\quad p"'] & \\ & fg' \end{tikzcd}$$

additive Peiffer branchings

$$\begin{tikzcd} & f'+g \\ f+g \arrow[ru, Rightarrow, "\alpha+g\quad p"] \arrow[rd, Rightarrow, "f+\beta\quad p"'] & \\ & f+g' \end{tikzcd}$$

where α and β are rewriting steps. The *overlapping* branchings are the remaining local branchings. The local branchings are compared by the strict order $\prec$ generated by $(\alpha, \beta) \prec \big(\lambda m\alpha m' + g, \lambda m\beta m' + g\big)$, for any local branching (α, β), where λ is in $\mathbb{K} \setminus \{0\}$, m, m' are monomial 1-cells in Σ_1^ℓ, g is a 1-cell in Σ_1^ℓ, no monomial in the basis decomposition appears in the basis decomposition of $ms(\alpha)m'$ and at least one of the two following conditions: 1/ either m or m' is not an identity monomial, 2/ the 1-cell g is not zero.

An overlapping local branching that is minimal for the order $\prec$ is called a *critical branching.* Note that the critical branchings have a monomial source. A branching (α, β) is *confluent* when there exists a pair (α', β') of 2-cells of Λ_2^+ with the following shape:

$$\begin{tikzcd} & g \arrow[rd, Rightarrow, "\alpha'", "*"'] & \\ f \arrow[ru, Rightarrow, "\alpha", "*"'] \arrow[rd, Rightarrow, "\beta"', "*"] & & f' \\ & g' \arrow[ru, Rightarrow, "\beta'"', "*"] & \end{tikzcd}$$

We say that Λ is *confluent* (resp. *locally confluent*) when all of its branchings (resp. local branchings) are confluent. We prove that a linear 2-polygraph is locally confluent if and only if all its critical branchings are confluent. A linear 2-polygraph Λ is confluent if and only if for any f in $I(\Lambda)$, $f \Rightarrow^* 0$. If moreover, the linear 2-polygraph Λ is terminating, then it is confluent if and only if we have the decomposition $\Sigma_1^\ell = \mathrm{ir}(\Lambda) \oplus I(\Lambda)$.

A linear 2-polygraph is said to be *convergent* when it is terminating and confluent. Such a polygraph is called a *convergent presentation* of the algebra $\mathbf{A}$. In this case, there is a canonical section $\mathbf{A} \to \Sigma_1^\ell$ sending f to its normal form denoted by $\widehat{f}$, so that $\widehat{f} = \widehat{g}$ holds in Σ_1^ℓ if and only if we have $f = g$ in $\mathbf{A}$. Thus, by the decomposition $\Sigma_1^\ell = \mathrm{ir}(\Lambda) \oplus I(\Lambda)$, the set of irreducible monomials $\mathrm{ir}_{\mathrm{m}}(\Lambda)$ forms a $\mathbb{K}$-linear basis of the algebra $\mathbf{A}$ via the canonical map $\mathrm{ir}(\Lambda) \longrightarrow \mathbf{A}$, called a *standard basis* of $\mathbf{A}$. In [4], we show that we recover the usual notions of Gröbner basis and of PBW basis when the rules in Λ_2 are compatible with a monomial order.

3 Polygraphic resolutions of algebras

In this section, we define polygraphic resolutions for algebras, which are extensions of presentations of an algebra in higher dimensions with a property of acyclicity. Such resolutions were introduced in [5] for monoids and categories.

Starting from a linear 2-polygraph Λ, we define the 2-spheres of Λ by the pairs (f, g) of elements in Λ_2^ℓ such that $s(f) = s(g)$ and $t(f) = t(g)$. A *linear extension* of Λ_2^ℓ is a vector space Λ_3, together with linear maps s and t from Λ_3 to Λ_2^ℓ sending an element h on a 2-sphere $(s(h), t(h))$. The data $(\Sigma_1, \Lambda_2, \Lambda_3)$ defines a *linear 3-polygraph.* Proceeding inductively with the notion of spheres and linear extensions, we define the notion of *linear n-polygraphs.*

A linear n-polygraph Λ is called *acyclic* when any k-sphere (f, g) can be filled by a $(k+1)$-cell A, that is $s(A) = f$ and $t(A) = g$, for all $1 \leq k < n$. A *polygraphic resolution* of $\mathbf{A}$ is an acyclic linear ∞-polygraph Λ, whose underlying linear 2-polygraph Λ_2 is a presentation of $\mathbf{A}$.

Consider a linear 2-polygraph Λ, which is *reduced*, that is for every 2-cell $\varphi : m \Rightarrow f$ in Λ_2, the 1-cell m is a normal form for $\Sigma_2 \setminus \{\varphi\}$ and the 1-cell f is irreducible for Λ_2. We define a *k-fold branching* by a k-tuple $(f_1, \ldots, f_k)$ of 2-cells of Λ_2^+ with the same source. As before, we can define the overlapping branchings, and an ordering relation via inclusion. We define the *critical* k-fold branchings as the minimal overlapping k-fold branchings. For instance, when $k = 3$, we get two possible shapes of such critical branchings.

$$\begin{tikzcd}
{} \arrow[r, "m_1"'] \arrow[rrr, bend left=50, "\varphi\Uparrow"] & {} \arrow[r, "m_2"] \arrow[rrr, bend right=50, "\Downarrow\psi"'] & {} \arrow[r, "m_3"] \arrow[rrr, bend left=50, "\Uparrow\chi"] & {} \arrow[r, "m_4"] & {} \arrow[r, "m"'] & {}
\end{tikzcd}
\quad \text{or} \quad
\begin{tikzcd}
{} \arrow[r, "m_1"'] \arrow[rr, bend left=50, "\varphi\Uparrow"] & {} \arrow[r, "m_2"] \arrow[rrr, bend right=50, "\Downarrow\psi"'] & {} \arrow[r, "m_3"] & {} \arrow[r, "m_4"] \arrow[rr, bend left=50, "\Uparrow\chi"] & {} \arrow[r, "m"'] & {}
\end{tikzcd}$$

where the m_i's are monomials and φ, ψ, χ are generating 2-cells. The following result states that a polygraphic resolution can be built using higher critical branchings of a convergent presentation.

Theorem 3.1. *[4, 4.2.10] Any convergent linear 2-polygraph Λ extends to an acyclic linear ∞-polygraph whose k-cells, for $k \geq 3$, are indexed by the critical $(k-1)$-fold branchings.*

4 Confluence for the Koszul property

An N-homogeneous algebra $\mathbf{A}$ is called *Koszul* if there exists a free resolution of graded right $\mathbf{A}$-modules

$$0 \longleftarrow \mathbb{K} \longleftarrow F_0 \longleftarrow F_1 \longleftarrow \ \ldots \ \longleftarrow F_{k-1} \longleftarrow F_k \longleftarrow \ \ldots$$

such that for any integer $k \geq 0$, the $\mathbf{A}$-module F_k has the form $N_k \otimes \mathbf{A}$ with all elements of N_k in weight $\ell_N(k)$, where $\ell_N(k) = lN$, if $k = 2l$, and $\ell_N(k) = lN + 1$, if $k = 2l + 1$.

In [4], we show that polygraphic resolutions for an algebra $\mathbf{A}$ induce free resolutions of $\mathbf{A}$-modules of the base field $\mathbb{K}$. As a consequence, a necessary condition for an algebra to be Koszul can be expressed in terms of polygraphic resolutions. For this purpose, the weight grading on a linear 2-polygraph is extended on higher cells. Given a weight function $\omega : \mathbb{N} \to \mathbb{N}$, a graded polygraphic resolution Λ is *ω-concentrated* when its k-cells are concentrated in weight $\omega(k)$. The necessary condition is formulated as follows:

Theorem 4.1. *[4, 5.3.4] Let $\mathbf{A}$ be an N-homogeneous algebra. If $\mathbf{A}$ has an ℓ_N-concentrated polygraphic resolution, then $\mathbf{A}$ is Koszul.*

As a consequence of this result, the confluence property can be used to prove that an algebra is Koszul. Indeed, we have

Proposition 4.2. *[4, 5.3.6] Let $\mathbf{A}$ be an algebra presented by a quadratic convergent 2-polygraph Λ, then Λ can be extended into an ℓ_2-concentrated polygraphic resolution. In particular, such an algebra is Koszul.*

Finally, the following proposition is often be useful, in particular when using the homotopy reduction procedure on convergent rewriting systems developed in [6].

Proposition 4.3. *[4, 5.3.8] If an N-homogeneous algebra $\mathbf{A}$ is presented by an acyclic ℓ_N-concentrated 3-polygraph Λ with $\Lambda_3 = \{0\}$, then $\mathbf{A}$ is Koszul.*

We now show on some examples how our method can be applied. Given an algebra **A** presented by generators and relations, the idea is first to obtain a convergent linear 2-polygraph Λ (using usual completion procedures if necessary) presenting **A**. Then we study the critical branchings of Λ to understand the polygraphic resolution and its properties.

Example 4.1. The linear 2-polygraph Λ defined in the Example 2.1 is convergent, without any critical branching. It follows that $(\Sigma_1, \mathbb{K}\Sigma_2, \{0\})$ is a ℓ_3-concentrated acyclic 3-polygraph, hence by Proposition 4.3, **A** is Koszul.

Example 4.2. Consider the algebra **A** with generators x, y, z and the relations $x^2 + yz = 0$, $x^2 + azy = 0$ where $a \in \mathbb{K} \setminus \{0; 1\}$. It is a Koszul algebra, see [7, Section 4.3], but has no convergent quadratic presentation. We prove that the linear 3-polygraph Λ with $\Sigma_1 = \{x, y, z\}$,

$$\Sigma_2 = \{\ yz \overset{\alpha}{\Longrightarrow} -x^2\ ,\ zy \overset{\beta}{\Longrightarrow} bx^2\ ,\ yx^2 \overset{\gamma}{\Longrightarrow} ax^2y\ ,\ zx^2 \overset{\delta}{\Longrightarrow} -bx^2z\ \},\ \Sigma_3 = \{A_1, B_1, C_1, D_1\}$$

$$\begin{tikzcd}
& byx^2 \arrow[dd, Rightarrow, "b\gamma"] \arrow[dd, phantom, "A_1"', shift right=4] \\
yzy \arrow[ur, Rightarrow, bend left, "y\beta"] \arrow[dr, Rightarrow, bend right, "\alpha y"'] & \\
& -x^2y
\end{tikzcd}
\quad
\begin{tikzcd}
& bx^2z \\
zyz \arrow[ur, Rightarrow, bend left, "\beta z"] \arrow[dr, Rightarrow, bend right, "z\alpha"'] & \\
& -zx^2 \arrow[uu, Rightarrow, "-\delta"'] \arrow[uu, phantom, "B_1", shift left=4]
\end{tikzcd}
\quad
\begin{tikzcd}
& bx^4 & \\
zyx^2 \arrow[ur, Rightarrow, bend left, "\beta x^2"] \arrow[dr, Rightarrow, bend right, "z\gamma"'] \arrow[rr, phantom, "C_1"] & & x^2zy \arrow[ul, Rightarrow, bend right, "x^2\beta"'] \\
& azx^2y \arrow[ur, Rightarrow, bend right, "a\delta y"'] &
\end{tikzcd}
\quad
\begin{tikzcd}
& -x^4 & \\
yzx^2 \arrow[ur, Rightarrow, bend left, "\alpha x^2"] \arrow[dr, Rightarrow, bend right, "y\delta"'] \arrow[rr, phantom, "D_1"] & & x^2yz \arrow[ul, Rightarrow, bend right, "x^2\alpha"'] \\
& -byx^2z \arrow[ur, Rightarrow, bend right, "-b\gamma z"'] &
\end{tikzcd}$$

is acyclic and forms a presentation of the algebra **A**. There are four critical triples with sources $yzyz$, $yzyx^2$, $zyzy$, $zyzx^2$. For instance, the 4-cell corresponding to the critical triple on $yzyz$ is

$$\begin{tikzcd}
& -x^2yz \arrow[dr, Rightarrow, bend left, "-x^2\alpha"] \arrow[dr, phantom, "-D_1"', shift right=3] & \\
yzyz \arrow[ur, Rightarrow, bend left, "\alpha yz"] \arrow[r, Rightarrow, "y\beta z"] \arrow[dr, Rightarrow, bend right, "yz\alpha"'] & byx^2z \arrow[u, Rightarrow, "b\gamma z"] \arrow[u, phantom, "A_1z", shift right=6] & x^4 \\
& -yzx^2 \arrow[u, Rightarrow, "-y\delta"'] \arrow[u, phantom, "yB_1", shift left=6] \arrow[ur, Rightarrow, bend right, "-\alpha x^2"'] &
\end{tikzcd}
\quad \overset{A_2}{\Rrightarrow} \quad
\begin{tikzcd}
& -x^2yz \arrow[dr, Rightarrow, bend left, "-x^2\alpha"] & \\
yzyz \arrow[ur, Rightarrow, bend left, "\alpha yz"] \arrow[dr, Rightarrow, bend right, "yz\alpha"'] \arrow[rr, phantom, "\equiv"] & & x^4 \\
& -yzx^2 \arrow[ur, Rightarrow, bend right, "-\alpha x^2"'] &
\end{tikzcd}$$

The 4-cell A_2 relates the 3-cell D_1 with the 3-cells A_1 and B_1. Using a homotopy reduction procedure as in [6], we can remove the cells which are not on the diagonal. Thus we obtain an ℓ_2-concentrated polygraphic resolution and by Proposition 4.3, it follows that **A** is Koszul.

References

[1] D.J. Anick. On the homology of associative algebras. *Trans. Amer. Math. Soc.*, 296(2):641–659, 1986.

[2] R. Berger. Koszulity for nonquadratic algebras. *J. Algebra*, 239(2):705–734, 2001.

[3] Edward L. Green. Noncommutative Gröbner bases, and projective resolutions. In *Computational methods for representations of groups and algebras (Essen, 1997)*, volume 173 of *Progr. Math.*, pages 29–60. Birkhäuser, Basel, 1999.

[4] Y. Guiraud, E. Hoffbeck, and P. Malbos. Linear polygraphs and koszulity of algebras. Preprint arXiv 1406.0815, 2014.

[5] Y. Guiraud and P. Malbos. Higher-dimensional normalisation strategies for acyclicity. *Adv. Math.*, 231(3-4):2294–2351, 2012.

[6] Y. Guiraud, P. Malbos, and S. Mimram. A homotopical completion procedure with applications to coherence of monoids. RTA 2013.

[7] A. Polishchuk and L. Positselski. *Quadratic algebras.* University Lecture Series, vol. 37. AMS, 2005.

[8] S. B. Priddy. Koszul resolutions. *Trans. Amer. Math. Soc.*, 152:39–60, 1970.

Critical Pairs in Network Rewriting

Lars Hellström

Division of Applied Mathematics, The School of Education, Culture and Communication, Mälardalen University, Box 883, 721 23 Västerås, Sweden; `lars.hellstrom@residenset.net`

Abstract

This extended abstract breifly introduces rewriting of networks (directed acyclic graphs with the extra structure needed to serve as expressions for PROducts and Permutations categories) and describes the critical pairs aspects of this theory. The author's interest in these comes from wanting to do equational reasoning in algebraic theories (such as Hopf algebras) that mix ordinary operations with co-operations; networks then serve as a formalism for expressions.

The main message is to point out two phenomena that arise in network rewriting. The first is that of non-convexity of rules, wherein the left hand side of a rule need not be syntactically similar to a symbol in any extension of the underlying signature. The second is one of critical pairs potentially arising where two redexes wrap around each other even when they do not intersect.

1 Introduction to networks

Words provide a natural model for expressions in an algebraic theory of all-unary operations; the corresponding abstract algebraic structure is the monoid, and the set of all words may be formalised as the free monoid. Generalising to operations of arbitrary arity, the natural model for expressions instead becomes that of terms. Adding the condition that the terms should be linear in the sense that each variable occurs exactly once, one arrives at a concept whose corresponding abstract algebraic structure is called an operad. Operads first became popular within topology, but have since become useful tools also in algebra in general, especially to study non-associative structures.

The generalisation from one to many is however not exhausted by operads or terms: if an n-ary operation would be implemented by a subroutine with 1 out-parameter and n in-parameters, then what kinds of expressions could be built from operations that syntactically are like subroutines with m out-parameters and n in-parameters? The corresponding abstract algebraic structure will be the PROP—or strict symmetric monoidal category (symocat) if one works with multiple atomic sorts—and the natural expression model will be that of networks [2].

A *network* is essentially like a term, expect that instead of having an underlying tree there is an underlying directed acyclic graph (DAG). Formally starting from a DAG, the extra data needed to turn it into a network are the following. (i) Each inner vertex is given a symbol from a doubly ranked alphabet. If the symbol $D(v)$ of vertex v has rank (m, n), then the in-degree of v must be n (the *arity*) and the out-degree of v must be m (the *coarity*). (ii) There is at each vertex a total ordering of the incoming edges, and a separate total ordering of the outgoing edges. (iii) There are two distinguished vertices 0 and 1 that represent the output and input respectively sides of the network; the arity of the network as a whole is the degree (all outgoing) of the input vertex 1, and the coarity of the network as a whole is the degree (all incoming) of the output vertex 0. In the special case that each symbol in the alphabet has coarity 1, the networks with coarity 1 are precisely the linear terms (networks of higher coarity would be to linear terms as forests are to trees). As expression models, networks are special cases of *share graphs* [1], but their built-in linearity—that each edge has exactly one tail and exactly one head implies each intermediate result is generated once and used once—make them valid as expressions in a much wider range of contexts, such as quantum computing and multilinear algebra where a classical duplication of information would violate fundamental axioms.

Just like a monoid offers a natural setting for evaluating a word as an expression, the natural setting for evaluating a network is an algebraic structure called a PROP. A *PRO* [4, Ch. V] is a set of doubly-ranked elements together with two composition operations: the serial composition $\circ$ which corresponds to ordinary composition of functions, and the parallel composition $\otimes$ which corresponds to letting two functions act separately on disjoint parts of a composite argument; a trivial example of the latter is to make $(f \otimes g)(x, y) := \bigl(f(x), g(y)\bigr)$, whereas other basic examples make $\otimes$ the tensor product of two linear maps. As operations on networks, $\circ$ amounts to joining the outputs of the right operand to the inputs of the left operand, whereas $\otimes$ simply places the operands side-by-side, exposing each input and output of either operand as an input or output of the combined network. A *PROP* is a PRO equipped with actions of permutations on the elements, which for networks correspond to permutating outputs among themselves and/or inputs among themselves. The formal definition of how to evaluate a network is rather technical [2, Def. 5.2], although the fact that it can be done can be taken as an alternative definition of PROP [2, Th. 5.17].

The multiple atomic sorts counterpart of a PRO is a monoidal category, whereas in terms of networks it would correspond to adding a planarity constraint in the sense of 'no crossing edges'. PROs may seem more elementary if coming from the abstract algebra point of view, but from the formalised expression perspective they rather constitute a curious restriction, in that they call for items of data to be located to points within a geometric space.

2 Network rewriting

Naively, a rewriting step consists of replacing one subexpression equal to the left hand side of a rule with the right hand side of that rule. Since networks are graphs (with extra structure), network rewriting is visually intuitive: you cut some edges, remove the piece that thereby got separated and match it to the left hand side of a rewrite rule, replace the piece with a new one equal to the rewrite rule right hand side, and splice together the edges at the cuts. What turns out to be a nonobvious matter is however that of what kind of piece qualifies as a subexpression: different established formula formalisms lead to different answers.

