IWC 2016

5th International Workshop on Confluence

Proceedings

Editors: Beniamino Accattoli & Ashish Tiwari

Sep 8-9, 2016, Obergurgl, Austria

Preface

This report contains the proceedings of the *5th International Workshop on Confluence (IWC 2016)*, which was held in Obergurgl, Austria on Sep 8-9, 2016. The workshop was part of the Computational Logic in the Alps event. Previous IWC workshops were held in Nagoya (2012), Eindhoven (2013), Vienna (2014), and Berlin (2015).

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 2016 received 12 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 *Florent Jacquemard* on *Some Results on Confluence: Decision and What to do Without*, and a second invited talk by Paul-Andre Mellies on *Five Basic Concepts of Axiomatic Rewriting Theory*. Their abstracts are also included in the report. Moreover, the 5th Confluence Competition (CoCo 2016) was held during the workshop and the results are available at http://coco.nue.riec.tohoku.ac.jp/2016/.

Several persons helped to make IWC 2016 a success. We are grateful to the members of the program committee for their work. We also thank the members of the Computational Logic in the Alps (CLA) organizing committee for hosting IWC 2016 in Obergurgl.

August 2016

Beniamino Accattoli
Ashish Tiwari

Organization

IWC 2016 was part of the Computational Logic in the Alps event (CLA 2016), which was organized by the Computational Logic group of the University of Innsbruck.

Program Committee Chairs

Beniamino Accattoli	INRIA, France
Ashish Tiwari	SRI International, USA

Program Committee

Beniamino Accattoli	INRIA, France
Bertram Felgenhauer	University of Innsbruck, Austria
Yves Guiraud	University of Paris 7, France
Nao Hirokawa	JAIST, Japan
Koji Nakazawa	Nagoya, Japan
Ashish Tiwari	SRI International, Menlo Park, USA

Local Organisation Committee

- Martina Ingenhaeff-Falkner
- René Thiemann
- Benjamin Winder

Table of Contents

Abstracts of Invited Talks

Contributed Papers

CoCo 2016 System Descriptions

Some results on confluence: decision and what to do without.

Florent Jacquemard[1*]

INRIA – Sorbonne Universités
STMS (IRCAM-CNRS-UPMC), Paris
florent.jacquemard@inria.fr

Abstract

We recall first some decidability results on the confluence of TRS, and related properties about unicity of normal forms. In particular we put it in perspective old proofs of undecidability of confluence for the class of flat systems with more recent results, in order to discuss the importance of linearity wrt these decision problems.

Second, we describe a case study on musical rhythm notation involving modeling rewrite systems which are not confluent. In this case, instead of applying rewrite rules directly, we enumerate the equivalence class of a given term using automata-based representations and dynamic programming.

1 Confluence (un)decidability

When term rewriting systems (TRS) are used as models in fields such as functional programming languages semantics, automated deduction or system or program verification, the application of rewrite rules can be highly non-deterministic. Confluence permits to relax from this problem by guaranteeing that divergent reduction will eventually converge to a canonical form, in case of termination. It is therefore an crucial property to decide for TRS.

Decidability of confluence for linear TRS. Confluence of TRS is undecidable in general, even for linear systems (every variable can occur at most once in every left- or right-hand-side of rules) [28]. It has been shown decidable for ground TRS (rewrite rules without variables) [18, 3] and for left-linear right-ground TRS [2]. Polynomial time decision procedures have been proposed years later for ground TRS [1, 22], for left-shallow-linear and right-ground TRS (every variable can occur at most once and at depth at most one in every left-hand-side of rule) [22], for linear-shallow TRS (every variable occurs at most once in each rule and at depth at most one) [22, 10], and for linear and shallow TRS (every variable occurs at most once and at depth at most one in each side rule but can occur twice in a rule) [7].

Uniqueness of Normal Forms. The decidability of several alternatives to confluence has been studied. A first alternative, *uniqueness of normal forms* ($\mathrm{UN}^=$), implied by confluence, expresses that no two distinct normal forms (irreducible terms) can be equivalent modulo the rewrite system considered. $\mathrm{UN}^=$ has been shown decidable for ground TRS [28], and for shallow TRS (without the restriction of linearity) [19]. It is also polynomial time decidable for shallow and linear TRS [24]. It is undecidable for right ground TRS [26], for linear, non-collapsing (the right-hand-side of rules cannot be a variable), variable-preserving, and depth-two TRS [25], for

[*] co-authors for the results mentioned in §1: Ichiro Mitsuhashi, Michio Oyamaguchi, Guillem Godoy, and in S2: Jean Bresson, Masahiko Sakai, Adrien Ycart, Adrien Maire, Pierre Donat Bouillud, Slawek Staworko.

left-linear and left-flat TRS with with depth-two right-hand sides of rules [19] as well as for right-ground, right-flat TRS [25].

A second alternative, *unique normalization* (UN) expresses that every term can reach at most one normal form using the TRS considered. $\mathrm{UN}^{=}$ implies UN but the converse is not true. UN is decidable in polynomial time for ground TRS [27], and also for for shallow and linear TRS [9]. On the negative side, UN is undecidable for right-ground TRS [23], for flat TRS (left- and right- hand side of rules have depth at most one) [8], for linear and right-flat TRS [11] and for flat and right-linear TRS [9].

Decidability of confluence for non-linear TRS The linearity is often considered as a yardstick when considering decision of properties of TRS such as confluence, reachability or joinability. For instance, tree automata based methods sometimes used in this context [18, 3, 2, 9] need, in case of non-linear TRS, generalized models with difficult decision problems.

Confluence is shown undecidable for flat (non-linear) TRS [14, 17] by reduction of reachability, also shown undecidable in this case (note that this is in contrast with $\mathrm{UN}^{=}$ [19]). The latter proofs have been simplified drastically in [8]. However, confluence has been shown decidable for some classes of TRS allowing non-linear rules, like right-ground TRS (without restriction on the left-hand-sides of rules) [16], and shallow and right-linear TRS [12].

The latter proof uses decidability of reachability and joinability, both implied by regularity preservation result. To our knowledge, it is an open question whether confluence is decidable for other classes of TRS preserving regularity such as right-linear and finite-path-overlapping TRS [21] (shallow right-linear TRS are a particular case) or Layer Transducing TRS [20]. It is also interesting to consider the decision of confluence for particular rewriting strategies *e.g.* bottom-up [5, 6]. Finally, it can be observed that collapsing (right-variable) rules are essential in shifted pairing like constructions for undecidability proofs [14, 17, 8]. It is also unknown whether confluence is decidable for shallow and non-collapsing TRS.

2 What to do when there is no confluence

Traditional music notation is since centuries the standard format for the communication, exchange, and preservation of musical works in Western musical practice. We have been working recently on modeling the notation of rhythm (durations), following an approach based on formal languages and term rewriting.

In common western music notation, durations values are expressed proportionally, by recursive subdivisions of a unit (*beat*). This hierarchical definition induces naturally tree-structured representations called *rhythm trees* (RT). Every position in a RT is associated to a duration value. In a simple variant (see Figure 1), the root position is associated a fixed duration value and every non-root position is associated the duration of its parent p_0 divided by the number of edges outgoing from p_0. Moreover, if a leaf position p labeled by $\circ$, the the duration of p is added to the duration of the next leaf p' in depth-first-traversal (if it exists). The other leafs may be labeled by symbols giving information on notes, rests *etc*, and the labels of inner positions are not significant (here we use named after their arity $2, 3, 4\ldots$). To a RT, we associate the sequence of durations of the non-o leaves (in *dfs*). To capture more complex rhythm notations, we use a dag representations not described here.

The RT representations are used in a new tool for the transcription of timestamped event sequences into a music notation [29]. It is implemented as a library of the algorithmic composition framework OpenMusic (Figure 2). We are also developing Music Information Retrieval

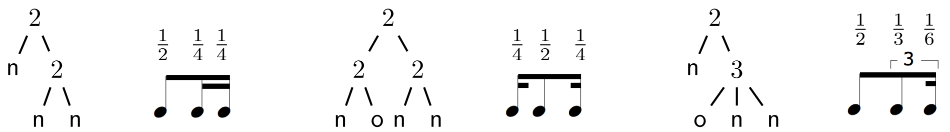


Figure 1: Rhythm Trees with associated duration sequences (symbol n represents a note).

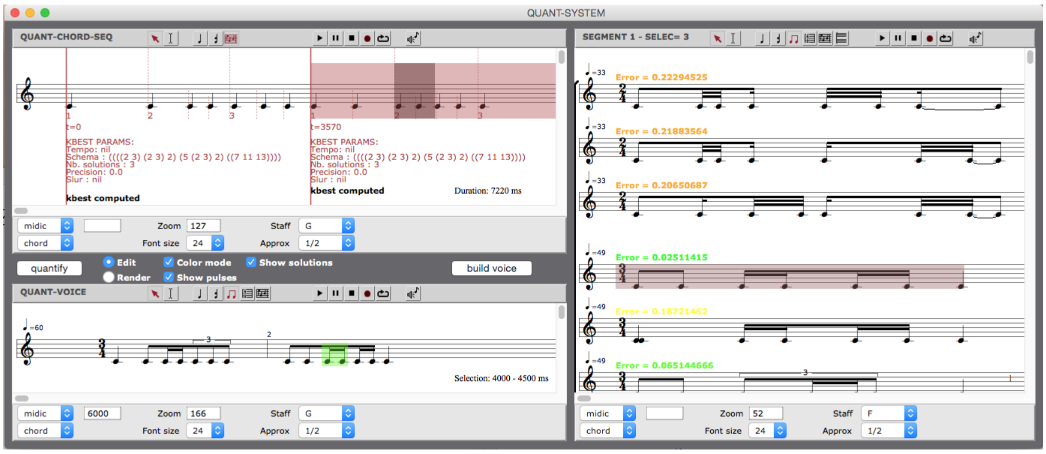


Figure 2: Transcription librairy for OpenMusic (Ircam). http://repmus.ircam.fr/cao/rq

tasks based on RT representations, in particular for querying bases of digital music scores (*e.g.* by *query by tapping*) and for musicologist research, using similarity measures and tree edit distances.

Structural theory of RT. For reasoning about rhythm notations in the above tasks, we define an equivalence between RT with term rewriting rules [15, 4]. For instance, the rules $2(\mathsf{o},\mathsf{n}) \to \mathsf{n}$, $3(\mathsf{o},\mathsf{o},\mathsf{n}) \to \mathsf{n}, \ldots$ and $2(\mathsf{o},\mathsf{o}) \to \mathsf{o}, \ldots$ comply with the semantics of o presented above, and rules of the form $3\big(2(x_1,x_2), 2(x_3,x_4), 2(x_5,x_6)\big) \to 2\big(3(x_1,x_2,x_3), 3(x_4,x_5,x_6)\big)$ can be used in order to simplify RT. The TRS containing these simple rules is not confluent. For instance, starting from $t = 3\big(2(\mathsf{o},\mathsf{o}), 2(\mathsf{n},\mathsf{o}), 2(\mathsf{o},\mathsf{n})\big)$, we have the following non-joinable critical peak:

$$3(\mathsf{o}, 2(\mathsf{n},\mathsf{o}), \mathsf{n}) \xleftarrow{*} t \to 2\big(3(\mathsf{o},\mathsf{o},\mathsf{n}), 3(\mathsf{o},\mathsf{o},\mathsf{n})\big) \xrightarrow{*} 2(\mathsf{n},\mathsf{n}).$$

Exploring sets of equivalent terms. Therefore, in order to to reason about sets of equivalent terms (in particular the set $[\![t]\!]$ of terms equivalent to a given RT t), instead of applying rewriting to reach a canonical normal form that does not exist, we use automata-based characterizations. Some techniques like *tree automata completion*, can be used to compute a tree automaton recognizing the rewrite closure of a given regular tree set (in particular recognizing $[\![t]\!]$ given $\{t\}$), by superposition of rewrite rules into tree automata transition rules. Such techniques have been used for verify safety properties of program or systems modeled as TRS (possibly not confluent) by reduction to the problem of emtiness of tree automata intersection (*regular tree model checking*).

With rewrite rules like the above ones, it is not easy to establish the termination of standard tree automata completion procedures. Even though in our case in practice we only need to consider terms of a bounded depth, hence finite set of terms, it is neither easy to reasonably bound the size of the automaton obtained this way. As an alternative, we have developed an ad hoc construction using the duration sequence associated to a given RT, and a tree automaton representing the family of RT that we want to consider. Once an automaton recognizing $[\![t]\!]$ is constructed, we use dynamic programming for the lazy enumeration of this set, according to a measure of tree complexity, following techniques of *k-best parsing* [13]. This way, we can enumerate efficiently the rhythms equivalent to a given rhythm, by increasing complexity.

References

[1] Hubert Comon, Guillem Godoy, and Robert Nieuwenhuis. The confluence of ground term rewrite systems is decidable in polynomial time. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science* (FOCS), pages 298–307. IEEE Computer Society, 2001.

[2] Max Dauchet, Thierry Heuillard, Pierre Lescanne, and Sophie Tison. Decidability of the confluence of finite ground term rewrite systems and of other related term rewrite systems. *Inf. Comput.*, 88(2):187–201, 1990.

[3] Max Dauchet and Sophie Tison. The theory of ground rewrite systems is decidable. In *Proceedings Fifth Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 242–248. IEEE Computer Society, 1990.

[4] Pierre Donat-Bouillud, Florent Jacquemard, and Masahiko Sakai. Towards an equational theory of rhythm notation. In *Music Encoding Conference*, 2015.

[5] Irène Durand, Géraud Sénizergues, and Marc Sylvestre. Termination of linear bounded term rewriting systems. In *Proceedings of the 21st Int. Conf. on Rewriting Techniques and Applications* (RTA), vol.6, pages 341–356 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 2010.

[6] Irène Durand and Marc Sylvestre. Left-linear Bounded TRSs are Inverse Recognizability Preserving. In *22nd RTA*, volume 10 of LIPIcs, pages 361–376, 2011.

[7] G. Godoy, A. Tiwari, and R. Verma. On the confluence of linear shallow term rewrite systems. In *20th Intl. Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2607 of *Lecture Notes in Computer Science*, pages 85–96. Springer, 2003.

[8] Guillem Godoy and Hugo Hernández. Undecidable properties for flat term rewrite systems. *Applicable Algebra in Engineering, Communication and Computing*, 20(2):187–205, 2009.

[9] Guillem Godoy and Florent Jacquemard. Unique normalization for shallow trs. In *20th International Conference on Rewriting Techniques and Applications (RTA)*, volume 5595 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2009.

[10] Guillem Godoy, Robert Nieuwenhuis, and Ashish Tiwari. Classes of term rewrite systems with polynomial confluence problems. *ACM Trans. Comput. Logic*, 5(2):321–331, 2004.

[11] Guillem Godoy and Sophie Tison. On the normalization and unique normalization properties of term rewrite systems. In *Proc. 21st International Conference on Automated Deduction (CADE)*, volume 4603 of *Lecture Notes in Computer Science*, pages 247–262. Springer, 2007.

[12] Guillem Godoy and Ashish Tiwari. Confluence of shallow right-linear rewrite systems. In *19th International Workshop of Computer Science Logic, CSL*, volume 3634 of *Lecture Notes in Computer Science*, pages 541–556. Springer, 2005.

[13] Liang Huang and David Chiang. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, pages 53–64, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[14] Florent Jacquemard. Reachability and confluence are undecidable for flat term rewriting systems. *Information Processing Letters*, 87(5):265–270, 2003.

[15] Florent Jacquemard, Pierre Donat-Bouillud, and Jean Bresson. A Structural Theory of Rhythm Notation based on Tree Representations and Term Rewriting. In *5th International Conference on Mathematics and Computation in Music (MCM)*, volume 9110 of *Lecture Notes in Artificial Intelligence*. Springer, 2015.

[16] Lukasz Kaiser. Confluence of right ground term rewriting systems is decidable. In *International Conference on Foundations of Software Science and Computation Structures (Fossacs)*, pages 470–489. Springer, 2005.

[17] I. Mitsuhashi, M. Oyamaguchi, and F. Jacquemard. The confluence problem for flat TRSs. In *Proc. 8th Intl. Conf. on Artificial Intelligence and Symbolic Computation (AISC'06)*, volume 4120 of *LNAI*, pages 68–81. Springer, 2006.

[18] M. Oyamaguchi. The Church-Rosser property for ground term rewriting systems is decidable. *Theoretical Computer Science*, 49:43–79, 1987.

[19] Nicholas Radcliffe and Rakesh M. Verma. Uniqueness of normal forms is decidable for shallow term rewrite systems. In *Conference on Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2010.

[20] H. Seki, T. Takai, Y. Fujinaka, and Y. Kaji. Layered Transducing Term Rewriting System and Its Recognizability Preserving Property. In *Int. Conf. on Rewriting Techniques and Applications (RTA)*, volume 2378 of *Lecture Notes in Computer Science*, pages 98–113. Springer, 2002.

[21] T. Takai, Y. Kaji, and H. Seki. Right-linear finite path overlapping term rewriting systems effectively preserve recognizability. In *11th Int. Conf. on Rewriting Techniques and Applications (RTA)*, volume 1833 of *Lecture Notes in Computer Science*, pages 246–260. Springer, 2000.

[22] A. Tiwari. Deciding confluence of certain term rewriting systems in polynomial time. In *IEEE Symposium on Logic in Computer Science (LICS)*, pages 447–456. IEEE Society, 2002.

[23] R. Verma. Complexity of normal form properties and reductions for rewriting problems. *Fundamenta Informaticae*, 2008.

[24] R. Verma and J. Zinn. A polynomial-time algorithm for uniqueness of normal forms of linear shallow term rewrite systems. In *Symposium on Logic in Computer Science LICS (short presentation)*, 2006.

[25] Rakesh Verma. New undecidability results for properties of term rewrite systems. *Electronic Notes in Theoretical Computer Science*, 290:69–85, 2008.

[26] Rakesh Verma. Complexity of normal form properties and reductions for term rewriting problems complexity of normal form properties and reductions for term rewriting problems. *Fundamenta Informaticae*, 92(1-2):145–168, 2009.

[27] Rakesh Verma and Ara Hayrapetyan. A new decidability technique for ground term rewriting systems with applications. *ACM Trans. Comput. Logic*, 6(1):102–123, January 2005.

[28] Rakesh M. Verma, Michael Rusinowitch, and Denis Lugiez. Algorithms and reductions for rewriting problems. *Fundam. Inf.*, 46:257–276, August 2001.

[29] Adrien Ycart, Florent Jacquemard, Jean Bresson, and Slawomir Staworko. A Supervised Approach for Rhythm Transcription Based on Tree Series Enumeration. In *Proceedings of the 42nd International Computer Music Conference (ICMC)*. ICMA, 2016.

Five Basic Concepts of Axiomatic Rewriting Theory

Paul-André Melliès

Institut de Recherche en Informatique Fondamentale (IRIF)
CNRS, Université Paris Diderot

Abstract

In this invited talk, I will review five basic concepts of Axiomatic Rewriting Theory, an axiomatic and diagrammatic theory of rewriting started 25 years ago in a LICS paper with Georges Gonthier and Jean-Jacques Lévy, and developed along the subsequent years into a fully fledged 2-dimensional theory of causality and residuation in rewriting. I will give a contemporary view on the theory, informed by my later work on categorical semantics and higher-dimensional algebra, and also indicate a number of current research directions in the field.

A good way to understand Axiomatic Rewriting Theory is to think of it as a 2-dimensional refinement of Abstract Rewriting Theory. Recall that an abstract rewriting system is defined as a set V of vertices (= terms) equipped with a binary relation $\to \subseteq V \times V$. This abstract formulation is convenient to formulate various notions of termination and of confluence, and to compare them, typically:

strong normalisation	vs.	weak normalisation
confluence	vs.	local confluence

Unfortunately, the theory is not sufficiently informative to capture more sophisticated structures and properties of rewriting systems related to causality and residuation, like

redexes and residuals
finite developments
standardisation
head rewriting paths

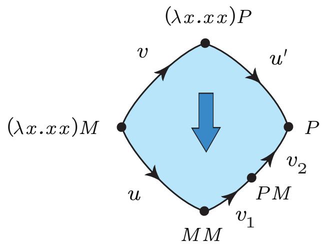
These structures and properties are ubiquitous in rewriting theory. They appear in conflict-free rewriting systems like the λ-calculus as well as in rewriting systems with critical pairs, like action calculi and bigraphs designed by Milner [9] as universal calculus integrating the λ-calculus, Petri nets and process calculi, or the $\lambda\sigma$-calculus introduced by Abadi, Cardelli, Curien and Lévy [1] to express in a single rewriting system the various evaluation strategies of an environment machine.

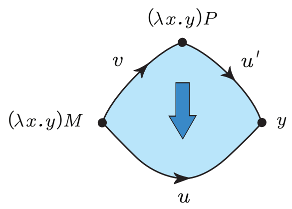
It thus makes sense to refine Abstract Rewriting Theory into a more sophisticated framework where the causal structures of computations could be studied for themselves, in a generic way. Intuitively, the causal structure of a rewriting path $f : M \twoheadrightarrow N$ is the cascade of elementary computations implemented by that path. In order to extract these elementary computations from the rewriting path f, one needs to trace operations (= redexes) inside it. This is achieved by permuting the order of execution of independent redexes executed by f. An axiomatic rewriting system is thus defined as a graph $G = (V, E, \partial_0, \partial_1)$ consisting of a set V of vertices (= the terms), a set E of edges (= the redexes) and a pair of source and target functions $\partial_0, \partial_1 : E \to V$ equipped moreover with a family of permutation tiles, satisfying a number of axiomatic properties.

1. Permutation tiles. The purpose of permutation tiles is to permute the order of execution of redexes. In our axiomatic setting, a permutation tile (f, g) is a pair of coinitial and cofinal rewriting paths of the form:

$$f = M \xrightarrow{v} P \xrightarrow{u'} N \qquad\qquad g = M \xrightarrow{u} Q \xrightarrow{h} \twoheadrightarrow N$$

where u, v, u' are redexes and h is a rewriting path. The intuition is that h computes the residuals of the redex v along the redex u. Two typical permutation tiles in the λ-calculus are the following one:





where $h = v_1 \cdot v_2$ on the left-hand side and $h = id$ on the right-hand side.

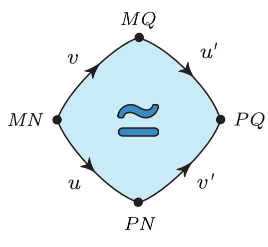
2. Standardisation cells. The permutation tiles are oriented, and generate a 2-dimensional rewriting system on the 1-dimensional rewriting paths. In order to distinguish this rewriting system from the original rewriting system, we call it the standardisation rewriting system. A standardisation path θ between 1-dimensional rewriting paths $f, g : M \twoheadrightarrow N$ is then written as

$$\theta \quad : \quad f \Rightarrow g \quad : \quad M \twoheadrightarrow N$$

The axioms of Axiomatic Rewriting Theory are designed to ensure that this 2-dimensional rewriting system is weakly normalising and confluent. In order to establish weak normalisation, one needs to clarify an important point: when should one consider that two standardisation paths

$$\theta, \theta' \quad : \quad f \Rightarrow g \quad : \quad M \twoheadrightarrow N$$

are equal? The question looks a bit esoteric, but it is in fact fundamental! By way of illustration, consider the following permutation tile in the λ-calculus:



(1)

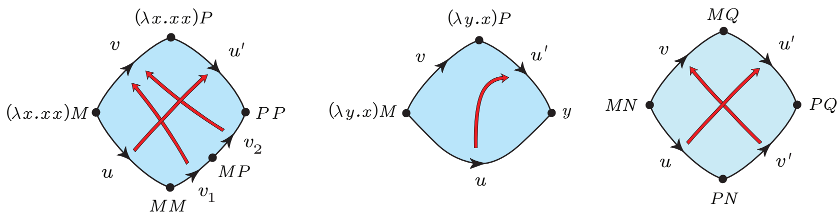
where the two β-redexes u and v should be considered as syntactically disjoint because u is a β-redex of the subterm M and v is a β-redex of the disjoint subterm N. If one does not want to give a left-to-right precedence to the β-redex u over the β-redex v, one should equip the axiomatic rewriting system with two permutation tiles

$$\theta_1 : v \cdot u' \Rightarrow u \cdot v' \qquad\qquad \theta_2 : u \cdot v' \Rightarrow v \cdot u'.$$

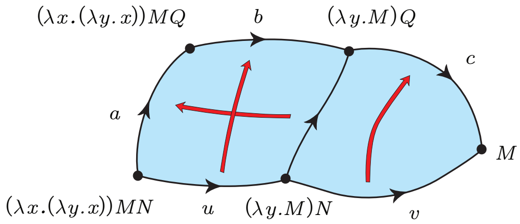
The task of the permutation tile θ_1 is to permute u before v, while the task of the permutation tile θ_2 is to permute v before u. It thus makes sense to require that their composite are equal to the identity in the standardisation rewriting system:

$$\theta_1; \theta_2 = id : v \cdot u' \Rightarrow v \cdot u' \qquad\qquad \theta_2; \theta_1 = id : u \cdot v' \Rightarrow u \cdot v'$$

Of course, this enforces that θ_1 and θ_2 are inverse. One declares in that case that the permutation tile (1) is reversible. A standardisation path $\theta : f \Rightarrow g$ consisting only of such reversible permutation tiles is called reversible, and one writes $\theta : f \simeq g$ in that case. A simple and elegant way to describe the equational theory on standardisation paths is to equip every permutation tile (f, g) with an ancestor function $\varphi : [n] \to [2]$ where $[k] = \{1, \ldots, k\}$ and n is the length of the path $g = u \cdot h$. The purpose of the function φ is to map the index of redex in $g = u \cdot h$ to the index of its ancestor $f = v \cdot u'$, in the following way:



By way of illustration, the permutation tiles equipped with their ancestor functions may be composed in the following way in the λ-calculus:



This leads us to identify two standardisation paths $\theta, \theta' : f \Rightarrow g$ when they produce the same ancestor function. A standardisation cell is then defined as an equivalence class of standardisation paths $\theta, \theta' : f \Rightarrow g$ modulo this equivalence relation. Note in particular that the equivalence relation identifies the standardisation path $\theta_1; \theta_2$ with the identity, and similarly for $\theta_2; \theta_1$.

In this way, one defines for every axiomatic rewriting system G a 2-category $\mathbf{Std}(G)$ of whose objects are the vertices (= terms) of G, whose morphisms are the paths (= rewriting paths) of G, and whose 2-cells are the standardisation cells. One declares that two rewriting paths $f, g : M \twoheadrightarrow N$ are equivalent modulo redex permutation (noted $f \sim g$) when f and g are in the same connected component of the hom-category $\mathbf{Std}(G)(M, N)$ of rewriting paths from M to N. This means that one can construct a zig-zag of standardisation paths between f and g. We also like to say that the rewriting paths f and g are homotopy equivalent when $f \sim g$.

3. Standard rewriting paths. A rewriting path $f : M \twoheadrightarrow N$ is called standard when every standardisation cell $\theta : f \Rightarrow g : M \twoheadrightarrow N$ is reversible. The standardisation theorem states that

Standardisation Theorem. For every rewriting path $f : M \twoheadrightarrow N$ there exists a standardisation cell $\theta : f \Rightarrow g$ to a standard rewriting path $g : M \twoheadrightarrow N$. Moreover, this standard rewriting path is unique in the sense that for every standardisation cell $\theta' : f \Rightarrow g'$ to a standard rewriting path $g' : M \twoheadrightarrow N$, there exists a reversible standardisation path $\theta'' : g' \simeq g$ such that $\theta = \theta'; \theta''$.

The theorem is established in any axiomatic rewriting system G using the elementary axioms on the permutation tiles provided by the theory. As a matter of fact, the property is even stronger: it states that there exists a unique standardisation cell θ from f to the standard rewriting path g. This means that every standard path $g : M \twoheadrightarrow N$ is a terminal object in its connected component of rewriting paths $f : M \twoheadrightarrow N$. See [3, 4, 8] for details.

4. External rewriting paths. An external rewriting path $e : M \twoheadrightarrow N$ is defined as a rewriting path such that for every standard rewriting path $f : N \twoheadrightarrow P$, the composite rewriting path $e \cdot f : M \twoheadrightarrow P$ is standard. Note in particular that every external rewriting path is standard. Accordingly, a rewriting path $m : M \twoheadrightarrow N$ is called internal when for every standardisation cell $\theta : m \Rightarrow e \cdot f$ where the rewriting path e is external, the rewriting path e is in fact the identity on M. One establishes the following property in every axiomatic rewriting system, see [6] for details:

Factorisation Theorem: For every rewriting path $f : M \twoheadrightarrow N$, there exists a unique external rewriting path $e : M \twoheadrightarrow P$ and a unique internal rewriting path $m : P \twoheadrightarrow N$ up to permutation equivalence such that $f \sim e \cdot m$. This factorization is moreover functorial.

5. Head-rewriting paths. The factorization theorem is supported by the intuition that only the external part $e : M \twoheadrightarrow P$ of a rewriting path $f : M \twoheadrightarrow N$ performs relevant computations, while the internal part $m : P \twoheadrightarrow N$ produces essentially useless extra computations. The factorization property plays a fundamental role in the theory. In particular, it enables us to establish a stability theorem which shows the existence of head-rewriting paths in every axiomatic rewriting system, even the rewriting system is non-deterministic and has critical pairs. The stability theorem states that under very general and natural assumptions on a set $\mathcal{H}$ of head-values, see [7], the following property holds:

Stability Theorem: For every term M of the axiomatic rewriting system, there exists a cone of external paths (called head-rewriting paths)

$$e_i \quad : \quad M \twoheadrightarrow V_i \qquad\qquad \text{with } V_i \in \mathcal{H}$$

indexed by $i \in I$, which satisfies the following universality property: for every rewriting path $f : M \twoheadrightarrow W$ reaching a head-value $W \in \mathcal{H}$, there exists a unique index $i \in I$ such that the rewriting path f factors as

$$f \sim e_i \cdot h \quad : \quad M \twoheadrightarrow W$$

for a given rewriting path $h : V_i \twoheadrightarrow W$. The rewriting path $h : V_i \twoheadrightarrow W$ is moreover unique modulo permutation equivalence. In the case of axiomatic rewriting systems without critical pairs, the theorem establishes the existence of a head-rewriting path $e : M \twoheadrightarrow V$ for every term M which can be rewritten to a head-value $W \in \mathcal{H}$. The stability theorem is particularly useful in rewriting systems with critical pairs. By way of illustration, it enables one to describe the head-rewriting paths $e_i : M \twoheadrightarrow V_i$ which transport a λ-term M to its head-normal forms in the $\lambda\sigma$-calculus, see [5] for details.

References

[1] Martin Abadi, Luca Cardelli, Pierre-Louis Curien, Jean-Jacques Lévy. Explicit substitutions. Conférence Principle Of Programming Languages, 1990.

[2] Georges Gonthier, Jean-Jacques Lévy, Paul-André Melliès. *An abstract standardisation theorem.* Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science (LiCS), Santa Cruz, 1992.

[3] Paul-André Melliès. Description abstraite des systèmes de réécriture. Thèse de doctorat, Université Paris VII, 1996.

[4] Paul-André Melliès. Axiomatic Rewriting Theory I: An axiomatic standardisation theorem. Chapter of the Festschrift in honour of Jan Willem Klop: *Processes, Terms and Cycles: Steps on the Road to Infinity.* Edited by A. Middeldorp, V. van Oostrom, F. van Raamsdonk and R. de Vrijer. Lecture Notes in Computer Science 3838, Springer Verlag (2006).

[5] Paul-André Melliès. Axiomatic Rewriting Theory II : The lambda-sigma calculus enjoys finite normalisation cones. Journal of Logic and Computation, vol 10 No. 3, pp. 461-487, 2000.

[6] Paul-André Melliès. Axiomatic Rewriting Theory III : A factorisation theorem in Rewriting Theory. Proceedings of the 7th Conference on Category Theory and Computer Science (CTCS), Santa Margherita Ligure, Lecture Notes in Computer Science 1290, pp. 49-68, Springer, 1997.

[7] Paul-André Melliès. Axiomatic Rewriting Theory IV : A stability theorem in Rewriting Theory. Proceedings of the 13th Annual Symposium on Logic in Computer Science (LiCS), Indianapolis, pp. 287-298, 1998.

[8] Paul-André Melliès. Une étude micrologique de la négation. Habilitation à Diriger des Recherches, 2016.

[9] Robin Milner. The Space and Motion of Communicating Agents. Cambridge University Press, March 2009.

Non-ω-overlapping TRSs are UN

Stefan Kahrs and Connor Smith

School of Computing
University of Kent
Canterbury, United Kingdom
{S.M.Kahrs,cls204}@kent.ac.uk

Abstract

This paper solves problem #79 of RTA's list of open problems [11] — in the positive. If the rules of a TRS do not overlap w.r.t. substitutions of infinite terms then the TRS has unique normal forms. We first reduce the problem to one of consistency for "similar" constructor term rewriting systems. To prove consistency, we define a relation $\Downarrow$ that is consistent by construction for all TRSs, and which — if transitive — would coincide with the rewrite system's equivalence relation $=_R$. We then prove the transitivity of $\Downarrow$ by coalgebraic reasoning. This involves showing that it is transitive on any finite Σ-coalgebra.

1 Introduction

Note: this is the short version of a paper that appeared in FSCD 2016 [6].

For over 40 years [10] it has been known that TRSs that are left-linear and non-overlapping are confluent, and for over 30 years [5] that non-overlapping on its own may not even give us unique normal forms:

Example 1. *By Huet [5]: $\{F(x,x) \to A, F(x,G(x)) \to B, C \to G(C)\}$. The term $F(C,C)$ possesses two distinct normal forms, A and B.*

However, in a certain sense the first two rules overlap semantically: the infinite term $G(G(\cdots))$ provides such an overlap, and in the world of infinitary rewriting [7] the term C even rewrites to that term in the limit.

The notion of overlap is based on the notion of substitution. By changing the codomain of the substitutions of concern from the set of finite terms to the set of infinitary (finite or infinite) ones we arrive at the notion of ω-overlap. This creates the question: *do non-ω-overlapping TRSs have unique normal forms*? This was first conjectured 27 years ago by Ogawa [9].

When making the step from a rewrite relation $\to_R$ to its equivalence closure $=_R$ one is typically interested in its consistency [2, p32ff], i.e. are there terms t, u such that $\neg(t =_R u)$? Both uniqueness of normal forms (UN) and consistency (CON) can be looked at as properties of open terms or ground terms. We stick in this paper to the versions on open terms, as these notions are unaffected by signature extensions.

For non-ω-overlapping systems UN and CON are closely related, as we can extend non-UN systems in a seemingly harmless way to make them fail CON too:

Example 2. *Add to the system of Example 1 the rewrite rules $H(A,x,y) \to x$ and $H(B,x,y) \to y$. The system remains non-overlapping but it is now inconsistent.*

As similar (non-ω-overlapping-preserving) modifications are always possible it suffices to look at the consistency problem instead.

Even if a TRS is non-ω-overlapping, the reduction relation $\to_R$ may still not be confluent (and so we need a different approach to show consistency); this follows from a well-known example by Klop [8]:

Example 3. $\{A \to C(A), C(x) \to D(x, C(x)), D(x,x) \to E\}$.

In this system we have $A \to_R^* E$ and $A \to_R^* C(E)$, but $C(E)$ and E have no common reduct.

One can reduce the consistency problem for arbitrary TRSs to (a very similar version of) the problem for constructor TRSs. The translation works by (i) doubling up the signature, so that for each function symbol F we have both a constructor version F_c and a destructor F_d; (ii) translating the rewrite rules to make them comply with the regime of Constructor TRSs; (iii) adding further rules that make former patterns regain pattern status. Overall, this translation preserves and reflects consistency of a TRS.

Example 4. *If we take the rewrite rules of Combinatory Logic, $A(A(K,x),y) \to x$ and $A(A(A(S,x),y),z) \to A(A(x,z),A(y,z))$ and apply the translation, we end up with the following system:*

$$
\begin{array}{cc}
A_d(A_c(K_c,x),y) \to x & A_d(A_c(A_c(S_c,x),y),z) \to A_d(A_d(x,z),A_d(y,z)) \\
A_d(K_c,x) \to A_c(K_c,x) & A_d(S_c,x) \to A_c(S_c,x) \\
K_d \to K_c & A_d(A_c(S_c,x),y) \to A_c(A_c(S_c,x),y) \\
 & S_d \to S_c
\end{array}
$$

The top two rules are the translated versions of the original rules, the ones below are their respective pattern rules.

In Example 4, an orthogonal TRS was translated into an orthogonal Constructor TRS. In general, this will not be the case, and non-ω-overlapping TRSs will not remain non-ω-overlapping either. However, all overlaps created by the translation are benign, to an extent that we choose to ignore the details here — which can be found in the full version [6].

At the heart of our overall proof is showing (for our rewrite systems in question) that the equivalence closure $=_R$ of single rewrite steps is a subrelation of a consistent relation $\Downarrow$ and therefore itself consistent. This relation $\Downarrow$ is defined using slightly stronger closure principles than those that characterise the joinability relation $\downarrow$, however they remain weak enough to ensure (for arbitrary TRSs) that $\Downarrow$ is consistent. Because $\Downarrow$ is closed under the same operations as $=_R$, *except for transitivity*, proving consistency of $=_R$ can be reduced to proving that $\Downarrow$ is transitive.

2 Term-Coalgebras, and relations on them

Relations on terms can more generally be viewed as relations on or between Σ-coalgebras. This can be useful to stratify the reasoning on terms, one finite Σ-coalgebra at a time.

In order to consider coalgebras of signatures Σ we would have to view signatures as functors on the category Set. However, we only need here the following special instance of this concept:

Definition 1. *Given a signature Σ, a* term-coalgebra *is a set $A \subseteq Ter^\infty(\Sigma, \emptyset)$ which is closed under subterms. It is called* finite *if it is a finite set, and* strongly finite *if in addition $A \subseteq Ter(\Sigma, \emptyset)$.*

More generally, Σ-coalgebras A would be characterized by a function $\upsilon : A \to \Sigma(A)$ which maps a node to a structure containing its root function symbol and the list of its subnodes. We also allow for variables in term-coalgebras by "freezing" them, i.e. when considered as a member of a term-coalgebra a variable is a nullary constructor. For heterogeneous relations between term-coalgebras we must therefore have that the variable set X is the same, so that they are coalgebras of the same functor.

One ingredient to define relations between or on term-coalgebras for a signature Σ we use the following notation: if $R \subseteq A \times B$, where A and B are term-coalgebras A and B then $\widetilde{R} \subseteq A \times B$ is defined as follows:

$$\begin{aligned}\forall t \in A.\, \forall u \in B.\, t\, \widetilde{R}\, u \iff & \exists F \in \Sigma.\, \exists a_1, \ldots, a_n \in A.\, \exists b_1, \ldots, b_n \in B.\\ & t = F(a_1, \ldots, a_n) \wedge u = F(b_1, \ldots, b_n) \wedge \forall i.\, a_i\; R\; b_i\end{aligned}$$

This concept was first used in [3, 4]; we modified it slightly by removing the reflexivity case. For constructur signatures, we use the notations $\overline{R}$ and $\widehat{R}$ to mean $\widetilde{R}$ for the subsignatures Σ_d and Σ_c, respectively. In particular, $t\; \widehat{id}\; t$ iff the root symbol of t is a constructor, and so $\widehat{R} \cdot \overline{S} = \emptyset$. We still use $\widetilde{R}$ for constructor signatures, to refer to the combined signature; hence $\widetilde{R} = \overline{R} \cup \widehat{R}$.

Definition 2. *A relation R between term-coalgebras is called Σ-closed iff $\widetilde{R} \subseteq R$.*

Note: this is standard terminology taken from [1], except that we generalise it to coalgebras.

If id_V is the coreflexive identity on variables then we can express consistency of a relation R relation-algebraically as $id_V \cdot R \cdot id_V \subseteq id_V$. However, we already have consistency issues when a relation R relates any terms topped with distinct constructors. Relations that do not do that we call "constructor-consistent": $\widehat{id} \cdot R \cdot \widehat{id} \subseteq \widehat{1}$, where "1" is top element of the lattice of relations. To reason about pattern matching we need something even stronger than that:

Definition 3. *A relation R between term-coalgebras is called* constructor-compatible *iff*

$$\widehat{id} \cdot R \cdot \widehat{id} \subseteq \widehat{R}.$$

Constructor-compatible relations are preserved by arbitrary union; consequently, relations defined as $\mu x.\, f(x)$ are constructor-compatible whenever the function f preserves this property.

Proposition 1. *Let $a \to b$ and $c \to d$ be two rewrite rules (of some constructor TRS) with only trivial ω-overlaps. Let $=_S$ be a constructor-compatible and Σ-closed equivalence. Then $\sigma(a) \overline{=_S}\, \theta(c)$ implies $\sigma(b) =_S \theta(d)$.*

The reason this is true is that (i) every equation derived via the ω-unification algorithm still holds in $=_S$, (ii) Σ-closed equivalences "are" algebras, so that we can "interpret" eventually the anti-unifiers of b and d in $=_S$-equivalent environments, giving $=_S$-equivalent results.

Given a TRS with signature Σ, and a term-coalgebra A, the relation $\Downarrow_A$ is a relation on A defined as follows:

$$\Downarrow_A \doteq \mu x.\, x^{-1} \cup \xrightarrow{\epsilon}_A \cdot\, x \cup \overline{x} \cdot x \cup id_A \cup \widehat{x}$$

The relation $\Downarrow$ bears some similarity to joinability, but has one stronger feature: we have $\overline{\Downarrow} \cdot \Downarrow \subseteq \Downarrow$. Like joinability, $\Downarrow$ is also constructor-compatible (and therefore consistent), for any rewrite system. It is also always Σ-closed.

3 Proof Graphs

We want to prove that in rewrite systems with only trivial ω-overlaps the relation $\Downarrow$ is transitive, in fact that all $\Downarrow_A$ are transitive. For this it suffices to look at strongly finite A. For such coalgebras we can build a "universal proof" for $\Downarrow_A$ as a union/find-structure, i.e. an equivalence $=_E$ where $t =_E u \iff t \Downarrow_A u$. Notice that there is some similarity to the reduction graphs found in [12], which are essentially proof graphs for a different invariant relation, i.e. for joinability. In other words, these concepts could be subjected to a generalisation, aiming at a more general technique to prove invariants of TRSs by reasoning with equivalences on finite coalgebras.

As nodes of this union/find-structure we use the nodes of the coalgebra A, edges express a relation $\to_e$ under which $\Downarrow_A$ is prefix-closed (i.e. $\to_e \cdot \Downarrow_A \subseteq \Downarrow_A$) — such as $\overline{\Downarrow_A}$ or $\stackrel{\epsilon}{\to}_A$. Because of that (and because $\Downarrow_A$ is reflexive and symmetric), any two elements of a connected component of that structure are automatically in $\Downarrow_A$-relation. For constructors, one can ensure that $=_E$ is Σ_c-closed (and remains constructor-compatible) by adding the necessary edges between constructor-topped terms.

To ensure $=_E$ is eventually Σ-closed we prioiritize adding edges of the form $\overline{\Downarrow_A}$ over root-rewrite steps. This way we can also ensure that any two nodes related by $\overline{\Downarrow_A}^*$ are related by $=_E$. This leaves for any equivalence class of $\overline{\Downarrow_A}^*$ at most one node to which we can attach a redex-contraction edge.

After doing that we have a complete proof graph, i.e. one where we cannot add any further (proof-carrying) edges to merge equivalence classes. The so-constructed relation $=_E$ is necessarily a constructor-compatible congruence relation. Moreover, when the constructor TRS is almost non-ω-overlapping then it coincides with $\Downarrow_A$.

The reason for the latter is a fixpoint argument: suppose a rewrite step $t \stackrel{\epsilon}{\to}_A u$ had *not* been added in the construction of the proof graph for $=_E$. Then we have $t\ \overline{\Downarrow_A}^*\ t' \stackrel{\epsilon}{\to}_A u'$ and $t' =_E u'$ for some t', u' — here t' is the representative redex of its $\overline{\Downarrow_A}$-equivalence class. Assuming (on subterms) that $=_E$ coincides with $\Downarrow_A$ we have $t\ \overline{=_E}\ t'$, and then we can apply the argument from Proposition 1 to get $u =_E u'$, and therefore $t =_E u$.

To avoid this somewhat problematic fixpoint argument, one can instead maintain the property that all edges of the form $t\ \overline{\Downarrow_A}\ u$ also satisfy $t\ \overline{=_E}\ u$. However, such edges would not be added to the graph inductively, but coinductively:

Example 5. *Take the TRS with rules $\{A \to F(A), B \to F(B), F(x) \to C\}$. Its (universal) proof graph for the coalgebra $\{A, B, F(A), F(B), C\}$ contains the rewrite steps for $A \stackrel{\epsilon}{\to} F(A)$, $B \stackrel{\epsilon}{\to} F(B)$ and $F(B) \stackrel{\epsilon}{\to} C$, as well as the "inner" step $F(A)\ \overline{=_E}\ F(B)$. In a way, this inner step justifies itself, because A and B become linked by the addition of this very edge.*

This kind of co-inductive construction is sound, since we *require* that $\overline{=_E}$ edges also satisfy $\overline{\Downarrow_A}$. For instance, that condition would fail in Example 5 if we replaced the rule $F(x) \to C$ with $F(x) \to x$.

Theorem 1. *Almost non-ω-overlapping Constructor TRSs have a consistent equational theory.*

4 Future Work

We would like to extend the result to wider ranges of TRSs, in particular non-ω-overlapping as the condition appears much stronger than needed. The reason is that we do not have to resolve ambiguities in the theories in which they arise, we only have to resolve them *eventually*.

Moreover, it would be nice to apply the proof graph technique to other invariants than $\Downarrow$, e.g. for invariants directly addressing confluence and unique normal forms — as $\Downarrow$ is primarily a consistency invariant. This is likely to require a generalisation of the concept of "proof graph extension". For the construction shown here we merely allowed to extend a proof graph with *edges*, merging its equivalence classes. Joinability ($\downarrow$) is the natural invariant for confluence, but for joinability proof graphs we would need the capability to extend a graph with nodes as well.

References

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

[2] Hendrik P. Barendregt. *The Lambda-Calculus, its Syntax and Semantics.* North-Holland, Amsterdam, 1984.

[3] Jörg Endrullis, Helle Hvid Hansen, Dimitri Hendriks, Andrew Polonsky, and Alexandra Silva. A coinductive treatment of infinitary rewriting. In *Workshop on Infinitary Rewriting*, page 8, 2013.

[4] Jörg Endrullis, Dimitri Hendriks, Helle Hvid Hansen, Andrew Polonsky, and Alexandra Silva. A coinductive framework for infinitary rewriting and equational reasoning. In *Rewriting Techniques and Applications*, 2015.

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

[6] Stefan Kahrs and Connor Smith. Non-Omega-Overlapping TRSs are UN. In Delia Kesner and Brigitte Pientka, editors, *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016)*, volume 52 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[7] Richard Kennaway and Fer-Jan de Vries. *Term Rewriting Systems*, chapter Infinitary Rewriting, pages 668–711. Cambridge University Press, 2003.

[8] Jan Willem Klop. *Combinatory Reduction Systems.* PhD thesis, Centrum voor Wiskunde en Informatica, 1980.

[9] Mizuhito Ogawa and Satoshi Ono. On the uniquely converging property of nonlinear term rewriting systems. Technical Report IEICE COMP89-7, NTT Software Laboratories, 1989.

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

[11] RTA. The RTA list of open problems.
`http://www.win.tue.nl/rtaloop/index.html`

[12] Masahiko Sakai, Michio Oyamaguchi, and Mizuhito Ogawa. Non-e-overlapping, weakly shallow, and non-collapsing trss are confluent. In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 111–126. Springer, 2015.

Efficiently Deciding Uniqueness of Normal Forms and Unique Normalization for Ground TRSs*

Bertram Felgenhauer

University of Innsbruck
bertram.felgenhauer@uibk.ac.at

Abstract

We present an almost linear time algorithm for deciding uniqueness of normal forms for ground TRSs, and a cubic time algorithm for deciding unique normalization for ground TRSs.

1 Introduction

It is known that $\mathsf{UN}^{=}$ and $\mathsf{UN}^{\rightarrow}$ are decidable in polynomial time for ground TRSs. In this note, we are interested in bounding the exponent of the polynomial, which is of great interest to implementers. As far as we know, the best previous result for $\mathsf{UN}^{=}$ is an almost quadratic algorithm by Verma et al. [9] with $O(||\mathcal{R}||^2 \log ||\mathcal{R}||)$ time complexity, where $||\mathcal{R}||$ denotes the sum of the sizes of the sides of $\mathcal{R}$. In Section 3 we present an algorithm that decides $\mathsf{UN}^{=}$ in $O(||\mathcal{R}|| \log ||\mathcal{R}||$ time. In fact our algorithm is closely related to another algorithm by Verma [8, Theorem 31], but some care is needed to achieve an almost linear bound.

In the case of $\mathsf{UN}^{\rightarrow}$ for ground TRSs, Verma [8] and Godoy and Jacquemard [4] have established that polynomial time algorithms exist, using tree automata techniques. No precise bound is given by these authors. In Section 4 we will sketch (due to limited space) an $O(||\mathcal{R}||^3)$ time algorithm for deciding $\mathsf{UN}^{\rightarrow}$, based on a rewriting analysis reminiscent of the cubic time algorithm for confluence in [3].

2 Preliminaries

We assume familiarity with term rewriting and (bottom-up) tree automata, see [1, 2]. Fix a finite signature Σ. A tree automaton $\mathcal{A} = (Q, Q_f, \Delta)$ consists of a finite set of states Q disjoint from Σ, a set of final states $Q_f \subseteq Q$, and a set Δ of transitions $f(q_1, \ldots, q_n) \to q$ and ϵ-transitions $p \to q$, where f is an n-ary function symbol and $q_1, \ldots, q_n, p, q \in Q$. A deterministic tree automaton is an automaton without ϵ-transitions whose transitions have distinct left-hand sides (we do not require deterministic tree automata to be completely defined). Note that Δ can be viewed as a ground TRS over an extended signature that contains Q as constants. We write $\to_{\mathcal{A}}$ for $\to_{\Delta}$, where we regard the transitions as rewrite rules. For a TRS $\mathcal{R}$ we define $\mathcal{R}^{-} = \{r \to \ell \mid \ell \to r \in \mathcal{R}\}$. We write $t \trianglelefteq \mathcal{R}$ if t is a subterm of a side of a rule in $\mathcal{R}$. The size $||\mathcal{R}||$ of $\mathcal{R}$ is the sum of the sizes of the sides of $\mathcal{R}$. The unique normal forms property (convertible normal forms are equal) and the unique normalization property (no term reaches two distinct normal forms) are denoted by $\mathsf{UN}^{=}$ and $\mathsf{UN}^{\rightarrow}$, respectively. For a relation $\to$, $\mathrel{\mathrlap{\to}\,{}^{\shortparallel}}$ denotes its parallel closure, $\to^!$ denotes reduction to a normal form, and $s{\downarrow}$ denotes a normal form of s. In particular, $s \to^! s{\downarrow}$.

*This research was supported by FWF project P27528.

3 Deciding $\mathsf{UN}^=$

We need some preparation before deciding $\mathsf{UN}^=$.