The monoidal category perspective suggests that a rule $l \to r$ can be applied to those expressions that are obtained by padding l using the two compositions $\circ$ and $\otimes$, i.e., that any rewrite step done using $l \to r$ can be written as

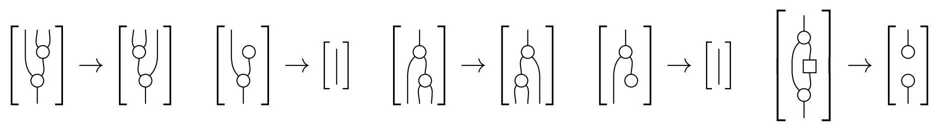
$$B \circ C_1 \otimes l \otimes C_2 \circ D \to B \circ C_1 \otimes r \otimes C_2 \circ D$$

for some B, C_1, C_2, and D, where $\otimes$ is taken to have higher priority than $\circ$. (Having permutations allows combining C_1 and C_2, but that is beside the point here.) This is also the subexpression concept one gets from a straightforward double pushout graph rewriting formalism where l and r are both being produced as images (under separate morphisms) of a marker symbol x, and the context graph network has the form $B \circ C_1 \otimes x \otimes C_2 \circ D$. It is however not the most general subexpression concept.

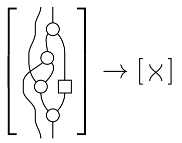
An alternative formula formalism for writing expressions in PROPs and symocats is the Abstract Index Notation [7], which is an abstract reinterpretation of the Einstein summation convention for tensors. Here, an expression is written as a formal product of factors which each carry zero or more sub- and superscripts, e.g. $\mu^a_{bc} S^b_d \Delta^{dc}_e$. The Einstein summation convention says that there is an implicit summation over any index letter appearing once as superscript and once as subscript in a product, whereas letters with one appearance in total are externally visible indices of the composed tensor, and two or more appearances of the same kind (super or sub) is forbidden. In terms of networks, the abstract index reinterpretation is that each factor is a vertex, the base letter of a factor is the operation symbol associated with that vertex, and the index letters name the edges that are incident with the vertex: superscripts are outgoing, subscripts incoming. Here, two appearances of an index letter means it is an internal edge, whereas just one means it is an external edge with the other end at the implicit output vertex 0 or input vertex 1 as appropriate; two or more appearances of the same kind are impossible for a directed edge. In abstract index notation, any subset of the factors would constitute a valid subexpression, so a rewrite step can be any $lA \to rA$, where l, r, and A are products of labelled factors such that the composite products lA and rA satisfy all syntactic constraints.

This is different from the previous subexpression concept in that it allows subexpressions to be *nonconvex*: a path may begin in a vertex/factor of l, pass through some vertex in the context A, and then return to the subexpression l, even if it must then be at a different vertex of l than that at which it started. This is not possible in a formalism that considers a subexpression to be something that is similar to a vertex, since any path leaving a vertex through one edge and then returning to it via another would constitute a cycle. Are nonconvex subexpressions useful for rewriting, though? The system for Hopf algebras considered in [3] demonstrate that they are very useful indeed.

The axioms for a Hopf algebra can be naturally stated as a network rewrite system, five of the rules in that being



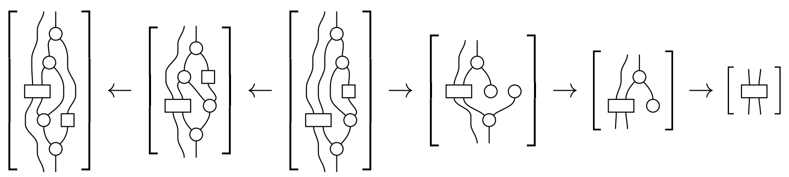
The brackets here serve as frames around a network when it appears as part of a formula, to clarify its graphical extent. For the interpretation of the various vertex types in the case of Hopf algebras, see [3]. Completing the rewrite system consisting of just the Hopf axioms does however produce several nonconvex rules. One of these derived rules (which follows directly from those five above) is:



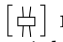
where the left input may have a depenence on the left output;

this 'may have a dependence on' phrase is stating that the left hand side may be matched against a nonconvex subexpression during rewriting, and it says how that nonconvexity may be realised.

The reason this rule should be nonconvex is that there is no step in its derivation which contradicts a dependence of left input on left output; any rewrite step made where this rule is matched against a nonconvex subexpression can alternatively be carried out as a sequence of forward and backward steps using the five axiom rules above, at each step only replacing convex subexpressions. One example of this would be



(1)

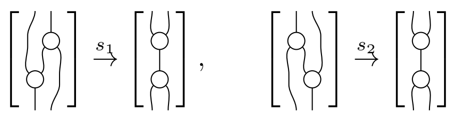
where the right input to left output edge of the $r = [\times]$ network is part of the top right edge of the  network, whereas the left input to right output edge of the r network is part of the bottom right edge. It can therefore be argued that nonconvex subexpressions are natural from a rewriting perspective: since a derived rule is somewhat like a prepackaged sequence of rewrite steps, and since there in a two-dimensional setting is no reason for the union of a number of polygons (say) to be convex (even if each component is convex and the union is connected), there is no reason to expect that derived rules will all be convex even if the rules they are derived from happen to be. Indeed, some experience with completing network rewriting systems suggest that derived rules will typically grow nonconvex fairly soon after they have become complicated enough to exhibit such features.

Allowing rewrite rules to match against nonconvex subexpressions does however raise the problem of how to avoid creating cycles. (Many technicalities become much easier if one allows cycles, including that of defining evaluation of a network—cf. the 'normal form expressions' in [6, Sec. 1.4]—but far from all interpretations of networks support cycles. In a computational context, a cycle could correspond to sending information backwards in time, or at best to some kind of fixpoint operation. For multilinear interpretations, cycles tend to be problematic as soon as one considers infinite-dimensional spaces.) The framework of [2] uses a filtration (indexed by boolean matrices) of the PROP of networks to keep track of which dependencies of inputs on outputs of a network are consistent with it appearing as a subnetwork of another network. Each rule has an associated *transferrence type*, and rules may only apply in contexts consistent with that type. Likewise, each critical pair has an associated transferrence type, and if it gives rise to a derived rule, then that will also be the transferrence type of that derived rule. In the author's opinion, nonconvex rules are well understood and cared for by the framework of [2].

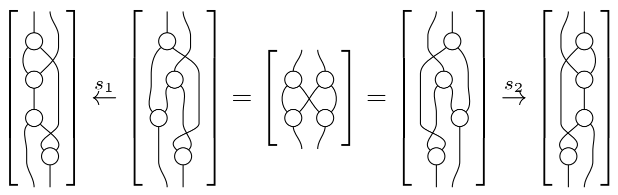
3 Critical pairs

The nice thing about the ability to handle nonconvex subexpressions in network rewriting is that the sites of critical ambiguities (i.e., critical pairs) need never include vertices that do not correspond to a vertex in the left hand side of at least one of the rules [2, proof of L. 10.13]; without this, (1) would give rise to a separate critical pair for every way of replacing the wide vertex with something else. In [5], Mimram seeks to achieve the same end of eliminating irrelevant vertices by formally bending edges; unlike the nonconvex subexpression concept, this cannot cope with an irrelevant vertex completely surrounded by vertices acted upon by a rewrite rule of the critical pair, but on the other hand it preserves the plane embedding which is a concern in that paper. As mentioned above, the network rewriting formalism of [2] does not consider data items to have a location in a geometric space, so concerns about planarity are meaningless.

An issue that at present is not fully understood is however that of critical pairs of 'wrap' type, which are due to two redexes wrapping around each other in such a way that reducing one will block reducing the other, even though they do not overlap. The simplest example of this is probably that the two rules



give rise to the critical pair



(2)

where a naive attempt at applying the other rewrite rule at either side would violate acyclicity. The left hand side of one rule can still be found as a subexpression after the other rule has been applied, but it is no longer a redex since the subexpression does no longer satisfy the transferrence constraints associated with the rule. This is different from at least the elementary sense of rewriting residual (wherein a redex is moved by but still remains a redex after the application of a separate rule; the network rewriting framework handle ordinary residuals under the name of 'montage ambiguities' [2, Def. 10.14]).

It is notable that this example of a wrap ambiguity does not rely on having nonconvex rules or subexpressions, so including them in the framework did not cause this problem. Indeed, it is rather the impression of the author that wrap ambiguities is a problem that nonconvex rules do not quite manage to solve, even though that they are in the vicinity of doing so; if networks are viewed as merely having a particular arity and coarity, then wrap ambiguities cannot be ruled out, but if they are instead viewed as having a particular transferrence type then there is a practical condition ('sharpness' [2, Def. 10.3]) that will guarantee that a rewrite system is free of wrap ambiguities. Perhaps a further refinement of the network rewriting framework can help eliminate them altogether.

On the other hand, the nontrivial derivation in (2) suggests that there really is something here that an automated completion procedure would need to explore. It is an open problem in the theory of network rewriting to enumerate all critical pairs where wrap ambiguities remain a possiblity.

References

[1] Masahito Hasegawa. *Models of sharing graphs.* Diss. PhD thesis, University of Edinburgh, Department of Computer Science, 1997.

[2] Lars Hellström. *Network Rewriting I: The Foundation*, 2012. arXiv:1204.2421v1 [math.RA].

[3] Lars Hellström. *Network Rewriting II: Bi- and Hopf Algebras.* Manuscript, 2014, 15 pp. urn:nbn:se:mdh:diva-25102

[4] Saunders MacLane. Categorical Algebra. *Bull. Amer. Math. Soc.* **71** (1965), 40–106.

[5] Samuel Mimram. *Computing Critical Pairs in 2-Dimensional Rewriting Systems.* arXiv:1004.3135v1 [cs.FL].

[6] Gheorghe Ştefănescu. *Network Algebra.* Springer, 2000. ISBN 1-85233-195-X.

[7] Wikipedia. *Abstract index notation — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/w/index.php?title=Abstract_index_notation` [Online; accessed 4-June-2014]

Normalization Equivalence of Rewrite Systems[*]

Nao Hirokawa[1], Aart Middeldorp[2] and Christian Sternagel[2]

[1] JAIST, Japan
hirokawa@jaist.ac.jp
[2] University of Innsbruck, Austria
{aart.middeldorp|christian.sternagel}@uibk.ac.at

Abstract

Métivier (1983) proved that every confluent and terminating rewrite system can be transformed into an equivalent canonical rewrite system. He also proved that equivalent canonical rewrite systems which are compatible with the same reduction order are unique up to variable renaming. In this note we present simple and formalized proofs of these results. The latter result is generalized to the uniqueness of *normalization equivalent* reduced rewrite systems.

1 Introduction

Consider the TRS $\mathcal{R}$ of combinatory logic with equality test, studied by Klop [3]:

$$\mathsf{S}xyz \to xz(yz) \qquad \mathsf{K}xy \to x \qquad \mathsf{I}x \to x \qquad \mathsf{D}xx \to \mathsf{E}$$

The TRS $\mathcal{R}$ is reduced, but neither terminating nor confluent. One might ask: *is there another reduced TRS $\mathcal{S}$ that computes the same normal forms for every starting term*? We refer to this property as *normalization equivalence* of two TRSs. According to the main result of this note, it turns out that $\mathcal{R}$ is unique up to variable renaming.

In the next section normalization equivalence is studied in an abstract setting. The concrete results on term rewrite systems are presented in Section 3. Throughout this note, we assume familiarity with basic notions and terminology of term rewriting.

All the proofs that are presented in the following have been formalized as part of IsaFoR[1] (see theory `Normalization_Equivalence`).

2 Abstract Normalization Equivalence

First, we introduce the two notions of equivalence that will be studied in this note.

Definition 2.1. Two ARSs $\mathcal{A}$ and $\mathcal{B}$ are *(conversion) equivalent* if $\leftrightarrow^*_\mathcal{A} = \leftrightarrow^*_\mathcal{B}$. If $\to^!_\mathcal{A} = \to^!_\mathcal{B}$ we say that $\mathcal{A}$ and $\mathcal{B}$ are *normalization equivalent.*

The following example shows that the two equivalence notions defined above are different.

Example 2.2. The ARSs

$$\mathcal{A}_1: \quad \mathsf{a} \longrightarrow \mathsf{b} \qquad\qquad \mathcal{B}_1: \quad \mathsf{a} \longleftarrow \mathsf{b}$$

[*]Supported by JSPS KAKENHI Grant Number 25730004 and the Austrian Science Fund (FWF) projects I963 and J3202.

[1]`http://cl-informatik.uibk.ac.at/software/ceta/`

are conversion equivalent but not normalization equivalent. The ARSs

$$\begin{tikzcd} \mathcal{A}_2\colon & \mathsf{a} \arrow[r] & \mathsf{b} \arrow[loop below] & & \mathcal{B}_2\colon & \mathsf{a} \arrow[loop below] & \mathsf{b} \arrow[loop below] \end{tikzcd}$$

are normalization equivalent but not conversion equivalent.

The easy proof (by induction on the length of conversions) of the following result is omitted.

Lemma 2.3. *Normalization equivalent terminating ARSs are equivalent.* □

Note that the termination assumption can be weakened to weak normalization. However, the present version suffices to prove the following lemma that we employ in our proof of Métivier's transformation result (Theorem 3.7).

Lemma 2.4. *Let $\mathcal{A}$ and $\mathcal{B}$ be ARSs such that $\to_{\mathcal{B}} \subseteq \to_{\mathcal{A}}^{+}$ and $\mathsf{NF}(\mathcal{B}) \subseteq \mathsf{NF}(\mathcal{A})$. If $\mathcal{A}$ is complete then $\mathcal{B}$ is complete and normalization equivalent to $\mathcal{A}$.*

Proof. From the inclusion $\to_{\mathcal{B}} \subseteq \to_{\mathcal{A}}^{+}$ we infer that $\mathcal{B}$ is terminating. Moreover, $\to_{\mathcal{B}}^{*} \subseteq \to_{\mathcal{A}}^{*}$ and, since $\mathsf{NF}(\mathcal{B}) \subseteq \mathsf{NF}(\mathcal{A})$, also $\to_{\mathcal{B}}^{!} \subseteq \to_{\mathcal{A}}^{!}$. For the reverse inclusion we reason as follows. Let $a \to_{\mathcal{A}}^{!} b$. Because $\mathcal{B}$ is terminating, $a \to_{\mathcal{B}}^{!} c$ for some $c \in \mathsf{NF}(\mathcal{B})$. So $a \to_{\mathcal{A}}^{!} c$ and thus $b = c$ from the confluence of $\mathcal{A}$. It follows that $\mathcal{A}$ and $\mathcal{B}$ are normalization equivalent. It remains to show that $\mathcal{B}$ is locally confluent. This follows from the sequence of inclusions

$${}_{\mathcal{B}}{\leftarrow} \cdot \to_{\mathcal{B}} \;\subseteq\; {}^{+}_{\mathcal{A}}{\leftarrow} \cdot \to_{\mathcal{A}}^{+} \;\subseteq\; \to_{\mathcal{A}}^{*} \cdot {}^{*}_{\mathcal{A}}{\leftarrow} \;\subseteq\; \to_{\mathcal{A}}^{!} \cdot {}^{!}_{\mathcal{A}}{\leftarrow} \;\subseteq\; \to_{\mathcal{B}}^{!} \cdot {}^{!}_{\mathcal{B}}{\leftarrow}$$

where we use the inclusion $\to_{\mathcal{B}} \subseteq \to_{\mathcal{A}}^{+}$, the confluence of $\mathcal{A}$, the termination of $\mathcal{A}$, and the normalization equivalence of $\mathcal{A}$ and $\mathcal{B}$. □

In the above lemma, completeness can be weakened to semi-completeness (i.e., the combination of confluence and weak normalization), which is not true for Theorem 3.7 as shown by Gramlich [1]. Again, the present version suffices for our purposes.

3 Normalization Equivalence

In this section we study normalization equivalence for TRSs.

Definition 3.1. A *variable substitution* is a substitution from $\mathcal{V}$ to $\mathcal{V}$. A *renaming* is a bijective variable substitution. A term s is a *variant* of a term t if $s = t\sigma$ for some renaming σ. If $\ell \to r$ is a rewrite rule and σ is a renaming then the rewrite rule $\ell\sigma \to r\sigma$ is a variant of $\ell \to r$. A TRS is said to be *variant-free* if it does not contain rewrite rules that are variants of each other.

TRSs are usually assumed to be variant-free. We make the same assumption, but see Example 3.6 below.

Given terms s and t, we write $s \doteq t$ if $s\sigma = t$ and $s = t\tau$ for some substitutions σ and τ. The following result is folklore; the proof has recently been formalized [2].

Lemma 3.2. *Two terms s and t are variants if and only if $s \doteq t$.* □

Definition 3.3. Two TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ over the same signature $\mathcal{F}$ are called *literally similar*, denoted by $\mathcal{R}_1 \doteq \mathcal{R}_2$, if every rewrite rule in $\mathcal{R}_1$ has a variant in $\mathcal{R}_2$ and vice-versa.

Definition 3.4. A TRS $\mathcal{R}$ is *left-reduced* if $\ell \in \mathsf{NF}(\mathcal{R} \setminus \{\ell \to r\})$ for every rewrite rule $\ell \to r$ in $\mathcal{R}$. We say that $\mathcal{R}$ is *right-reduced* if $r \in \mathsf{NF}(\mathcal{R})$ for every rewrite rule $\ell \to r$ in $\mathcal{R}$. A *reduced* TRS is left- and right-reduced. A reduced complete TRS is called *canonical.*

Theorem 3.7 below states that we can always eliminate redundancy in a complete TRS. This is achieved by the two-stage transformation defined below.

Definition 3.5. Given a complete TRS $\mathcal{R}$, the TRSs $\dot{\mathcal{R}}$ and $\ddot{\mathcal{R}}$ are defined as follows:

$$\dot{\mathcal{R}} = \{\ell \to r{\downarrow_{\mathcal{R}}} \mid \ell \to r \in \mathcal{R}\}$$
$$\ddot{\mathcal{R}} = \{\ell \to r \in \dot{\mathcal{R}} \mid \ell \in \mathsf{NF}(\dot{\mathcal{R}} \setminus \{\ell \to r\})\}$$

The TRS $\dot{\mathcal{R}}$ is obtained from $\mathcal{R}$ by normalizing the right-hand sides. To obtain $\ddot{\mathcal{R}}$ we remove the rules of $\dot{\mathcal{R}}$ whose left-hand sides are reducible with another rule of $\dot{\mathcal{R}}$.

Example 3.6. Consider the TRS $\mathcal{R}_1$ consisting of the four rules

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

Then the first transformation yields $\dot{\mathcal{R}}_1$

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

and the second one $\ddot{\mathcal{R}}_1$

$$\mathsf{a} \to \mathsf{c} \qquad \mathsf{b} \to \mathsf{c}$$

Note that $\ddot{\mathcal{R}}_1$ is *not* equivalent to $\mathcal{R}_1$. This is caused by the fact that the result of the first transformation is no longer variant-free.

The proof of the following theorem depends on the implicit assumption that TRSs are always variant-free. However, even for variant-free $\mathcal{R}$, $\dot{\mathcal{R}}$ does not necessarily have this property (as shown by Example 3.6 above). Thus, in our formalization, we explicitly remove variants of rules as part of the $\dot{\mathcal{R}}$ transformation.