3.1 Currying

Currying allows us to turn an arbitrary TRS into one over constants and a single binary function symbol, thereby bounding the maximum arity of the resulting TRS.

In order to curry a TRS $\mathcal{R}$, we change all function symbols in Σ to be constants, and add a fresh, binary function symbol $\circ$, which we write as a left-associative infix operator. We define

$$(f(t_1,\ldots,t_n))^\circ = f \circ t_1^\circ \circ \cdots \circ t_n^\circ$$

The curried version of $\mathcal{R}$ is given by $\mathcal{R}^\circ = \{\ell^\circ \to r^\circ \mid \ell \to r \in \mathcal{R}\}$.

For ground systems, currying reflects and preserves $\mathsf{UN}^\to$ and $\mathsf{UN}^=$. For reflection, a direct simulation argument works ($s \to_\mathcal{R} t$ implies $s^\circ \to_{\mathcal{R}^\circ} t^\circ$, and s° is a $\mathcal{R}^\circ$-normal form if and only if s is an $\mathcal{R}$-normal form). For preservation, Kenneway et al. [5] show that $\mathsf{UN}^\to$ is preserved by currying for left-linear systems, and that $\mathsf{UN}^=$ is preserved by currying for arbitrary TRSs.

3.2 Recognizing Normal Forms

With $Q = Q_f = \{[s] \mid s \trianglelefteq \mathcal{R}\}$ and $\Delta = \{f([s_1],\ldots,[s_n]) \to [f(s_1,\ldots,s_n)] \mid f(s_1,\ldots,s_n) \trianglelefteq \mathcal{R}\}$ we obtain a deterministic tree automaton that accepts the subterms of $\mathcal{R}$. We modify this automaton to recognize normal forms. To this end, let $\star$ be a fresh constant and let

$$\begin{aligned}
Q' = Q'_f &= \{[s] \mid s \trianglelefteq \mathcal{R} \text{ and } s \text{ is } \mathcal{R}\text{-normal form}\} \cup \{[\star]\}\\
\Delta' &= \{f([s_1],\ldots,[s_n]) \to [f(s_1,\ldots,s_n)] \mid [f(s_1,\ldots,s_n)] \in Q'_f\} \cup\\
&\quad\ \{f([s_1],\ldots,[s_n]) \to [\star] \mid f \in \Sigma,\ [s_1],\ldots,[s_n] \in Q'_f,\ f(s_1,\ldots,s_n) \ntrianglelefteq \mathcal{R}\}
\end{aligned}$$

The state $[\star]$ accepts those $\mathcal{R}$-normal forms that are not subterms of $\mathcal{R}$.

Proposition 1. *The automaton $\mathcal{N}_\mathcal{R} = (Q', Q'_f, \Delta')$ recognizes the $\mathcal{R}$-normal forms over Σ.* ☐

3.3 Congruence Closure

Congruence closure (introduced by Nelson and Oppen [6]; a clean and fast implementation can be found in [7]) is an efficient method for deciding convertibility of ground terms modulo a set of ground equations $\mathcal{R}$.

The congruence closure consists of two phases. In the first phase, the procedure determines the congruence classes (hence the name) among the subterms of the given set of equations, where two subterms s and t are identified if and only if they are convertible, $s \leftrightarrow^*_\mathcal{R} t$. We write $[s]_\mathcal{R}$ for the convertibility class of s. In the second phase, given two terms u and v, we compute the normal forms with respect to rules

$$\mathcal{C} = \{f([s_1]_\mathcal{R},\ldots,[s_n]_\mathcal{R}) \to [f(s_1,\ldots,s_n)]_\mathcal{R} \mid f(s_1,\ldots,s_n) \trianglelefteq \mathcal{R}\}$$

The terms u and v are $\mathcal{R}$-convertible if and only if $u{\downarrow_\mathcal{C}} = v{\downarrow_\mathcal{C}}$. We observe the following.

Proposition 2. *If we regard $[s]_\mathcal{R}$ for subterms $s \trianglelefteq \mathcal{R}$ as fresh constants, the set $\mathcal{C}$ is an orthogonal, ground TRS whose rules, as transitions of a tree automaton, are deterministic.* ☐

1: compute $\mathcal{C}_\mathcal{R}$ and a representation of $\mathcal{N}_\mathcal{R}$
2: **for all** constants $c \trianglelefteq \mathcal{R}$ that are normal forms **do**
3: push $([c]_\mathcal{R}, [c])$ to *worklist*
4: **while** *worklist* not empty **do**
5: $(p, q) \leftarrow$ pop *worklist*
6: **if** $\mathit{seen}(p)$ is defined **then**
7: **return** $\mathsf{UN}^=(\mathcal{R})$ is false
8: $\mathit{seen}(p) \leftarrow q$
9: **for all** transitions $p_1 \circ p_2 \to p_r \in \mathcal{C}_\mathcal{R}$ with $p \in \{p_1, p_2\}$ **do**
10: **if** $q_1 = \mathit{seen}(p_1)$ and $q_2 = \mathit{seen}(p_2)$ are defined **then**
11: **if** there is a transition $q_1 \circ q_2 \to q_r \in \mathcal{N}_\mathcal{R}$ **then**
12: push (p_r, q_r) to *worklist*
13: **return** $\mathsf{UN}^=(\mathcal{R})$ is true

Figure 1: Deciding $\mathsf{UN}^=(\mathcal{R})$

Consequently, we may represent $\mathcal{C}$ as a deterministic tree automaton $\mathcal{C}_\mathcal{R} = (Q, Q_f, \Delta)$ with $Q = Q_f = \{[s]_\mathcal{R} \mid s \trianglelefteq \mathcal{R}\}$ and $\Delta = \mathcal{C}$. Each state $[s]_\mathcal{R}$ accepts precisely the terms convertible to s. Note that the automaton is not completely defined in general: Only terms s that allow a conversion $s \leftrightarrow^*_\mathcal{R} t$ with a root step are accepted.

3.4 Checking $\mathsf{UN}^=$

Given a ground TRS $\mathcal{R}$, we want to decide $\mathsf{UN}^=(\mathcal{R})$, that is, whether any two $\mathcal{R}$-convertible $\mathcal{R}$-normal forms are equal.

First note that if we have two distinct convertible normal forms $s \leftrightarrow^*_\mathcal{R} t$ such that the conversion does not contain a root step, then there are strict subterms of s and t that are convertible and distinct. Therefore, $\mathsf{UN}^=(\mathcal{R})$ reduces to the question whether any state of $\mathcal{C}_\mathcal{R}$, the automaton produced by the congruence closure of $\mathcal{R}$, accepts more than one normal form. Let $\mathcal{C}_\mathcal{R} \cap \mathcal{N}_\mathcal{R}$ be the result of the product construction on $\mathcal{C}_\mathcal{R}$ and $\mathcal{N}_\mathcal{R}$. We can decide $\mathsf{UN}^=$ by enumerating accepting runs $t \to^*_{\mathcal{C}_\mathcal{R} \cap \mathcal{N}_\mathcal{R}} (q_1, q_2)$ in a bottom-up fashion until either

- we obtain two distinct accepting runs ending in (q_1, q_2) and (q_1', q_2') with $q_1 = q_2$, in which case $\mathsf{UN}^=(\mathcal{R})$ does not hold; or
- we have exhausted all runs, in which case $\mathsf{UN}^=(\mathcal{R})$ holds.

Assume that $\mathcal{R}$ is curried. The enumeration of accepting runs can be performed by the algorithm in Figure 1. The correctness of the procedure hinges on two key facts: First, the automaton $\mathcal{C}_\mathcal{R} \cap \mathcal{N}_\mathcal{R}$ is deterministic, which means that distinct runs result from distinct terms. Secondly, the set of $\mathcal{R}$ normal forms is closed under subterms, so we can skip non-normal forms in the enumeration.

Theorem 3. *The algorithm in Figure 1 is correct and runs in $O(||\mathcal{R}|| \log ||\mathcal{R}||)$ time.*

Proof. We have already argued correctness, so let us focus on the complexity. Let $n = ||\mathcal{R}||$. First we compute $\mathcal{C}_\mathcal{R}$ using the congruence closure algorithm from [7] in $O(n \log n)$ time. While $\mathcal{N}_\mathcal{R}$ has quadratically many transitions, we can define the transitions as a partial function using $O(n \log n)$ time for preparation and $O(\log n)$ time per invocation of the transition function. This bound relies on currying, for constant size left-hand sides, and on perfect sharing of terms, for $O(\log n)$ subterm tests. This covers line 1 of the algorithm. Lines 2 to 3 take $O(n)$ time. Note that lines 8 to 12 are executed at most once per state of $\mathcal{C}_\mathcal{R}$, i.e., $O(n)$ times. The enumeration on line 9 can be precomputed in $O(n)$ time, by creating an array of lists of transitions indexed by the states of $\mathcal{C}_\mathcal{R}$ and adding each transition $q_1 \circ q_2 \to q_r \in \mathcal{C}_\mathcal{R}$ to the lists indexed by q_1 and q_2 (if $q_1 \neq q_2$). Because each transition is added to at most two lists, lines 10 to 12 are executed at most twice per transition in $\mathcal{C}_\mathcal{R}$, so $O(n)$ times. The check on line 11 takes $O(\log n)$ time per iteration, so $O(n \log n)$ time in total. Finally, we note that line 12 is executed $O(n)$ times, so no more than $O(n)$ items are ever added to the worklist, which means that lines 4 to 7 are executed $O(n)$ times. Overall the algorithm executes in $O(n \log n)$ time, as claimed. ☐

4 Deciding $\mathsf{UN}^\rightarrow$

4.1 Preparation: Flattening, Rewrite Closure, Meetable Constants

To simplify the reachability analysis in the $\mathsf{UN}^\rightarrow$ property, we flatten the ground TRS $\mathcal{R}$, which we assume to be curried. To this end, we add fresh constants $[s]$ for $s \trianglelefteq \mathcal{R}$, and take the rules

$$\mathcal{E} = \{f([s_1],\dots,[s_n]) \to [f(s_1,\dots,s_n)] \mid f(s_1,\dots,s_n) \trianglelefteq \mathcal{R}\}$$

The flattened TRS is $\mathcal{R}' = \{[\ell] \to [r] \mid \ell \to r \in \mathcal{R}\} \cup \mathcal{E}$. This system simulates rewriting by $\mathcal{R}$.

Proposition 4. ${}_{\mathcal{E}^-}\!\!\overset{!}{\leftarrow} \cdot \to_{\mathcal{R}'} \cdot \to^!_{\mathcal{E}^-} \subseteq \to_\mathcal{R} \subseteq \to^*_\mathcal{E} \cdot \to_{\mathcal{R}'} \cdot {}_{\mathcal{E}}\!\!\overset{*}{\leftarrow}$.

In the following, p and q range over $[t]$ with $t \trianglelefteq \mathcal{R}$. Following [3, Section 3.2], we define the rewrite closure $\mathcal{F}$ of $\mathcal{R}'$ inductively by the following inference rules:

$$\frac{t \trianglelefteq \mathcal{R}}{[t] \to [t] \in \mathcal{F}}\ \text{refl} \qquad \frac{p_1 \circ p_2 \to p \in \mathcal{E} \quad p_1 \to q_1 \in \mathcal{F} \quad p_2 \to q_2 \in \mathcal{F} \quad q_1 \circ q_2 \to q \in \mathcal{E}}{p \to q \in \mathcal{F}}\ \text{comp}$$

$$\frac{p \to q \in \mathcal{R}'}{p \to q \in \mathcal{F}}\ \text{base} \qquad \frac{p \to q \in \mathcal{F} \quad q \to r \in \mathcal{F}}{p \to r \in \mathcal{F}}\ \text{trans}$$

Proposition 5 ([3, Lemma 3.4]). $\to^*_\mathcal{R} \subseteq \to^*_{\mathcal{E}\cup\mathcal{F}} \cdot {}_{\mathcal{E}\cup\mathcal{F}^-}\!\!\overset{*}{\leftarrow}$. ☐

We say two constants p and q are meetable if $p\ {}_{\mathcal{E}\cup\mathcal{F}}\!\!\overset{*}{\leftarrow} \cdot \to^*_{\mathcal{E}\cup\mathcal{F}} q$. In this case we write $p \uparrow q$. The relation $\uparrow$ is dual to $\downarrow$ in [3, Section 3.5] and can be computed as follows.

$$\frac{t \trianglelefteq \mathcal{R}}{[t] \uparrow [t]}\ \text{refl} \qquad \frac{p_1 \circ p_2 \to p \in \mathcal{E} \quad p_1 \uparrow q_1 \quad p_2 \uparrow q_2 \quad q_1 \circ q_2 \to q \in \mathcal{E}}{p \uparrow q}\ \text{comp}$$

$$\frac{q \to p \in \mathcal{F} \quad q \uparrow r}{p \uparrow r}\ \text{trans}_l \qquad \frac{p \uparrow q \quad q \to r \in \mathcal{F}}{p \uparrow r}\ \text{trans}_r$$

4.2 Peak Analysis

Using the rewrite closure, any peak $s\ {}_\mathcal{R}\!\!\overset{*}{\leftarrow} \cdot \to^*_\mathcal{R} t$ between normal forms s and t can be decomposed as

$$s \xrightarrow[\mathcal{E}\cup\mathcal{F}^-]{*} \cdot \xleftarrow[\mathcal{E}\cup\mathcal{F}]{*} \cdot \xrightarrow[\mathcal{E}\cup\mathcal{F}]{*} \cdot \xleftarrow[\mathcal{E}\cup\mathcal{F}^-]{*} t$$

If s and t are chosen to be of minimal size, then there must be a root step. Hence, without loss of generality, there is a constant q such that

$$s \xrightarrow[\mathcal{E}\cup\mathcal{F}^-]{*} q \xleftarrow[\mathcal{E}\cup\mathcal{F}]{*} \cdot \xrightarrow[\mathcal{E}\cup\mathcal{F}]{*} \cdot \xleftarrow[\mathcal{E}\cup\mathcal{F}^-]{*} t \tag{1}$$

Note the special case $s \to^*_{\mathcal{E}\cup\mathcal{F}^-} q\ {}_{\mathcal{E}\cup\mathcal{F}^-}\!\!\overset{*}{\leftarrow} t$, which implies that any q is reachable from at most one normal form using rules from $\mathcal{E} \cup \mathcal{F}^-$.

4.3 Checking $\mathsf{UN}^\to$

The computation consists of several steps. Using the relation $\uparrow$, (1) becomes

$$s \xrightarrow[\mathcal{E}\cup\mathcal{F}^-]{*} q \xleftarrow[\mathcal{E}\cup\mathcal{F}]{*} C[q_1,\ldots,q_n] \uparrow^{\shortparallel} C[p_1,\ldots,p_n] \xleftarrow[\mathcal{E}\cup\mathcal{F}^-]{*} t \tag{2}$$

First, we compute the partial function $w(q)$ that maps q to the normal form s with $s \to^*_{\mathcal{E}\cup\mathcal{F}^-} q$, the first part of the conversion (2). If any q is reachable from more than one normal form, $\mathsf{UN}^\to$ does not hold. To perform this computation efficiently, we make use of the automaton $\mathcal{N}_\mathcal{R}$ that recognizes normal forms. The code is similar to Figure 1, lines 2 to 13, but using the automaton $\mathcal{A} = (Q, Q_f, \Delta)$ given by $Q = Q_f = \{[s] \mid s \trianglelefteq \mathcal{R}\}$ and $\Delta = \mathcal{E} \cup \mathcal{F}^-$ instead of $\mathcal{C}_\mathcal{R}$. Because the product automaton is no longer deterministic, we actually have to compute witnesses and only fail in line 7 if the witnesses are different. Furthermore, in addition to lines 9 to 12, we need a similar loop processing the ϵ-transitions (from $\mathcal{F}^-$). The latter change increases the complexity from $O(||\mathcal{R}|| \log ||\mathcal{R}||$ to $O(||\mathcal{R}||^2)$.

Secondly, we analyze the right part of the conversion (2). To this end, we compute the partial function $w'(q)$ that maps q to the normal form t with $q \;{}_{\mathcal{E}\cup\mathcal{F}}{\overset{*}{\leftarrow}} \cdot \uparrow^{\shortparallel} \cdot {}_{\mathcal{E}\cup\mathcal{F}^-}{\overset{*}{\leftarrow}}\; t$, or to ∞ if there is more than one such normal form. Note that by (2) with $C = \Box$, we have $w(p) = w'(q)$ or $w'(q) = \infty$ whenever $p \uparrow q$ and $w(q)$ is defined. Extending the base cases to larger contexts requires analyzing the $q \;{}_{\mathcal{E}\cup\mathcal{F}}{\overset{*}{\leftarrow}}\; C[q_1,\ldots,q_n]$ sequence and can be done almost the same way as the computation of $w(q)$, using $\mathcal{E}\cup\mathcal{F}$ instead of $\mathcal{E}\cup\mathcal{F}^-$. The complexity of this computation is still $O(||\mathcal{R}||^2)$, despite a subtlety: whereas $w(q)$ is updated at most once, each $w'(q)$ may be updated twice: to record a witness, and to record that there is more than one witness.

The system has the $\mathsf{UN}^\to$ property if $w'(q) = w(q)$ whenever $w(q)$ is defined. Overall, the computation is dominated by the computation of the rewrite closure and the meetable constants, which take $O(||\mathcal{R}||^3)$ time [3]. Hence, $\mathsf{UN}^\to$ can be decided in cubic time.

References

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

[2] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications.* 2007. URL: http://tata.gforge.inria.fr.

[3] B. Felgenhauer. Deciding confluence of ground term rewrite systems in cubic time. In *Proc. 23rd International Conference on Rewriting Techniques and Applications*, volume 15 of *Leibniz International Proceedings in Informatics*, pages 165–175, 2012.

[4] G. Godoy and F. Jacquemard. Unique normalization for shallow TRS. In *Proc. 20th International Conference on Rewriting Techniques and Applications*, volume 5595 of *Lecture Notes in Computer Science*, pages 63–77, 2009.

[5] R. Kennaway, J.W. Klop, M. Ronan Sleep, and F.-J. de Vries. Comparing curried and uncurried rewriting. *Journal of Symbolic Computation*, 21(1):15–39, 1996.

[6] G. Nelson and D.C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, 1980.

[7] R. Nieuwenhuis and A. Oliveras. Fast congruence closure and extensions. *Information and Computation*, 205(4):557–580, 2007.

[8] R. Verma. Complexity of normal form properties and reductions for term rewriting problems. *Fundamenta Informaticae*, 92(1–2):145–168, 2009.

[9] R.M. Verma, M. Rusinowitch, and D. Lugiez. Algorithms and reductions for rewriting problems. *Fundamenta Informaticae*, 46(3):257–276, 2001.

Reducing Joinability to Confluence: How to Preserve Linearity and Shallowness*

Luis Moraes and Rakesh Verma

University of Houston, Houston, TX, USA
{ltdemoraes,rverma}@uh.edu

Abstract

We show how joinability can be reduced to confluence. In particular, our reduction preserves both linearity and shallowness through a non-trivial construction. This allows us to extend the scope of previous undecidability results.

1 Introduction

Term rewrite systems are a framework for modeling computations through a collection of rewrite rules, $l \to r$. For general term rewrite systems, many properties of interest are undecidable. However, if we impose restrictions on the format of the rules that can be defined, these properties can become decidable. Linear rules restrict variables to appear only once on each side. Shallow rules restrict variables to appear only at depth zero or one. For term rewrite systems that are both linear and shallow (composed of rules that conform to these constraints) nearly all properties of interest are decidable.

Undecidability results are also interesting since they show we cannot do any better without losing decidability. For instance, in [3] joinability is shown to be undecidable for linear and left-shallow systems. If a reduction exists between properties, we can transfer decidability results between them. A reduction usually introduces new rules that may alter which restrictions still hold. In [2] joinability was reduced to confluence; however, the reduction did not preserve linearity and shallowness. We show it is possible to preserve both properties, thus displaying the deep connection between joinability and confluence. We present a non-trivial reduction that extends the results in [3] which could not be extended by previous reductions.

2 Preliminaries

A *signature* is a set of distinct *function symbols* each assigned with an *arity*. For example, $\Sigma := \{\gamma : n\}$ for $n \in \mathbb{N}$. The arity of a function symbol indicates the number of subterms, i.e. arguments, it requires. Function symbols of arity zero are called *constants*. Let X be a set of variables – constants not appearing in Σ. We now recursively define the set of Σ-*terms*:

Definition 1. *Let $T(\Sigma, X)$ be the set of Σ-terms. $X \subset T(\Sigma, X)$. For all $f : n \in \Sigma$ and $t_1 \ldots t_n \in T(\Sigma, X)$ we have $f(t_1 \cdots t_n) \in T(\Sigma, X)$.*

Terms have a tree structure. Each function symbol in a term appears at a unique *position*. These are distinguished by the path taken from the *root*, the position of the outermost function symbol. We denote as $t|_p$ the *subterm* of t found at position p. The symbol λ is used for the root position: $t|_\lambda = t$. Other positions are represented as a sequence of natural numbers.

*Research supported in part by NSF Grants DUE 1241772 and CNS 1319212

The subterm at position p of $t = f(t_1 \cdots t_n)$ is found recursively: $t|_p = t|_{i.p'} = (t|_i)|_{p'} = t_i|_{p'}$ where $1 \leq i \leq n$. Let $Pos(t)$ denote the set of positions found within the term t. The *height* of a term t is defined as follows: if t is a constant or variable, $height(t) = 0$. Otherwise, for $t = f(t_1 \cdots t_n)$, $height(t) = 1 + \max(height(t_i))$ for $1 \leq i \leq n$. A term is called *flat* if it has height zero or one. We define the *depth* of a subterm according to the position in which it is found. A subterm found at λ is at depth zero. If $t = f(t_1 \cdots t_n)$ is a subterm found at depth d, then each t_i is found at depth $d+1$. A term is called *shallow* if variables only occur at depth zero or one. A term is called *linear* if each variable occurs only once.

A rewrite *rule* is defined as $l \to r$; the term l found on the left-hand side (LHS) is rewritten to the term r found on the right-hand side (RHS). Rules can be applied to any subterm. Rules are considered *flat*, *shallow*, or *linear* if both l and r satisfy these conditions. We write $l \xrightarrow{*} r$ if l reaches r through zero or more rewrite steps. A sequence of terms obtained through successive rewrite steps is called a *derivation*: $u_1 \to u_2 \to \cdots \to u_{n-1} \to u_n$. If there exists a term z such that $s \xleftarrow{*} z \xrightarrow{*} t$ we write $s \uparrow t$ (joins above). Similarly, if there exists a term z such that $s \xrightarrow{*} z \xleftarrow{*} t$ we write $s \downarrow t$ (joins below).

A term rewrite system (TRS) $\mathcal{R}$ is defined through a signature and a set of rewrite rules. A TRS is confluent if for any pair of terms s, t that have a common ancestor these terms can be joined, i.e. $\forall s, t. \quad s \uparrow t \implies s \downarrow t$.

3 Reduction

We are given a joinability problem: a TRS $\mathcal{R}$ and terms s, t whose joinability is of interest. In other words, we would like to know if $s \downarrow t$. We will create a new TRS $\mathcal{R}'$ that is an extension of the original. Our construction will guarantee that $\mathcal{R}'$ will be confluent if and only if the terms s and t join under $\mathcal{R}$. We shall construct $\mathcal{R}'$ incrementally. Each step towards the final TRS shall be denoted through intermediary symbols and rules, Σ_i and $\mathcal{R}_i$.

3.1 Construction

First, we flatten s and t (see [1] for an example, although we only use backward rules). The constant counterparts to s and t shall be denoted c_s and c_t. The flattening procedure generates flat rules ($\mathcal{R}_{flat}$) and constants (Σ_{flat}) that allow $c_s \xrightarrow{*} s$ and $c_t \xrightarrow{*} t$ in a manner similar to a tree automaton. For all rules in $\mathcal{R}_{flat}$, the LHS is a constant outside the original signature. The joinability relation between terms of the original rewrite system is unchanged since $\mathcal{R}_{flat}$ rules do not apply to those terms. Note that $c_s \downarrow c_t$ if and only if $s \downarrow t$ (we can easily modify a proof of one into a proof of the other). We introduce the constant α to serve as a common ancestor to both c_s and c_t. This guaratees that, if $\mathcal{R}_1$ is confluent, c_s and c_t must join.

$$\Sigma_1 := \Sigma \cup \Sigma_{flat} \cup \{\alpha\!:\!0\} \qquad \mathcal{R}_1 := \mathcal{R} \cup \mathcal{R}_{flat} \cup \{\alpha \to c_s,\, \alpha \to c_t\}$$

Now we create a set of function symbols h_i that can serve as a substitute for any other function symbol. These function symbols use the first B subterms as a code for which symbol is being represented. Let M be the maximum arity among all function symbols in Σ_1. For all values i where $0 \leq i \leq M$, we create a new function symbol h_i with arity $B + i$. Thus, each arity possible in Σ_1 has a corresponding h_i function symbol in Σ_2.

$$\Sigma_{code} := \{h_i\!:\!B + i \;\mid\; 0 \leq i \leq M\} \qquad \Sigma_2 := \Sigma_1 \cup \Sigma_{code}$$

We assign to each non-h_i function symbol a binary string. Let B be the minimum length for each string to be unique. Now we can abstract any function symbol $f:i$ as an h_i symbol where the first B positions correspond to f's binary string. The trick is to use c_s and c_t to denote 0 and 1.

$$\mathcal{R}_{code} := \{f(x_1 \cdots x_n) \to h_n(c_{f_1} \cdots c_{f_B}, x_1 \cdots x_n)\} \qquad \mathcal{R}_2 := \mathcal{R}_1 \cup \mathcal{R}_{code}$$

for all $f \in \Sigma_1$ where $c_{f_i} \in \{c_s, c_t\}$. The c_s and c_t subterms that serve as substitute for 0 and 1 shall be called *binary string subterms*.