Theorem 3.7. *If $\mathcal{R}$ is a complete TRS then $\ddot{\mathcal{R}}$ is a normalization and conversion equivalent canonical TRS.*

The proof by Métivier [4, Theorem 7] is hard to reconstruct. The proof in [5, Exercise 7.4.7] involves 13 steps with lots of redundancy. The proof below uses induction on the well-founded encompassment order $\rhd$ and has been formalized. Since subsumption as well as encompassment have not been part of IsaFoR before, we had to amend this situation. See theory `Encompassment` for details.

Proof. Let $\mathcal{R}$ be a complete TRS. The inclusions $\ddot{\mathcal{R}} \subseteq \dot{\mathcal{R}} \subseteq \to_{\mathcal{R}}^{+}$ are obvious from the definitions. Since $\mathcal{R}$ and $\dot{\mathcal{R}}$ have the same left-hand sides, their normal forms coincide. We show that $\mathsf{NF}(\ddot{\mathcal{R}}) \subseteq \mathsf{NF}(\dot{\mathcal{R}})$. To this end we show that $\ell \notin \mathsf{NF}(\ddot{\mathcal{R}})$ whenever $\ell \to r \in \dot{\mathcal{R}}$ by induction on ℓ with respect to the well-founded order $\rhd$. If $\ell \to r \in \ddot{\mathcal{R}}$ then $\ell \notin \mathsf{NF}(\ddot{\mathcal{R}})$ trivially holds. So suppose $\ell \to r \notin \ddot{\mathcal{R}}$. By definition of $\ddot{\mathcal{R}}$, $\ell \notin \mathsf{NF}(\dot{\mathcal{R}} \setminus \{\ell \to r\})$. So there exists a rewrite rule $\ell' \to r' \in \dot{\mathcal{R}}$ different from $\ell \to r$ such that $\ell \unrhd \ell'$. We distinguish two cases.

- If $\ell \rhd \ell'$ then we obtain $\ell' \notin \mathsf{NF}(\ddot{\mathcal{R}})$ from the induction hypothesis and hence $\ell \notin \mathsf{NF}(\ddot{\mathcal{R}})$ as desired.

- If $\ell \doteq \ell'$ then by Lemma 3.2 there exists a renaming σ such that $\ell = \ell'\sigma$. Since $\dot{\mathcal{R}}$ is right-reduced by construction, r and r' are normal forms of $\dot{\mathcal{R}}$. The same holds for $r'\sigma$ because normal forms are closed under renaming. We have $r \;{}_{\dot{\mathcal{R}}}\!\leftarrow \ell = \ell'\sigma \to_{\dot{\mathcal{R}}} r'\sigma$. Since $\dot{\mathcal{R}}$ is confluent as a consequence of Lemma 2.4, $r = r'\sigma$. Hence $\ell' \to r'$ is a variant of $\ell \to r$, contradicting the assumption that TRSs are variant-free.

From Lemma 2.4 we infer that the TRSs $\dot{\mathcal{R}}$ and $\ddot{\mathcal{R}}$ are complete and normalization equivalent to $\mathcal{R}$. The TRS $\ddot{\mathcal{R}}$ is right-reduced because $\ddot{\mathcal{R}} \subseteq \dot{\mathcal{R}}$ and $\dot{\mathcal{R}}$ is right-reduced. From $\mathsf{NF}(\ddot{\mathcal{R}}) = \mathsf{NF}(\dot{\mathcal{R}})$ we easily infer that $\ddot{\mathcal{R}}$ is left-reduced. It follows that $\ddot{\mathcal{R}}$ is canonical. It remains to show that $\ddot{\mathcal{R}}$ is not only normalization equivalent but also (conversion) equivalent to $\mathcal{R}$. This is an immediate consequence of Lemma 2.3. □

For our next result we need the following technical lemma.

Lemma 3.8. *Let $\mathcal{R}$ be a right-reduced TRS and let s be a reducible term which is minimal with respect to $\rhd$. If $s \to_{\mathcal{R}}^{+} t$ then $s \to t$ is a variant of a rule in $\mathcal{R}$*

Proof. Let $\ell \to r$ be the rewrite rule that is used in the first step from s to t. So $s \unrhd \ell$. By assumption, $s \rhd \ell$ does not hold and thus $s \doteq \ell$. According to Lemma 3.2 there exists a renaming σ such that $s = \ell\sigma$. We have $s \to_{\mathcal{R}} r\sigma \to_{\mathcal{R}}^{*} t$. Because $\mathcal{R}$ is right-reduced, $r \in \mathsf{NF}(\mathcal{R})$. Since normal forms are closed under renaming, also $r\sigma \in \mathsf{NF}(\mathcal{R})$ and thus $r\sigma = t$. It follows that $s \to t$ is a variant of $\ell \to r$. □

In our formalization, the above proof is the first spot where we actually need that $\mathcal{R}$ satisfies the variable condition (more precisely, right-hand sides of rules do not introduce fresh variables).

The next result is the main result of this note.

Theorem 3.9. *Normalization equivalent reduced TRSs are unique up to literal similarity.*

Proof. Let $\mathcal{R}$ and $\mathcal{S}$ be normalization equivalent reduced TRSs. Suppose $\ell \to r \in \mathcal{R}$. Because $\mathcal{R}$ is right-reduced, $r \in \mathsf{NF}(\mathcal{R})$ and thus $\ell \neq r$. Hence $\ell \to_{\mathcal{S}}^{+} r$ by normalization equivalence. Because $\mathcal{R}$ is left-reduced, ℓ is a minimal (with respect to $\rhd$) $\mathcal{R}$-reducible term. Another application of normalization equivalence yields that ℓ is minimal $\mathcal{S}$-reducible. Hence $\ell \to r$ is a variant of a rule in $\mathcal{S}$ by Lemma 3.8. □

We show that the corresponding result of Métivier [4, Theorem 8] is an easy consequence of Theorem 3.9. Here a TRS $\mathcal{R}$ is said to be compatible with a reduction order $>$ if $\ell > r$ for every rewrite rule $\ell \to r$ of $\mathcal{R}$.

Theorem 3.10. *Let $\mathcal{R}$ and $\mathcal{S}$ be equivalent canonical TRSs. If $\mathcal{R}$ and $\mathcal{S}$ are compatible with the same reduction order then $\mathcal{R} \doteq \mathcal{S}$.*

Proof. Suppose $\mathcal{R}$ and $\mathcal{S}$ are compatible with the reduction order $>$. We show that $\to_{\mathcal{R}}^{!} \subseteq \to_{\mathcal{S}}^{!}$. Let $s \to_{\mathcal{R}}^{!} t$. We show that $t \in \mathsf{NF}(\mathcal{S})$. Let u be the unique $\mathcal{S}$-normal form of t. We have $t \to_{\mathcal{S}}^{!} u$ and thus $t \leftrightarrow_{\mathcal{R}}^{*} u$ because $\mathcal{R}$ and $\mathcal{S}$ are equivalent. Since $t \in \mathsf{NF}(\mathcal{R})$, we have $u \to_{\mathcal{R}}^{!} t$. If $t \neq u$ then both $t > u$ (as $t \to_{\mathcal{S}}^{!} u$) and $u > t$ (as $u \to_{\mathcal{R}}^{!} t$), which is impossible. Hence $t = u$ and thus $t \in \mathsf{NF}(\mathcal{S})$. Together with $s \leftrightarrow_{\mathcal{S}}^{*} t$, which follows from the equivalence of $\mathcal{R}$ and $\mathcal{S}$, we conclude that $s \to_{\mathcal{S}}^{!} t$. We obtain $\to_{\mathcal{S}}^{!} \subseteq \to_{\mathcal{R}}^{!}$ by symmetry. Hence $\mathcal{R}$ and $\mathcal{S}$ are normalization equivalent and the result follows from Theorem 3.9. □

References

[1] B. Gramlich. On interreduction of semi-complete term rewriting systems. *Theoretical Computer Science*, 258(1-2):435–451, 2001. doi:10.1016/S0304-3975(00)00030-X.

[2] N. Hirokawa, A. Middeldorp, and C. Sternagel. A new and formalized proof of abstract completion. In *Proc. 5th International Conference on Interactive Theorem Proving*, volume 8558 of *Lecture Notes in Computer Science*, 2014. To appear.

[3] J.W. Klop. *Combinatory Reduction Systems.* PhD thesis, Utrecht University, 1980.

[4] Y. Métivier. About the rewriting systems produced by the Knuth-Bendix completion algorithm. *Information Processing Letters*, 16(1):31–34, 1983. doi:10.1016/0020-0190(83)90009-1.

[5] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 2003.

Certification of Confluence Proofs using CeTA[*]

Julian Nagele and René Thiemann

University of Innsbruck, Austria, {julian.nagele|rene.thiemann}@uibk.ac.at

1 Introduction

CeTA was originally developed as a tool for certifying termination proofs [5], which have to be provided as certificates in the CPF-format. Given a certificate CeTA will either answer `CERTIFIED`, or return a detailed error message why the proof was `REJECTED`. Its correctness is formally proven as part of IsaFoR, the **Isa**belle **F**ormalization **o**f **R**ewriting: IsaFoR contains executable "check"-functions for each formalized proof technique together with formal proofs that whenever such a check is accepted, the technique is applied correctly. Isabelle's code-generator is then used to obtain CeTA.[1] By now, CeTA can also be used for certifying confluence and non-confluence proofs. In this system description, we give an overview on what kind of proofs are supported, and what information has to be given in the certificates. As we will see, only little information is required and so we hope that CSI [8] will not stay the only confluence tool that can produce certificates.

2 Terminating Term Rewrite Systems (TRSs)

It is well known that confluence of terminating TRSs is decidable by checking joinability of all critical pairs. The latter can be decided by reducing both terms of a critical pair to arbitrary normal forms and then checking if these are equal. This technique is also supported in CeTA, where in the certificate one just has to provide the termination proof and CeTA automatically constructs all critical pairs and checks their joinability by rewriting to normal forms. Alternatively one can also specify to check joinability by an automatic breadth-first search. Finally one can completely provide the joining sequences for all critical pairs in the certificate. Although the latter results in more verbose certificates, which are harder to produce, they are faster to check as no search is required for certification. For example, for $\mathcal{R} = \mathcal{R}_{\mathsf{ack}} \cup \{\mathsf{f}(x) \to x, \mathsf{a} \to \mathsf{ack}(1000, 1000), \mathsf{a} \to \mathsf{f}(\mathsf{ack}(1000, 1000))\}$, where $\mathcal{R}_{\mathsf{ack}}$ is a convergent TRS for the Ackermann-function, all critical pairs are joinable, but rewriting to normal form won't work.

3 Certificates for Confluence

IsaFoR contains formalizations of two techniques that ensure confluence and do not demand termination: strongly closed and linear TRSs as well as weakly orthogonal TRSs are confluent.

For the latter, the certificate only consists of the statement that the TRS is weakly orthogonal, which is a syntactic criterion that can easily be checked by CeTA. For the former criterion, the interesting part is to ensure that a given TRS $\mathcal{R}$ is strongly closed, i.e., for every critical pair (s, t) there are terms u and v such that $s \to^*_{\mathcal{R}} u \;{}_{\mathcal{R}}^{=}\!\!\leftarrow t$ and $s \to^{=}_{\mathcal{R}} v \;{}_{\mathcal{R}}^{*}\!\!\leftarrow t$. Clearly, rewriting to normal forms is of little use here, so we just offer a breadth-first search in CeTA. In the certificate one just has to provide a bound on the length of the joining derivations. The reason for requiring the explicit bound is that in Isabelle all functions have to be total.In contrast to

[*]Supported by the Austrian Science Fund (FWF), projects P22467 and P22767.

[1]At `http://cl-informatik.uibk.ac.at/software/ceta/` one can access CeTA, IsaFoR, and the CPF-format.

Section 2, here $\mathcal{R}$ is not necessarily terminating, and thus an unbounded breadth-first search might be non-terminating, whereas an explicit bound on the depth easily ensures totality.

At this point, let us recall our notions of TRSs and critical pairs: as usual a TRS $\mathcal{R}$ is just a set of rewrite rules. However we do not assume the following standard variable conditions:

$$VC_{lhs}(\mathcal{R}) = \forall \ell \to r \in \mathcal{R}.\, \ell \notin \mathcal{V} \qquad VC_{\supseteq}(\mathcal{R}) = \forall \ell \to r \in \mathcal{R}.\, \mathcal{V}(\ell) \supseteq \mathcal{V}(r)$$

The critical pairs of a TRS $\mathcal{R}$ are defined as

$$CP(\mathcal{R}) = \{(r\sigma, C[r']\sigma) \mid \ell \to r \in \mathcal{R},\, \ell' \to r' \in \mathcal{R},\, \ell = C[u],\, u \notin \mathcal{V},\, mgu(u, \ell') = \sigma\}$$

where it is assumed that the variables in $\ell \to r$ and $\ell' \to r'$ have been renamed apart. We do not exclude root overlaps of a rule with itself, which gives rise to trivial critical pairs of the form $(r\sigma, r\sigma)$. Therefore, most techniques in IsaFoR that rely on critical pairs immediately try to remove all trivial critical pairs, i.e., they consider $\{(s,t) \in CP(\mathcal{R}) \mid s \neq t\}$ instead of $CP(\mathcal{R})$. So, in practice these additional critical pairs do not play a role. However, for TRSs that do not satisfy the variable conditions they are essential. For example, for the TRS $\mathcal{R}_1 = \{\mathsf{a} \to y\}$ over signature $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ we have $CP(\mathcal{R}) = \{(x, y)\}$, whereas without root-overlaps with the same rule there would be no critical pair and we might wrongly conclude confluence via orthogonality.

The confluence criterion of weak orthogonality not only implicitly demands $VC_{\supseteq}(\mathcal{R})$, but explicitly demands $VC_{lhs}(\mathcal{R})$. In contrast, none of the variable conditions is required for strongly closed and linear TRSs. Hence, the following two TRSs are confluent via this criterion: $\mathcal{R}_2 = \{x \to \mathsf{f}(x), y \to \mathsf{g}(y)\}$ is strongly closed as there are no critical pairs, and $\mathcal{R}_3 = \{\mathsf{a} \to \mathsf{f}(x), \mathsf{f}(x) \to \mathsf{b}\}$ is strongly closed as the only non-trivial critical pair is $(\mathsf{f}(x), \mathsf{f}(y))$, which is obviously joinable in one step to b. Also $\mathcal{R}_4 = \{\mathsf{a} \to \mathsf{f}(x), \mathsf{f}(x) \to \mathsf{b}, x \to \mathsf{f}(\mathsf{g}(x))\}$—which satisfies neither of the variable conditions—is strongly closed and linear, and thus confluent. Similarly as for weak orthogonality, the addition of root overlaps w.r.t. the same rule is essential, as otherwise the non-confluent and linear TRS $\mathcal{R}_1$ would be strongly closed.

4 Disproving Confluence via Non-Joinable Forks

One way to disprove confluence of an arbitrary, possibly non-terminating TRS $\mathcal{R}$ is to provide a non-joinable fork, i.e., $s \to_{\mathcal{R}}^* t_1$ and $s \to_{\mathcal{R}}^* t_2$ such that t_1 and t_2 have no common reduct. To certify these proofs, in CeTA we demand the concrete derivations from s to t_1 and t_2 and additionally a certificate that t_1 and t_2 are not joinable, which is clearly the more interesting part. To this end, we generalize the notion of non-joinability to two TRSs, which allows us to conveniently and modularly formalize several existing techniques for non-joinability. Initially, $\mathcal{R}_1 = \mathcal{R}_2 = \mathcal{R}$ and any change on one of the TRSs is currently internally computed by CeTA.

$$NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1, t_2) = (\neg\exists u.\, t_1 \to_{\mathcal{R}_1}^* u \wedge t_2 \to_{\mathcal{R}_2}^* u)$$

4.1 Grounding

Clearly, $NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1\sigma, t_2\sigma)$ implies $NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1, t_2)$ for some arbitrary substitution σ. This substitution has to be provided in the certificate and can be used replace each variable in t_1 and t_2 by some fresh constant. Grounding can be beneficial for other non-joinability techniques.

4.2 Tcap and Unification

The function $tcap_{\mathcal{R}}$ can approximate an upper part of a term where no rewriting with $\mathcal{R}$ is possible, and thus, remains unchanged by rewriting. Hence, it suffices to check that $tcap_{\mathcal{R}_1}(t_1)$

is not unifiable with $tcap_{\mathcal{R}_2}(t_2)$ to ensure $NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1,t_2)$.

Since $tcap_{\mathcal{R}_i}$ replaces variables by fresh ones, it is beneficial to apply grounding beforehand [8]. To this end, CeTA computes a suitable grounding substitution, if some t_i is not a ground term. Because of grounding, this criterion fully subsumes the criterion, that two different normal forms are not joinable. Nevertheless one can also refer to the latter criterion in certificates.

4.3 Usable Rules for Reachability

In [1] the usable rules for reachability $\mathcal{U}_r$ have been defined (via some inductive definition of auxiliary usable rules $\mathcal{U}_0$). They have the crucial property that $t \to_{\mathcal{R}}^* s$ implies $t \to_{\mathcal{U}_r(\mathcal{R},t)}^* s$. This property immediately shows the following theorem.

Theorem 1. *$NJ_{\mathcal{U}_r(\mathcal{R}_1,t_1),\mathcal{U}_r(\mathcal{R}_2,t_2)}(t_1,t_2)$ implies $NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1,t_2)$.*

Whereas the crucial property was easily formalized within IsaFoR following the original proof, it was actually more complicated to provide an implementation of usable rules that turns the inductive definition of $\mathcal{U}_0$ into executable code. Note that we did not have this problem in previous work on usable rules [3] where we explicitly demand that the set of usable rules is provided in the certificate. However, due to our implementation of usable rules, we no longer require the set of usable rules in the certificate.

4.4 Discrimination Pairs

In [1] term orders are utilized to prove non-joinability. To be precise, $(\succsim, \succ)$ is a discrimination pair iff $\succsim$ is a rewrite order, $\succ$ is irreflexive, and $\succsim \circ \succ \subseteq \succ$.[2] We formalized the following theorem, which in combination with Theorem 1 completely simulates [1, Theorem 12].

Theorem 2. *If $(\succsim, \succ)$ is a discrimination pair, $\mathcal{R}_1^{-1} \cup \mathcal{R}_2 \subseteq \succsim$, and $t_1 \succ t_2$ then $NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1,t_2)$.*

Proof. We perform a proof by contradiction, so assume $t_1 \to_{\mathcal{R}_1}^* u$ and $t_2 \to_{\mathcal{R}_2}^* u$ and hence $t_2 \to_{\mathcal{R}_1^{-1}\cup\mathcal{R}_2}^* t_1$. Then by the preconditions we obtain $t_2 \succsim^* t_1 \succ t_2$. Iteratively applying $\succsim \circ \succ \subseteq \succ$ yields $t_2 \succ t_2$ in contradiction to irreflexivity of $\succ$. ☐

We have also proven within IsaFoR that every reduction pair is a discrimination pair, and thus one can use all reduction pairs that are available in CeTA in the certificate.

4.5 Argument Filters

In [1] it is shown that argument filters π are useful for non-confluence proofs. The essence is

Observation 3. *$NJ_{\pi(\mathcal{R}_1),\pi(\mathcal{R}_2)}(\pi(t_1),\pi(t_2))$ implies $NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1,t_2)$.*

Consequently, one may show non-joinability by applying an argument filter and then continue on the filtered problem. At this point we can completely simulate [1, Theorem 14]: apply usable rules, apply argument filter, apply usable rules, apply discrimination pair.

4.6 Interpretations

Let $\mathcal{F}$ be some signature. Let $\mathcal{A}$ be a weakly monotone $\mathcal{F}$-algebra $(A, (f^{\mathcal{A}})_{f\in\mathcal{F}}, \geq)$, i.e., $f^{\mathcal{A}} : A^n \to A$ for each n-ary symbol $f \in \mathcal{F}$, $\geq$ is a partial order, and for all a, b, f, $a \geq b$ implies

[2]Note, that unlike what is said in [1], one does not require $\succ \circ \succsim \subseteq \succ$.

$f^{\mathcal{A}}(\dots,a,\dots) \geq f^{\mathcal{A}}(\dots,b,\dots)$. $\mathcal{A}$ is a quasi-model for $\mathcal{R}$ iff $[\![\ell]\!]_{\mathcal{A},\alpha} \geq [\![r]\!]_{\mathcal{A},\alpha}$ for all $\ell \to r \in \mathcal{R}$ and every valuation $\alpha : \mathcal{V} \to A$. Let α_d be some default valuation.

Theorem 4. *If $\mathcal{A}$ is a quasi-model of $\mathcal{R}_1^{-1} \cup \mathcal{R}_2$ and $[\![t_2]\!]_{\mathcal{A},\alpha_d} \not\geq [\![t_1]\!]_{\mathcal{A},\alpha_d}$ then $NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1,t_2)$.*

Proof. Similar as for Theorem 2. Given $t_2 \to^*_{\mathcal{R}_1^{-1}\cup\mathcal{R}_2} t_1$ and the quasi-model condition we conclude $[\![t_2]\!]_{\mathcal{A},\alpha_d} \geq [\![t_1]\!]_{\mathcal{A},\alpha_d}$. This is an immediate contradiction to $[\![t_2]\!]_{\mathcal{A},\alpha_d} \not\geq [\![t_1]\!]_{\mathcal{A},\alpha_d}$. $\square$

This proof was easy to formalize as it could reuse the formalization of semantic labeling [4], which also includes algorithms to check the quasi-model conditions as well as a format for models in the certificate. Here, CeTA is currently restricted to algebras over finite domains. Moreover, the valuation α_d cannot be specified in the certificate. However, by previously applying grounding, the choice of α_d does not matter any longer.

Note that in contrast to [1, Theorem 10], we only require $[\![t_2]\!]_{\mathcal{A},\alpha_d} \not\geq [\![t_1]\!]_{\mathcal{A},\alpha_d}$ instead of $[\![t_2]\!]_{\mathcal{A},\alpha_d} \not\geq [\![t_1]\!]_{\mathcal{A},\alpha_d} \wedge [\![t_1]\!]_{\mathcal{A},\alpha_d} \geq [\![t_2]\!]_{\mathcal{A},\alpha_d}$. This has an immediate advantage, namely that we can derive [1, Corollary 6] as a consequence: instantiate $\geq$ by equality, then weak monotonicity is always guaranteed, the quasi-model condition becomes a model condition, and $[\![t_2]\!]_{\mathcal{A},\alpha_d} \not\geq [\![t_1]\!]_{\mathcal{A},\alpha_d}$ is equivalent to $[\![t_1]\!]_{\mathcal{A},\alpha_d} \neq [\![t_2]\!]_{\mathcal{A},\alpha_d}$. Moreover, the usable rules can easily be integrated as a preprocessing step in the same way as we did for discrimination pairs.

Further note that [1, Corollary 6] can also simulate [1, Theorem 5], by just taking the quotient algebra. Therefore, by Theorems 1, 2, and 4, and Observation 3 we can now simulate all non-joinability criteria of [1] and CeTA can also certify all example proofs of [1].

4.7 Tree Automata

A bottom-up tree automaton $\mathcal{A}$ is a quadruple $(\mathcal{Q}, \mathcal{F}, \Delta, \mathcal{Q}_f)$ with states $\mathcal{Q}$, signature $\mathcal{F}$, transitions Δ, and final states $\mathcal{Q}_f$, and $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{T}(\mathcal{F})$ denotes the accepted regular tree language. We say that $\mathcal{A}$ is closed under $\mathcal{R}$ if $\{t \mid s \in \mathcal{L}(\mathcal{A}), s \to_{\mathcal{R}} t\} \subseteq \mathcal{L}(\mathcal{A})$.

Observation 5. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be tree automata. If $t_i \in \mathcal{L}(\mathcal{A}_i)$ and $\mathcal{A}_i$ is closed under $\mathcal{R}_i$ for $i = 1, 2$, and $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2) = \varnothing$ then $NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1,t_2)$.*

For checking these non-joinability certificates, CeTA implemented standard tree automata algorithms for membership, intersection, and emptiness. The most difficult part is checking whether $\mathcal{A}$ is closed under $\mathcal{R}$ for some $\mathcal{A}$ and $\mathcal{R}$. Here, CeTA provides three alternatives. One can refer to Genet's criterion of compatibility, or use the more liberal condition of state-compatibility [2], which requires an additional compatibility relation in the certificate, or one can just refer to the decision procedure [2], which currently requires a deterministic automaton as input. Since all of the conditions have been formalized under the condition $VC_{\supseteq}(\mathcal{R})$, Observation 5 can only be applied if both TRSs satisfy this variable condition. Moreover, grounding is an essential preprocessing step, since tree automata only accept ground terms.

Example 6. *Let $\mathcal{R}_5 = \{\mathsf{a} \to \mathsf{b}_1, \mathsf{a} \to \mathsf{b}_2, x \to \mathsf{f}(x)\}$. Non-confluence can easily be shown since the critical pair $(\mathsf{b}_1, \mathsf{b}_2)$ is not joinable: Take the automata $\mathcal{A}_i = (\{1\}, \mathcal{F}, \{\mathsf{f}(1) \to 1, \mathsf{b}_i \to 1\}, \{1\})$, which satisfy all conditions of Observation 5.*

5 Modularity of Confluence

In [6] it was proven that confluence is a modular property for disjoint unions of TRSs. Whereas a certificate for applying this proof technique is trivial by just providing the decomposition, we cannot certify these proofs, since currently a formalization of this modularity result is missing.

However, we at least formalized the easy direction of the modularity theorem that non-confluence of one of the TRSs implies non-confluence of the disjoint union, and we can thus certify non-confluence proofs in a modular way. We base our certifier on the following theorem. Here, we assume an infinite set of symbols[3] and finite signatures $\mathcal{F}(\mathcal{R})$ and $\mathcal{F}(\mathcal{S})$ of the TRSs.

Theorem 7. *Let $\mathcal{F}(\mathcal{R}) \cap \mathcal{F}(\mathcal{S}) = \varnothing$, let $VC_{\supseteq}(\mathcal{R})$, let $VC_{lhs}(\mathcal{S})$. Then $\neg CR(\mathcal{R})$ implies $\neg CR(\mathcal{R} \cup \mathcal{S})$.*

Proof. By assuming $\neg CR(\mathcal{R})$ there are s, t, u such that $s \rightarrow^*_{\mathcal{R}} t$, $s \rightarrow^*_{\mathcal{R}} u$, and $NJ_{\mathcal{R},\mathcal{R}}(t, u)$. Since $\mathcal{F}(\mathcal{R}) \cap \mathcal{F}(\mathcal{S}) = \varnothing$, w.l.o.g. we assume $\mathcal{F}(s) \cap \mathcal{F}(\mathcal{S}) = \varnothing$.[4] By $VC_{\supseteq}(\mathcal{R})$ we conclude that also $(\mathcal{F}(t) \cup \mathcal{F}(u)) \cap \mathcal{F}(\mathcal{S}) = \varnothing$ must hold. Assume that t and u are joinable by $\mathcal{R} \cup \mathcal{S}$. By looking at the function symbols and using $VC_{lhs}(\mathcal{S})$ we conclude that the joining sequences cannot use any rule from $\mathcal{S}$. Hence, t and u are joinable by $\mathcal{R}$, a contradiction to $NJ_{\mathcal{R},\mathcal{R}}(t, u)$. ☐

There is an asymmetry in the modularity theorem, namely that $\mathcal{R}$ and $\mathcal{S}$ have to satisfy different variable conditions. Note that in general it is not possible to weaken these conditions as can be seen by the following two examples of [7, Example 20 and example in Section 5.3]. If $\mathcal{R} = \{\mathsf{a} \to \mathsf{b}, \mathsf{a} \to \mathsf{c}\}$ and $\mathcal{S} = \{x \to \mathsf{d}\}$ (or if $\mathcal{R} = \{\mathsf{f}(x, y) \to \mathsf{f}(z, z), \mathsf{f}(\mathsf{b}, \mathsf{c}) \to \mathsf{a}, \mathsf{b} \to \mathsf{d}, \mathsf{c} \to \mathsf{d}\}$ and $\mathcal{S} = \{\mathsf{g}(y, x, x) \to y, \mathsf{g}(x, x, y) \to y\}$) then $\neg CR(\mathcal{R})$, but $CR(\mathcal{R} \cup \mathcal{S})$. Hence $VC_{lhs}(\mathcal{S})$ (or $VC_{\supseteq}(\mathcal{R})$) cannot be dropped from Theorem 7. The relaxation on the variable conditions sometimes is helpful:

Example 8. *Consider the non-confluent $\mathcal{R}_5$ of Example 6 and $\mathcal{S} = \{\mathsf{g}(x) \to y\}$. By Theorem 7 and $\neg CR(\mathcal{R}_5)$ we immediately conclude $\neg CR(\mathcal{R}_5 \cup \mathcal{S})$. Note that the proof in Example 6 is not applicable to $\mathcal{R}_5 \cup \mathcal{S}$, since $VC_{\supseteq}(\mathcal{R}_5 \cup \mathcal{S})$ does not hold.*

Acknowledgments We thank Thomas Sternagel for his formalized breadth-first search algorithm, and Bertram Felgenhauer and Harald Zankl for integrating CPF-export into CSI. The authors are listed in alphabetical order regardless of individual contributions or seniority.

References

[1] T. Aoto. Disproving confluence of term rewriting systems by interpretation and ordering. In *FroCoS*, volume 8152 of *LNCS*, pages 311–326, 2013.

[2] B. Felgenhauer and R. Thiemann. Reachability analysis with state-compatible automata. In *LATA*, volume 8370 of *LNCS*, pages 347–359, 2014.

[3] C. Sternagel and R. Thiemann. Certified subterm criterion and certified usable rules. In *RTA*, volume 6 of *LIPIcs*, pages 325–340, 2010.

[4] C. Sternagel and R. Thiemann. Modular and certified semantic labeling and unlabeling. In *RTA*, volume 10 of *LIPIcs*, pages 329–344, 2011.

[5] R. Thiemann and C. Sternagel. Certification of termination proofs using CeTA. In *TPHOLs*, volume 5674 of *LNCS*, pages 452–468, 2009.

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

[7] V. van Oostrom. Modularity of confluence. In *IJCAR*, volume 5195 of *LNCS*, pages 348–363, 2008.

[8] H. Zankl, B. Felgenhauer, and A. Middeldorp. CSI – A confluence tool. In *CADE*, volume 6803 of *LNAI*, pages 499–505, 2011.

[3] Note that in IsaFoR function symbols do not come with a fixed arity.

[4] Here is exactly the point where in the formalization we use the assumptions of finite signatures and an infinite set of symbols. Then it is always possible to rename all symbols in $\mathcal{F}(s) \cap \mathcal{F}(\mathcal{S})$ into fresh ones.

On Proving Confluence of Conditional Term Rewriting Systems via the Computationally Equivalent Transformation*

Naoki Nishida[1], Makishi Yanagisawa[1] and Karl Gmeiner[2]

[1] Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 4648603, Japan
nishida@is.nagoya-u.ac.jp makishi@trs.cm.is.nagoya-u.ac.jp
[2] Institute of Computer Science, UAS Technikum Wien
gmeiner@technikum-wien.at

Abstract

This paper improves the existing criterion for proving confluence of a normal conditional term rewriting system (CTRS) via the Şerbănuţă-Roşu transformation, a computationally equivalent transformation of CTRSs into unconditional term rewriting systems (TRS), showing that a weakly left-linear normal CTRS is confluent if the transformed TRS is confluent. Then, we discuss usefulness of the optimization of the Şerbănuţă-Roşu transformation, which has informally been proposed in the literature.

1 Introduction

Conditional term rewriting is known to be much more complicated than unconditional term rewriting in the sense of analyzing properties, e.g., operational termination [8], confluence [16], reachability [4], and so on. A popular approach to the analysis of conditional term rewriting systems (CTRS) is to transform a CTRS into an unconditional term rewriting system (TRS) that is an overapproximation of the CTRS in terms of reduction. This approach enables us to use techniques for the analysis of TRSs, which are well investigated in the literature. For example, if the transformed TRS is terminating then the CTRS is operationally terminating [3].

There are two approaches to such transformations: *unravelings* [9, 10] proposed by Marchiori (see, e.g., [5, 11]), and a transformation [17] proposed by Viry (see, e.g., [14, 5]). The latest transformation based on Viry's approach is a *computationally equivalent* transformation proposed by Şerbănuţă and Roşu [14, 15], called *the SR transformation*. This converts a left-linear confluent normal CTRS into a TRS which is computationally equivalent to the CTRS. This means that the converted TRS can be used to exactly simulate reduction sequences of the CTRS to normal forms — there is no reduction to from possible initial terms to normal forms, which does not hold on the original CTRS. Another interesting use of the SR transformation is to prove confluence of a left-linear normal CTRS: if the converted TRS is confluent *on reachable terms*, then the CTRS is confluent [14]. However, as far as we know, there are no formal method to prove confluence on reachable terms.

In this paper, we revisit the SR transformation from the viewpoint of proving confluence of CTRSs, especially normal CTRSs. First, we improve the existing criterion [14] for proving confluence of normal CTRSs via the SR transformation, showing that a *weakly left-linear* normal CTRS is confluent if the transformed TRS is confluent on reachable terms. Then, by an example, we show uselessness of the improved criterion for the case that we attempt to use confluence on

*The research in this paper is partly supported by the Austrian Science Fund (FWF) international project I963 and the Japan Society for the Promotion of Science (JSPS).

arbitrary terms instead of confluence on reachable terms. Finally, we discuss usefulness of the optimization of the SR transformation, which has informally been proposed in the literature [14].

2 The SR transformation for Normal CTRSs

In this section, we briefly recall the SR transformation [14] for normal CTRSs. For the sake of readability, as in [14], we restrict our interest to normal *1*-CTRSs, any rule of which has *at most one condition.*

This paper assumes familiarity of readers with the basic notions and notations of term rewriting [2, 13]. We only give the definition of *normal CTRSs.* Throughout the paper, we use $\mathcal{V}$ as a countably infinite set of *variables.* An *(oriented) conditional rewrite rule* over a signature $\mathcal{F}$ is a triple (l, r, c), denoted by $l \to r \Leftarrow c$, such that the *left-hand side* l is a non-variable term in $\mathcal{T}(\mathcal{F}, \mathcal{V})$, the *right-hand side* r is a term in $\mathcal{T}(\mathcal{F}, \mathcal{V})$, and the *conditional part* c is either an empty sequence or a pair $s \twoheadrightarrow t$ of terms s and t over $\mathcal{F}$. We may abbreviate it to $l \to r$ if the conditional part is the empty sequence. A *conditional term rewriting system* (CTRS) is a finite set $\mathcal{R}$ of conditional rewrite rules, and it is called a *normal 1-CTRS* if, for every rule $l \to r \Leftarrow s \twoheadrightarrow t \in \mathcal{R}$, $\mathcal{V}ar(l) \supseteq \mathcal{V}ar(r) \cup \mathcal{V}ar(s)$ and the term t is a ground normal form w.r.t. the *underlying unconditional system* $\mathcal{R}_u = \{l \to r \mid l \to r \Leftarrow c \in \mathcal{R}\}$. The sets of *defined symbols* and *constructors* of $\mathcal{R}$ are denoted by $\mathcal{D}_\mathcal{R}$ and $\mathcal{C}_\mathcal{R}$, respectively.

In the following, the word "conditional rule" is used for representing rules having exactly one condition. We often denote a sequence $o_i, o_{i+1}, \ldots, o_j$ of objects by $\boldsymbol{o}_{i..j}$. Moreover, for the application of a mapping op to $\boldsymbol{o}_{i..j}$, we denote $op(o_i), \ldots, op(o_j)$ by $op(\boldsymbol{o}_{i..j})$, e.g., for a sequence $\boldsymbol{t}_{i..j}$ of terms and a substitution θ, we denote $t_i\theta, \ldots, t_j\theta$ by $\theta(\boldsymbol{t}_{i..j})$.

Before transforming a CTRS $\mathcal{R}$, we extend the signature of $\mathcal{R}$ as follows: we leave constructors of $\mathcal{R}$ without any change; the arity of an n-ary defined symbol f is expanded to $n+m$ where f has m conditional rules in $\mathcal{R}$, and we replace f by $\bar{\mathsf{f}}$ with the arity $n+m$; a fresh constant $\perp$ and a fresh unary symbol $\langle\cdot\rangle$ are introduced. We denote the extended signature by $\overline{\mathcal{F}}$: $\overline{\mathcal{F}} = \{\mathsf{c} \mid \mathsf{c} \in \mathcal{C}_\mathcal{R}\} \cup \{\bar{\mathsf{f}} \mid \mathsf{f} \in \mathcal{D}_\mathcal{R}\} \cup \{\perp, \langle\cdot\rangle\}$. We introduce the mapping $\mathtt{ext}(\cdot)$ to extend the arguments of defined symbols by applying to terms inductively as follows: $\mathtt{ext}(x) = x$ for $x \in \mathcal{V}$; $\mathtt{ext}(\mathsf{c}(\boldsymbol{t}_{1..n})) = \mathsf{c}(\mathtt{ext}(\boldsymbol{t}_{1..n}))$ for $\mathsf{c}/n \in \mathcal{C}_\mathcal{R}$; $\mathtt{ext}(\mathsf{f}(\boldsymbol{t}_{1..n})) = \bar{\mathsf{f}}(\mathtt{ext}(\boldsymbol{t}_{1..n}), \boldsymbol{z}_{1..m})$ for $\mathsf{f}/n \in \mathcal{D}_\mathcal{R}$ where $\mathtt{arity}_{\overline{\mathcal{F}}}(\bar{\mathsf{f}}) = n+m$ and $z_1, \ldots, z_m$ are fresh variables. The expanded arguments of $\bar{\mathsf{f}}$ are used for evaluating the corresponding conditions, and the fresh constant $\perp$ is introduced to the expanded arguments of defined symbols, which does not store any evaluation. We define a mapping $\bar{\cdot}$ from $T(\mathcal{F}, \mathcal{V})$ to $T(\overline{\mathcal{F}}, \mathcal{V})$, which extends the arguments of defined symbols and puts $\perp$ to all the expanded arguments by applying to terms inductively as follows: $\overline{x} = x$ for $x \in \mathcal{V}$; $\overline{\mathsf{c}(\boldsymbol{t}_{1..n})} = \mathsf{c}(\overline{\boldsymbol{t}_{1..n}})$ for $\mathsf{c} \in \mathcal{C}_\mathcal{R}$; $\overline{\mathsf{f}(\boldsymbol{t}_{1..n})} = \bar{\mathsf{f}}(\overline{\boldsymbol{t}_{1..n}}, \perp, \ldots, \perp)$ for $\mathsf{f} \in \mathcal{D}_\mathcal{R}$.