If we replace all non-h_i symbols with h_i, we are left with a term that only makes use of h_i, c_s, and c_t – we call these *code terms*. Likewise, given a term with one or more h_i symbols, we can replace them with the $f:i$ symbols they represent thus obtaining an h_i-free term – we call these *pure terms*. Finally, the terms that are neither code terms nor pure terms are referred to as *partial code terms* since some, but not all, of the symbols have been replaced.

Suppose we have a correspondence $\{a : 00,\ b : 01,\ f : 10\}$. Thus, a can be represented as $h_0(c_s, c_s)$, b as $h_0(c_s, c_t)$, and f as $h_1(c_t, c_s, x)$. Let $c_s \xrightarrow{*} z \xleftarrow{*} c_t$. Then, many terms already join, for instance we have $a \to h_0(c_s, c_s) \xrightarrow{*} h_0(z, z) \xleftarrow{*} h_0(c_s, c_t) \leftarrow b$. Similarly, $f(a) \xrightarrow{*} h_1(z, z, h_0(z, z)) \xleftarrow{*} f(b)$. However, to ensure confluence, we want every term to join if $c_s \downarrow c_t$. Thus, we must address the following case: $f(a) \xrightarrow{*} h_1(z, z, h_0(z, z)) \neq h_0(z, z) \xleftarrow{*} b$. It is apparent that only *structurally equivalent* code terms can be joined under $\mathcal{R}_2$ if $c_s \downarrow c_t$.

Definition 2. *Two terms t_1, t_2 are* structurally equivalent *if $Pos(t_1) = Pos(t_2)$.*

We introduce a *dummy symbol* δ to work around this issue. We adjust the value of B and the binary strings assigned to each function symbol in order to accomodate δ. We also create an $\mathcal{R}_{code}$ rule for δ. The dummy symbol will be used to extend the structure of a code term. Since what indicates the function symbol is the binary string, we keep the string intact while adding a new position occupied by the dummy symbol.

$$\mathcal{R}_{ex} := \{h_n(x_1 \cdots x_{B+n}) \to h_{n+1}(x_1 \cdots x_{B+n}, \delta)\}$$
$$\Sigma' := \Sigma_2 \cup \{\delta : 0\} \qquad \mathcal{R}' := \mathcal{R}_2 \cup \mathcal{R}_{ex}$$

for $0 \le n \le M - 1$. Now we can join terms of different arity, for instance: $a \xrightarrow{*} h_0(z, z) \to h_1(z, z, \delta) \xrightarrow{*} h_1(z, z, h_0(z, z)) \xleftarrow{*} f(a)$. These rewrites can be chained to extend the arity of a code term by more than one: $a \xrightarrow{*} h_1(z, z, \delta) \to h_2(z, z, \delta, \delta) \xrightarrow{*} h_2(z, z, h_0(z, z), h_0(z, z)) \xleftarrow{*} g(a, a)$. Furthermore, once the dummy symbol is rewritten to a code term, new positions beneath it can be generated in the same manner: $a \xrightarrow{*} h_1(z, z, \delta) \xrightarrow{*} h_1(z, z, h_0(z, z)) \to h_1(z, z, h_1(z, z, \delta)) \xrightarrow{*} h_1(z, z, h_1(z, z, h_0(z, z))) \xleftarrow{*} f(f(a))$.

We are now ready to prove the correctness of this construction.

3.2 Proof of Correctness

These lemmas apply to $\mathcal{R}'$. First we show how $c_s \downarrow c_t$ leads to all pairs of terms being joinable.

Lemma 1. *Every term $t \in \mathcal{T}(\Sigma', X)$ reaches a code term.*

Proof. We proceed by induction on $height(t)$. If t is a constant, then we can rewrite it to an h_0 term with the appropriate binary string. Assume the lemma holds for $height(t) < n$. For t of height n, note its children will have height at most $n - 1$. We rewrite each child to a code term by the inductive hypothesis if the child is not a code term already. Then, we perform an $\mathcal{R}_{code}$ rewrite at the root if necessary. Thus, t reaches a code term. $\square$

Lemma 2. *Let P be the set of positions for an arbitrary code term t. Given a code term s such that $Pos(s) \subseteq P$, we can rewrite s into s' such that $Pos(s') = P$.*

Proof. We proceed by induction on $height(t)$. The minimum height of t is that of a constant's code term. Because $Pos(s) \subseteq P$ we know s is also a code term for a constant, thus structurally equivalent. Assume the lemma holds for $height(t) < n$. For t of height n, note its children will have height at most $n - 1$. We perform $\mathcal{R}_{ex}$ rewrites on s at λ until it has the same number of children as t at that position. We convert the new dummy symbols into code terms. By the inductive hypothesis, we rewrite each child $s|_i$ to a term structurally equivalent to $t|_i$. Both terms now have the same set of positions. □

For any pair of code terms (s, t) we can rewrite them to have the set of positions $Pos(s) \cup Pos(t)$. This set of positions corresponds to that of some code term reachable by δ. We arrive at the following corollary:

Corollary 1. *Any pair of code terms can be rewritten into structurally equivalent code terms.*

Lemma 3. *If $c_s \downarrow c_t$ then any two terms can be joined.*

Proof. Suppose we have two terms: u, v. We know every term reaches a code term by Lemma 1 so u, v reach code terms u', v'. If not already structurally equivalent we can make them so by Corollary 1. It should be evident that structurally equivalent code terms are easily joined by rewriting each c_s, c_t to the term that joins them. □

We must now show the joinability relation between c_s and c_t remains unchanged under $\mathcal{R}'$. For the following lemmas it is important to understand how h_i terms behave in a derivation. Once a subterm is rewritten by $\mathcal{R}_{code}$ that subterm is "locked": no other rewrites can be performed at that position except $\mathcal{R}_{ex}$ rewrites, which preserve subterms. In fact, the h_i symbol works as a cap for the subterm: either rewrites are performed on subterms beneath it, or the whole h_i term must be beneath a variable (thus preserving its subterms).

Definition 3. *A* minimal proof *of joinability between two terms t_1, t_2 is a pair of derivations demonstrating $t_1 \xrightarrow{*} z \xleftarrow{*} t_2$ for some z such that there exists no other pair with a fewer number of rewrite steps.*

Lemma 4. *A minimal proof of joinability for $c_s \downarrow c_t$ performs no rewrites on binary string subterms.*

Proof. Suppose we had a minimal proof of joinability for $c_s \downarrow c_t$ which did rewrite binary string subterms. Consider the two corresponding h_i subterms (one for each derivation) at their inception, before any rewrites are performed on their binary string subterms.

Case 1: The binary strings are the same. Clearly the rewrites on the binary string subterms are superfluous and may be safely removed from the derivations. The resulting proof is shorter, thus violating the minimality of the proof in question.

Case 2: The binary strings are different. Since any rewrites that modify the binary string subterms must be performed at or beneath their position, these by themselves constitute another proof of $c_s \downarrow c_t$. This proof is shorter since it is contained inside the proof in question, thus violating minimality. □

Lemma 5. *A minimal proof of joinability for $c_s \downarrow c_t$ performs no $\mathcal{R}_{ex}$ rewrites.*

Proof. Suppose we had a minimal proof of joinability for $c_s \downarrow c_t$ which did perform $\mathcal{R}_{ex}$ rewrites. Consider the two corresponding h_i subterms (one for each derivation) after extending their arity with dummy symbols.

Case 1: Both terms have a dummy symbol in the same position. Clearly the rewrites are superfluous and may be safely removed from the derivations. The resulting proof is shorter, thus violating the minimality of the proof in question.

Case 2: One term has a dummy symbol and the other does not. Once converted to code terms, the binary strings for these subterms will not match. However, all binary strings must match since we cannot rewrite binary string subterms per Lemma 4. ☐

Let π be a mapping from code terms (partial or otherwise) to pure terms. Given a term t, $\pi(t) = t'$ is the term obtained when each h_i symbol is replaced by its corresponding function symbol in Σ_1. This mapping is well defined only for (partial) code terms which have not introduced any dummy symbols and whose binary string subterms are all c_s or c_t.

Lemma 6. *$c_s \downarrow c_t$ under $\mathcal{R}_1$ iff $c_s \downarrow c_t$ under $\mathcal{R}'$.*

Proof. If a proof of joinability exists, then a minimal proof of joinability exists. By Lemmas 4 and 5 we are guaranteed the mapping π is well defined on terms of the minimal proof. Given a proof that relies on $\mathcal{R}_{code}$ rewrites we can obtain a proof without them by applying π to every term of each derivation.

If $u_i \to u_{i+1}$ is a step in one of the derivations that uses a rule from $\mathcal{R}_{code}$, then it can be erased since $\pi(u_i) = \pi(u_{i+1})$. Similarly, if $u_i \to u_{i+1}$ is a step in one of the derivations that uses a rule from $\mathcal{R}_1$, then $\pi(u_i) \to \pi(u_{i+1})$ since h_i symbols can only occur beneath a variable (or otherwise in a position that does not interfere). Finally, nonlinearity is not an issue since subterms that match before applying π will match afterwards as well. ☐

Theorem 1. *Joinability reduces to confluence while preserving linearity and shallowness restrictions.*

Proof. ($\Longrightarrow$) If $s \downarrow t$ under $\mathcal{R}$ then by Lemma 3 any two terms join under $\mathcal{R}'$. In particular, terms with a common ancestor join. Thus, $\mathcal{R}'$ is confluent. Since all the new rules are linear and flat, the resulting TRS preserves linearity and shallowness.

($\Longleftarrow$) If $\mathcal{R}'$ is confluent, then $c_s \downarrow c_t$ since they have a common ancestor. By Lemma 6 we know $s \downarrow t$ under $\mathcal{R}$ (same as $c_s \downarrow c_t$ under $\mathcal{R}_1$). ☐

Other restrictions are also preserved such as: $Var(r) \subset Var(l)$ and $l \notin X$ for rules $l \to r$. These restrictions come into play when extending undecidability results through the reduction (such as [2]).

References

[1] Guillem Godoy, Ashish Tiwari, and Rakesh Verma. On the confluence of linear shallow term rewrite systems. In *STACS 2003, Symposium on Theoretical Aspects of Computer Science*, volume 2607 of *Lecture Notes in Computer Science*, pages 85–96, 2003.

[2] Rakesh Verma. Complexity of normal form properties and reductions for term rewriting problems. *Fundamenta Informaticae*, 92:145–168, 2009.

[3] Rakesh Verma. New undecidability results for properties of term rewrite systems. *Electronic Notes in Theoretical Computer Science*, 290:69–85, 2012.

Confluence Properties on Open Terms in the First-Order Theory of Rewriting*

Franziska Rapp and Aart Middeldorp

Department of Computer Science, University of Innsbruck, Austria
{franziska.rapp|aart.middeldorp}@uibk.ac.at

Abstract

FORT is a decision and synthesis tool for the first-order theory of rewriting for finite left-linear right-ground rewrite systems. We report on an extension that distinguishes between ground and open terms for properties related to confluence.

1 Introduction

In a recent paper [5] we introduced FORT, a decision and synthesis tool for the first-order theory of rewriting induced by finite left-linear right-ground rewrite systems. In this theory one can express well-known properties like termination (SN), normalization (WN), and confluence (CR), but also properties like strong confluence ($\mathsf{SCR}\colon \forall\, s\, \forall\, t\, \forall\, u\, (s \to t \wedge s \to u \implies \exists\, v\, (t \to^{=} v \wedge u \to^{*} v)))$ and the normal form property ($\mathsf{NFP}\colon \forall\, s\, \forall\, t\, \forall\, u\, (s \to t \wedge s \to^{!} u \implies t \to^{!} u))$. The decision procedure is based on tree automata techniques (Dauchet and Tison [3]). Tree automata operate on *ground* terms. Consequently, variables in formulas range over ground terms and hence the properties that FORT is able to decide are restricted to ground terms. Whereas for termination and normalization this makes no difference, for other properties it does, even for *left-linear right-ground rewrite systems* as will be shown below. This raises the question how one can use FORT to decide properties on open terms. We show that for properties related to confluence it suffices to add one or two fresh constants. We furthermore provide sufficient conditions which obviate the need for additional constants. The proofs of these results are presented in the next section. The results are incorporated in version 0.2 of FORT, which we briefly describe in Section 3. We also provide a few rewrite systems that were synthesized by FORT. Section 4 contains a comparison with AGCP (Aoto and Toyama [1]), a new tool for checking ground-confluence of many-sorted rewrite systems.

We assume familiarity with first-order term rewriting [2]. In this paper we consider the following properties, besides SCR and NFP:

$$\begin{aligned}
\mathsf{CR}\colon&\ \forall\, s\, \forall\, t\, \forall\, u\, (s \to^{*} t \wedge s \to u \implies t \downarrow u)\\
\mathsf{WCR}\colon&\ \forall\, s\, \forall\, t\, \forall\, u\, (s \to t \wedge s \to u \implies t \downarrow u)\\
\mathsf{UN}\colon&\ \forall\, s\, \forall\, t\, \forall\, u\, (s \to^{!} t \wedge s \to^{!} u \implies t = u)\\
\mathsf{UNC}\colon&\ \qquad \forall\, t\, \forall\, u\, (t \leftrightarrow^{*} u \wedge \mathsf{NF}(t) \wedge \mathsf{NF}(u) \implies t = u)
\end{aligned}$$

Let $\mathfrak{P} = \{\mathsf{CR}, \mathsf{SCR}, \mathsf{WCR}, \mathsf{NFP}, \mathsf{UNC}, \mathsf{UN}\}$. In FORT 0.2 these properties are considered over all terms. Let $\mathfrak{R}$ consist of all $(\mathcal{F}, \mathcal{R})$ where $\mathcal{R}$ is a finite left-linear right-ground TRSs over the finite signature $\mathcal{F}$ which contains at least one constant.

*This work is supported by FWF (Austrian Science Fund) project P27528.

2 Ground versus Non-Ground Properties

The properties supported in FORT 0.1 are restricted to *ground* terms. So CR in FORT 0.1 stands for ground-confluence, which is different from confluence, even for left-linear right-ground TRSs. The TRS

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

is ground-confluent since all ground terms rewrite to b, but not confluent: $\mathsf{b} \leftarrow \mathsf{f}(x,\mathsf{a}) \to \mathsf{f}(x,\mathsf{b})$ with normal forms b and $\mathsf{f}(x,\mathsf{b})$. The same example shows that for no property $P \in \mathfrak{P}$, $\mathsf{G}P$ implies P, where $\mathsf{G}P$ denotes the property P restricted to ground terms. So how can we check a property $P \in \mathfrak{P}$ using tree automata techniques? The following result provides the answer.

Lemma 1. *If* $(\mathcal{F},\mathcal{R}) \in \mathfrak{R}$ *then*

1. $(\mathcal{F},\mathcal{R}) \vDash P \iff (\mathcal{F} \cup \{c\},\mathcal{R}) \vDash \mathsf{G}P$ *for all* $P \in \mathfrak{P} \setminus \{\mathsf{UNC}\}$
2. $(\mathcal{F},\mathcal{R}) \vDash \mathsf{UNC} \iff (\mathcal{F} \cup \{c,c'\},\mathcal{R}) \vDash \mathsf{GUNC}$

with fresh constants c *and* c'.

Proof. For the only-if directions we observe that all properties $P \in \mathfrak{P}$ are preserved under signature extension [4]. Moreover, $(\mathcal{G},\mathcal{R}) \vDash P$ implies $(\mathcal{G},\mathcal{R}) \vDash \mathsf{G}P$ for all TRSs $(\mathcal{G},\mathcal{R})$ and properties $P \in \mathfrak{P}$. For the if-direction, we first consider $P \in \mathfrak{P} \setminus \{\mathsf{UNC}\}$. Suppose $(\mathcal{F} \cup \{c\},\mathcal{R}) \vDash \mathsf{G}P$ and let σ be the substitution that maps all variables to the constant c. Because $\mathcal{R}$ is left-linear and c does not appear in the rules of $\mathcal{R}$, the following property holds for all terms $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$:

(a) if $t\sigma \to_{\mathcal{R}} u$ then $t \to_{\mathcal{R}} u'$ with $u'\sigma = u$.

Moreover,

(b) if $t \to_{\mathcal{R}} u$ and $p \in \mathcal{P}\mathsf{os}_{\mathcal{V}}(u)$ then $u(p) = t(p)$.

This property relies on the right-groundness of $\mathcal{R}$, which entails that the redex contracted in $t \to_{\mathcal{R}} u$ cannot be above position p. The above properties allow us to prove $(\mathcal{F},\mathcal{R}) \vDash P$ for $P \in \{\mathsf{CR},\mathsf{SCR},\mathsf{WCR}\}$. Here we consider $P = \mathsf{SCR}$ and let $s \to_{\mathcal{R}} t$ and $s \to_{\mathcal{R}} u$. Closure under substitutions yields $s\sigma \to_{\mathcal{R}} t\sigma$ and $s\sigma \to_{\mathcal{R}} u\sigma$. Because $(\mathcal{F} \cup \{c\},\mathcal{R})$ satisfies GSCR, we obtain a ground term $v \in \mathcal{T}(\mathcal{F} \cup \{c\})$ such that $s\sigma \to_{\mathcal{R}}^{=} v$ and $t\sigma \to_{\mathcal{R}}^{*} v$. Property (a) yields terms $v_1, v_2 \in \mathcal{T}(\mathcal{F},\mathcal{V})$ such that $t \to_{\mathcal{R}}^{=} v_1$ and $u \to_{\mathcal{R}}^{*} v_2$ with $v_1\sigma = v = v_2\sigma$. If $v_1 \neq v_2$ then there must be a position $p \in \mathcal{P}\mathsf{os}_{\mathcal{V}}(v_1) \cap \mathcal{P}\mathsf{os}_{\mathcal{V}}(v_2)$ such that $v_1(p) \neq v_2(p)$. Repeated application of (b) yields $v_1(p) = t(p) = s(p)$ and $v_2(p) = u(p) = s(p)$, which is impossible. Hence $v_1 = v_2$ and thus $(\mathcal{F},\mathcal{R}) \vDash \mathsf{SCR}$. The proofs for $P = \mathsf{CR}$ and $P = \mathsf{WCR}$ are very similar. For $P \in \{\mathsf{UN},\mathsf{NFP}\}$ we need the following additional observation:

(c) if t is a normal form then $t\sigma$ is a normal form.

Consider $P = \mathsf{UN}$ and let $s \to_{\mathcal{R}}^{!} t$ and $s \to_{\mathcal{R}}^{!} u$ with $s \in \mathcal{T}(\mathcal{F},\mathcal{V})$. We obtain $s\sigma \to_{\mathcal{R}}^{!} t\sigma$ and $s\sigma \to_{\mathcal{R}}^{!} u\sigma$ from (c), and thus $t\sigma = u\sigma$ because $(\mathcal{F} \cup \{c\},\mathcal{R})$ satisfies GUN. We need to show $t = u$. If this does not hold then there must be a position $p \in \mathcal{P}\mathsf{os}_{\mathcal{V}}(t) \cap \mathcal{P}\mathsf{os}_{\mathcal{V}}(u)$ such that $t(p) \neq u(p)$. This contradicts $t(p) = s(p)$ and $u(p) = s(p)$, which we obtain from (b). Next consider $P = \mathsf{NFP}$. So let $s \to_{\mathcal{R}} t$ and $s \to_{\mathcal{R}}^{!} u$. We obtain $s\sigma \to_{\mathcal{R}} t\sigma$ and $s\sigma \to_{\mathcal{R}}^{!} u\sigma$ as before. Hence $t\sigma \to^{*} u\sigma$ because GNFP holds. From property (a) we obtain a term u' such that $t \to_{\mathcal{R}}^{*} u'$ and $u'\sigma = u\sigma$. Let p be any position in $\mathcal{P}\mathsf{os}_{\mathcal{V}}(u') \cap \mathcal{P}\mathsf{os}_{\mathcal{V}}(u)$. Repeated application of property (b) yields $u'(p) = t(p) = s(p) = u(p)$. Hence $u' = u$ and thus $t \to_{\mathcal{R}}^{*} u$ as desired.

Finally we consider $P = \mathsf{UNC}$. So suppose $(\mathcal{F} \cup \{c, c'\}, \mathcal{R}) \vDash \mathsf{GUNC}$ and let $t \leftrightarrow_{\mathcal{R}}^* u$ with normal forms $t, u \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. If t and u are ground then $t = u$ by GUNC. If one of the two terms is ground, say t, and $t \neq u$ then $t \neq u\sigma$ and $t \leftrightarrow_{\mathcal{R}}^* u\sigma$ for the same substitution σ as before, contradicting GUNC. If both t and u are non-ground and $t \neq u$ then, because $t\sigma = u\sigma$ by GUNC and (c), there has to be a position $p \in \mathcal{P}\mathsf{os}_{\mathcal{V}}(t) \cap \mathcal{P}\mathsf{os}_{\mathcal{V}}(u)$ such that $t(p) \neq u(p)$. In this case a contradiction is obtained by considering the substitution σ' that maps $t(p)$ to c and all other variables to c'. ☐

The following example shows that adding a single fresh constant is insufficient for UNC.

Example 1. *The left-linear right-ground TRS $\mathcal{R}$ consisting of the rules*

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

does not satisfy UNC *since* $\mathsf{f}(x, \mathsf{b}) \leftarrow \mathsf{f}(x, \mathsf{a}) \to \mathsf{f}(\mathsf{b}, \mathsf{b}) \leftarrow \mathsf{f}(y, \mathsf{a}) \to \mathsf{f}(y, \mathsf{b})$ *is a conversion between distinct normal forms. Adding a single fresh constant* c *is not enough to violate* GUNC *as the last two rewrite rules ensure that* $\mathsf{f}(\mathsf{c}, \mathsf{b})$ *is the only ground instance of* $\mathsf{f}(x, \mathsf{b})$ *that is a normal form. Adding another fresh constant* c', GUNC *is lost:* $\mathsf{f}(\mathsf{c}, \mathsf{b}) \leftarrow \mathsf{f}(\mathsf{c}, \mathsf{a}) \to \mathsf{f}(\mathsf{b}, \mathsf{b}) \leftarrow \mathsf{f}(\mathsf{c}', \mathsf{a}) \to \mathsf{f}(\mathsf{c}', \mathsf{b})$.

For termination (SN) and normalizion (WN) there is no need to add fresh constants, since these properties hold if and only if they hold for all ground terms. For other properties that can be expressed in the first-order theory of rewriting, one or two fresh constants may be insufficient. Consider e.g. the formula φ:

$$\neg\, \exists\, s\, \exists\, t\, \exists\, u\, \forall\, v\, (v \leftrightarrow^* s \vee v \leftrightarrow^* t \vee v \leftrightarrow^* u)$$

which is satisfied on open terms (with respect to any $(\mathcal{F}, \mathcal{R}) \in \mathfrak{R}$). For the TRS consisting of the rule $\mathsf{f}(x) \to \mathsf{a}$ and two additional constants c and c', φ does not hold for ground terms because every ground term is convertible to a, c or c'. It is tempting to believe that adding a fresh unary symbol g in addition to a fresh constant c, in order to create infinitely many ground normal forms which can replace variables that appear in open terms, is sufficient for any property P. The formula $\forall\, s\, \forall\, t\, (s \to t \implies s \xrightarrow{\epsilon} t)$ and the TRS consisting of the rule $\mathsf{a} \to \mathsf{b}$ show that this is incorrect.

It is interesting to note that the two properties in the preceding paragraph are not *component-closed* [6], unlike the properties in $\mathfrak{P}$. This observation can be used to generalize Lemma 1 to confluence-related properties outside $\mathfrak{P}$. The following result shows that for the properties in $\mathfrak{P}$ it is not always necessary to add fresh constants. Here a *monadic* signature consists of constants and unary function symbols.

Lemma 2. *Let $(\mathcal{F}, \mathcal{R}) \in \mathfrak{R}$ such that $\mathcal{R}$ is ground or $\mathcal{F}$ is monadic. For all $P \in \mathfrak{P}$*

$$(\mathcal{F}, \mathcal{R}) \vDash P \quad \iff \quad (\mathcal{F}, \mathcal{R}) \vDash \mathsf{G}P$$

Proof. First assume that $\mathcal{R}$ is ground. In this case only ground subterms can be rewritten. Given a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, we write $t = C[\![t_1, \ldots, t_n]\!]$ if $t = C[t_1, \ldots, t_n]$ and $t_1, \ldots, t_n$ are the maximal ground subterms of t. So all variables appearing in t occur in C. The following property is obvious:

1. if $t = C[\![t_1, \ldots, t_n]\!] \to_{\mathcal{R}}^* u$ then $u = C[\![u_1, \ldots, u_n]\!]$ and $t_i \to_{\mathcal{R}}^* u_i$ for all $1 \leqslant i \leqslant n$.

Suppose $(\mathcal{F}, \mathcal{R}) \vDash \mathsf{GCR}$ and consider $s \to_{\mathcal{R}}^* t$ and $s \to_{\mathcal{R}}^* u$ with $s \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. Writing $s = C[\![s_1, \ldots, s_n]\!]$, we obtain $t = C[\![t_1, \ldots, t_n]\!]$ and $u = C[\![u_1, \ldots, u_n]\!]$ with $s_i \to_{\mathcal{R}}^* t_i$ and $s_i \to_{\mathcal{R}}^* u_i$ for all $1 \leqslant i \leqslant n$. GCR yields $t_i \downarrow u_i$ for all $1 \leqslant i \leqslant n$. Hence $t \downarrow u$ as desired. The proofs for the other properties in $\mathfrak{P}$ are equally easy. For UNC note that $\leftrightarrow^*_{\mathcal{R}}$ coincides with $\to^*_{\mathcal{R}\,\cup\,\mathcal{R}^{-1}}$ for the *ground* TRS $\mathcal{R} \cup \mathcal{R}^{-1}$, where $\mathcal{R}^{-1}$ is obtained from $\mathcal{R}$ by reversing the rewrite rules.

Next suppose that $\mathcal{F}$ is monadic. Let $(\mathcal{F}, \mathcal{R}) \vDash \mathsf{G}P$ and let σ be the substitution that maps all variables to some arbitrary but fixed ground term. In this case the following property holds:

2. if $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ and $t \to u$ then $u \in \mathcal{T}(\mathcal{F})$ and $t\sigma \to u$.

We consider $P = \mathsf{NFP}$ and $P = \mathsf{UNC}$ and leave the proof for the other properties to the reader. Let $s \to_{\mathcal{R}} t$ and $s \to^!_{\mathcal{R}} u$. We obtain $s\sigma \to_{\mathcal{R}} t$ and $s\sigma \to^!_{\mathcal{R}} u$ from property 2. (Note that $s \neq u$.) Hence $t \to^*_{\mathcal{R}} u$ follows from GNFP. Let $t \leftrightarrow^*_{R} u$ with normal forms t and u. If t and u are ground terms then we obtain $t = u$ from GUNC (after applying the substitution σ to all intermediate terms in the conversion between t and u). Otherwise, the conversion between t and u must be empty due to property 2 and the fact that t and u are normal forms. Hence also in this case $t = u$. ☐

FORT indeed benefits from this optimization. Checking for GCR of the TRS

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

whose signature is monadic takes 1.73 seconds if a fresh constant is added, compared to 0.85 seconds if Lemma 2 is used.

3 Synthesis Experiments with FORT 0.2

The results of the previous section are incorporated in version 0.2 of FORT. Compared to version 0.1, the properties in $\mathfrak{P}$ now refer to open terms and we reserve $\mathsf{G}P$ with $P \in \mathfrak{P}$ for properties on ground terms. The property SCR, which is new in version 0.2, can also be used for parallel rewriting ($\mathsf{SCR}(\twoheadrightarrow\!\!\!\!\!\shortmid\;)$) and the same holds for the diamond property ($\diamond(\twoheadrightarrow\!\!\!\!\!\shortmid\;)$), which is another addition in FORT 0.2. Further additions can be found in the online documentation of FORT. Precompiled binaries to run FORT 0.2 from the command line are available from

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

We report on some synthesis experiments that we performed in FORT 0.2, based on the following diagram which summarizes the relationships between properties P and $\mathsf{G}P$ for $P \in \mathfrak{P}$:

$$\begin{tikzcd}
\mathsf{WCR} \arrow[ddd, Rightarrow] & & \mathsf{CR} \arrow[ll, Rightarrow] \arrow[r, Rightarrow] \arrow[ddd, Rightarrow] & \mathsf{NFP} \arrow[r, Rightarrow] \arrow[ddd, Rightarrow] & \mathsf{UNC} \arrow[r, Rightarrow] \arrow[ddd, Rightarrow] & \mathsf{UN} \arrow[ddd, Rightarrow] \\
& \mathsf{SCR} \arrow[ul, Rightarrow] \arrow[ur, Rightarrow] \arrow[d, Rightarrow] & & & & \\
& \mathsf{GSCR} \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & & & & \\
\mathsf{GWCR} & & \mathsf{GCR} \arrow[ll, Rightarrow] \arrow[r, Rightarrow] & \mathsf{GNFP} \arrow[r, Rightarrow] & \mathsf{GUNC} \arrow[r, Rightarrow] & \mathsf{GUN}
\end{tikzcd}$$

The following TRSs were produced by FORT 0.2 on the given formulas when restricting the signature (using the option `-f "f:2 a:0 b:0"`) to a binary function symbol f and two constants a and b:

`GWCR & ~WCR & ~GCR`	$\mathsf{a} \to \mathsf{b}$	$\mathsf{f}(x, \mathsf{a}) \to \mathsf{a}$	$\mathsf{a} \to \mathsf{f}(\mathsf{a}, \mathsf{a})$
`GCR & ~CR & ~GSCR`	$\mathsf{a} \to \mathsf{b}$	$\mathsf{f}(x, \mathsf{a}) \to \mathsf{b}$	$\mathsf{b} \to \mathsf{f}(\mathsf{a}, \mathsf{a})$
`GNFP & ~NFP & ~GCR`	$\mathsf{a} \to \mathsf{b}$	$\mathsf{f}(x, \mathsf{a}) \to \mathsf{f}(\mathsf{a}, \mathsf{a})$	$\mathsf{f}(\mathsf{b}, \mathsf{b}) \to \mathsf{f}(\mathsf{a}, \mathsf{a})$
`GUNC & ~UNC & ~GNFP`	$\mathsf{a} \to \mathsf{a}$	$\mathsf{f}(x, \mathsf{a}) \to \mathsf{a}$	$\mathsf{f}(\mathsf{b}, x) \to \mathsf{b}$

tool	yes (∅ time)	no (∅ time)	maybe (∅ time)	timeout	total time
AGCP	8 (0.02 s)	–	56 (0.19 s)	1	71 s
FORT	42 (0.37 s)	14 (3.31 s)	–	9	602 s

Table 1: Comparison of AGCP and FORT 0.2 on 65 left-linear right-ground TRSs.

The reader is encouraged to verify that the synthesized TRSs indeed satisfy the indicated properties. We do not know whether there exist TRSs over the restricted signature that satisfy `GUN & ~UN & ~GUNC`. Human expertise was used to produce a witness over a larger signature, which was subsequently simplified using the decision mode of FORT 0.2:

$$\begin{array}{lllll} \mathsf{b} \to \mathsf{a} & \mathsf{c} \to \mathsf{c} & \mathsf{d} \to \mathsf{c} & \mathsf{f}(x,\mathsf{a}) \to \mathsf{A} & \mathsf{f}(x,\mathsf{A}) \to \mathsf{A} \\ \mathsf{b} \to \mathsf{c} & & \mathsf{d} \to \mathsf{e} & \mathsf{f}(x,\mathsf{e}) \to \mathsf{A} & \mathsf{f}(\mathsf{c},x) \to \mathsf{A} \end{array}$$

4 Comparison

The tool AGCP[1] uses rewriting induction to automatically prove ground-confluence of many-sorted TRSs (Aoto & Toyama [1]). In Table 1 we compare FORT 0.2 and AGCP on the 65 left-linear right-ground TRSs from the combined confluence[2] and termination[3] problem databases. These TRSs were presented to AGCP as many-sorted TRSs having exactly one sort. It is interesting to note that there is no difference between confluence and ground-confluence on this database. We used a 60 seconds time limit. Unlike FORT, AGCP is not restricted to left-linear right-ground TRSs. Moreover, AGCP is much faster than FORT. In the near future, we plan to extend FORT to many-sorted TRSs in order to allow a fairer comparison to AGCP.

Acknowledgments Discussions with Bertram Felgenhauer and Vincent van Oostrom helped to improve the paper. The same holds for the remarks by the anonymous reviewers.

References

[1] T. Aoto and Y. Toyama. Ground confluence prover based on rewriting induction. In *Proc. 1st FSCD*, volume 52 of *LIPIcs*, pages 33:1–33:12, 2016. doi: 10.4230/LIPIcs.FSCD.2016.33.

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

[3] M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *Proc. 5th LICS*, pages 242–248, 1990. doi: 10.1109/LICS.1990.113750.

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

[5] F. Rapp and A. Middeldorp. Automating the first-order theory of left-linear right-ground term rewrite systems. In *Proc. 1st FSCD*, volume 52 of *LIPIcs*, pages 36:1–36:12, 2016. doi: 10.4230/LIPIcs.FSCD.2016.36.

[6] H. Zantema. Termination of term rewriting: Interpretation and type elimination. *JSC*, 17(1):23–50, 1994. doi: 10.1006/jsco.1994.1003.

[1] `http://www.nue.ie.niigata-u.ac.jp/tools/agcp/`
[2] `http://cops.uibk.ac.at/`
[3] `http://termination-portal.org/wiki/TPDB`

Ground Confluence Proof with Pattern Complementation*

Takahito Aoto[1] and Yoshihito Toyama[2]

[1] Faculty of Engineering, Niigata University
aoto@ie.niigata-u.ac.jp
[2] RIEC, Tohoku University
toyama@nue.riec.tohoku.ac.jp

Abstract

In (Aoto&Toyama, FSCD 2016), we gave a procedure for proving ground confluence of many-sorted TRSs based on rewriting induction for proving bounded ground convertibility. The procedure needs to find a strongly quasi-reducible terminating set of rules from the given input TRS to make the rewriting induction work. It turns out, however, that such a subset of rewrite rules is often not present in the input TRS. In this note, we propose an improvement of the procedure; in the new procedure, firstly the lack of defining patterns is detected using pattern complementation procedure (Lazrek et al., I&C 1990), and then possible defining rules that can be appended to obtain a strongly quasi-reducible terminating TRS are searched. The new procedure is useful to prove ground confluence of some TRSs automatically which have been failed in the previous procedure.

1 Introduction

A term rewriting system (TRS for short) is ground confluent if all ground terms are confluent. Procedures for proving ground confluence have been studied in e.g. [2, 5, 3, 4]. In [1], a procedure for proving ground confluence of many-sorted TRSs based on rewriting induction, aiming for proving bounded ground convertibility. For making the rewriting induction work, the procedure needs to find a strongly quasi-reducible terminating set of rules from the given input TRS. It turns out, however, that such a subset of rewrite rules is often not present in the input TRS.

In this note, we propose an improvement of the procedure. In our new procedure, firstly the lack of defining patterns is detected using pattern complementation procedure [6]. Then the rewrite rules to define such pattern are searched by combining multiple rewrite steps of the input TRS. Then founded rewrite rules are added to the input TRS to form a strongly quasi-reducible terminating subset of rewrite rules so that the rewriting induction can work on it. We also report a result of preliminary experiment.

2 Preliminaries

We assume basic familiarity with (many-sorted) term rewriting (e.g. [7]).

The transitive reflexive (reflexive, symmetric, reflexive symmetric, equivalence) closure of a relation $\to$ is denoted by $\xrightarrow{*}$ (resp. $\xrightarrow{=}$, $\leftrightarrow$, $\xleftrightarrow{=}$, $\xleftrightarrow{*}$). For any quasi-order $\succsim$, we put $\succ = \succsim \setminus \precsim$ and $\approx = \succsim \cap \precsim$. A quasi-order $\succsim$ is *well-founded* if so is its strict part $\succ$.

Let $\mathcal{S}$ be a set of *sorts*. Each *many-sorted function* f is equipped with its *sort declaration* $f : \alpha_1 \times \cdots \times \alpha_n \to \alpha_0$, where $\alpha_0, \ldots, \alpha_n \in \mathcal{S}$ $(n \geq 0)$. The set of *terms* over the set of many-sorted function symbols $\mathcal{F}$ and the set of variables $\mathcal{V}$ is denoted by $\mathrm{T}(\mathcal{F}, \mathcal{V})$. The set of function symbols (variables) contained in a term t is denoted by $\mathcal{F}(t)$ (resp. $\mathcal{V}(t)$). The set of *ground*

*This work is partially supported by JSPS KAKENHI (Nos. 15K00003, 25280025).

terms over $\mathcal{G} \subseteq \mathcal{F}$ is denoted by $\mathrm{T}(\mathcal{G})$. A *ground substitution* is a mapping from $\mathcal{V}$ to $\mathrm{T}(\mathcal{F})$. A *rewrite relation (quasi-order)* is a relation (resp. quasi-order) on terms *closed under contexts and substitutions*. A rewrite relation (quasi-order) is a *reduction* relation (resp. quasi-order) if it is well-founded. (Indirected) *equations* $l \doteq r$ and $r \doteq l$ are identified. A directed equation $l \to r$ is a *rewrite rule* if $l \notin \mathcal{V}$ and $\mathcal{V}(l) \supseteq \mathcal{V}(r)$ hold. A (many-sorted) *term rewriting system* (*TRS* for short) is a finite set of rewrite rules. The smallest rewrite relation containing $\mathcal{R}$ is denoted by $\to_{\mathcal{R}}$. The set of *critical pairs* of a TRS $\mathcal{R}$ is denoted by $\mathrm{CP}(\mathcal{R})$.

Terms s and t are *joinable* w.r.t. the rewrite relation $\to_{\mathcal{R}}$ (denoted by $s \downarrow_{\mathcal{R}} t$) if $s \xrightarrow{*}_{\mathcal{R}} u$ and $t \xrightarrow{*}_{\mathcal{R}} u$ for some u. A TRS $\mathcal{R}$ is *(ground) confluent* if $s \downarrow_{\mathcal{R}} t$ holds for any (ground) terms s, t such that $u \xrightarrow{*}_{\mathcal{R}} s$ and $u \xrightarrow{*}_{\mathcal{R}} t$ for some (resp. ground) term u. Terms s and t are *ground convertible* if $s\sigma_g \overset{*}{\leftrightarrow}_{\mathcal{R}} t\sigma_g$ holds for any ground substitution σ_g. An equation $s \doteq t$ is an *inductive theorem* of a TRS $\mathcal{R}$, or *inductively valid* in $\mathcal{R}$, if s and t are ground convertible. We write $\mathcal{R} \models_{ind} E$ for a set E of equations (pairs, rewrite rules) if every equation $s \doteq t$ is an inductive theorem for any $s \doteq t$ ($\langle s, t\rangle$, $s \to t$) in E.

We consider a partition of function symbols into the set $\mathcal{D}$ of *defined symbols*, and the set $\mathcal{C}$ of *constructors* i.e. $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$. Terms in $\mathrm{T}(\mathcal{C}, V)$ are *constructor terms*. Then a mapping from $\mathcal{V}$ to $\mathrm{T}(\mathcal{C})$ is called a *ground constructor substitution*. The set of *ground basic terms* is defined by $\mathrm{T_B}(\mathcal{D}, \mathcal{C}) = \{f(c_1, \ldots, c_n) \mid f \in \mathcal{D}, c_i \in \mathrm{T}(\mathcal{C})\}$. A TRS $\mathcal{R}$ is said to be *quasi-reducible* if no ground basic terms are normal. Clearly, if $\mathcal{R}$ is a quasi-reducible terminating TRS then for any ground term s there exists t such that $s \xrightarrow{*} t \in \mathrm{T}(\mathcal{C})$.

3 Ground Confluence Proof by Rewriting Induction

In [1], the authors give a system of rewriting induction for proving ground confluence of many-sorted term rewriting systems. The procedure is described as follows:

GCR Procedure 1

Input: TRS $\mathcal{R}$
Output: `YES` or `MAYBE`

1. Compute (possibly multiple) candidates for the partition $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$ of function symbols.
2. Compute (possibly multiple) candidates for strongly quasi-reducible $\mathcal{R}_0 \subseteq \mathcal{R}$.
3. Find a reduction quasi-order $\succsim$ such that $\mathcal{R}_0 \subseteq \succ$.
4. Run rewriting induction for proving bounded ground convertibility of $\mathrm{CP}(\mathcal{R}_0)$ with $\succsim$.
5. Run rewriting induction for proving $\mathcal{R}_0 \models_{ind} (\mathcal{R} \setminus \mathcal{R}_0)$.
6. Return `YES` if it succeeds in steps 4 and 5, otherwise `MAYBE`.

Note here that strong quasi-reducibility [1] and quasi-reducibility coincide when constructors are free, i.e. $\mathcal{D} = \{l(\epsilon) \mid l \to r \in \mathcal{R}\}$. To make the explanation simple, here after we only consider free constructors.

Proposition 1 ([1])**.** *If GCR Procedure 1 returns* `YES` *then* $\mathcal{R}$ *is ground confluent.*

As indicated above, for the procedure shown to work, it is required that there exists (strongly) quasi-reducible and terminating subset $\mathcal{R}_0 \subseteq \mathcal{R}$. Experiments in [1], however, reveal that there are cases that there does not exist such an $\mathcal{R}_0$.

Example 2 (Cops 128)**.** *Let* $\mathcal{F} = \{\mathsf{plus} : \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Nat}, \mathsf{s} : \mathsf{Nat} \to \mathsf{Nat}, \mathsf{0} : \mathsf{Nat}\}$ *and*

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{plus}(\mathsf{0}, y) & \to & y & (a) \\ \mathsf{plus}(x, \mathsf{s}(y)) & \to & \mathsf{s}(\mathsf{plus}(x, y)) & (b) \\ \mathsf{plus}(x, y) & \to & \mathsf{plus}(y, x) & (c) \end{array} \right\}$$

Then there exists no quasi-reducible and terminating subsets of $\mathcal{R}$*.*

A natural candidate of quasi-reducible terminating $\mathcal{R}_0$ here would be

$$\mathcal{R}_0 = \left\{ \begin{array}{lcll} \mathsf{plus}(\mathsf{0}, y) & \to & y & (a) \\ \mathsf{plus}(\mathsf{s}(x), y) & \to & \mathsf{s}(\mathsf{plus}(x, y)) & (b') \end{array} \right\}$$

Indeed, the rewrite rule (b') is equationally valid as

$$\mathsf{plus}(\mathsf{s}(x), y) \to_{(c)} \mathsf{plus}(y, \mathsf{s}(x)) \to_{(b)} \mathsf{s}(\mathsf{plus}(y, x)) \to_{(c)} \mathsf{s}(\mathsf{plus}(x, y))$$

However, the rewrite rule (b') is not contained in $\mathcal{R}$ and thus the procedure given in [1] fails to prove ground confluence of this system.

4 Ground Confluence Proof with Pattern Complementation

A finite set P of basic terms is called a *pattern*. Intuitively, the set P can be regarded as expressing a set of ground terms given as $\mathrm{Inst}(P) = \{p\sigma_{gc} \mid p \in P, \sigma_{gc} : \mathcal{V} \to \mathrm{T}(\mathcal{C})\}$. A finite set Q of terms is said to be a *complement* (w.r.t. $\mathrm{T_B}(\mathcal{D}, \mathcal{C})$) of P if $\mathrm{Inst}(P) \uplus \mathrm{Inst}(Q) = \mathrm{T_B}(\mathcal{D}, \mathcal{C})$. We denote Q as $\mathrm{T_B}(\mathcal{D}, \mathcal{C}) \ominus P$.

A pattern P is *linear* if so are all its elements. Theorem 1 of [6] gives an algorithm to compute Q from P (complementation algorithm) for any linear pattern P.

Example 3. *Let* $\mathcal{R}$ *be TRS in Example 2. Let* $P_0 = \{\mathsf{plus}(\mathsf{0}, y)\}$ *and* $P_1 = \{\mathsf{plus}(x, \mathsf{s}(y))\}$*. Then* $\mathrm{T_B}(\mathcal{D}, \mathcal{C}) \ominus P_0 = \{\mathsf{plus}(\mathsf{s}(x), y)\}$ *and* $\mathrm{T_B}(\mathcal{D}, \mathcal{C}) \ominus P_1 = \{\mathsf{plus}(x, \mathsf{0})\}$*. Furthermore, we have* $\mathrm{T_B}(\mathcal{D}, \mathcal{C}) \ominus (P_0 \cup P_1) = \{\mathsf{plus}(\mathsf{s}(x), \mathsf{0})\}$*.*

GCR Procedure 2

Input: TRS $\mathcal{R}$
Output: `YES` or `MAYBE`

1. Compute (possibly multiple) candidates for the partition $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$ of function symbols.
2. Find left-linear $\mathcal{R}_0 \subseteq \mathcal{R}$ and a reduction quasi-order $\succsim$ such that $\mathcal{R}_0 \subseteq \succ$.
3. Compute a complement $P = \mathrm{T_B}(\mathcal{D}, \mathcal{C}) \ominus lhs(\mathcal{R}_0)$, where $lhs(\mathcal{R}_0) = \{l \mid l \to r \in \mathcal{R}_0\}$. For each $p \in P$ find p' such that $p \xrightarrow{*}_{\mathcal{R}} p'$ and $p \succ p'$. Let $\mathcal{R}_1 = \mathcal{R}_0 \cup \{p \to p' \mid p \in P\}$.
4. Run rewriting induction for proving bounded ground convertibility of $\mathrm{CP}(\mathcal{R}_1)$ with $\succsim$.
5. Run rewriting induction for proving $\mathcal{R}_1 \models_{ind} (\mathcal{R} \setminus \mathcal{R}_0)$.
6. Return `YES` if it succeeds in steps 4 and 5, otherwise `MAYBE`.

Table 1: Preliminary experiments

problem	added equation(s)	steps		
		#1	#2	#3
Cops 128	$+(\mathsf{s}(x),0) \to \mathsf{s}(x)$	×	✓	✓
Cops 130	$\left\{\begin{array}{l}\mathsf{and3}(\mathsf{F},\mathsf{T},\mathsf{T}) \to \mathsf{F}\\ \mathsf{and3}(\mathsf{F},\mathsf{F},\mathsf{T}) \to \mathsf{F}\\ \mathsf{and3}(\mathsf{T},\mathsf{F},\mathsf{T}) \to \mathsf{F}\end{array}\right\}$	×	×	✓
Cops 133	$+(0,\mathsf{s}(x)) \to \mathsf{s}(x)$	×	✓	✓
Cops 137	$\mathsf{max}(0,\mathsf{s}(y)) \to \mathsf{s}(y)$	×	✓	✓
Cops 140	$+(\mathsf{s}(x),0) \to \mathsf{s}(x)$	×	✓	✓
Cops 146	$+(0,\mathsf{s}(x)) \to \mathsf{s}(x)$	×	✓	✓
Cops 165	$\mathsf{max}(0,\mathsf{s}(y)) \to \mathsf{s}(y)$	×	✓	✓
Cops 174	$+(0,\mathsf{s}(x)) \to \mathsf{s}(x)$	×	✓	✓
Cops 180	$+(\mathsf{s}(x),0) \to \mathsf{s}(x)$	×	✓	✓
Cops 186	$+(0,\mathsf{s}(x)) \to \mathsf{s}(x)$	×	✓	✓
Cops 197	$\mathsf{or}(\mathsf{F},\mathsf{T}) \to \mathsf{T}$	×	✓	✓
Cops 210	$+(\mathsf{s}(x),0) \to \mathsf{s}(x)$	×	✓	✓
Cops 234	$\mathsf{eq}(0,0) \to \mathsf{true}$	✓	✓	✓
	total time (seconds)	32.694	32.620	33.052

Theorem 4. *If GCR Procedure 2 returns* `YES` *then $\mathcal{R}$ is ground confluent.*

Proof. Let $\mathcal{R}' = \mathcal{R} \cup (\mathcal{R}_1 \setminus \mathcal{R}_0)$. Then we have $\to_{\mathcal{R}} \subseteq \to_{\mathcal{R}'} \subseteq \xrightarrow{*}_{\mathcal{R}}$ and thus the ground confluence of $\mathcal{R}$ follows from that of $\mathcal{R}'$. □

Example 5. *Let $\mathcal{R}$ be a TRS given in Example 2. Suppose $\precsim$ be the lpo based on the precedence $\mathsf{plus} \succ \mathsf{s} \succ \mathsf{0}$. Then the GCR Procedure 2 puts $\mathcal{R}_0 = \{(a),(b)\}$ and one has $P = \mathrm{T_B}(\mathcal{D},\mathcal{C}) \ominus lhs(\mathcal{R}_0) = \{\mathsf{plus}(\mathsf{s}(x),0)\}$. By $\mathsf{plus}(\mathsf{s}(x),0) \to \mathsf{plus}(0,\mathsf{s}(x)) \to \mathsf{s}(x)$, one gets $\mathcal{R}_1 = \mathcal{R}_0 \cup \{\mathsf{plus}(\mathsf{s}(x),0) \to \mathsf{s}(x)\}$. Then $\mathrm{CP}(\mathcal{R}_1) = \emptyset$ and one successfully proves $\mathcal{R}_1 \models_{ind} \{(c)\}$ by rewriting induction. Thus $\mathcal{R}$ is proved to be ground confluent.*

5 Implementation and Experiment

A preliminary implementation has been done on AGCP so that when no strongly quasi-reducible subset is found it computes a complement of the defining patterns and adds defining rules. We used rewrite steps of length up to 7 to find p' such that $p \xrightarrow{*}_{\mathcal{R}} p'$ in the Step 3 of the GCR procedure 2. We tested our preliminary implementation on the collection of 121 ground confluence problems given in [1].

We found that 13 new examples can be handled using our preliminary implementation. The summary is presented in Table 1. Here, the column below "steps" shows results when length of rewrite steps to find p' are changed. Here, ✓ shows success and and × shows failure. All these examples are proved by ≤ 3 steps, one needs 3 steps only for Cops 130. Total time indicates the time required for running the prover on the collection of 121 ground confluence problems. Tests are performed on a PC with one 2.50GHz CPU and 4G memory. We impose 5 (resp. 1) seconds time limit rewriting induction proof (resp. computation of constructors). It turns out that changing the length from 1 step to 3 does not affect the total running time. However, with length 7, the total time exceeds 2 minutes and with length 8 we cannot complete the experiment within 10 minutes.