The SR transformation [14] is formally defined as follows.

Definition 1 ($\mathbb{SR}$)**.** *Let $\mathsf{f}/n \in \mathcal{D}_\mathcal{R}$ that has m conditional rules in $\mathcal{R}$. Then, $\mathbb{SR}(\mathsf{f}(\boldsymbol{w}_{1..n}) \to r)$ $= \{\ \bar{\mathsf{f}}(\mathtt{ext}(\boldsymbol{w}_{1..n}), \boldsymbol{z}_{1..m}) \to \langle\overline{r}\rangle\ \}$ and, for the i-th conditional rule of* f,

$$\mathbb{SR}(\mathsf{f}(\boldsymbol{w}_{1..n}) \to r_i \Leftarrow s_i \twoheadrightarrow t_i) = \left\{ \begin{array}{l} \bar{\mathsf{f}}(\boldsymbol{w'}_{1..n}, \boldsymbol{z}_{1..i-1}, \perp, \boldsymbol{z}_{i+1..m}) \to \bar{\mathsf{f}}(\boldsymbol{w'}_{1..n}, \boldsymbol{z}_{1..i-1}, \langle\overline{s_i}\rangle, \boldsymbol{z}_{i+1..m}), \\ \bar{\mathsf{f}}(\boldsymbol{w'}_{1..n}, \boldsymbol{z}_{1..i-1}, \langle t_i\rangle, \boldsymbol{z}_{i+1..m}) \to \langle\overline{r_i}\rangle \end{array} \right\}$$

where $z_1, \ldots, z_m$ are fresh variables, and $w'_j = \mathtt{ext}(w_j)$ for all $1 \leq j \leq n$. The set of auxiliary rules is defined as follows:

$$\begin{array}{rl} \mathcal{R}_{aux} = & \{\ \langle\langle x\rangle\rangle \to \langle x\rangle\ \} \cup \{\ \mathsf{c}(\boldsymbol{x}_{1..i-1}, \langle x_i\rangle, \boldsymbol{x}_{i+1..n}) \to \langle\mathsf{c}(\boldsymbol{x}_{1..n})\rangle \mid \mathsf{c}/n \in \mathcal{C}_\mathcal{R},\ 1 \leq i \leq n\ \} \\ & \cup \{\ \bar{\mathsf{f}}(\boldsymbol{x}_{1..i-1}, \langle x_i\rangle, \boldsymbol{x}_{i+1..n}, \boldsymbol{z}_{1..m}) \to \langle\bar{\mathsf{f}}(\boldsymbol{x}_{1..n}, \perp, \ldots, \perp)\rangle \mid \mathsf{f}/n \in \mathcal{D}_\mathcal{R},\ 1 \leq i \leq n\ \} \end{array}$$

where $z_1, \ldots, z_m$ are fresh variables. The transformation $\mathbb{SR}$ is defined as follows: $\mathbb{SR}(\mathcal{R}) = \bigcup_{\rho \in \mathcal{R}} \mathbb{SR}(\rho) \cup \mathcal{R}_{aux}$. Note that $\mathbb{SR}(\mathcal{R})$ is a TRS over $\overline{\mathcal{F}}$. The backtranslation mapping $\hat{\cdot}$ for $\bar{\cdot}$ is *defined as follows:* $\widehat{x} = x$ *for* $x \in \mathcal{V}$*;* $\widehat{\mathsf{c}(\boldsymbol{t'}_{1..n})} = \mathsf{c}(\widehat{\boldsymbol{t'}_{1..n}})$ *for* $\mathsf{c} \in \mathcal{C}_{\mathcal{R}}$*;* $\widehat{\overline{\mathsf{f}}(\boldsymbol{t'}_{1..n}, \boldsymbol{u}_{1..m})} = \mathsf{f}(\widehat{\boldsymbol{t'}_{1..n}})$ *for* $\mathsf{f} \in \mathcal{D}_{\mathcal{R}}$*;* $\widehat{\langle t' \rangle} = \widehat{t'}$*. A term* t' *in* $T(\overline{\mathcal{F}}, \mathcal{V})$ *is called* reachable *if there exists a term* $s \in T(\mathcal{F}, \mathcal{V})$ *such that* $\langle \overline{s} \rangle \to^*_{\mathbb{SR}(\mathcal{R})} t'$*. We say that* $\mathbb{SR}$ *(and also* $\mathbb{SR}(\mathcal{R})$*) is* sound for (reduction of) $\mathcal{R}$ *if, for all terms* s *in* $T(\mathcal{F}, \mathcal{V})$ *and terms* $t' \in T(\overline{\mathcal{F}}, \mathcal{V})$*,* $\langle \overline{s} \rangle \to^*_{\mathbb{SR}(\mathcal{R})} t'$ *implies* $s \to^*_{\mathcal{R}} \widehat{t'}$*.*

Note that $\widehat{\cdot}$ is not defined for $\bot$, but $\widehat{\cdot}$ is a total function for reachable terms and their *structural* subterms [14].

$\mathbb{SR}$ is *complete* for all normal CTRSs [14], i.e., for all terms s and t in $T(\mathcal{F}, \mathcal{V})$, $s \to^*_{\mathcal{R}} t$ implies $\langle \overline{s} \rangle \to^*_{\mathbb{SR}(\mathcal{R})} \langle \overline{t} \rangle$. On the other hand, $\mathbb{SR}$ is not sound for all normal CTRSs [14]. The first rule in $\mathcal{R}_{aux}$ removes the nest of $\langle \cdot \rangle$, the second rule is used for shifting $\langle \cdot \rangle$ upward, and the third rules are used for both shifting $\langle \cdot \rangle$ upward and resetting the evaluation of conditions at the expanded arguments of $\overline{\mathsf{f}}$ (see [14] for the detail of the role of $\langle \cdot \rangle$ and its rules).

Example 2. Consider the following normal CTRS, a variant of the one in [14]:

$$\mathcal{R}_1 = \{\ \mathsf{e}(0) \to \mathsf{true},\quad \mathsf{e}(\mathsf{s}(x)) \to \mathsf{true} \Leftarrow \mathsf{e}(x) \twoheadrightarrow \mathsf{false},\quad \mathsf{e}(\mathsf{s}(x)) \to \mathsf{false} \Leftarrow \mathsf{e}(x) \twoheadrightarrow \mathsf{true}\ \}$$

$\mathcal{R}_1$ is transformed by $\mathbb{SR}$ as follows:

$$\mathbb{SR}(\mathcal{R}_1) = \left\{ \begin{array}{ll} \overline{\mathsf{e}}(0, z_1, z_2) \to \langle \mathsf{true} \rangle, & \\ \overline{\mathsf{e}}(\mathsf{s}(x), \bot, z_2) \to \overline{\mathsf{e}}(\mathsf{s}(x), \langle \overline{\mathsf{e}}(x, \bot, \bot) \rangle, z_2), & \overline{\mathsf{e}}(\mathsf{s}(x), \langle \mathsf{false} \rangle, z_2) \to \langle \mathsf{true} \rangle, \\ \overline{\mathsf{e}}(\mathsf{s}(x), z_1, \bot) \to \overline{\mathsf{e}}(\mathsf{s}(x), z_1, \langle \overline{\mathsf{e}}(x, \bot, \bot) \rangle), & \overline{\mathsf{e}}(\mathsf{s}(x), z_1, \langle \mathsf{true} \rangle) \to \langle \mathsf{false} \rangle, \\ \langle \langle x \rangle \rangle \to \langle x \rangle, \quad \mathsf{s}(\langle x \rangle) \to \langle \mathsf{s}(x) \rangle, & \overline{\mathsf{e}}(\langle x \rangle, z_1, z_2) \to \langle \overline{\mathsf{e}}(x, \bot, \bot) \rangle \end{array} \right\}$$

$\mathbb{SR}(\mathcal{R}_1)$ is confluent on reachable terms, but $\mathbb{SR}(\mathcal{R}_1)$ is not confluent.

3 Proving Confluence of CTRSs via the Transformation

It has been shown that if $\mathbb{SR}(\mathcal{R})$ is left-linear and confluent on reachable terms, then $\mathcal{R}$ is confluent [14]. Note that by definition, $\mathcal{R}$ is left-linear iff so is $\mathbb{SR}(\mathcal{R})$. As described in [14], in this claim, left-linearity is assumed in order to ensure soundness. Here, we give a proof so as to relax the left-linearity to soundness.

Theorem 3. *If* $\mathbb{SR}(\mathcal{R})$ *is sound for* $\mathcal{R}$ *and confluent on reachable terms, then* $\mathcal{R}$ *is confluent.*

Proof. Let s, t_1, and t_2 be terms in $T(\mathcal{F}, \mathcal{V})$ such that $t_1 \leftarrow^*_{\mathcal{R}} s \to^*_{\mathcal{R}} t_2$. It follows from completeness of $\mathbb{SR}$ that $\langle \overline{t_1} \rangle \leftarrow^*_{\mathbb{SR}(\mathcal{R})} \langle \overline{s} \rangle \to^*_{\mathbb{SR}(\mathcal{R})} \langle \overline{t_2} \rangle$. Since $\mathbb{SR}(\mathcal{R})$ is confluent on reachable terms, there exists a term $u' \in T(\overline{\mathcal{F}}, \mathcal{V})$ such that $\langle \overline{t_1} \rangle \to^*_{\mathbb{SR}(\mathcal{R})} u' \leftarrow^*_{\mathbb{SR}(\mathcal{R})} \langle \overline{t_2} \rangle$. It follows from soundness of $\mathbb{SR}(\mathcal{R})$ that $t_1 \to^*_{\mathcal{R}} \widehat{u'} \leftarrow^*_{\mathcal{R}} t_2$. Therefore, $\mathcal{R}$ is confluent. $\square$

Theorem 3 means that soundness conditions of $\mathbb{SR}$ are very useful in proving confluence of normal CTRSs. A normal CTRS $\mathcal{R}$ is called *weakly left-linear* [6] if every conditional rewrite rule having at least one condition is left-linear, and for every unconditional rule, any non-linear variable in the left-hand side does not occur in the right-hand side. Note that a left-linear CTRS is weakly left-linear. Note also that $\mathcal{R}$ is weakly left-linear iff so is $\mathbb{SR}(\mathcal{R})$. It has been shown that $\mathbb{SR}$ is sound for weakly left-linear normal CTRSs [12]. Thus, Theorem 3 provides us a new sufficient condition for confluence of normal CTRSs.

Theorem 4. *A weakly left-linear normal CTRS* $\mathcal{R}$ *is confluent if* $\mathbb{SR}(\mathcal{R})$ *is confluent on reachable terms.*

It would be difficult to directly prove that $\mathbb{SR}(\mathcal{R})$ is confluent on reachable terms. A trivial sufficient condition for the property is confluence on arbitrary terms.

Lemma 5. $\mathbb{SR}(\mathcal{R})$ *is confluent on reachable terms if* $\mathbb{SR}(\mathcal{R})$ *is confluent.*

Due to Lemma 5, to prove confluence of $\mathcal{R}$, instead of proving confluence on reachable terms, we try to prove confluence of $\mathbb{SR}(\mathcal{R})$. Unfortunately, this approach looks impractical.

Example 6. As described in Example 2, $\mathbb{SR}(\mathcal{R}_1)$ is not confluent. Since $\mathcal{R}_1$ is a basic example of normal CTRSs, the combination of Theorem 4 and Lemma 5 looks unpractical.

It is often that critical pairs between rules in $\mathbb{SR}(\mathcal{R}) \setminus \mathcal{R}_{aux}$ and $\mathcal{R}_{aux}$ are not joinable. For this reason, confluence of $\mathbb{SR}(\mathcal{R})$ is too severe to prove termination of $\mathcal{R}$, and thus, $\mathbb{SR}$ is not so useful to prove confluence of normal CTRSs.

Let us consider a direct proof for confluence on reachable terms again. When $\mathbb{SR}(\mathcal{R})$ is terminating, it would be sufficient to prove that any critical pair is joinable if an instance of the critical peak is reachable. For example, to prove confluence of $\mathbb{SR}(\mathcal{R}_1)$ on reachable terms, it would be sufficient to prove that any instance of $\overline{\mathsf{e}}(\mathsf{s}(x), \langle\mathsf{false}\rangle, \langle\mathsf{true}\rangle)$ is not reachable. However, it is in general undecidable whether a term is reachable.

Instead of such unreachability, an optimization has already been discussed in [14]. We recall the optimization via the following example.

Example 7. Consider $\mathcal{R}_1$ in Example 2 again. The overlapping rules $\mathsf{e}(\mathsf{s}(x)) \to \mathsf{true} \Leftarrow \mathsf{e}(x) \twoheadrightarrow \mathsf{false}$ and $\mathsf{e}(\mathsf{s}(x)) \to \mathsf{false} \Leftarrow \mathsf{e}(x) \twoheadrightarrow \mathsf{true}$ have the same initial terms $\mathsf{e}(x)$ of conditions to be evaluated. For this reason, we do not need two extra arguments of $\overline{\mathsf{e}}$, and then $\mathbb{SR}(\mathcal{R}_1)$ is optimized as follows, where we denote the optimization of $\mathbb{SR}$ by $\mathbb{SR}_{opt}$:

$$\mathbb{SR}_{opt}(\mathcal{R}_1) = \left\{ \begin{array}{ll} \overline{\mathsf{e}}(\mathsf{0}, z) \to \langle\mathsf{true}\rangle, & \overline{\mathsf{e}}(\mathsf{s}(x), \bot) \to \overline{\mathsf{e}}(\mathsf{s}(x), \langle\overline{\mathsf{e}}(x, \bot)\rangle), \\ \overline{\mathsf{e}}(\mathsf{s}(x), \langle\mathsf{false}\rangle) \to \langle\mathsf{true}\rangle, & \overline{\mathsf{e}}(\mathsf{s}(x), \langle\mathsf{true}\rangle) \to \langle\mathsf{false}\rangle, \qquad \ldots \end{array} \right\}$$

$\mathbb{SR}_{opt}(\mathcal{R}_1)$ is still not confluent since there are two critical pairs which are not joinable: $(\overline{\mathsf{e}}(\langle\mathsf{s}(x)\rangle, \langle\mathsf{false}\rangle), \langle\mathsf{true}\rangle)$ and $(\overline{\mathsf{e}}(\langle\mathsf{s}(x)\rangle, \langle\mathsf{true}\rangle), \langle\mathsf{false}\rangle)$. As described in [14], the introduced unary symbol $\langle\cdot\rangle$ and its related rules are not necessary for constructor CTRSs in the sense of soundness. Then, by removing them from $\mathbb{SR}(\mathcal{R}_1)$, we obtain the following orthogonal TRS:

$$\{\ \overline{\mathsf{e}}(\mathsf{0}, z) \to \mathsf{true},\ \ \overline{\mathsf{e}}(\mathsf{s}(x), \bot) \to \overline{\mathsf{e}}(\mathsf{s}(x), \overline{\mathsf{e}}(x, \bot)),\ \ \overline{\mathsf{e}}(\mathsf{s}(x), \mathsf{false}) \to \mathsf{true},\ \ \overline{\mathsf{e}}(\mathsf{s}(x), \mathsf{true}) \to \mathsf{false}\ \}$$

Note that the resulting TRS above is equivalent to that obtained by [1]. Therefore, from Theorem 4 and Lemma 5, $\mathcal{R}_1$ is confluent.

The optimization is useful in proving confluence of $\mathcal{R}_2$, but it is not always successful.

Example 8. Consider the following constructor normal CTRS, a variant of $\mathcal{R}_1$:

$$\mathcal{R}_2 = \left\{ \begin{array}{lll} \mathsf{e}(\mathsf{0}) \to \mathsf{true}, & \mathsf{e}(\mathsf{s}(x)) \to \mathsf{true} \Leftarrow \mathsf{o}(x) \twoheadrightarrow \mathsf{true}, & \mathsf{e}(\mathsf{s}(x)) \to \mathsf{false} \Leftarrow \mathsf{e}(x) \twoheadrightarrow \mathsf{true}, \\ \mathsf{o}(\mathsf{0}) \to \mathsf{false}, & \mathsf{o}(\mathsf{s}(x)) \to \mathsf{true} \Leftarrow \mathsf{e}(x) \twoheadrightarrow \mathsf{true}, & \mathsf{o}(\mathsf{s}(x)) \to \mathsf{false} \Leftarrow \mathsf{o}(x) \twoheadrightarrow \mathsf{true} \end{array} \right\}$$

For $\mathcal{R}_2$, $\mathbb{SR}_{opt}$ does not differ from $\mathbb{SR}$, i.e., $\mathbb{SR}(\mathcal{R}_2) = \mathbb{SR}_{opt}(\mathcal{R}_2)$ even when $\langle\cdot\rangle$ is not introduced. $\mathbb{SR}(\mathcal{R}_2)$ with/without $\langle\cdot\rangle$ is not confluent, and thus, useful to prove confluence of $\mathcal{R}_2$.

Finally, we consider the case of non-constructor-based CTRSs.

Example 9. Consider the following normal CTRS over the signature $\{\mathsf{0}, \mathsf{s}, \circ, \mathsf{true}, \mathsf{false}\}$ [14]:

$$\mathcal{R}_3 = \left\{ \begin{array}{lll} \multicolumn{3}{l}{\circ(x, \circ(y, l)) \to \circ(y, \circ(x, l)) \Leftarrow x < y \twoheadrightarrow \mathsf{true},} \\ \mathsf{0} < \mathsf{0} \to \mathsf{false}, & \mathsf{0} < \mathsf{s}(\mathsf{0}) \to \mathsf{true}, & \mathsf{0} < \mathsf{s}(\mathsf{s}(x)) \to \mathsf{0} < \mathsf{s}(x), \\ \mathsf{s}(\mathsf{0}) < \mathsf{0} \to \mathsf{false}, & \mathsf{s}(\mathsf{s}(x)) < \mathsf{0} \to \mathsf{s}(x) < \mathsf{0}, & \mathsf{s}(x) < \mathsf{s}(y) \to x < y \end{array} \right\}$$

$\mathcal{R}_3$ is operationally terminating and the critical pair $(\circ(x, \circ(z, \circ(y, l)), \circ(y, \circ(x, \circ(z, l)))) \Leftarrow y < z \twoheadrightarrow \mathsf{true}, x < y \twoheadrightarrow \mathsf{true}$ is joinable. Thus, $\mathcal{R}_3$ is confluent. The CTRS $\mathcal{R}_3$ is transformed by $\mathbb{SR}$ into the following TRS: $\mathbb{SR}(\mathcal{R}_3) = \mathbb{SR}_{opt}(\mathcal{R}_3) = \{\ \overline{\circ}(x, \overline{\circ}(y, l, c), \bot) \to \overline{\circ}(x, \overline{\circ}(y, l, c), \langle x \overline{<} y\rangle),$ $\overline{\circ}(x, \overline{\circ}(y, l, c), \langle\mathsf{true}\rangle) \to \langle\overline{\circ}(y, \overline{\circ}(x, l, \bot))\rangle,\ \ldots\ \}$. $\mathbb{SR}(\mathcal{R}_3)$ is not confluent because there is some critical pairs which are not joinable, e.g., $(\langle\overline{\circ}(x, \overline{\circ}(y, l, c), \bot)\rangle, \langle\overline{\circ}(y, \overline{\circ}(x, l, \bot))\rangle)$. Notice that $\overline{\circ}(\langle x\rangle, l, z_1) \to \langle\overline{\circ}(x, l, \bot)\rangle \in \mathbb{SR}(\mathcal{R}_3)$.