Example 6 (Cops 130)**.** *Let* $\mathcal{F} = \{\mathsf{and3} : \mathsf{Bool} \times \mathsf{Bool} \times \mathsf{Bool} \to \mathsf{Bool}, \mathsf{T} : \mathsf{Bool}, \mathsf{F} : \mathsf{Bool}\}$ *and*

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{and3}(x, y, \mathsf{F}) & \to & \mathsf{F} & (a) \\ \mathsf{and3}(\mathsf{T}, \mathsf{T}, \mathsf{T}) & \to & \mathsf{T} & (b) \\ \mathsf{and3}(x, y, z) & \to & \mathsf{and3}(y, z, x) & (c) \end{array} \right\}$$

Let $\mathcal{D} = \{\mathsf{and3}\}$ *and* $\mathcal{C} = \{\mathsf{T}, \mathsf{F}\}$*. Take* $\mathcal{R}_0 = \{(a), (b)\}$*. Then one obtains* $\mathrm{T_B}(\mathcal{D}, \mathcal{C}) \ominus lhs(\mathcal{R}_0) = \{\mathsf{and3}(\mathsf{F}, \mathsf{T}, \mathsf{T}), \mathsf{and3}(\mathsf{F}, \mathsf{F}, \mathsf{T}), \mathsf{and3}(\mathsf{T}, \mathsf{F}, \mathsf{T})\}$*. Then* $\mathsf{and3}(\mathsf{F}, \mathsf{T}, \mathsf{T}) \to_{\mathcal{R}} \mathsf{and3}(\mathsf{T}, \mathsf{T}, \mathsf{F}) \to_{\mathcal{R}} \mathsf{F}$ *and* $\mathsf{and3}(\mathsf{F}, \mathsf{F}, \mathsf{T}) \to_{\mathcal{R}} \mathsf{and3}(\mathsf{F}, \mathsf{T}, \mathsf{F}) \to_{\mathcal{R}} \mathsf{F}$*. But* $\mathsf{and3}(\mathsf{T}, \mathsf{F}, \mathsf{T}) \to_{\mathcal{R}} \mathsf{and3}(\mathsf{F}, \mathsf{T}, \mathsf{T}) \to_{\mathcal{R}} \mathsf{and3}(\mathsf{T}, \mathsf{T}, \mathsf{F}) \to_{\mathcal{R}} \mathsf{F}$*. Let us consider multiset path ordering with* $\mathsf{and3} \succ \mathsf{T} \succ \mathsf{F}$*. Then considering 2 steps at* $p \xrightarrow{*} p'$ *in the Step 3 of the procedure does not suffice as* $\mathsf{and3}(\mathsf{T}, \mathsf{F}, \mathsf{T}) \not\succ_{mpo} \mathsf{and3}(\mathsf{T}, \mathsf{T}, \mathsf{F})$*. By considering 3 steps at* $p \xrightarrow{*} p'$ *in the Step 3 of the procedure, we obtain a rewrite rule* $\mathsf{and3}(\mathsf{T}, \mathsf{F}, \mathsf{T}) \to \mathsf{F}$ *such that* $\mathsf{and3}(\mathsf{T}, \mathsf{F}, \mathsf{T}) \succ_{mpo} \mathsf{F}$*.*

Sometimes computation of $p \xrightarrow{*} p'$ diverges. The next example illustrates this.

Example 7. *Cops 62 contains the following rewrite rules:*

$$\begin{array}{lcllllll} \mathsf{mod}(0, y) & \to & 0 & (a) & \quad \mathsf{mod}(x, 0) & \to & x & (c) \\ \mathsf{mod}(x, \mathsf{s}(y)) & \to & \mathsf{if}(<(x, \mathsf{s}(y)), x, \mathsf{mod}(-(x, \mathsf{s}(y)), \mathsf{s}(y))) & (b) & & & & \end{array}$$

Then $\mathcal{R}_0 = \{\ldots, (a), (c), \ldots\}$ *and* $(b) \notin \mathcal{R}_0$ *due to ordering restriction. Then* $\mathsf{mod}(\mathsf{s}(x), \mathsf{s}(y)) \in P$*, and the procedure searches some rewrite rule* $\mathsf{mod}(\mathsf{s}(x), \mathsf{s}(y)) \to r$*. However, the set* $\{r \mid \mathsf{mod}(\mathsf{s}(x), \mathsf{s}(y)) \xrightarrow{*}_{\mathcal{R}} r\}$ *is infinite and there is no* r *satisfying* $\mathsf{mod}(\mathsf{s}(x), \mathsf{s}(y)) \succ r$*.*

6 Conclusion

We have shown how the procedure for proving ground confluence of many-sorted TRSs in [1] is improved by constructing new rewrite rules necessary for making the rewriting induction work. We have presented our new procedure and shown its correctness. We have reported on our preliminary implementation and experiment. There are 13 new examples for which ground confluence can be proved from the collection of 121 examples, where the previous procedure can prove 86 problems.

References

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

[2] K. Becker. Proving ground confluence and inductive validity in constructor based equational specifications. In *Proc. of 4th TAPSOFT*, volume 668 of *LNCS*, pages 46–60. Springer-Verlag, 1993.

[3] A. Bouhoula. Simultaneous checking of completeness and ground confluence for algebraic specifications. *ACM Transactions on Computational Logic*, 10(2):20:1–33, 2009.

[4] H. Ganzinger. Ground term confluence in parametric conditional equational specifications. In *Proc. of 4th STACS*, volume 247 of *LNCS*, pages 286–298, 1987.

[5] R. Göbel. Ground confluence. In *Proc. of 2nd RTA*, volume 256 of *LNCS*, pages 156–167, 1987.

[6] A. Lazrek, P. Lescanne, and J. J. Thiel. Tools for proving inductive equalities, relative completeness, and ω-completeness. *Information and Computation*, 84:47–70, 1990.

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

An Algebraic Approach to Confluence and Completion

Cyrille Chenavier

INRIA, équipe πr^2
Laboratoire IRIF
Université Paris-Diderot
chenavier@pps.univ-paris-diderot.fr

Abstract

We propose a functional description of rewriting systems where reduction rules are represented by linear maps called reduction operators. We exhibit a lattice structure on the set of reduction operators. Using this structure we formulate the equivalent notions of confluence and Church-Rosser property. We relate these notions to the classical ones coming from abstract rewriting theory. We also give an algebraic formulation of confluence.

1 Introduction

Reduction operators were introduced by Berger for finite dimensional vector spaces. His motivation was to study the homology of a special class of algebras called *finitely generated homogeneous algebras*. The elements of these algebras are non-commutative polynomials over a finite number of variables modulo the congruence spanned by a set of oriented homogeneous relations having the same degree. By degree, we mean the one induced by the length of non-commutative monomials. The latter are identified with words. Berger considered the linear endomorphism mapping every left-hand side of a rewrite rule to its right-hand side. This is an endomorphism of the vector space spanned by words whose length is the degree of the rewriting rules. The number of variables being finite, this vector space is of finite dimension. It turns out that the endomorphism described previously is a reduction operator. Berger also proved that the set of reduction operators admits a lattice structure. We point out that in order to obtain this structure, he needs to consider finite dimensional vector spaces. He deduces from this structure a lattice formulation of the confluence for homogeneous rewriting systems. Using this point of view, one can study the homological property of *Koszulness* ([1, 2, 3, 8, 6]). For the definition of Koszulness, we refer the reader to [10] and [3].

In the next section, we propose to develop a notion of reduction operator for non-necessarily finite dimensional vector spaces. Our motivation is that we want to use the theory of reduction operators to study non-homogeneous algebras. For such algebras, we do not have any bound for the degree of a word appearing in a rewrite rule. Hence, the operator described in the previous paragraph is an endomorphism of the vector space spanned by all words which is infinite dimensional. We consider vector spaces admitting a basis equipped with a well-founded total order. We introduce a lattice structure on the set of reduction operators associated to such a vector space and deduce an algebraic formulation of the confluence. This formulation generalises the one obtained by Berger for finite sets.

In the last section, we relate our notion of confluence to the classical one coming from rewriting theory. For that, we formulate the notion of Church-Rosser property in terms of reduction operators which turns out to be equivalent to the one of confluence. We also formulate the completion in terms of reduction operators. Classical completion algorithms exist: the Knuth-Bendix completion algorithm [7] for term rewriting or the Buchberger algorithm [4, 5, 9] for Gröbner bases. These algorithms add new rules to a rewriting system to obtain an equivalent one which moreover is convergent. For reduction operators, our purpose is to complete a set F of such operators. A *completion* of F is a confluent set F' containing F which is such that the lower bounds of the sets F and F' are equal. We also study the question of the existence of a completion. To this end, we introduce an operator C^F called the *F-complement* and state that the set $F \cup \left\{C^F\right\}$ is a completion of F.

2 Reduction Operators

2.1 First Definitions

2.1.1. Notations. We denote by $\mathbb{K}$ a commutative field. Throughout the paper, we fix a well-ordered set $(G, <)$. We denote by $\mathbb{K}G$ the vector space spanned by G: the non-zero elements are the finite formal linear combinations of elements of G with coefficients in $\mathbb{K}$. For every $v \in \mathbb{K}G \setminus \{0\}$, there exist a unique finite subset S_v of G, called the *support* of v, and a unique family of non zero scalars $(\lambda_g)_{g\in S_v}$ such that

$$v = \sum_{g\in S_v} \lambda_g g.$$

2.1.2. Partial Order on the Vectors. The order on G being total, for every $v \in \mathbb{K}G$, the set S_v admits a greatest element, written $\lg(v)$. The element $\lg(v)$ is the *leading generator* of v. We extend the order $<$ on G into a partial order on $\mathbb{K}G$ in the following way: we have $u < v$ if $u = 0$ and v is different from 0 or if $\lg(u) < \lg(v)$.

2.1.3. Reduction Operators. A linear endomorphism T of $\mathbb{K}G$ is a *reduction operator relative to* $(G, <)$ if it is idempotent and if for every $g \in G$, we have $T(g) \leq g$. We denote by $\mathbf{RO}(G, <)$ the set of reduction operators relative to $(G, <)$. Given a reduction operator T, a generator g is said to be *T-reduced* if $T(g)$ is equal to g. We denote by $\mathrm{Red}(T)$ the set of T-reduced generators and by $\mathrm{Nred}(T)$ the complement of $\mathrm{Red}(T)$ in G.

2.1.4. Remark. Let T be a reduction operator relative to $(G, <)$. The image of T is equal to $\mathbb{K}\mathrm{Red}(T)$.

2.2 Lattice Structure and Confluence

Our aim is to equip the set $\mathbf{RO}(G, <)$ with a lattice structure. To define it, let $\mathscr{L}(\mathbb{K}G)$ be the set of subspaces of $\mathbb{K}G$. The following proposition extends the one obtained by Berger when G is finite

2.2.1. Proposition. *The map*

$$\begin{aligned} \mathbf{RO}(G, <) &\longrightarrow \mathscr{L}(\mathbb{K}G), \\ T &\longmapsto \ker(T) \end{aligned}$$

is a bijection.

2.2.2. Remark. The hard part of the proof is to show that the restriction of the kernel map to the set of reduction operators is onto. When G is finite and V is a subspace of $\mathbb{K}G$, the gaussian elimination enables us to construct the reduction operator with kernel V. For the non-necessarily finite case, we need to consider an inductive construction to obtain this operator. We point out that when G is finite this inductive construction is not the same algorithm than the gaussian elimination.

2.2.3. Lattice Structure. The application mapping a subspace of $\mathbb{K}G$ to the operator whose kernel is this subspace is written θ. We consider the binary relation on $\mathbf{RO}(G,<)$ defined by

$$T_1 \preceq T_2 \text{ if and only if } \ker(T_2) \subset \ker(T_1).$$

This relation is reflexive and transitive. From Proposition 2.2.1, it is also anti-symmetric. Hence, it is an order relation on $\mathbf{RO}(G,<)$. Let us equip $\mathbf{RO}(G,<)$ with a lattice structure. The lower bound $T_1 \wedge T_2$ and the upper bound $T_1 \vee T_2$ of two elements T_1 and T_2 of $\mathbf{RO}(G,<)$ are defined in the following manner:

$$\begin{cases} T_1 \wedge T_2 = \theta\left(\ker(T_1) + \ker(T_2)\right), \\ T_1 \vee T_2 = \theta\left(\ker(T_1) \cap \ker(T_2)\right). \end{cases}$$

Our aim is to formulate the notion of confluence using this lattice structure. For that, we need the following:

2.2.4. Lemma. *Let T_1 and T_2 be two reduction operators relative to $(G,<)$. Then, we have:*

$$T_1 \preceq T_2 \Longrightarrow \mathrm{Red}(T_1) \subset \mathrm{Red}(T_2).$$

2.2.5. Obstructions. Let F be a subset of $\mathbf{RO}(G,<)$. We let

$$\mathrm{Red}(F) = \bigcap_{T\in F} \mathrm{Red}(T) \text{ and } \wedge F = \theta\left(\sum_{T\in F} \ker(T)\right).$$

For every $T \in F$, we have $\wedge F \preceq T$. Thus, from Lemma 2.2.4, the set $\mathrm{Red}(\wedge F)$ is included in $\mathrm{Red}(T)$ for every $T \in F$, so that it is included in $\mathrm{Red}(F)$. We write

$$\mathrm{Obs}^F_{\mathrm{red}} = \mathrm{Red}(F) \setminus \mathrm{Red}(\wedge F). \tag{1}$$

2.2.6. Confluence. A subset F of $\mathbf{RO}(G,<)$ is said to be *confluent* if $\mathrm{Obs}^F_{\mathrm{red}}$ is the empty set.

3 Rewriting Properties and Completion

3.1 Reduction Operators and Abstract Rewriting

In this section, we explain how our notion of confluence is related to the one coming from rewriting theory. For that, consider the abstract rewriting system $\left(\mathbb{K}G, \underset{F}{\longrightarrow}\right)$ defined by $v \underset{F}{\longrightarrow} v'$ if and only if there exists $T \in F$ such that v does not belong to $\mathrm{im}(T)$ and v' is equal to $T(v)$.

3.1.1. Church-Rosser Property. We denote by $\langle F\rangle$ the submonoid of $(\mathrm{End}(\mathbb{K}G), \circ)$ spanned by F. Let v and v' be two elements of $\mathbb{K}G$. We say that v *rewrites into* v' if there exists $R \in \langle F\rangle$ such that v' is equal to $R(v)$. We say that F has the *Church-Rosser property* if for every $v \in \mathbb{K}G$, v rewrites into $\wedge F(v)$. The following result is the analogous of the Church-Rosser theorem for reduction operators:

3.1.2. Theorem. *A subset of* $\mathbf{RO}(G, <)$ *is confluent if and only if it has the Church-Rosser property.*

3.1.3. Equivalence Relations. We denote by $\xleftrightarrow[F]{*}$ the reflexive transitive symmetric closure of $\xrightarrow[F]{}$. We easily show that we have $v \xleftrightarrow[F]{*} v'$ if and only if $v - v'$ belongs to the kernel of $\wedge F$. We deduce that F has the Church-Rosser property if and only if it is so for $\xrightarrow[F]{}$. From Theorem 3.1.2, we get:

3.1.4. Proposition. *Let F be a subset of* $\mathbf{RO}(G, <)$. *Then, F is confluent if and only if it is so for* $\xrightarrow[F]{}$.

3.1.5. Multi-Set Order. Given an element v of $\mathbb{K}G$, let S_v be the support of v. We introduce the order $<_{\mathrm{mul}}$ on $\mathbb{K}G$ defined in the following way: we have $v <_{\mathrm{mul}} v'$ if for every $g \in S_v$ such that g does not belong to $S_{v'}$, there exists an element $g' \in S_{v'}$ not appearing in S_v, such that $g < g'$.

3.1.6. Obstructions and Abstract Rewriting. For every $v \in \mathbb{K}G$, $\wedge F(v)$ is the smallest element $v' \in \mathbb{K}G$ for $<_{\mathrm{mul}}$ such that $v - v'$ belongs to the kernel of $\wedge F$. Hence, denoting by $[v]$ the class of v for $\xleftrightarrow[F]{*}$, we deduce from Proposition 3.1.4 that $\wedge F(v)$ is the smallest element of $[v]$ for $<_{\mathrm{mul}}$. In particular, $\mathrm{Obs}^F_{\mathrm{red}}$ being the set of elements of G fixed by every element of T but not fixed by $\wedge F$, $\mathbb{K}\mathrm{Obs}^F_{\mathrm{red}}$ is the set of normal forms for $\xrightarrow[F]{}$ which are not minimal in their equivalence classes.

3.2 Completion

In this section, we investigate the notion of completion in terms of reduction operators.

3.2.1. Definition. Let F be a subset of $\mathbf{RO}(G, <)$.

1. A *completion of* F is a subset F' of $\mathbf{RO}(G, <)$, such that
   (a) F' is confluent,
   (b) $F \subset F'$ and $\wedge F' = \wedge F$.
2. A *complement of* F is an element C of $\mathbf{RO}(G, <)$ such that
   (a) $(\wedge F) \wedge C = \wedge F$,
   (b) $\mathrm{Obs}^F_{\mathrm{red}} \subset \mathrm{Nred}(C)$.

   A complement is said to be *minimal* if the inclusion 2b is an equality.

The link between a complement and a completion is the following:

3.2.2. Proposition. *Let $C \in \mathbf{RO}(G,<)$ such that $(\wedge F) \wedge C$ is equal to $\wedge F$. The set $F \cup \{C\}$ is a completion of F if and only if C is a complement of F.*

3.2.3. Remark. The operator $\wedge F$ is a complement of F. However, in general, this complement is not minimal. Our aim is to exhibit a minimal complement.

3.2.4. The F-Complement. Letting $\vee \overline{F} = \theta(\mathbb{K}\mathrm{Red}(F))$, the operator

$$C^F = (\wedge F) \vee \left(\vee \overline{F}\right),$$

is the *F-complement.*

3.2.5. Theorem. *Let F be a subset of $\mathbf{RO}(G,<)$. The F-complement is a minimal complement of F.*

References

[1] Roland Berger. Confluence and Koszulity. *J. Algebra*, 201(1):243–283, 1998.

[2] Roland Berger. Weakly confluent quadratic algebras. *Algebr. Represent. Theory*, 1(3):189–213, 1998.

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

[4] Bruno Buchberger. Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal. *Universitat Innsbruck, Austria, Ph. D. Thesis*, 1965.

[5] Bruno Buchberger. History and basic features of the critical-pair/completion procedure. *J. Symbolic Comput.*, 3(1-2):3–38, 1987. Rewriting techniques and applications (Dijon, 1985).

[6] Cyrille Chenavier. Confluence Algebras and Acyclicity of the Koszul Complex. *Algebr. Represent. Theory*, 19(3):679–711, 2016.

[7] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 263–297. Pergamon, Oxford, 1970.

[8] Benoit Kriegk and Michel Van den Bergh. Representations of non-commutative quantum groups. *Proc. Lond. Math. Soc. (3)*, 110(1):57–82, 2015.

[9] Teo Mora. An introduction to commutative and noncommutative Gröbner bases. *Theoret. Comput. Sci.*, 134(1):131–173, 1994. Second International Colloquium on Words, Languages and Combinatorics (Kyoto, 1992).

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

Decreasing Diagrams: Two Labels Suffice

Jörg Endrullis[1], Jan Willem Klop[1,2], and Roy Overbeek[1]

[1] Vrije Universiteit Amsterdam,
Department of Computer Science,
Amsterdam, the Netherlands
[2] Centrum Wiskunde & Informatica (CWI),
Amsterdam, the Netherlands

Abstract

The decreasing diagrams technique is one of the strongest and most versatile methods for proving confluence of abstract reductions systems. The technique employs a labelling of the steps $\to$ with labels from a well-founded order $(I, <)$ in order to conclude confluence of the underlying unlabelled relation.

Our point of departure was the following natural question: *How does the size of the label set I influence the strength of the decreasing diagrams technique?* In particular, what class of abstract reduction systems can be proven confluent using decreasing diagrams with 1 label, 2 labels, 3 labels, and so on? Surprisingly, we find that 2 labels are sufficient to prove confluence for every abstract rewrite system having the cofinality property, thus in particular every confluent, countable system.

1 Introduction

A binary relation $\to$ is called *confluent* if two co-initial reductions can always be extended to co-final reductions, that is:

$$\forall abc.\ \big(b \twoheadleftarrow a \twoheadrightarrow c \Rightarrow \exists d.\ b \twoheadrightarrow d \twoheadleftarrow c\big)\ .$$

The method of choice for proving confluence is the *decreasing diagrams technique*. The power of decreasing diagrams is witnessed by the fact that many well-known confluence criteria are direct consequences of decreasing diagrams: the lemma of Hindley–Rosen [3, 7], Rosen's request lemma [7], Newman's lemma [6], and Huet's strong confluence lemma [4]. Moreover, Van Oostrom has shown [10] that the decreasing diagrams technique is complete for systems having the cofinality property [8, p. 766]. Thus, in particular for every confluent, countable abstract reduction system, the confluence property can be proven using the decreasing diagrams technique.

What makes the decreasing diagrams technique so powerful? The freedom to label the steps sets decreasing diagrams apart from all other confluence criteria, with the exception of the equally powerful weak diamond property [1, 2] by De Bruijn. This suggests that the power of these techniques arises from the labelling. This naturally leads to the following questions. *Can the size of the label set I serve as a measure of difficulty of a confluence problem? What class of abstract reduction systems can be proven confluent using decreasing diagrams with* 1 *label,* 2 *labels,* 3 *labels and so on?*

More generally, we can define classes DCR_α for every ordinal α as follows.

Definition 1.1. Let *DCR* denote the class of abstract reduction systems whose confluence can be proven using decreasing diagrams. For ordinals α, let DCR_α denote the class for which confluence can be proven with label set $\{\beta \mid \beta < \alpha\}$ ordered by the usual order $<$ on ordinals.

Note that *DCR* implies DCR_α for some ordinal α. The reason is that any partial well-founded order can be transformed into a total well-founded order (thus an ordinal).

Clearly, we have $DCR_\alpha \subseteq DCR_\beta$ whenever $\alpha < \beta$. *But which of these inclusions are strict?* From the completeness proof in [10] it follows that all abstract reduction systems having the *cofinality property*, including all countable systems, belong to DCR_ω.

Surprisingly, we find that all systems with the cofinality property are already in the class DCR_2. In particular, for proving confluence of countable abstract reduction systems it always suffices to label steps with 0 or 1 using the order $0 < 1$.

2 Preliminaries

Let A be a set. For a relation $\to \subseteq A \times A$ we write $\to^*$ or $\twoheadrightarrow$ for its reflexive transitive closure. We write $\equiv$ for the empty step, that is, $\equiv = \{(a,a) \mid a \in A\}$, and we define $\to^{\equiv} = \to \cup \equiv$.

Definition 2.1 (Abstract Reduction System)**.** An *abstract reduction system (ARS)* $\mathcal{A} = (A, \to)$ consists of a non-empty set A together with a binary relation $\to \subseteq A \times A$. For $B \subseteq A$ we define $\mathcal{A}|_B$, the *restriction of* $\mathcal{A}$ *to* B, by $\mathcal{A}|_B = (B, \to \cap (B \times B))$.

Definition 2.2 (Confluence)**.** An ARS $(A, \to)$ is *confluent (CR)* if $\twoheadleftarrow \cdot \twoheadrightarrow \subseteq \twoheadrightarrow \cdot \twoheadleftarrow$, that is, every pair of finite, co-initial rewrite sequences can be joined to a common reduct.

Definition 2.3 (Countable)**.** An ARS $(A, \to)$ is *countable (CNT)* if there exists a surjective function from the set of natural numbers $\mathbb{N}$ to A.

Definition 2.4 (Cofinal Reduction)**.** Let $\mathcal{A} = (A, \to)$ be an ARS. A set $B \subseteq A$ is *cofinal* in $\mathcal{A}$ if for every $a \in A$ we have $a \twoheadrightarrow b$ for some $b \in B$. A finite or infinite reduction sequence $b_0 \to b_1 \to b_2 \to \cdots$ is *cofinal* in $\mathcal{A}$ if the set $B = \{ b_i \mid i \geq 0 \}$ is cofinal in $\mathcal{A}$.

Definition 2.5 (Cofinality Property)**.** An ARS $\mathcal{A} = (A, \to)$ has the *cofinality property (CP)* if for every $a \in A$, there exists a reduction $a \equiv b_0 \to b_1 \to b_2 \to \cdots$ that is cofinal in $\mathcal{A}|_{\{b \mid a \twoheadrightarrow b\}}$.

Lemma 2.6. *Let $\mathcal{A} = (A, \to)$ be a confluent ARS and $a \in A$. If a rewrite sequence is cofinal in $\mathcal{A}|_{\{b \mid a \twoheadrightarrow b\}}$, then it is also cofinal in $\mathcal{A}|_{\{b \mid a \leftrightarrow^* b\}}$.* □

Theorem 2.7 (Klop [5])**.** *Every confluent countable ARS has the cofinality property.* □

Next, we introduce indexed ARSs and the decreasing diagrams technique.

Definition 2.8 (Indexed ARS)**.** An *indexed ARS* $\mathcal{A} = (A, \{\to_\alpha\}_{\alpha \in I})$ consists of a non-empty set A, and a family $\{\to_\alpha\}_{\alpha \in I}$ of relations $\to_\alpha \subseteq A \times A$ indexed by some set I.

For an indexed ARS $\mathcal{A} = (A, \{\to_\alpha\}_{\alpha \in I})$ and a relation $< \subseteq I \times I$, we define

$$\to \;=\; \bigcup_{\alpha \in I} \to_\alpha \qquad\qquad \to_{<\beta} \;=\; \bigcup_{\alpha < \beta} \to_\alpha \qquad\qquad \to_{\leq\beta} \;=\; \bigcup_{\alpha \leq \beta} \to_\alpha$$

Moreover, we use $\to_{<\alpha \cup <\beta}$ as shorthand for $(\to_{<\alpha} \cup \to_{<\beta})$.

Definition 2.9 (Decreasing Church–Rosser [9])**.** An ARS $\mathcal{A} = (A, \rightsquigarrow)$ is called *decreasing Church–Rosser (DCR)* if there exists an ARS $\mathcal{B} = (A, \{\to_\alpha\}_{\alpha \in I})$ indexed by a well-founded partial order $(I, <)$ such that $\rightsquigarrow = \to$ and every peak $c \leftarrow_\beta a \to_\alpha b$ can be joined by reductions of the form shown in Figure 1.