4 Conclusion

In this paper, we showed that a weakly left-linear normal CTRS is confluent if the transformed TRS is confluent (on reachable terms). Then, by an example, we showed uselessness of the improved criterion for the case that we attempt to use confluence on arbitrary terms instead of confluence on reachable terms. Finally, we discussed usefulness of the optimization of the SR transformation. We will make an experiment to evaluate usefulness of the optimization in terms of proving confluence of CTRSs. The discussion is not a sufficient evidence for usefulness of the optimization, and thus, we will formalize the optimization, adapt the claims which hold on $\mathbb{SR}$ to the optimization, and compare the optimization with that of unravelings [7]. We will also develop a more practical method to prove confluence of a CTRS by using critical pairs of the transformed TRS.

References

[1] S. Antoy, B. Brassel, and M. Hanus. Conditional narrowing without conditions. In *Proc. PPDP 2003*, pp. 20–31, ACM, 2003.

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

[3] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving termination of membership equational programs. In *Proc. PEPM 2004*, pp. 147–158, ACM, 2004.

[4] G. Feuillade and T. Genet. Reachability in conditional term rewriting systems. *Electr. Notes Theor. Comput. Sci.*, 86(1):133–146, 2003.

[5] K. Gmeiner and B. Gramlich. Transformations of conditional rewrite systems revisited. In *Proc. WADT 2008*, vol. 5486 of *LNCS*, pp. 166–186. Springer, 2009.

[6] K. Gmeiner, B. Gramlich, and F. Schernhammer. On (un)soundness of unravelings. In *Proc. RTA 2010*, vol. 6 of *LIPIcs*, pp. 119–134, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2010.

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

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

[9] M. Marchiori. Unravelings and ultra-properties. In *Proc. ALP 1996*, vol. 1139 of *LNCA*, pp. 107–121, Springer, 1996.

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

[11] N. Nishida, M. Sakai, and T. Sakabe. Soundness of unravelings for conditional term rewriting systems via ultra-properties related to linearity. *Logical Methods in Computer Science*, 8(3):1–49, 2012.

[12] N. Nishida, M. Yanagisawa, and K. Gmeiner. On proving soundness of the computationally equivalent transformation for normal conditional term rewriting systems by using unravelings. In *Proc. WPTE 2014*, vol. 40 of *OASIcs*, issue 1, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014 (to appear).

[13] E. Ohlebusch. *Advanced Topics in Term Rewriting.* Springer, 2002.

[14] T.-F. Şerbănuţă and G. Roşu. Computationally equivalent elimination of conditions. In *Proc. RTA 2006*, vol. 4098 of *LNCS*, pp. 19–34, Springer, 2006.

[15] T.-F. Şerbănuţă and G. Roşu. Computationally equivalent elimination of conditions. Technical Report UIUCDCS-R-2006-2693, Department of Computer Science, University of Illinois at Urbana-Champaign, 2006.

[16] T. Suzuki, A. Middeldorp, and T. Ida. Level-confluence of conditional rewrite systems with extra variables in right-hand sides. In *Proc. RTA 1995*, vol. 914 of *LNCS*, pp. 179–193, Springer, 1995.

[17] P. Viry. Elimination of conditions. *J. Symb. Comput.*, 28(3):381–401, 1999.

Confluence and Critical-Pair-Closing Systems[*]

Michio Oyamaguchi[1] and Nao Hirokawa[2]

[1] Nagoya University, Japan
[2] JAIST, Japan

Abstract

In this note we introduce *critical-pair-closing systems* which are aimed at analyzing confluence of term rewriting systems. Based on the notion two new confluence criteria are presented. One is a generalization of weak orthogonality based on relative termination, and another is a partial solution to the RTA Open Problem #13.

1 Introduction

For confluence of a left-linear term rewriting system (TRS) joinability of every critical pair is necessary but not sufficient [4]. In this note we focus attention on rewrite steps that are used for closing critical pairs. For brevity we adopt Dershowitz's critical pair notation [2].

Definition 1.1. A TRS $\mathcal{C}$ is *critical-pair-closing* for a TRS $\mathcal{R}$ if $\mathcal{C} \subseteq \widehat{\mathcal{R}}$ and ${}_{\mathcal{R}}{\leftarrow} \rtimes {\rightarrow}_{\mathcal{R}} \subseteq {\downarrow}_{\mathcal{C}}$. Here $\widehat{\mathcal{R}} = \{\ell\sigma \to r\sigma \mid \ell \to r \in \mathcal{R}$ and $x\sigma$ is a ground normal form for all $x \in \mathcal{D}\mathsf{om}(\sigma)\}$.

We present two confluence criteria. One is the criterion that a left-linear TRS $\mathcal{R}$ is confluent if there exists a confluent critical-pair-closing system $\mathcal{C}$ for $\mathcal{R}$ such that $\to_{\mathcal{C}_{\mathsf{d}}/\mathcal{R}}$ is terminating. Here $\mathcal{C}_{\mathsf{d}}$ is the set of duplicating rewrite rules of $\mathcal{C}$ and $\to_{\mathcal{C}_{\mathsf{d}}/\mathcal{R}} = \to_{\mathcal{R}}^{*} \cdot \to_{\mathcal{C}_{\mathsf{d}}} \cdot \to_{\mathcal{R}}^{*}$. In other words, if a left-linear TRS $\mathcal{R}$ that admits a confluent critical-pair-closing system $\mathcal{C}$ is not confluent, there exists an infinite rewrite sequence of $\mathcal{R}$ that contains infinitely many $\mathcal{C}_{\mathsf{d}}$-steps. Another criterion is that a left-linear TRS $\mathcal{R}$ is confluent if there is a terminating critical-pair-closing system $\mathcal{C}$ for $\mathcal{R}$ with ${}_{\mathcal{R}}\xleftarrow{>\epsilon} \rtimes \longrightarrow_{\mathcal{R}} \subseteq \mathrel{⇻}_{\mathcal{C}} \cdot {}_{\mathcal{C}}^{*}{\leftarrow}$. The symbol ${}_{\mathcal{R}}\xleftarrow{>\epsilon} \rtimes \longrightarrow_{\mathcal{R}}$ stands for the set of all *inside* critical pairs of $\mathcal{R}$ induced from non-root-overlaps. As a corollary, a left-linear TRS $\mathcal{R}$ is confluent if ${}_{\mathcal{R}}{\leftarrow} \rtimes {\rightarrow}_{\mathcal{R}} \subseteq \mathrel{⇻}_{\mathcal{C}} \cup {}_{\mathcal{C}}{⇺}$ holds for some terminating subsystem $\mathcal{C}$ of $\mathcal{R}$. This is regarded as a partial solution to one of variations of the RTA Open Problem 13: *Is a left-linear TRS $\mathcal{R}$ confluent if ${}_{\mathcal{R}}{\leftarrow} \rtimes {\rightarrow}_{\mathcal{R}} \subseteq \mathrel{⇻}_{\mathcal{R}} \cup {}_{\mathcal{R}}{⇺}$ holds?* Both criteria subsume weak orthogonality, considering the case $\mathcal{C} = \varnothing$.

2 Confluence Criteria

In this section we prove the two criteria stated in the introduction. Both rely on the following confluence criterion for abstract rewriting systems (ARSs). Let $>_1, >_2$ be strict orders and $\gtrsim_1$ a preorder such that $\gtrsim_1 \cdot >_1 \cdot \gtrsim_1 \subseteq >_1$. We define the *lexicographic product* $((>_1, \gtrsim_1), >_2)_{\text{lex}}$ as follows: $(a, b)\ ((>_1, \gtrsim_1), >_2)\ (c, d)$ if either $a >_1 c$, or $a \gtrsim_1 c$ and $b >_2 d$. The lexicographic product is well-founded, whenever $>_1$ and $>_2$ are well-founded. Let $\mathcal{A} = (A, \{\to_\alpha\}_{\alpha \in I})$ be a labeled ARS equipped with a well-founded order $>$ on the label set I. The union of $\to_\beta$ for all $\beta < \alpha$ is denoted by $\to_{\vee\alpha}$.

Lemma 2.1. *Let $\mathcal{B}$ be a confluent ARS with $\to_{\mathcal{B}} \subseteq \to_{\mathcal{A}}^{*}$. The labeled ARS $\mathcal{A}$ is confluent if*

[*]Supported by JSPS KAKENHI Grant Number 25730004.

$$\begin{tikzcd}
s \arrow[r, "n"', "\mathcal{R}"', "+\!\!\!\to" description] \arrow[d, "\mathcal{C}-\mathcal{C}_{\mathsf{d}}"'] & u \arrow[d, dotted, two heads, "\mathcal{C}"] \\
t \arrow[r, dotted, "n", "\mathcal{R}"'] & \cdot
\end{tikzcd}
\qquad
\begin{tikzcd}
s \arrow[r, "n", "\mathcal{R}"'] \arrow[d, "\mathcal{C}_{\mathsf{d}}"'] & u \arrow[d, dotted, two heads, "\mathcal{C}"] \\
t \arrow[r, dotted, "\mathcal{R}"'] & \cdot
\end{tikzcd}
\qquad
\begin{tikzcd}
s \arrow[r, "\mathcal{R}"'] \arrow[d, "\mathcal{C}"'] & v \arrow[r, "n-1", "\mathcal{R}"'] \arrow[d, dotted, two heads, "\mathcal{C}"'] & u \\
t \arrow[r, dotted, two heads, "\mathcal{C}"'] & \cdot &
\end{tikzcd}$$

non-overlapping cases overlapping case

Figure 1: The proof of Lemma 2.3.

- ${}_{\alpha}\!\leftarrow \cdot \to_{\beta} \subseteq (\to_{\mathcal{B}}^{*} \cdot \to_{\mathcal{A}} \cdot {}_{\mathcal{A}}\!\leftarrow \cdot {}_{\mathcal{B}}^{*}\!\leftarrow) \cup ({}_{\vee\alpha}\!\leftarrow \cdot \leftrightarrow_{\mathcal{B}}^{*} \cdot \to_{\vee\beta})$ *for all labels* $\alpha, \beta \in I$*, and*
- ${}_{\mathcal{B}}\!\leftarrow \cdot \to_{\alpha} \subseteq (\to_{\alpha} \cdot {}_{\mathcal{B}}^{*}\!\leftarrow) \cup (\leftrightarrow_{\mathcal{B}}^{*} \cdot \to_{\vee\alpha} \cdot {}_{\mathcal{B}}^{*}\!\leftarrow)$ *for all labels* $\alpha \in I$*.*

Proof. Let $\rightarrowtail\ =\ \to_{\mathcal{B}}^{*} \cdot \to_{\mathcal{A}}$. Since $\to_{\mathcal{A}} \subseteq\ \rightarrowtail\ \subseteq \to_{\mathcal{A}}^{*}$, it is enough to prove the diamond property of $\rightarrowtail$. The inclusion ${}_{\alpha}\!\leftarrow \cdot \to_{\mathcal{B}}^{m} \cdot {}_{\mathcal{B}}^{n}\!\leftarrow \cdot \to_{\beta} \subseteq\ \rightarrowtail \cdot \leftarrowtail$ is shown by induction on $(\{\alpha, \beta\}, m + n)$ with respect to $((>^{\mathrm{mul}}, =), >)_{\mathrm{lex}}$. Hence, $\leftarrowtail \cdot \rightarrowtail\ \subseteq {}_{\mathcal{A}}\!\leftarrow \cdot \to_{\mathcal{B}}^{*} \cdot {}_{\mathcal{B}}^{*}\!\leftarrow \cdot \to_{\mathcal{A}} \subseteq\ \rightarrowtail \cdot \leftarrowtail$. □

2.1 A criterion based on relative termination

We prove correctness of the first criterion. For a set U of parallel redex occurrences, the parallel step of $\mathcal{R}$ that contracts U is referred to as $\overset{U}{+\!\!\!\to}_{\mathcal{R}}$. We write $s \overset{n}{+\!\!\!\to}_{\mathcal{R}} t$ if $s \overset{U}{+\!\!\!\to}_{\mathcal{R}} t$ and $|U| \leqslant n$ for some U. For technical convenience we assume $\overset{n}{+\!\!\!\to}_{\mathcal{R}} = \varnothing$ when $n < 0$.

Lemma 2.2. *Suppose that a left-linear TRS $\mathcal{R}$ admits a critical-pair-closing system $\mathcal{C}$. If $t\ {}_{\mathcal{R}}\overset{m}{\leftarrow\!\!\!+}\ s \overset{n}{+\!\!\!\to}_{\mathcal{R}} u$ then $t +\!\!\!\to_{\mathcal{R}} \cdot {}_{\mathcal{R}}\!\leftarrow\!\!\!+\ u$, or there exist v and w such that $t\ {}_{\mathcal{R}}\overset{m-1}{\longleftarrow\!\!\!+}\ v \leftrightarrow_{\mathcal{C}}^{*} w \overset{n-1}{+\!\!\!\longrightarrow}_{\mathcal{R}} u$ and $v\ {}_{\mathcal{R}}\!\leftarrow s \to_{\mathcal{R}} w$.*

Lemma 2.3. *Suppose that $\mathcal{C}$ is a confluent critical-pair-closing system for a left-linear $\mathcal{R}$. If $t\ {}_{\mathcal{C}}\!\leftarrow s \overset{n}{+\!\!\!\to}_{\mathcal{R}} u$ then one of the following conditions holds.*

(a) $t \overset{n}{+\!\!\!\to}_{\mathcal{R}} \cdot {}_{\mathcal{C}}^{*}\!\leftarrow u$,

(b) $t \leftrightarrow_{\mathcal{C}}^{*} v +\!\!\!\to_{\mathcal{R}} \cdot {}_{\mathcal{C}}^{*}\!\leftarrow u$ *and* $s \to_{\mathcal{C}_{\mathsf{d}}/\mathcal{R}} v$ *hold for some* v,

(c) $t \leftrightarrow_{\mathcal{C}}^{*} v \overset{n-1}{+\!\!\!\longrightarrow}_{\mathcal{R}} \cdot {}_{\mathcal{C}}^{*}\!\leftarrow u$ *and* $s \to_{\mathcal{R}}^{*} v$ *hold for some* v.

Proof. If $n = 0$ then (a) holds trivially. Otherwise, one of the three diagrams in Figure 2.1 holds. In any case one of conditions (a–c) holds. □

Theorem 2.4. *A left-linear TRS $\mathcal{R}$ is confluent if $\mathcal{C}_{\mathsf{d}}/\mathcal{R}$ is terminating for some confluent critical-pair-closing system $\mathcal{C}$ for $\mathcal{R}$.*

Proof. We define the labeled ARS $\mathcal{A}$ on terms as follows: $s \to_{\alpha} t$ if $s \overset{n}{+\!\!\!\to}_{\mathcal{R}} t$ and $\alpha = (s, n)$. Moreover, we define the ARS $\mathcal{B}$ as $\to_{\mathcal{B}} = \to_{\mathcal{C}}$. Lemmata 2.2 and 2.3 ensure the conditions of Lemma 2.1 for the well-founded order $((\to_{\mathcal{C}_{\mathsf{d}}/\mathcal{R}}^{+}, \to_{\mathcal{R}}^{*}), >)_{\mathrm{lex}}$ on labels. Therefore, confluence of $\mathcal{A}$ is obtained. Hence, $\mathcal{R}$ is confluent. □

We illustrate use of Theorem 2.4 with an example for which ACP v0.40 [1] and CSI v0.4.1 [7] fail to show confluence.

Example 2.5. We show confluence of the left-linear TRS $\mathcal{R}$

$$\mathsf{f}(\mathsf{h}(x,\mathsf{d}),y) \to \mathsf{f}(\mathsf{h}(y,\mathsf{d}),x) \qquad\qquad \mathsf{h}(\mathsf{c},x) \to \mathsf{h}(x,x)$$

by successive application of Theorem 2.4.

(i) The left-linear TRS $\mathcal{C}$

$$1\colon\ \mathsf{f}(\mathsf{h}(x,\mathsf{d}),\mathsf{c}) \to \mathsf{f}(\mathsf{h}(\mathsf{c},\mathsf{d}),x) \qquad\qquad 2\colon\ \mathsf{h}(\mathsf{c},\mathsf{d}) \to \mathsf{h}(\mathsf{d},\mathsf{d})$$

is critical-pair-closing for $\mathcal{R}$, and $\mathcal{C}_{\mathsf{d}}/\mathcal{R}$ is terminating as $\mathcal{C}_{\mathsf{d}} = \varnothing$. Thus, it is sufficient to show that $\mathcal{C}$ is confluent.

(ii) Let $\mathcal{C}' = \{2\}$. The TRS $\mathcal{C}'$ is critical-pair-closing for $\mathcal{C}$. Because $\mathcal{C}'_{\mathsf{d}} = \varnothing$, termination of $\mathcal{C}'_{\mathsf{d}}/\mathcal{R}$ is trivial. It remains to show that $\mathcal{C}'$ is confluent.

(iii) Since $\mathcal{C}'$ admits no critical pair, the empty TRS $\varnothing$ is a confluent critical-pair-closing system for $\mathcal{C}'$. Termination of $\varnothing_{\mathsf{d}}/\mathcal{R}_1$ is trivial, and therefore $\mathcal{C}'$ is confluent.

Hence, $\mathcal{R}$ is confluent. Note that taking $\mathcal{C}$ from *instances* of rules in $\mathcal{R}$ is essential.

The following examples explain why none of confluence of $\mathcal{C}$, termination of $\mathcal{C}_{\mathsf{d}}/\mathcal{R}$, or the ground normal form condition of $\widehat{\mathcal{R}}$ can be removed from the conditions of Theorem 2.4.

Example 2.6. Consider the *non-confluent* left-linear TRS $\mathcal{R}$ taken from [4]:

$$\mathsf{b} \to \mathsf{a} \qquad\qquad \mathsf{b} \to \mathsf{c} \qquad\qquad \mathsf{c} \to \mathsf{b} \qquad\qquad \mathsf{c} \to \mathsf{d}$$

Let $\mathcal{C} = \mathcal{R}$. While $\mathcal{C}$ is not confluent, $\mathcal{C}$ is critical-pair-closing and termination of $\mathcal{C}_{\mathsf{d}}/\mathcal{R}$ follows from $\mathcal{C}_{\mathsf{d}} = \varnothing$.

Example 2.7. Consider the *non-confluent* left-linear TRS $\mathcal{R}$ from [3]:

$$1\colon\ \mathsf{f}(\mathsf{a},\mathsf{a}) \to \mathsf{b} \qquad 2\colon\ \mathsf{f}(x,\mathsf{b}) \to \mathsf{f}(x,x) \qquad 3\colon\ \mathsf{f}(\mathsf{b},x) \to \mathsf{f}(x,x) \qquad 4\colon\ \mathsf{a} \to \mathsf{b}$$

Let $\mathcal{C} = \{1,2,3\}$. Then, $\mathcal{C}$ is confluent and critical-pair-closing for $\mathcal{R}$, while $\mathcal{C}_{\mathsf{d}}/\mathcal{R}$ is not terminating. Let $\mathcal{C}' = \{1,2',3'\}$, where $2'\colon \mathsf{f}(\mathsf{a},\mathsf{b}) \to \mathsf{f}(\mathsf{a},\mathsf{a})$ and $3'\colon \mathsf{f}(\mathsf{b},\mathsf{a}) \to \mathsf{f}(\mathsf{a},\mathsf{a})$. Confluence of $\mathcal{C}'$, termination of $\mathcal{C}'_{\mathsf{d}}/\mathcal{R}$, and ${}_{\mathcal{R}}{\leftarrow}\rtimes{\rightarrow}_{\mathcal{R}} \subseteq {\downarrow_{\mathcal{C}'}}$ hold, while $\mathcal{C}' \subseteq \widehat{\mathcal{R}}$ does not hold.

2.2 A criterion based on termination

Next, we show the second criterion.

Lemma 2.8. *Every terminating critical-pair-closing system $\mathcal{C}$ for a TRS $\mathcal{R}$ is confluent.*

Proof. By the definition of $\widehat{\mathcal{R}}$ every critical pair of $\mathcal{C}$ is an instance of some critical pair of $\mathcal{R}$. Therefore, ${}_{\mathcal{C}}{\leftarrow}\rtimes{\rightarrow}_{\mathcal{C}} \subseteq {\downarrow_{\mathcal{C}}}$. Hence, Knuth and Bendix' criterion entails confluence. □

In order to analyze local peaks we recall Huet's measure which was used for proving correctness of the Parallel Closedness Theorem [4].

Definition 2.9. Let $\gamma : t \;{}_{\mathcal{R}}\overset{U}{\nleftarrow\!\!\!\!\leftarrow}\; s \;\overset{V}{\nrightarrow\!\!\!\!\rightarrow}_{\mathcal{R}}\; u$. The *weight* $w(\gamma)$ of the local peak γ is given by

$$w(\gamma) = \sum_{p \in W} |s|_p|$$

where, $W = \{p \in U \mid q \leqslant p \text{ for some } q \in V\} \cup \{p \in V \mid q < p \text{ for some } q \in U\}$. Note that if $W = \varnothing$ then $w(\gamma) = 0$.

$$\begin{tikzcd}
s \arrow[r, "p", "\mathcal{C}"'] \arrow[d, "\epsilon"', "\mathcal{R}"] & \cdot \arrow[r, "U-\{p\}", "\mathcal{C}"', "\shortparallel" marking] \arrow[d, dotted, two heads, "\mathcal{C}"'] \arrow[dr, phantom, "\text{Lem. 2.8}"] & t \arrow[d, dotted, two heads, "\mathcal{C}"] \\
u \arrow[r, dotted, two heads, "\mathcal{C}"'] & \cdot \arrow[r, dotted, two heads, "\mathcal{C}"'] & \cdot
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
s \arrow[r, "p", "\mathcal{R}"'] \arrow[d, "\epsilon"', "\mathcal{C}"] & \cdot \arrow[rr, "V-\{p\}", "\mathcal{R}"', "\shortparallel" marking] \arrow[d, dotted, "="', "\mathcal{C}"] \arrow[drr, phantom, "\text{I.H.}"] & & u \arrow[d, dotted, two heads, "\mathcal{C}"] \\
t \arrow[r, dotted, two heads, "\mathcal{C}"'] & \cdot \arrow[r, dotted, two heads, "\mathcal{C}"'] & \cdot \arrow[r, dotted, "\mathcal{R}"', "\shortparallel" marking] & \cdot
\end{tikzcd}$$

(b)

Figure 2: The proof of Lemma 2.10.

Lemma 2.10. *Let $\mathcal{C}$ be a terminating critical-pair-closing system for a left-linear TRS $\mathcal{R}$, and suppose ${}_{\mathcal{R}}\xleftarrow{>\epsilon} \rtimes \longrightarrow_{\mathcal{R}} \subseteq \mathrel{+\!\!\!\!\twoheadrightarrow}_{\mathcal{C}} \cdot {}_{\mathcal{C}}^{*}\!\leftarrow$. The following inclusions hold.*

(a) ${}_{\mathcal{C}}\!\leftarrow \cdot \mathrel{+\!\!\!\!\twoheadrightarrow}_{\mathcal{R}} \subseteq \to_{\mathcal{C}}^{*} \cdot \mathrel{+\!\!\!\!\twoheadrightarrow}_{\mathcal{R}} \cdot {}_{\mathcal{C}}^{*}\!\leftarrow$

(b) ${}_{\mathcal{R}}\!\mathrel{\twoheadleftarrow\!\!\!\!+} \cdot \mathrel{+\!\!\!\!\twoheadrightarrow}_{\mathcal{R}} \subseteq \to_{\mathcal{C}}^{*} \cdot \mathrel{+\!\!\!\!\twoheadrightarrow}_{\mathcal{R}} \cdot {}_{\mathcal{R}}\!\mathrel{\twoheadleftarrow\!\!\!\!+} \cdot {}_{\mathcal{C}}^{*}\!\leftarrow$

Proof. We only prove (a) since (b) can be proved in a similar way. We show the stronger claim that $\gamma : t \ {}_{\mathcal{C}}\!\overset{U}{\mathrel{\twoheadleftarrow\!\!\!\!+}} \ s \ \overset{V}{\mathrel{+\!\!\!\!\twoheadrightarrow}}_{\mathcal{R}} \ u$ implies $t \to_{\mathcal{C}}^{*} \cdot \mathrel{+\!\!\!\!\twoheadrightarrow}_{\mathcal{R}} \cdot {}_{\mathcal{C}}^{*}\!\leftarrow u$, by induction on $(w(\gamma), s)$ with respect to $((>,=),\rhd)_{\text{lex}}$. Here $\rhd$ stands for the proper superterm relation. We distinguish four cases, depending on U and V:

- If U and V admit no critical overlap, the Parallel Moves Lemma entails $t \mathrel{+\!\!\!\!\twoheadrightarrow}_{\mathcal{R}} \cdot {}_{\mathcal{C}}\!\mathrel{\twoheadleftarrow\!\!\!\!+} u$.
- If neither U nor V contains ϵ, the induction hypotheses for immediate subterms of s apply.
- If some $p \in U$ and $\epsilon \in V$ form a critical overlap, (a) in Figure 2.2 holds.
- If $\epsilon \in U$ and some $p \in V$ with $p > \epsilon$ form a critical overlap, (b) in Figure 2.2 holds. □

Theorem 2.11. *A left-linear TRS $\mathcal{R}$ is confluent if ${}_{\mathcal{R}}\xleftarrow{>\epsilon} \rtimes \longrightarrow_{\mathcal{R}} \subseteq \mathrel{+\!\!\!\!\twoheadrightarrow}_{\mathcal{C}} \cdot {}_{\mathcal{C}}^{*}\!\leftarrow$ holds for some terminating critical-pair-closing system $\mathcal{C}$ of $\mathcal{R}$.*

Proof. Define the labeled ARS $\mathcal{A}$ as follows: $s \to_{\alpha} t$ if $s \mathrel{+\!\!\!\!\twoheadrightarrow}_{\mathcal{R}} t$ and $\alpha = s$. The claim follows from Lemmata 2.1, 2.8, and 2.10 by taking $\to_{\mathcal{B}} = \to_{\mathcal{C}}$ and $> = \to_{\mathcal{C}}^{+}$. □

Example 2.12. Consider the left-linear TRS $\mathcal{R}$ from [6]:

1:	$x - 0 \to x$	7:	$\mathsf{gcd}(x,0) \to x$	13:	$\mathsf{if}(\mathsf{false},x,y) \to x$
2:	$0 - x \to 0$	8:	$\mathsf{gcd}(0,x) \to x$	14:	$\mathsf{if}(\mathsf{true},x,y) \to x$
3:	$\mathsf{s}(x) - \mathsf{s}(y) \to x - y$	9:	$\mathsf{gcd}(x,y) \to \mathsf{gcd}(y,\mathsf{mod}(x,y))$		
4:	$x < 0 \to \mathsf{false}$	10:	$\mathsf{mod}(x,0) \to x$		
5:	$0 < \mathsf{s}(y) \to \mathsf{true}$	11:	$\mathsf{mod}(0,y) \to 0$		
6:	$\mathsf{s}(x) < \mathsf{s}(y) \to x < y$	12:	$\mathsf{mod}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{if}(x < y, \mathsf{s}(x), \mathsf{mod}(x-y,\mathsf{s}(y)))$		

Let $\mathcal{C} = \{7, 8, 10, 11\}$. The system $\mathcal{C}$ is terminating and critical-pair-closing for $\mathcal{R}$. Since $\mathcal{R}$ admits no inside critical pairs, ${}_{\mathcal{R}}\xleftarrow{>\epsilon} \rtimes \longrightarrow_{\mathcal{R}} \subseteq \mathrel{+\!\!\!\!\twoheadrightarrow}_{\mathcal{C}} \cdot {}_{\mathcal{C}}^{*}\!\leftarrow$ holds. Hence, $\mathcal{R}$ is confluent.

The termination assumption of $\mathcal{C}$ cannot be dropped from Theorem 2.11, as seen below.

Example 2.13. Recall the TRS $\mathcal{R}$ of Example 2.6. There are no inside critical pairs. Let $\mathcal{C} = \mathcal{R}$. Then $\mathcal{C}$ is a critical-pair-closing system of $\mathcal{R}$. But $\mathcal{R}$ is not confluent.

Table 1: Experiments on 192 left-linear TRSs

	Th.2.4	Th.2.11	[3, Th.3]
confluence proved	**44**	**37**	**29**
timeout (30 sec)	28	43	0

3 Concluding Remarks

We have introduced the notion of critical-pair-closing subsystems for left-linear TRSs and shown two new confluence criteria for left-linear TRSs. The first criterion (Theorem 2.4) is a generalization of weak orthogonality based on relative termination, and the second criterion (Theorem 2.11) is related to the long-standing open problem stated in the introduction.

Table 1 summaries experimental results of the two criteria on left-linear TRSs in Cops Nos. 1–390[1]. As in Example 2.5, Theorem 2.4 was tested as a stand-alone criterion. For the comparison sake we also tested [3, Theorem 3], which is another generalization of weak orthogonality based on relative termination. A suitable critical-pair-closing system is searched from subsets of an input TRS $\mathcal{R}$ by enumeration. We have not supported use of $\widehat{\mathcal{R}}$ yet. Termination and relative termination were checked by using TT2 v1.16 [5], and joinability of critical pairs for Theorems 2.4 and [3, Theorem 3] were checked by $\to^k \cdot {}^m\!\leftarrow$ for $k, m \leqslant 4$. The tests were single-threaded run on a system equipped with an Intel Core i7-4500U with 1.8 GHz using a timeout of 30 seconds. Although the current implementation is still naive and inefficient, the numbers of proofs in the table clearly show effectiveness of our criteria.

All the three criteria are incomparable. For instance, Cops Nos. 22, 19, and 14 can be handled only by Theorem 2.4, Theorem 2.11, and [3, Theorem 3], respectively. It is worthwhile to investigate whether critical-pair-closing systems and *critical pair systems* [3] can be combined.

Acknowledgements. We thank the anonymous reviewers for their valuable comments.

References

[1] T. Aoto, J. Yoshida, and Y. Toyama. Proving confluence of term rewriting systems automatically. In *Proc. 21st RTA*, volume 5595 of *LNCS*, pages 93–102, 2009.

[2] N. Dershowitz. Open. Closed. Open. In J. Giesl, editor, *Proc. 16th RTA*, volume 3467 of *LNCS*, pages 276–393, 2005.

[3] N. Hirokawa and A. Middeldorp. Decreasing diagrams and relative termination. *Journal of Automated Reasoning*, 47(4):481–501, 2011.

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

[5] M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean Termination Tool 2. In *Proc. 20th RTA*, volume 5595 of *LNCS*, pages 295–304, 2009.

[6] M. Oyamaguchi and Y. Ohta. On the Church-Rosser property of left-linear term rewriting systems. *IEICE Transactions on Information and Systems*, E86-D(1):131–135, 2003.

[7] H. Zankl, B. Felgenhauer, and A. Middeldorp. CSI - a confluence tool. In *Proc. 23rd CADE*, volume 6803 of *LNAI*, pages 499–505, 2011.

[1]`http://coco.nue.riec.tohoku.ac.jp/`

Non-E-overlapping and weakly shallow TRSs are confluent (Extended abstract)

Masahiko Sakai[1], Michio Oyamaguchi[1] and Mizuhito Ogawa[2]

[1] Graduate School of Information Science, Nagoya University,
sakai@is.nagoya-u.ac.jp and oyamaguchi@za.ztv.ne.jp
[2] Japan Advanced Institute of Science and Technology,
mizuhito@jaist.ac.jp

1 Introduction

Confluence of term rewriting systems (TRSs) is undecidable, even for flat TRSs [MOJ06] or length-two string rewrite systems [SW08]. Two decidable subclasses are known: right-linear and shallow TRSs by tree automata techniques [GT05] and terminating TRSs [KB70]. Most of sufficient conditions are for either terminating TRSs [KB70] (extended to TRSs with relative termination [HA11, KH12]) or left-linear non-overlapping TRSs (and their extensions) [Ros73, Hue80, Toy87, Oos95, Oku98, OO97]. For non-linear TRSs, a goal is RTA open problem **58** "*strongly non-overlapping and right-linear TRSs are confluent*". A best known result strengthens the *right-linear* assumption to *simple-right-linear* [TO95, OOT95], which means that each rewrite rule is right-linear and no left-non-linear variables appear in the right hand side. Other trials by depth-preserving conditions are found in [GOO98].

We have proposed a different methodology, called a *reduction graph* [SO10]. It has shown that "*weakly non-overlapping, shallow, and non-collapsing TRSs are confluent*". An original idea comes from observation that, when non-E-overlapping, peak-elimination uses only "*copies*" of reductions in an original rewrite sequences. Thus, if we focus on terms appearing in peak elimination, they are finitely many. We regard a rewrite relation over these terms as a directed graph, and we construct a confluent directed acyclic graph (DAG) in a bottom-up manner, in which the shallow assumption works. The keys are, a connected convergent DAG always has a unique normal form (if it is finite), and convergence is preserved if we add an arbitrary reduction starting from that normal form.

This paper briefly sketches that "*non-E-overlapping and weakly-shallow TRSs are confluent*" by extending *reduction graph* in our previous work [SO10] by introducing *constructor expansion.* A term is weakly shallow if each defined function symbol appears either at the root or in the ground subterms, and a TRS is weakly shallow if the both sides of rules are weakly shallow. The non-E-overlapping property is undecidable for weakly shallow TRSs [MOM12] and a decidable sufficient condition is the strongly non-overlapping condition. A Turing machine can be simulated by a weakly shallow TRS (p.27 in [Klo93]); thus the word problem is undecidable, in contrast to shallow TRSs [CHJ94].

Basic definitions and notations

We follow standard definitions and terminology of graphs and TRSs [BN98]. As notational convention, V for a finite set (often of terms), F is a finite set of function symbols, D and C are the sets of *defined* and *constructor symbols* in F, respectively. X is the set of variables. We use s, t, u, v, w for terms, x, y for variables, p, q for positions, σ, θ for substitutions, $\ell \to r$ for a rewrite rule, and R for a TRS.

An *abstract reduction system* (ARS) is a directed graph $G = \langle V, \to\rangle$ with $\to \subseteq V \times V$. For $V', V'' \subseteq V$, $\to|_{V' \times V''} = \to \cap (V' \times V'')$. We write V_G and $\to_G$ to emphasize G. An edge $v \to u$ is an *out-edge* of v and an *in-edge* of u. A node v is $\to$-normal if it has no out-edges. Let $G = \langle V, \to\rangle$ and $G' = \langle V', \to'\rangle$. The union $G \cup G'$ is $\langle V \cup V', \to \cup \to'\rangle$. We say G is *finite* if V is finite, G is *convergent* if G is confluent and terminating, G' *includes* G (denoted by $G' \supseteq G$) if $V' \supseteq V$ and $\to' \supseteq \to$, and G' *weakly subsumes* G (denoted by $G' \sqsupseteq G$) if $V' \supseteq V$ and $\leftrightarrow'^* \supseteq \to$.

We use sub(t) for the set of *direct subterms* of a term t defined as sub$(t) = \emptyset$ if t is a variable and sub$(t) = \{t_1, \dots, t_n\}$ if $t = f(t_1, \dots, t_n)$. $s \xrightarrow[R]{p} t$ is a *top reduction* if $p = \varepsilon$. Otherwise, it is a *non-top* *reduction*, written as $s \overset{\varepsilon\leq}{\underset{R}{\to}} t$. We use $T|_f$ to denote the subset of $T \subseteq \mathrm{T}(F, X)$ and $f \in F$ that consists of the terms in T with the root symbol f. For $F' \subseteq F$, we use $T|_{F'}$ to denote $\cup_{f \in F'} T|_f$.

A *weakly shallow term* is a term in which defined function symbols appear only either at the root or in the ground subterms (i.e., $p \neq \varepsilon$ and $\mathrm{root}(s|_p) \in D$ imply that $s|_p$ is ground). A rewrite rule $\ell \to r$ is *weakly shallow* if ℓ and r are weakly shallow. A TRS is *weakly shallow* if each rewrite rule is weakly shallow. We assume that a TRS has finitely many rewrite rules.

Let $\ell_1 \to r_1, \ell_2 \to r_2 \in R$. If there exist substitutions θ_1, θ_2 for $p \in \mathrm{Pos}_X(\ell_1)$ such that $\ell_1|_p\theta_1 = \ell_2\theta_2$ (resp. $\ell_1|_p\theta_1 \overset{\varepsilon\leq}{\underset{R}{\leftrightarrow}}{}^* \ell_2\theta_2$), $(r_1\theta_1, (\ell_1\theta_1)[r_2\theta_2]_p)$ is a *critical pair* (resp. *E-critical pair*) except that $p = \varepsilon$ and the two rules are identical (up to renaming variables). A TRS R is *overlapping* (resp. *E-overlapping*, strongly overlapping) if there exists a critical pair (resp. E-critical pair, critical pair of linearizatoin of R). Note that when a TRS is left-linear, they are equivalent.

2 Extensions of convergent abstract reduction systems

Definition 2.1. For ARSs $G_1 = \langle V_1, \to_1\rangle$ and $G_2 = \langle V_2, \to_2\rangle$, we say that $G_1 \cup G_2$ is the *hierarchical combination of G_2 with G_1*, denoted by $G_1 \gtrdot G_2$, if $\to_1 \subseteq (V_1 \setminus V_2) \times V_1$.

Lemma 2.2. *Let $G_1 \gtrdot G_2$ be a convergent hierarchical combination of ARSs. If a convergent ARS G_3 weakly subsumes G_2 and $G_1 \gtrdot G_3$ is a hierarchical combination, then $G_1 \gtrdot G_3$ is convergent.*

Definition 2.3. Let $G = \langle V, \to\rangle$ be a convergent ARS and $v \neq v'$. Let G' be obtained by:

$$\begin{cases} \langle V \cup \{v'\}, \to \cup \{(v, v')\}\rangle & \text{if } v \in V \text{ is } \to\text{-normal, and } v' \notin V \\ \langle V, \to \cup \{(v, v')\}\rangle & \text{if } v \in V \text{ is } \to\text{-normal, } v' \in V \text{ and } v' \not\leftrightarrow^* v \\ \langle V, \to \setminus \{(v', v'') \mid v' \to v''\} \cup \{(v, v')\}\rangle & \text{if } v \in V \text{ is } \to\text{-normal, } v' \in V\text{, and } v' \leftrightarrow^* v \\ \langle V \cup \{v, v'\}, \to \cup \{(v, v')\}\rangle & \text{if } v \notin V \\ \text{Undefined} & \text{otherwise} \end{cases}$$

We denote G' by $\langle V, \to\rangle \multimap (v \to v')$ if G' is defined (i.e., the first four cases). We denote $G \multimap (v_0 \to v_1) \multimap (v_1 \to v_2) \multimap \cdots \multimap (v_{n-1} \to v_n)$ as $G \multimap (v_0 \to v_1 \to \cdots \to v_n)$.