Theorem 2.10 (Decreasing Diagrams – De Bruijn [1] & Van Oostrom [9])**.** *If an ARS is decreasing Church–Rosser, then it is confluent.* □

3 Decreasing Diagrams with Two Labels

In this section we show that two labels suffice for proving confluence using decreasing diagrams for any abstract reduction system having the cofinality property.

Let $\mathcal{A} = (A, \to)$ be an ARS having the cofinality property. Note that, for defining the labelling, we can consider connected components separately. Thus assume that $\mathcal{A}$ consists of a single connected component, that is, for every $a, b \in A$ we have $a \leftrightarrow^* b$. By the cofinality property and Lemma 2.6 there exists a rewrite sequence

$$m_0 \to m_1 \to m_2 \to m_3 \to \cdots$$

that is cofinal in $\mathcal{A}$; we call this rewrite sequence the *main road*. Without loss of generality we may assume that the main road contains no cycles, that is, $m_i \not\equiv m_j$ whenever $i \neq j$.

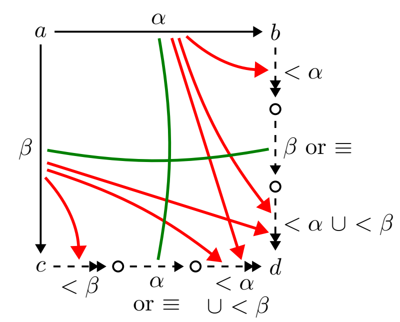


Figure 1: Decreasing elementary diagram; green *lines* stand for weak decrease ($\geq$), red *arrows* for strict decrease ($>$). Furthermore, multiple incoming arrows represent choice, thus weakening the requirements.

The idea of labelling the steps in $\mathcal{A}$ is as follows. For every node $a \in A$, we label precisely one of the outgoing edges with 0 and all others with 1. The edge labelled with 0 must be part of a shortest path from a to the main road. For the case that a lies on the main road, the step labelled 0 must be the step on the main road. This is illustrated in Figure 2.

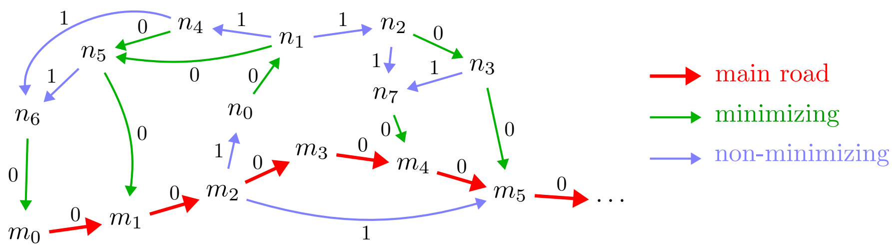


Figure 2: Example labelling.

Note that there is a choice about which edge to label with 0 whenever there are multiple outgoing edges that all start a shortest path to the main road. To resolve this choice, the following definition assumes a well-order $<$ on the universe A, whose existence is guaranteed by the well-ordering theorem. Then, whenever there is a choice, we choose the edge for which the target is minimal in this order.

Remark 3.1. Recall that the Axiom of Choice is equivalent to the well-ordering theorem. In many practical cases, however, the existence of such a well-order does not require the Axiom of Choice. If the universe is countable, then such a well-order can be derived directly from the surjective counting function $f : \mathbb{N} \to A$.

In the following definition we follow the proof in [8, Proposition 14.2.30, p. 766] employing the notion of a cofinal sequence and the rewrite distance from a point to this sequence. While the proof in [8] labels steps by their distance to the target node, we need a more sophisticated labelling.

Definition 3.2. Let $\mathcal{A} = (A, \to)$ be an ARS and $M : m_0 \to m_1 \to m_2 \to \cdots$ be a finite or infinite rewrite sequence in $\mathcal{A}$. For $a, b \in A$, we write

(i) $a \in M$ if $a \equiv m_i$ for some $i \geq 0$, and

(ii) $(a \to b) \in M$ if $a \equiv m_i$ and $b \equiv m_{i+1}$ for some $i \geq 0$.

If M is cofinal in $\mathcal{A}$, we define the *distance* $d(a, M)$ as the least natural number $n \in \mathbb{N}$ such that $a \to^n m$ for some $m \in M$. If M is clear from the context, we write $d(a)$ for $d(a, M)$.

Definition 3.3 (Labelling with Two Labels)**.** Let $\mathcal{A} = (A, \to)$ be an ARS equipped with a well-order $<$ on A such that there exists a cofinal reduction $M : m_0 \to m_1 \to m_2 \to \cdots$ that is free of cycles (that is, for all $i < j$, $m_i \not\equiv m_j$).

We say that a step $a \to b$ is

(i) *on the main road* if $(a \to b) \in M$;

(ii) *minimizing* if $d(a) = d(b) + 1$ and $b' \geq b$ for every $a \to b'$ with $d(b') = d(b)$.

We define an indexed ARS $\mathcal{A}_{\{0,1\}} = (A, \{\to_i\}_{i \in I})$ where $I = \{\, 0, 1 \,\}$ as follows:

$$a \to_0 b \iff a \to b \text{ and this step is on the main road or minimizing}$$
$$a \to_1 b \iff a \to b \text{ and this step is not on the main road and not minimizing}$$

for every $a, b \in A$.

Lemma 3.4. *Let $\mathcal{A} = (A, \to)$ be an ARS with a cofinal rewrite sequence $M : m_0 \to m_1 \to \cdots$ that is free of cycles (that is, for all $i < j$, $m_i \not\equiv m_j$). Furthermore, let $<$ be a well-order over A. Then for $\mathcal{A}_{\{0,1\}} = (A, \to_0, \to_1)$ we have:*

(i) $\to \;=\; \to_0 \cup \to_1$ *;*

(ii) for every $a, b \in M$ we have $a \twoheadrightarrow_0 \cdot \twoheadleftarrow_0 b$;

(iii) for every $a \in A$, there is at most one $b \in A$ such that $a \to_0 b$;

(iv) for every $a \notin M$, there exists $b \in A$ with $a \to_0 b$ and $d(a) > d(b)$;

(v) for every $a \in A$, there exists $m \in M$ such that $a \twoheadrightarrow_0 m$;

(vi) every peak $c \leftarrow_\beta a \to_\alpha b$ can be joined according as in Figure 1.

Proof. Properties (i) and (ii) follow from the definitions.

For (iii) assume that $b \leftarrow_0 a \to_0 c$. We show that $b \equiv c$. The steps $a \to b$ and $a \to c$ are either minimizing or on the main road. We distinguish cases $a \in M$ and $a \notin M$:

(i) Assume that $a \in M$. Then $d(a) = 0$, and thus neither $a \to b$ nor $a \to c$ is a minimizing step. Hence $(a \to b) \in M$ and $(a \to c) \in M$. Since M is free of cycles, we get $b \equiv c$.

(ii) If $a \notin M$, both steps $a \to b$ and $a \to c$ must be minimizing. If $d(b) \neq d(c)$, then we have either $d(a) \neq d(b) + 1$ or $d(a) \neq d(c) + 1$, contradicting minimization. Thus $d(b) = d(c)$. Then by minimization we have $b \geq c$ and $c \geq b$, from which we obtain $b \equiv c$.

For (iv), consider an element $a \notin M$. Let $B = \{b' \mid a \to b' \wedge d(a) = d(b') + 1\}$. By definition of the distance $d(\cdot)$, $B \neq \varnothing$. Define b as the least element of B in the well-order $<$ on A. It follows that $a \to b$ is a minimization step. Hence $a \to_0 b$ and $d(a) > d(b)$. Property (v) follows directly from (iv) using induction on the distance.

For (vi), consider a peak $c \leftarrow_\beta a \to_\alpha b$. If $b \equiv c$, then the joining reductions are empty steps. Thus assume that $b \not\equiv c$. By (iii) we have either $\alpha = 1$ or $\beta = 1$. By (v) there exist $m_b, m_c \in M$ such that $b \twoheadrightarrow_0 m_b$ and $c \twoheadrightarrow_0 m_c$. By (ii) we have $m_b \twoheadrightarrow_0 \cdot \twoheadleftarrow_0 m_c$. Hence $b \twoheadrightarrow_0 \cdot \twoheadleftarrow_0 c$. These joining reductions are of the form required by Figure 1 since $\twoheadrightarrow_0 = \twoheadrightarrow_{<\alpha \cup <\beta}$. ◻

Theorem 3.5. *If an ARS $\mathcal{A} = (A, \to)$ satisfies the cofinality property, then there exists an indexed ARS $(A, (\to_\alpha)_{\alpha \in \{0,1\}})$ such that $\to\, = \,\to_0 \cup \to_1$ and every peak $c \leftarrow_\beta a \to_\alpha b$ can be joined according to the elementary decreasing diagram in Figure 1.*

Proof. It suffices to consider a connected component of $\mathcal{A}$. Let $\mathcal{B} = (B, \to)$ be a connected component of $\mathcal{A}$: we have $a \leftrightarrow^* b$ for all $a, b \in B$. By the cofinality property and Lemma 2.6, there exists a cofinal reduction $m_0 \to m_1 \to \cdots$ in $\mathcal{B}$. By the well-ordering theorem, there exists a well-order $<$ over B. Then $\mathcal{B}$ has the required properties by Lemma 3.4(vi). □

We note that Theorem 3.5 also holds for De Bruijn's weak diamond property. However, when restricting the index set I to a single label, the decreasing diagram technique is equivalent to $\leftarrow \cdot \to \;\subseteq\; \to^{\equiv} \cdot \leftarrow^{\equiv}$, i.e. the *diamond property* for $\to \cup \equiv$, while the weak diamond property with one label is equivalent to *strong confluence* $\leftarrow \cdot \to \;\subseteq\; \to^{\equiv} \cdot \twoheadleftarrow$.

4 Conclusion

We have shown that all abstract reduction systems with the cofinality property (in particular, all confluent, countable systems) can be proven confluent using the decreasing diagrams technique with the almost trivial label set $I = \{0, 1\}$.

This raises the question *whether there is an confluent, uncountable system that needs more than 2 labels to establish confluence using decreasing diagrams?* In other words, is there an uncountable system that is *DCR* but not *DCR*$_2$?

Is there a confluent, uncountable system that is CR but not DCR$_2$*?* It is a long-standing open problem whether the method of decreasing diagrams is complete for proving confluence of uncountable systems [9], that is, whether *CR* implies *DCR*.

In general: *which of the inclusions $DCR_\alpha \subseteq DCR_\beta$ with $\alpha < \beta$ are strict?*

References

[1] N.G. de Bruijn. A Note on Weak Diamond Properties. Memorandum 78–08, Eindhoven University of Technology, 1978.

[2] J. Endrullis and J.W. Klop. De Bruijn's weak diamond property revisited. *Indagationes Mathematicae*, 24(4):1050 – 1072, 2013. In memory of N.G. (Dick) de Bruijn (1918–2012).

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

[4] G.P. Huet. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. *Journal of the ACM*, 27(4):797–821, 1980.

[5] J.W. Klop. *Combinatory Reduction Systems*, volume 127 of *Mathematical centre tracts*. Mathematisch Centrum, 1980.

[6] M.H.A. Newman. On Theories with a Combinatorial Definition of "Equivalence". *Annals of Mathematics*, 42(2):223–243, 1942.

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

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

[9] V. van Oostrom. Confluence by Decreasing Diagrams. *Theoretical Computer Science*, 126(2):259–280, 1994.

[10] V. van Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, Vrije Universiteit Amsterdam, 1994.

Coherence of quasi-terminating decreasing 2-polygraphs

Clément Alleaume and Philippe Malbos

Univ de Lyon, Université Claude Bernard Lyon 1, CNRS UMR 5208,
Institut Camille Jordan, 43 blvd. du 11 novembre 1918, F-69622 Villeurbanne cedex, France
`clement.alleaume@univ-st-etienne.fr`, `malbos@math.univ-lyon1.fr`

Abstract

Craig Squier introduced a combinatorial method based on rewriting in order to describe relations amongst relations for presentations of monoids. From a rewriting system he constructed a 2-dimensional combinatorial complex whose 2-cells are generated by relations induced by the rewriting rules. When the rewriting system is confluent and terminating, he characterized the homotopy of this complex in term of confluence diagrams induced by the critical branchings. In this work, we weaken the termination hypothesis and we prove such a result for quasi-terminating decreasing rewriting systems.

1 Introduction

At the end of the eighties, using a homological argument, Squier showed that there are finitely presented monoids with a decidable word problem that do not admit a finite convergent presentation, [7, 8]. He linked the existence of a finite convergent presentation for a finitely presented monoid to a homological property by showing that the critical branchings of a convergent string rewriting system generate the module of the 2-homological syzygies. Using this homological property he proved that there are finitely presented monoids with a decidable word problem that cannot be presented by a finite convergent string rewriting system. In [9], he linked the existence of a finite convergent presentation to a new finiteness condition of finitely presented monoids, called *finite derivation type* (FDT), that extends the properties of being finitely generated and finitely presented. FDT for a monoid is a finiteness property on a 2-dimensional combinatorial complex associated to a presentation of the monoid.

Squier's 2-dimensional complex. To a rewriting system Σ, Squier associated a 2-dimensional cellular complex $\mathcal{D}(\Sigma)$, defined independently by Kilibarda [5] and Pride [6]. The complex $\mathcal{D}(\Sigma)$ has only one 0-cell, its 1-cells are the strings in the free monoid Σ_1^* and the 2-cells are induced by the rewriting rules $\alpha : u \Rightarrow v$ in Σ_2 and their inverses $\alpha^- : v \Rightarrow u$ in Σ_2^-. There is a 2-cell in $\mathcal{D}(\Sigma)$ between each pair of words with shape wuw' and wvw' such that $\Sigma_2 \sqcup \Sigma_2^-$ contains a relation $u \Rightarrow v$. This 2-dimensional complex is extended with 3-cells, called *Peiffer confluences*, filling all the 2-spheres of the form of the following diagram where $u_1 \Rightarrow u_1'$ and $u_2 \Rightarrow u_2'$ are in $\Sigma_2 \sqcup \Sigma_2^-$ and v a string in Σ_1^*. These 3-cells make homotopic the 2-cells corresponding to the application of rewriting step on non-overlapping strings.

$$\begin{tikzcd}[row sep=small] & vu_1'v'u_2v'' \arrow[dr, Rightarrow] \arrow[dd, phantom, "\Downarrow"] & \\ vu_1v'u_2v'' \arrow[ur, Rightarrow] \arrow[dr, Rightarrow] & & vu_1'v'u_2'v'' \\ & vu_1v'u_2'v'' \arrow[ur, Rightarrow] & \end{tikzcd}$$

Homotopy bases. A *homotopy basis of* Σ is defined as a set Σ_3 of additional 3-cells that makes the complex $\mathcal{D}(\Sigma)$ aspherical, that is any 2-dimensional sphere can be "filled up" by the 3-cells of Σ_3. The presentation Σ is called FDT if it is finite and it admits a finite homotopy basis. Squier proved that the FDT property is an invariant property for finitely presented monoids, that is, if Σ and Υ are two finite presentations of the same monoid, then Σ has FDT if and only if Υ has FDT. Hence, the FDT property is an intrinsic property of finitely presented monoids.

Squier's completion. Given a convergent presentation Σ, Squier showed that it is sufficient to consider one 3-cell filling the confluence diagram induced by each critical branching to get a homotopy basis of Σ. Such a set of 3-cells is called a *family of generating confluences of* Σ. In others words, any diagram defined by two parallel rewriting paths can be filled up by confluence diagrams induced by the critical branchings and by Peiffer confluences. This result corresponds to a homotopical version of Newman's Lemma. In particular, when the presentation is finite, it has finitely many critical branchings, hence a finite family of generating confluences. This is a way to prove that finite convergent presentations are FDT, [9]. Squier used this result to give another proof that there exist finitely presented monoids with a decidable word problem that do not admit a finite convergent presentation.

Squier's completion without termination. After these works, Squier's construction of homotopy bases was applied to solve coherence problems, in particular for monoidal categories [2], Artin monoids [1], or plactic monoids [3]. Squier's construction starts from a convergent presentation. However in many situations it can be improved to compute a homotopy basis from a non terminating presentation. A characterization of a homotopy basis of a non terminating confluent rewriting system in term of confluence diagrams of critical branchings is still an open problem. In this work, we weaken the termination hypothesis and give a construction of homotopy bases for decreasing and quasi-terminating string rewriting systems. As an example, we construct a homotopy basis of the monoid $\mathbf{B}_3^+ = \langle s,t \mid sts = tst \rangle$. This monoid does not admit a finite convergent presentation with only two generators, [4], and Squier's construction can only be used by adding a redundant generator. We show that a construction of a homotopy basis of $\mathbf{B}_3^+$ without addition of generator can however be done with the following confluent and quasi-terminating presentation $\Sigma(\mathbf{B}_3^+) = \langle s,t \mid sts \Rightarrow tst,\ tst \Rightarrow sts \rangle$. We obtain a homotopy basis of the monoid $\mathbf{B}_3^+$ containing five 3-cells. This presentation can be reduced homotopically to obtain an empty homotopy basis. We prove more generally that any decreasing quasi-terminating string rewriting system such that Peiffer confluences are decreasing has a homotopy basis given by a family of elementary loops and generating decreasing confluences of critical branching.

2 Decreasing polygraphs

In this section, we define the notion of decreasing 2-polygraph from the corresponding notion for abstract rewriting systems introduced by van Oostrom in [10] and we recall in this context van Oostrom's theorem showing that decreasingness implies confluence.

2-polygraphs. A 1-*polygraph* Σ is a directed graph made of a set of 0-cells Σ_0, a set of 1-cells Σ_1 and source and target maps $s_0, t_0 : \Sigma_1 \to \Sigma_0$. We denote by Σ_1^* the free category on Σ_1. A *globular extension* of the free category Σ_1^* is a set Σ_2 equipped with two maps $s_1, t_1 : \Sigma_2 \to \Sigma_1^*$ such that, for every β in Σ_2, the pair $(s_1(\beta), t_1(\beta))$ is a 1-*sphere* in the category Σ_1^*, that is, $s_0 s_1(\beta) = s_0 t_1(\beta)$ and $t_0 s_1(\beta) = t_0 t_1(\beta)$. A 2-*polygraph* is a triple $\Sigma = (\Sigma_0, \Sigma_1, \Sigma_2)$, where (Σ_0, Σ_1) is a 1-polygraph and Σ_2 is a globular extension of Σ_1^*, whose elements are called the 2-*cells* of the 2-polygraph. We denote by Σ_2^* the free 2-category on Σ and by $\Sigma_2^\top$ the free $(2,1)$-category on Σ, that is the 2-category in which all the 2-cells are invertible.

Rewriting sequences. A *rewriting step* of Σ is a 2-cell of Σ_2^* of the form $u\varphi v$ where u and v are 1-cells in Σ_1^* and φ is a 2-cell of Σ_2. We denote Σ_{stp} the set of rewriting steps of Σ. For any 1-cells u and v, we say u *rewrites into* v if there is a 2-cell in Σ_2^* from u to v. The 2-polygraph Σ is *terminating* if there is no sequence $(u_n)_{n\in\mathbb{N}}$ such that for each $n \in \mathbb{N}$, there is a rewriting step from the 1-cell u_n to the 1-cell u_{n+1}. The 2-polygraph Σ is *quasi-terminating* if for each sequence $(u_n)_{n\in\mathbb{N}}$ such that for each $n \in \mathbb{N}$ there is a rewriting step from u_n to u_{n+1}, the sequence $(u_n)_{n\in\mathbb{N}}$ contains an infinite number of occurrences of the same 1-cell. Any 2-cell f in the 2-category Σ_2^* can be decomposed as a composite of rewriting steps: $f = f_1 \star_1 \ldots \star_1 f_p$, with f_i in Σ_{stp}. Note that, this decomposition is unique up to Peiffer relations. We define the *support* of the 2-cell f as the multiset, denoted by $\mathrm{Supp}(f)$, consisting of the rules of the rewriting steps occurring in any decomposition of f.

Branchings. A *(finite) branching* of Σ is a pair (f, g) of (finite) rewriting sequences of Σ such that $s_1(f) = s_1(g)$. The branching (f, g) is *local* (resp. *aspherical*) if f and g are in Σ_{stp} (resp. $f = g$). A *Peiffer branching* of Σ is a local branching (fv, ug) with 1-source uv where u, v are 1-cells and f, g are in Σ_{stp}. An *overlapping branching* of Σ is a local branching that is not aspherical or Peiffer. An overlapping branching is called a *critical branching* if its source is minimal. The 2-polygraph Σ is said to be *confluent* if each branching (f, g) of Σ can be completed by two 2-cells $f' : t_1(f) \Rightarrow v$ and $g' : t_1(g) \Rightarrow v$ with a common target.

Loops. A *loop* of Σ is a 2-cell f in Σ_2^* such that $s_1(f) = t_1(f)$. Two loops f and g are *equivalent* if there exists a circular permutation σ such that $f = f_1 \star_1 \ldots \star_1 f_p$ and $g = f_{\sigma(1)} \star_1 \ldots \star_1 f_{\sigma(p)}$. We denote by $\mathcal{L}(f)$ the set of loops of Σ equivalent to the loop f. A loop f of Σ is *minimal* if $f = g \star_1 h \star_1 k$, with h a loop, implies that h is either an identity or equal to f. A loop f is *elementary* if it is minimal and there is no nonidentity loops g and h such that $f = g \star_0 h$.

Labeled 2-polygraphs. A 2-polygraph Σ is *labeled* by a set W if there is a map $\psi : \Sigma_{stp} \longrightarrow W$ that associates to a rewriting step f a *label* $\psi(f)$. Given a rewriting sequence $f = f_1 \star_1 \ldots \star_1 f_k$, we denote by $L^W(f) = \{\psi(f_1), \ldots, \psi(f_k)\ \}$ the set of labels of f. If the set W is provided with a well-founded order $\prec$, we say $(W, \psi, \prec)$ is a *well-founded labeling* of the 2-polygraph Σ.

Decreasing 2-polygraph. Consider a 2-polygraph Σ with a well-founded labelling $(W, \psi, \prec)$. A local branching (f, g) of Σ is *decreasing* if there is a confluence diagram of the following form and such that the following properties hold

$$\begin{tikzcd}
\cdot \arrow[rrr, Rightarrow, "f"] \arrow[ddd, Rightarrow, "g"'] & & & \cdot \arrow[d, Rightarrow, "f'"] \\
& & & \cdot \arrow[d, Rightarrow, "g''"] \\
& & & \cdot \arrow[d, Rightarrow, "h_1"] \\
\cdot \arrow[r, Rightarrow, "g'"'] & \cdot \arrow[r, Rightarrow, "f''"'] & \cdot \arrow[r, Rightarrow, "h_2"'] & \cdot
\end{tikzcd}$$

- **i)** for each $k \in L^W(f')$, we have $k \prec \psi(f)$,
- **ii)** for each $k \in L^W(g')$, we have $k \prec \psi(g)$,
- **iii)** f'' is an identity or a rewriting step labeled by $\psi(f)$,
- **iv)** g'' is an identity or a rewriting step labeled by $\psi(g)$,
- **v)** for each $k \in L^W(h_1) \cup L^W(h_2)$, we have $k \prec \psi(f)$ or $k \prec \psi(g)$.

Such a diagram is then called a *decreasing confluence diagram.* A 2-polygraph Σ is *decreasing* if there exists a well-founded labeling $(W, \psi, \prec)$ of Σ making all its local branching decreasing.

Decreasingness of Peiffer branchings. Given a Peiffer branching $(\alpha v, u\beta)$, we will call *Peiffer confluence* of this branching the confluence diagram on the right. If the 2-polygraph Σ is decreasing, all its Peiffer branchings can be completed into a decreasing confluence diagram. But, the Peiffer confluences for this branching is not necessarily decreasing.

$$\begin{tikzcd}
uv \arrow[r, Rightarrow, "\alpha v"] \arrow[d, Rightarrow, "u\beta"'] & u'v \arrow[d, Rightarrow, "u'\beta"] \\
uv' \arrow[r, Rightarrow, "\alpha v'"'] & u'v'
\end{tikzcd}$$

Example. Any 2-polygraph Σ such that any local branching (f, g) is confluent using two rewriting steps $f' : t_1(f) \Rightarrow v$ and $g' : t_1(g) \Rightarrow v$ is decreasing. An order on Σ_{stp} making all local branchings decreasing is the empty order. In particular, the 2-polygraph $\Sigma(\mathbf{B}_3^+)$ is decreasing and not terminating.

As in the case of abstract rewriting systems, we show the following result.

Theorem 1. *A decreasing 2-polygraph such that Peiffer confluences are decreasing is confluent.*

3 Coherence by decreasingness

In this section, we introduce two globular extensions for 2-polygraphs. The first one is the globular extension of loops, containing for each equivalence class of elementary loop, a 3-cell from an elementary loop to the corresponding identity 2-cell. This globular extension allows us to construct a 3-cell from any loop to an identity 2-cell. The second globular extension we introduce is the extension of generating decreasing confluences, containing for each critical branching of Σ a decreasing confluence diagram. The union of those two globular extensions is called van Oostrom-Squier's completion. We prove that any van Oostrom-Squier's completion of a quasi-terminating 2-polygraph Σ such that all Peiffer confluences are decreasing is a coherent presentation of the category presented by Σ.

Coherent presentations. A *coherent presentation* of a monoid $\mathbf{M}$ is a 2-polygraph Σ presenting $\mathbf{M}$ extended by a globular extension Σ_3 of the free $(2,1)$-category $\Sigma_2^\top$, such that Σ_3 is a homotopy basis. That is, for every 2-sphere (f,g) of $\Sigma_2^\top$, there exists a 3-cell from f to g in the free $(3,1)$-category over the $(3,1)$-polygraph $(\Sigma_2^\top, \Sigma_3)$.