Proposition 2.4. *Let $G = \langle V, \to\rangle$ be a convergent ARS. Let $v_0, v_1, \ldots, v_n$ satisfy $v_i \neq v_j$ (for $i \neq j$), and the following conditions:*

i) if $v_0 \in V$, then v_0 is $\to$-normal and $v_i \in V$ implies $v_i \leftrightarrow^* v_0$ for each $i (< n)$,

ii) if $v_0 \notin V$, then $v_1, \cdots, v_{n-1} \notin V$.

Then, $G' = G \multimap (v_0 \to v_1 \to \cdots \to v_n)$ is convergent, and satisfies $G' \sqsupseteq G$.

3 Reduction graphs

Definition 3.1 ([SO10])**.** A finite ARS $G = \langle V, \to\rangle$ is an *R-reduction graph* if $V \subseteq \mathrm{T}(F, X)$ and $\to \subseteq \underset{R}{\to}$.

For an R-reduction graph $G = \langle V, \to\rangle$, *top-edges*, *inner-edges*, and *strict inner-edges* are given as $\overset{\varepsilon}{\to} = \to \cap \overset{\varepsilon}{\underset{R}{\to}}$, $\overset{\varepsilon\leq}{\to} = \to \cap \overset{\varepsilon\leq}{\underset{R}{\to}}$, and $\overset{\neq\varepsilon}{\to} = \to \setminus \overset{\varepsilon}{\underset{R}{\to}}$, respectively. We use G^{ϵ}, $G^{\varepsilon<}$, and $G^{\neq\varepsilon}$ to denote $\langle V, \overset{\varepsilon}{\to}\rangle$ $\langle V, \overset{\varepsilon\leq}{\to}\rangle$, and $\langle V, \overset{\neq\varepsilon}{\to}\rangle$, respectively. Remark that an edge $(s, t) \in \to$ may be both $\overset{\varepsilon}{\to}$ and $\overset{\varepsilon\leq}{\to}$, e.g., $(f(a, a), f(b, a))$ for $R = \{a \to b, f(x, x) \to f(b, a)\}$. For an R-reduction graph $G = \langle V, \to\rangle$ and $F' \subseteq F$, we represent $G|_{F'} = \langle V, \to|_{F'}\rangle$ where $\to|_{F'} = \to|_{V|_{F'} \times V}$.

Definition 3.2. Let $G = \langle V, \to \rangle$ be an R-reduction graph. The *direct-subterm reduction-graph* $\mathrm{sub}(G)$ of G is $\langle \mathrm{sub}(V), \mathrm{sub}(\to) \rangle$ where $\langle \mathrm{sub}(V), \mathrm{sub}(\to) \rangle = \langle \bigcup_{t \in V} \mathrm{sub}(t), \{(s_i, t_i) \mid f(s_1, \ldots, s_n) \overset{\varepsilon<}{\to} f(t_1, \ldots, t_n),\ s_i \neq t_i\} \rangle$. An R-reduction graph $G = \langle V, \to \rangle$ is *subterm-closed* if $sub(V) \subseteq V$ and $\mathrm{sub}(\overset{\neq\varepsilon}{\to}) \subseteq \leftrightarrow^*$.

Lemma 3.3. *Let $G = \langle V, \to \rangle$ be a subterm-closed R-reduction graph. Assume that $p \in \mathrm{Pos}(s)$ for a term s and $s[t]_p \leftrightarrow^* s[t']_p$, in which any reductions do not occur above p. Then $t \leftrightarrow^* t'$.*

Definition 3.4. Let $G = \langle V, \to \rangle$ be an R-reduction graph and F' $(\subseteq F)$. The *F'-monotonic extension* is

$$M_{F'}(G) = \langle V_1, \to_1 \rangle \quad \text{for} \quad \left\{ \begin{array}{lcl} V_1 & = & \{f(s_1, \ldots, s_n) \mid f \in F',\ s_1, \ldots, s_n \in V\}, \\ \to_1 & = & \{(f(\cdots s \cdots), f(\cdots t \cdots)) \in V_1 \times V_1 \mid s \to t\}. \end{array} \right.$$

When G is subterm-closed, an *C-expansion* $\overline{M_C}(G)$ is the hierarchical combination $G|_D \gg M_C(G)$ $(= G|_D \cup M_C(G))$. The k-times application of $\overline{M_C}$ to G is denoted by $\overline{M_C}^k(G)$.

Example 3.5. As a running example, we use a TRS $R_2 = \{f(x, g(x)) \to g^3(x),\ c \to g(c)\ \}$ with $C = \{g\}$ and $D = \{c, f\}$. Consider a subterm-closed R_2-reduction graph $G = \langle \{c, g(c), g^2(c)\}, \{(c, g(c))\} \rangle$. For easy description, we also denote as $G = \{c \to g(c),\ \ g^2(c)\}$. Then, $M_C(G) = \{g(c) \to g^2(c),\ \ g^3(c)\}$, $\overline{M_C}(G) = \{c \to g(c) \to g^2(c),\ \ g^3(c)\}$, $\overline{M_C}^3(G) = \{c \to g(c) \to g^2(c) \to g^3(c) \to g^4(c),\ \ g^5(c)\}$.

Lemma 3.6. *For a subterm-closed R-reduction graph G and $m > k \geq 0$, (1) $G \sqsubseteq \overline{M_C}^k(G)$, (2) $\overline{M_C}^k(G)$ is subterm-closed, (3) $\overline{M_C}^k(G)$ is convergent, if G is convergent, and (4) $\overline{M_C}^k(G) \sqsubseteq \overline{M_C}^m(G)$.*

4 Constructor expansion

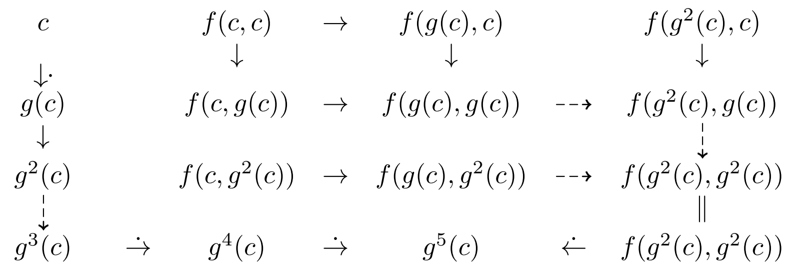
In Section 4 and 5, given an R-reduction graph G_0, we show how to inductively construct a convergent and subterm-closed R-reduction graph G_4 with $G_0 \sqsubseteq G_4$. Note that Section 5 assumes that a TRS R is non-E-overlapping and weakly shallow. Throughout these sections, we fix the notations.

- Given an R-reduction graph $G_0 = \langle V_0, \to_0 \rangle$ as an input.
- $G = \langle V, \to \rangle$ is used to denote a convergent and subterm-closed R-reduction graph that weakly subsumes $\mathrm{sub}(G_0)$ (by induction hypothesis).
- $G_1 = \langle V_1, \to_1 \rangle$ denotes a convergent R-reduction graph with $M_F(G) \sqsubseteq G_1$ (by Lemma 4.1).
- $G_{2_i} = \langle V_{2_i}, \to_{2_i} \rangle$ denotes $M_F(\overline{M_C}^i(G))$ for $i \geq 0$.
- T denotes a subgraph of $(G_0^\epsilon \cup G^\epsilon) \setminus (G_0^{\epsilon<} \cup G^{\epsilon<})$ such that T modulo $\leftrightarrow_1^*$ is acyclic and preserves connectivity of $(G_0^\epsilon \cup G^\epsilon) \setminus (G_0^{\epsilon<} \cup G^{\epsilon<})$ modulo $\leftrightarrow_1^*$.
- We repeatedly expand G_1 (by Lemma 5.1 and 5.2) by adding edges of T from nodes with out-edges only to sink order, and construct a convergent and subterm-closed G_4 with $G_0 \sqsubseteq G_4$.

If $G_i = \langle V_i, \to_i \rangle$ is convergent, we refer the normal form (in G_i) of a term $u (\in V_i)$ by $u{\downarrow_i}$.

Lemma 4.1. *For a convergent and subterm-closed R-reduction graph G, there exist k (≥ 0) and an R-reduction graph G_1 satisfying the following conditions: i) G_1 is convergent, and consists of non-top edges, ii) $G_1 \sqsubseteq G_{2_k}$, iii) $u \leftrightarrow_{2_i}^* v$ implies $u \leftrightarrow_1^* v$ for each $u, v \in V_1$ and i (≥ 0), and iv) $M_F(G) \sqsubseteq G_1$.*

Example 4.2. Consider R_2 in Example 3.5. Let $G_0 = \{f(g(c), c) \leftarrow f(c, c) \to f(c, g(c)) \overset{\varepsilon}{\to} g^3(c)\}$. The subterm graph $\mathrm{sub}(G_0)$ is equal to G in Example 3.5, and is convergent and subterm-closed. Then, Lemma 4.1 starts from $M_F(G)$, which is displayed by the solid arrows in Figure 1. An example of G_1 is constructed by augmenting the dashed edges with $k = 1$.



$$\begin{tikzcd}
c \arrow[d, dotted] & f(c,c) \arrow[r] \arrow[d] & f(g(c),c) \arrow[d] & f(g^2(c),c) \arrow[d] \\
g(c) \arrow[d] & f(c,g(c)) \arrow[r] & f(g(c),g(c)) \arrow[r, dashed] & f(g^2(c),g(c)) \arrow[d, dashed] \\
g^2(c) \arrow[d, dashed] & f(c,g^2(c)) \arrow[r] & f(g(c),g^2(c)) \arrow[r, dashed] & f(g^2(c),g^2(c)) \arrow[d, equal] \\
g^3(c) \arrow[r, dotted] & g^4(c) \arrow[r, dotted] & g^5(c) & f(g^2(c),g^2(c)) \arrow[l, dotted]
\end{tikzcd}$$

Figure 1: G_4 (dot arrows) and G_1 (dashed arrows) starting from G_0 (solid arrows) in Example 3.5

5 Merging top edges to the direct-subterm graph

Let $G_1 = \langle V_1, \to_1\rangle$ and $T_1 = \langle V_{T_1}, \to_{T_1}\rangle$ be R-reduction graphs with $V_{T_1} \subseteq V_1$. The *component graph*, denoted by T_1/G_1, of T_1 with G_1, is the graph $\langle V, \to\rangle$ having connected components of G_1 as nodes and $\to_{T_1/G_1}$ as edges such that $V = \{[v]_{\leftrightarrow_1^*} \mid v \in V_1\}$ and $\to_{T_1/G_1} = \{([u]_{\leftrightarrow_1^*}, [v]_{\leftrightarrow_1^*}) \mid (u,v) \in \to_{T_1}\}$. If clear from the context, we simply denote $[v]$ instead of $[v]_{\leftrightarrow_1^*}$.

Lemma 5.1. *Let G be a convergent subterm-closed R-reduction graph, $G_1 = \langle V_1, \to_1\rangle$, and k as in Lemma 4.1. Let $\to_S, \to_T \subseteq V_1 \times V_1$ such that $\to_S = \overset{\epsilon}{\to}_S$, $\to_T = \overset{\epsilon}{\to}_T$, $\overset{\varepsilon}{\to}_G \subseteq (\leftrightarrow_S \cup \leftrightarrow_T \cup \leftrightarrow_1)^{\varepsilon}$, and*

v) The component graph $(S \cup T)/G_1$ is acyclic, where out-edges are at most one for each node. Moreover, if $[u]_{\leftrightarrow_1^*}$ has an in-edge in T/G_1 then it has no edges in S/G_1.

vi) u is $\to_1$-normal for each $(u,v) \in S$.

If $T \neq \emptyset$, there is a tuple (S', T', G_1', k') such that $|T| > |T'|$ and the conditions i) to vi), (1) $\leftrightarrow_1^* \subseteq \leftrightarrow_{1'}^*$ and (2) $(\leftrightarrow_T \cup \leftrightarrow_S)^* \subseteq (\leftrightarrow_{T'} \cup \leftrightarrow_{S'} \cup \leftrightarrow_{1'})^*$ hold. We denote it by $(S,T,G_1,k) \vdash (S',T',G_1',k')$.

A convergent reduction graph $G_4 = \langle V_4, \to_4\rangle$ with $G_0 \sqsubseteq G_4$ is obtained from $S = \phi$, T (after $\vdash$ in Lemma 5.1 is preprocessed), and G_1 by repeated applications of $\vdash_l$, $\vdash_r$, and $\vdash_e$ below. For $(\ell\sigma, r\sigma) \in T$, there are $h \geq k$ and a substitution θ with $(\ell\sigma)\downarrow_1 = u_0(\overset{\varepsilon<}{\underset{R}{\to}} \cap \leftrightarrow_{2_h}^*)u_1(\overset{\varepsilon<}{\underset{R}{\to}} \cap \leftrightarrow_{2_h}^*)\cdots(\overset{\varepsilon<}{\underset{R}{\to}} \cap \leftrightarrow_{2_h}^*)u_n = \ell\theta$.

$$\text{Let} \quad \begin{cases} (S,T,G_1,k) \vdash_l (S,T,G_{1^l},h) & \text{by } G_{1^l} = G_1 \multimap (u_0 \to \cdots \to u_n). \\ (S,T,G_{1^l},h) \vdash_r (S,T,G_{1'},k') & \text{for } w \in V_1 \text{ such that } w \text{ is } \to_{1^l}\text{-normal, and } w \leftrightarrow_{2_{k'}}^* r\theta. \\ (S,T,G_{1'},k') \vdash_e (S',T',G_{1'},k') & \text{for } S' = S \cup \{(\ell\theta, r\theta)\} \text{ and } T' = T \setminus \{(\ell\sigma, r\sigma)\}. \end{cases}$$

Lemma 5.2. *Let $G_0 = \langle V_0, \to_0\rangle$ be an R-reduction graph. Then, there exists a convergent and subterm-closed R-reduction graph G_4 with $G_0 \sqsubseteq G_4$.*

Example 5.3. Let us consider to apply Lemma 5.2 on G_0 in Example 4.2. First, we take a convergent subterm-closed R_2-reduction graph that weakly subsumes $\mathrm{sub}(G_0)$. This graph is essentially the same as G in Example 3.5, containing some garbage. For simplicity, we use G in Example 3.5. As in Example 4.2, we obtain G_1 and $k = 1$. Let $T = (G_0^{\varepsilon} \cup G^{\varepsilon}) \setminus (G_0^{\varepsilon<} \cup G^{\varepsilon<})$, where G_0^{ε} and G^{ε} have the only edges $f(c, g(c)) \to g^3(c)$ and $c \to g(c)$, respectively.

The conversion $\vdash$ is applied twice, corresponding to two edges in T. The edge $c \to g(c)$ in T is simply moved to S. For the edge $f(c, g(c)) \to g^3(c)$ in T, $\vdash_l$ adds $f(g^2(c), g^2(c)) \to f(g^2(c), g^3(c))$ to G_1. $\vdash_r$ adds $g^3(c) \to g^4(c) \to g^5(c)$ to G_1 and increases k to 3. $\vdash_e$ adds $f(g^2(c), g^3(c)) \to g^5(c)$ to S. They are denoted by dotted arrows. Since $M_C(\overline{M_C}^3(G))$ is $\{g(c) \to g^3(c) \to \cdots \to g^4(c) \to g^5(c),\ \ g^6(c)\}$, $G_4 = (S \cup G_1|_D) \gg M_C(\overline{M_C}^2(G))$ is as in Figure 1, in which some garbage nodes are not presented.

Main Theorem *Non-E-overlapping and weakly-shallow TRSs are confluent.*

References

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

[CHJ94] H. Comon, M. Haberstrau, and J.-P.Jouannaud, *Syntacticness, cycle-syntacticness, and shallow theories*, Information and Computation, *111*, pp.154-191, 1994.

[GT05] G. Godoy and A. Tiwari. *Confluence of shallow right-linear rewrite systems.* CSL 2005, *LNCS 3634*, 541–556, 2005.

[GOO98] H. Gomi, M. Oyamaguchi, and Y. Ohta. *On the Church-Rosser property of root-E-overlapping and strongly depth-preserving term rewriting systems.* IPSJ, *39(4)*, 992–1005, 1998.

[Gra96] B. Gramlich. *Confluence without termination via parallel critical pairs.* CAAP'96, *LNCS 1059*, 211–225, 1996.

[HA11] N. Hirokawa and A. Middeldorp. *Decreasing Diagrams and Relative Termination.* J. Autom. Reasoning 47(4), 481-501, 2011.

[Hue80] G. Huet. *Confluent reductions: abstract properties and applications to term rewriting systems.* J. ACM, *27*, 797–821, 1980.

[KH12] D. Klein and N. Hirokawa. *Confluence of Non-Left-Linear TRSs via Relative Termination.* LPAR-18, *LNCS 7180*, 258-273, 2012.

[KB70] D. E. Knuth and P. B. Bendix. *Simple word problems in universal algebras.* Computational Problems in Abstract Algebra (Ed. J. Leech), 263–297, 1970.

[Klo93] J. W. Klop. *Term Rewriting Systems*, in *Handbook of Logic in Computer Science, Vol.2, Oxford University Press*, 1-116, 1993.

[MOJ06] I. Mitsuhashi, M. Oyamaguchi and F. Jacquemard. *The Confluence Problem for Flat TRSs.* AISC 2006, *LNCS 4120*, 68–81, 2006.

[MOM12] I. Mitsuhashi, M. Oyamaguchi and K. Matsuura. *On the E-overlapping Property of Weak Monadic TRSs.* IPSJ, *53(10)*, 2313–2327, 2012.

[OO89] M. Ogawa and S. Ono. *On the uniquely converging property of nonlinear term rewriting systems.* Tech. Rep. of IEICE, *COMP 89-7*, 61–70, 1989.

[OOT95] Y. Ohta, M. Oyamaguchi and Y. Toyama. *On the Church-Rosser Property of Simple-right-linear TRS's.* IEICE, *J78-D-I(3)*, 263–268, 1995 (in Japanese).

[Oku98] S. Okui. *Simultaneous Critical Pairs and Church-Rosser Property.* RTA'98, *LNCS 1379*, 2–16, 1998.

[Oos95] V. van Oostrom. *Development closed critical pairs.* HOA'95, *LNCS 1074*, 185–200, 1995.

[OO97] M. Oyamaguchi and Y. Ohta. *A new parallel closed condition for Church-Rosser of left-linear term rewriting systems.* RTA'97, *LNCS 1232*, 187–201, 1997.

[Ros73] B. K. Rosen. *Tree-manipulating systems and Church-Rosser theorems.* J. ACM, *20*, 160–187, 1973.

[SW08] M. Sakai and Y. Wang. *Undecidable Properties on Length-Two String Rewriting Systems.* ENTCS, *204*, 53–69, 2008.

[SO10] M. Sakai and M. Ogawa. *Weakly-non-overlapping non-collapsing shallow term rewriting systems are confluent.* Information Processing Letters, *110*, 810–814, 2010.

[Toy87] Y. Toyama. *Commutativity of term rewriting systems.* Programming of future generation computer *II*, 393–407, 1988.

[TO95] Y. Toyama and M. Oyamaguchi. *Church-Rosser property and unique normal form property of non-duplicating term rewriting systems.* CTRS'95, *LNCS 968*, 316–331, 1995.