The cellular extension of loops. Let Σ be a 2-polygraph. Let $\mathcal{L}$ be the set of equivalence classes of elementary loops of Σ_2^*. We will denote by $\mathcal{L}(\Sigma)_3$ the globular extension of the $(2,1)$-category $\Sigma_2^\top$ made of a family of 3-cells $A_\alpha : \alpha \Rrightarrow 1_{s_1(\alpha)}$ indexed by exactly one $\alpha \in E$ for each E in $\mathcal{L}$. We will denote by $\mathcal{L}(\Sigma)^\top$ the free $(3,1)$-category over the $(3,1)$-polygraph $(\Sigma, \mathcal{L}(\Sigma)_3)$.

Generating decreasing confluences. Let Σ be a decreasing 2-polygraph for a well-founded labeling $(W, \Psi, \prec)$. A *family of generating decreasing confluences of* Σ is a globular extension of the $(2,1)$-category $\Sigma_2^\top$ that contains, for every critical branching (f,g) of Σ, one 3-cell $D_{f,g}$ of the form on the right and where the confluence diagram $(f \star_1 f', g \star_1 g')$ is decreasing. Any decreasing 2-polygraph admits such a family of generating decreasing confluences. Indeed, any critical branching is local and thus confluent by decreasingness hypothesis. Note that such a family is not unique in general.

$$\begin{tikzcd} & v \arrow[r, Rightarrow, "f'"] \arrow[dd, phantom, "\Downarrow D_{f,g}"] & u' \\ u \arrow[ur, Rightarrow, "f"] \arrow[dr, Rightarrow, "g"'] & & \\ & v' \arrow[uur, Rightarrow, "g'"'] & \end{tikzcd}$$

van Oostrom-Squier's completion. Let Σ be a decreasing 2-polygraph for a well-founded labeling $(W, \Psi, \prec)$. A *van Oostrom-Squier's completion* of Σ is a $(3,1)$-polygraph, denoted by $\mathcal{D}(\Sigma)$, and defined by $\mathcal{D}(\Sigma) = \langle \Sigma \,|\, \mathcal{O}(\Sigma) \cup \mathcal{L}(\Sigma)_3 \rangle$, where $\mathcal{O}(\Sigma)$ is a chosen family of generating decreasing confluences. A van Oostrom-Squier's decreasing completion of the 2-polygraph $\Sigma(\mathbf{B}_3^+)$ is given in Appendix.

Decreasingness from quasi-termination. Let Σ be a confluent and quasi-terminating 2-polygraph. For any 1-cell u, we fix a semi-normal form $\widetilde{u}$ of u, that is a 1-cell $\widetilde{u}$ such that u rewrites into $\widetilde{u}$ and for any 1-cell $\widetilde{u}'$ such that u rewrites into $\widetilde{u}'$, the 1-cell $\widetilde{u}'$ rewrites into $\widetilde{u}$. We call *distance* from u to $\widetilde{u}$, denoted by $d(u, \widetilde{u})$, the length of the shortest rewriting sequence from u to $\widetilde{u}$. We choose $\widetilde{u}$ such that, for any 1-cells v, w and any 1-cells u_1, u_2 that rewrite into $\widetilde{u}$ such that $d(u_1, \widetilde{u}) \geq d(u_2, \widetilde{u})$, we have $d(vu_1w, v\widetilde{u}w) \geq d(vu_2w, v\widetilde{u}w)$. We define a *labeling to the semi-normal form*, labeling SNF for short, $(\psi, \mathbb{N})$ on Σ by setting, for any 2-cell f, $\psi(f) = d(t_1(f), \widetilde{t_1(f)})$. In this way, Σ is decreasing for the labelling ψ. Indeed, for any local branching leading to 1-cells u_1 and u_2, we have chosen a common semi-normal form $\widetilde{u}$. We can choose rewriting paths from u_1 to $\widetilde{u}$ and from u_2 to $\widetilde{u}$ of minimal length. Those paths yield a confluence diagram, decreasing by construction. In particular, a labeling SNF makes all Peiffer branchings decreasing. But, it does not necessarily make the Peiffer confluences decreasing. In particular, it is not the case when the source uv of the Peiffer confluence is already the chosen semi-normal form.

Theorem 2. *Let Σ be a decreasing quasi-terminating 2-polygraph for a labeling SNF such that all Peiffer confluences are decreasing. Any van Oostrom-Squier's completion of Σ is a coherent presentation of the category presented by Σ.*

This theorem does not apply to $\Sigma(\mathbf{B}_3^+)$ because no labeling SNF of $\Sigma(\mathbf{B}_3^+)$ makes all its Peiffer confluences decreasing. However, all Peiffer branchings of $\Sigma(\mathbf{B}_3^+)$ have a decreasing diagram which can be tiled by two loops. This proves that van Oostrom-Squier's completion of $\Sigma(\mathbf{B}_3^+)$ defined above is a homotopy basis of the free $(2,1)$-category $\Sigma(\mathbf{B}_3^+)_2^\top$.

A counterexample without quasi-termination. Quasi-termination is a required condition in Theorem 2. Indeed, consider Σ with no loop and containing two families $(f_j^i)_{i\in\mathbb{N},i\in\mathbb{N},ij=0}$ and $(g_j^i)_{i\in\mathbb{N},i\in\mathbb{N},ij=0}$ of 2-cells such that (see Figure A in Appendix)

- the sequences $(f_n^0)_{n\in\mathbb{N}}$, $(f_0^n)_{n\in\mathbb{N}}$, $(g_n^0)_{n\in\mathbb{N}}$ and $(g_0^n)_{n\in\mathbb{N}}$ are infinite rewriting paths,
- for any odd integer n, we have $t_1(f_0^n) = t_1(g_0^n)$ and $t_1(f_n^0) = t_1(g_n^0)$,
- for any even integer n, we have $t_1(f_0^n) = t_1(f_n^0)$ and $t_1(g_0^n) = t_1(g_n^0)$.

A family of generating confluences containing the 2-sphere $(f_0^0 \star_1 f_0^1, g_0^0 \star_1 g_0^1)$ cannot be used to construct a 3-cell from $f_0^0 \star_1 f_1^0$ to $g_0^0 \star_1 g_1^0$. Indeed, the 2-sphere $(f_0^0 \star_1 f_1^0, g_0^0 \star_1 g_1^0)$ is tiled by an infinite family of 2-spheres containing the 2-sphere $(f_0^0 \star_1 f_0^1, g_0^0 \star_1 g_0^1)$ and all 2-spheres of the form $(f_0^n \star_1 f_0^{n+1}, f_n^0 \star_1 f_{n+1}^0)$ and of the form $(g_0^n \star_1 g_0^{n+1}, g_n^0 \star_1 g_{n+1}^0)$.

This does not make possible to construct a homotopy basis of $\Sigma_2^\top$ by choosing only one generating confluence for each critical branching. The 2-polygraph Σ is not quasi-terminating because the source of the 2-cell f_0^0 does not have any semi-normal form.

Decreasingness from termination. For a convergent 2-polygraph Σ, we define the label Ψ by setting for each rewriting step $\psi = u\varphi v$, $\Psi(\psi)$ is distance from $t_1(\psi)$ to its normal form. This label makes Σ decreasing. Moreover, Σ being terminating it doe not have loop. As a consequence of Theorem 2, we have Squier's Theorem for convergent 2-polygraphs:

Corollary 1 ([9])**.** *Let Σ be a convergent 2-polygraph. Any Squier's completion $\mathcal{S}(\Sigma)$ of Σ is a coherent presentation of the category presented by Σ.*

References

[1] Stéphane Gaussent, Yves Guiraud, and Philippe Malbos. Coherent presentations of Artin monoids. *Compos. Math.*, 151(5):957–998, 2015.

[2] Yves Guiraud and Philippe Malbos. Coherence in monoidal track categories. *Math. Structures Comput. Sci.*, 22(6):931–969, 2012.

[3] Nohra Hage and Philippe Malbos. Coherent presentations of plactic monoids. preprint, 2016.

[4] Deepak Kapur and Paliath Narendran. A finite Thue system with decidable word problem and without equivalent finite canonical system. *Theoret. Comput. Sci.*, 35(2-3):337–344, 1985.

[5] Vesna Kilibarda. On the algebra of semigroup diagrams. *Internat. J. Algebra Comput.*, 7(3):313–338, 1997.

[6] Stephen J. Pride. Low-dimensional homotopy theory for monoids. *Internat. J. Algebra Comput.*, 5(6):631–649, 1995.

[7] Craig Squier and Friedrich Otto. The word problem for finitely presented monoids and finite canonical rewriting systems. In *Rewriting techniques and applications (Bordeaux, 1987)*, volume 256 of *Lecture Notes in Comput. Sci.*, pages 74–82. Springer, Berlin, 1987.

[8] Craig C. Squier. Word problems and a homological finiteness condition for monoids. *J. Pure Appl. Algebra*, 49(1-2):201–217, 1987.

[9] Craig C. Squier, Friedrich Otto, and Yuji Kobayashi. A finiteness condition for rewriting systems. *Theoret. Comput. Sci.*, 131(2):271–294, 1994.

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

Appendix

Example A. Consider the 2-polygraph $\Sigma(\mathbf{B}_3^+)$. It has the following four critical branchings: $(\alpha t, s\beta)$, $(t\alpha, \beta s)$, $(\alpha ts, st\beta)$ and $(ts\alpha, \beta st)$. Each of these critical branchings is confluent using two rewriting steps. Thus, a van Oostrom-Squier's decreasing completion is given by

$$
\begin{tikzcd}[column sep=small, row sep=small]
& tst^2 \arrow[dd, phantom, "\Downarrow D_{\alpha t,s\beta}"] \arrow[dr, Rightarrow, bend left, "\beta t"] & \\
stst \arrow[ur, Rightarrow, bend left, "\alpha t"] \arrow[dr, Rightarrow, bend right, "s\beta"'] & & stst \\
& s^2ts \arrow[ur, Rightarrow, bend right, "s\alpha"'] &
\end{tikzcd}
\quad
\begin{tikzcd}[column sep=small, row sep=small]
& sts^2 \arrow[dd, phantom, "\Downarrow D_{t\alpha,\beta s}"] \arrow[dr, Rightarrow, bend left, "\alpha s"] & \\
tsts \arrow[ur, Rightarrow, bend left, "\beta s"] \arrow[dr, Rightarrow, bend right, "t\alpha"'] & & tsts \\
& t^2st \arrow[ur, Rightarrow, bend right, "t\beta"'] &
\end{tikzcd}
\quad
\begin{tikzcd}[column sep=small, row sep=small]
& tst^2s \arrow[dd, phantom, "\Downarrow D_{\alpha ts,st\beta}"] \arrow[dr, Rightarrow, bend left, "\beta ts"] & \\
ststs \arrow[ur, Rightarrow, bend left, "\alpha ts"] \arrow[dr, Rightarrow, bend right, "st\beta"'] & & ststs \\
& st^2st \arrow[ur, Rightarrow, bend right, "st\alpha"'] &
\end{tikzcd}
\quad
\begin{tikzcd}[column sep=small, row sep=small]
& sts^2t \arrow[dd, phantom, "\Downarrow D_{ts\alpha,\beta st}"] \arrow[dr, Rightarrow, bend left, "\alpha st"] & \\
tstst \arrow[ur, Rightarrow, bend left, "\beta st"] \arrow[dr, Rightarrow, bend right, "ts\alpha"'] & & tstst \\
& ts^2ts \arrow[ur, Rightarrow, bend right, "ts\beta"'] &
\end{tikzcd}
\quad
\begin{tikzcd}[column sep=small, row sep=small]
& tst \arrow[d, phantom, "\Downarrow E_{\alpha\star_1\beta}"] \arrow[dr, Rightarrow, bend left, "\beta"] & \\
sts \arrow[ur, Rightarrow, bend left, "\alpha"] \arrow[rr, equal] & {} & sts
\end{tikzcd}
$$

where $D_{\alpha t,s\beta}$, $D_{t\alpha,\beta s}$, $D_{\alpha ts,st\beta}$ and $D_{ts\alpha,\beta st}$ are the generating decreasing confluences and $E_{\alpha\star_1\beta}$ is an elementary loop of Σ.

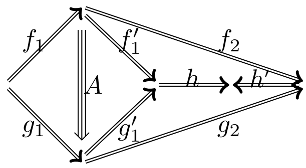
Lemma 1. *Let Σ be a decreasing 2-polygraph. Let b be a loop in Σ_2^*. Then there exists a 3-cell from f to $1_{s_1(f)}$ in $\mathcal{L}(\Sigma)^\top$.*

Proof. Any elementary loop f is equivalent to an elementary loop e such that the 3-cell $e \Rrightarrow 1_{s_1(e)}$ is in $\mathcal{L}(\Sigma)^\top$. As a consequence, there exists a 3-cell from f to $1_{s_1(f)}$ in $\mathcal{L}(\Sigma)^\top$. If f is minimal, f is a 0-composition of elementary loops and identities. As a consequence, there exists a 3-cell from f to $1_{s_1(f)}$ in $\mathcal{L}(\Sigma)^\top$. In the general case, a non identity loop f can be written as $f_1 \star_1 f' \star_1 f_2$ where f' is a minimal loop and f_1 and f_2 are 2-cells such that $f_1 \star_1 f_2$ is a loop. Thus, there exist a 3-cell from f to $f_1 \star_1 f_2$. The support of $f_1 \star_1 f_2$ is strictly included in the support of f. This proves the lemma by induction on the support of f. $\square$

Lemma 2. *Let Σ be a quasi-terminating 2-polygraph decreasing for a labeling SNF such that all Peiffer confluences are decreasing. Let $\mathcal{D}(\Sigma)$ be a van Oostrom-Squier's completion of Σ associated. For any 2-sphere (f,g) in Σ_2^*, there exists a 3-cell from f to g in $\mathcal{D}(\Sigma)^\top$.*

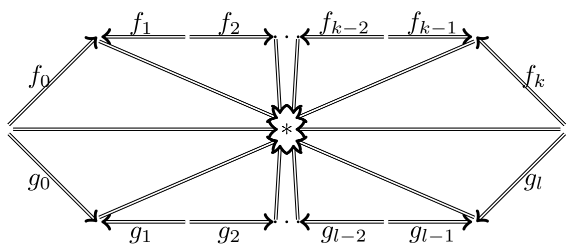
Proof. We proceed in two steps.
Step 1. We prove that, for every local branching $(f,g) : u \Rightarrow (v,v')$ of Σ, there exist 2-cells $f' : v \Rightarrow u'$ and $g' : v' \Rightarrow u'$ in Σ_2^* and a 3-cell $A : f \star_1 f' \Rrightarrow g \star_1 g'$ in $\mathcal{D}(\Sigma)^\top$. In the case of an aspherical or Peiffer branching, we can choose f' and g' such that $f \star_1 f' = g \star_1 g'$ holds in Σ_2^* and A is an identity 3-cell. If (f,g) is an overlapping branching that is not critical, we have $(f,g) = (whw', wkw')$ with (h,k) a critical branching. We consider the 3-cell $D_{h,k} : h \star_1 h' \Rrightarrow k \star_1 k'$ of $\mathcal{O}(\Sigma)$ corresponding to the generating decreasing confluence of the critical branching (h,k). Let define the 2-cells $f' = wh'w'$ and $g' = wk'w'$ and the 3-cell $A = wD_{h,k}w'$. The 2-polygraph Σ having a labeling SNF, the confluence diagram corresponding to the 3-cell A is decreasing.
Step 2. Let (f,g) be a 2-sphere in Σ_2^*. This 2-sphere defines a branching on $s_1(f) = s_1(g)$. We prove the lemma by well-founded induction on the measure $|(f,g)|$ of the branching (f,g), defined in the next paragraph. If f or g is an identity 2-cell, say $g = 1$, the 2-cell f is a loop. By Lemma 1, there exists a 3-cell $E : f \Rrightarrow 1_{s_1(f)}$ obtained by composition of 3-cells of $\mathcal{L}(\Sigma)_3$. Else, we have decompositions $f = f_1 \star_1 f_2$ and $g = g_1 \star_1 g_2$, where (f_1, g_1) is a local branching. Note that f_2 or g_2 can be equal to an identity 2-cell. The branching (f_1, g_1) is confluent by decreasingness. By Step 1, there exists a 3-cell $A : f_1 \star_1 f_1' \Rrightarrow g_1 \star_1 g_1'$ in $\mathcal{D}(\Sigma)^\top$ where the confluence diagram $(f_1 \star_1 f_1', g_1 \star_1 g_1')$ is decreasing. Peiffer confluences being decreasing, this diagram is decreasing even if (f_1, g_1) is a Peiffer branching. The branching (f_1', f_2) is confluent

by decreasingness, hence there exist 2-cells h and h' as indicated in the following diagram:



By the following Lemma 3, we have $|(f_1', f_2)| \prec |(f, g_1)|$. Moreover, by the following Lemma 4, we have $|(f, g_1)| = |(f_1, g_1)| = |(f, g)|$, hence $|(f_1', f_2)| \prec |(f, g)|$, where $\prec$ is the order on measures of branchings. Hence by induction hypothesis, there exists a 3-cell $B : f_1' \star_1 h \Rrightarrow f_2 \star_1 h'$ in $\mathcal{D}(\Sigma)^\top$. In the same way, we prove that there exists a 3-cell $C : g_1' \star_1 h \Rrightarrow g_2 \star_1 h'$ in $\mathcal{D}(\Sigma)^\top$. This concludes the induction and proves that there is a 3-cell from f to g. $\square$

Proof of Theorem 2. Let (f, g) be a 2-sphere of $\Sigma_2^\top$. By definition of $\Sigma_2^\top$, the 2-cell $f \star_1 g^-$ can be decomposed into a zigzag



where the 2-cells $f_0, \ldots, f_k, g_0, \ldots, g_l$ are 2-cells in Σ_2^*. Note that some of those 2-cells can be identities. By confluence of Σ, this 2-sphere can be filled up by a family of 2-spheres of Σ_2^*. By Lemma 2, these 2-spheres can be filled up by 3-cells of $\mathcal{D}(\Sigma)^\top$ whose the composition gives a 3-cell of $\mathcal{D}(\Sigma)^\top$ from f to g.

Measure of 2-cells and branchings. Let Σ be a 2-polygraph with a well-founded labeling $(W, \psi, \prec)$. Consider i in W and a 1-cell $w = w_1 \ldots w_n$ in the free monoid W^*, with w_i in W. We denote by $w^{\geq i}$ the 1-cell w written without the 1-cells labeled by j such that $j \prec i$, that is

$$w^{\geq i} = w_{\overline{1}} \ldots w_{\overline{n}},$$

where $w_{\overline{k}} = w_k$ if $\psi(w_k) \prec i$ and $w_{\overline{k}} = 1$ else. Given a 1-cell w' in W^*, we denote by $w^{(w')}$ the 1-cell defined by

$$w^{(w')} = \overline{w}_1 \ldots \overline{w}_n$$

such that for each $0 \leqslant k \leqslant n$, we have $\overline{w}_k = 1$ if the label i_k of w_k verifies $i_k \prec j$ for some $j \in L^W(w')$ and $\overline{w}_k = w_k$ otherwise.

Let Σ be a decreasing 2-polygraph for a well-founded labeling $(W, \Psi, \prec)$. Following [10], we consider the measure $|\cdot|$ from the free monoid W^* to the multiset $\mathrm{Mult}(W)$ over W defined as follows:

i) if 1 is the empty word of the free monoid W^*, then $|1|$ is the empty multiset,

ii) for every i in W, the multiset $|i|$ is the singleton $\{i\}$,

iii) for every i in W and every 1-cell w in W^*, twe have $|iw| = |i| \cup |w^{(i)}|$.

The measure $|\cdot|$ is extended to the set of finite rewriting sequences of Σ by setting, for any rewriting sequence $f_1 \star_1 \ldots \star_1 f_n$, with f_i labeled by k_i, for all i, we have

$$|f_1 \ldots f_n| = |k_1 \ldots k_n|,$$

were the $k_1 \ldots k_n$ is a product in the monoid W^*. Finally, the measure $|\cdot|$ is extended to the set of finite branchings (f, g) of Σ, by setting

$$|(f, g)| = |f| \cup |g|.$$

Note that for every words w_1 and w_2 in W^*, we have: $|w_1 w_2| = |w_1| \cup |w_2^{(w_1)}|$.

We define a strict order $\prec'$ on the multisets over W. For any multisets M and N, we define $M \prec' N$ if there exist multisets X, Y and Z such that:

$$M = Z \cup X, \qquad N = Z \cup Y, \qquad Y \text{ is not empty},$$

and for every i in W such that $M(i) \neq 0$, there exists j in W such that $N(j) \neq 0$ and $i \prec j$. The order $\prec'$ is well-founded because $\prec$ is. We call $\preccurlyeq'$ the symmetric closure of $\prec'$.

Lemma 3 ([10], Lemma 3.6.)**.** *Let Σ be a decreasing 2-polygraph. For every diagram in Σ_2^**

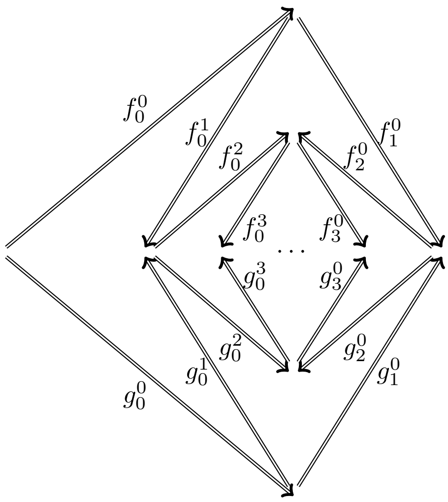
$$\begin{tikzcd} {} \arrow[r, Rightarrow, "\gamma_1"] \arrow[d, Rightarrow, "\delta_0"'] & {} \arrow[r, Rightarrow, "\gamma_2"] \arrow[d, Rightarrow, "\delta_1"] & {} \\ {} \arrow[r, Rightarrow, "\tau"'] & {} \end{tikzcd}$$

such that $|\gamma_1 \star_2 \delta_1| \preccurlyeq' |\delta_0, \gamma_1|$ and γ_1 is not an identity, we have $|\delta_1, \gamma_2| \prec' |\delta_0, \gamma_1 \star_2 \gamma_2|$.

Lemma 4. *Let Σ be a decreasing quasi-terminating 2-polygraph with the labeling SNF. Then, for all branchings (f_1, g_1) and all decreasing confluence diagrams leading to the semi-normal form, we have $|(f_1, g_1)| = |(f_1 \star_1 f_1', g_1 \star_1 g_1')|$.*

Proof. All labels k of the rewriting sequence f_1' verify $k \prec \psi(f_1)$. All labels k of the rewriting sequence g_1' verify $k \prec \psi(g_1)$. Thus, $|f_1 \star_1 f_1'| = |f_1|$ and $|g_1 \star_1 g_1'| = |g_1|$. This implies $|(f_1, g_1)| = |(f_1 \star_1 f_1', g_1 \star_1 g_1')|$. ☐

Figure A.



A Short Mechanized Proof of the Church-Rosser Theorem by the Z-property for the $\lambda\beta$-calculus in Nominal Isabelle*

Julian Nagele, Vincent van Oostrom, and Christian Sternagel

University of Innsbruck, Austria
`{julian.nagele,vincent.van-oostrom,christian.sternagel}@uibk.ac.at`

Abstract

We present a short proof of the Church-Rosser property for the lambda-calculus enjoying two distinguishing features: firstly, it employs the Z-property, resulting in a short and elegant proof; and secondly, it is formalized in the nominal higher-order logic available for the proof assistant Isabelle/HOL.

1 Introduction

Dehornoy proved confluence for the rule of self-distributivity $xyz \to xz(yz)$[1] by means of a novel method [3], the idea being to give a map $\bullet$ that is *monotonic* with respect to $\to^*$ and that yields for each object an *upper bound* on all objects reachable from it in a single step. Later, this method was extracted and dubbed the Z-property [4], and applied to prove confluence of various rewrite systems, in particular the $\lambda\beta$-calculus.

Here we present our Isabelle/HOL [8] formalization of part of the above mentioned work,[2] in particular that the $\lambda\beta$-calculus is confluent since it enjoys the Z-property and that the latter property is equivalent to an abstract version of Takahashi's confluence method [10]. We achieve a rigorous treatment of terms modulo α-equivalence by employing Nominal Isabelle [12], a nominal higher-order logic based on Isabelle/HOL. Our formalization is available from the *archive of formal proofs* [5]. Below, Isabelle code-snippets are in blue and hyperlinked.

2 Nominal λ-terms

In our formalization λ-terms are represented by the following nominal data type, where the annotation "**binds** x **in** t" indicates that the equality of such abstraction terms is up to renaming of x in t:

```
nominal_datatype term =
  Var name
| App term term
| Abs x::name t::term binds x in t
```

For the sake of readability we will use standard notation, i.e., x instead of *Var* x, $s\ t$ instead of *App* $s\ t$, and $\lambda x.\, t$ instead of *Abs* $x\ t$, in the remainder. When defining (recursive) functions on λ-terms, we may have to take care of so-called *freshness constraints*. A freshness constraint is written $x \sharp t$ and states that x does not occur in t, or equivalently, x is fresh for t.

Definition 1. Capture-avoiding substitution is defined recursively by the following equations:

*This work was partially supported by FWF (Austrian Science Fund) projects P27502 and P27528.

[1] Confluence of this single-rule term rewrite system is non-trivial: presently no tool can prove it automatically.

[2] The formalization follows the pen-and-paper proof *exactly*, except for one mistake in Lemma 9 (Rhs).

$y\,[x := s] = (\textbf{if } x = y \textbf{ then } s \textbf{ else } y)$
$(t\ u)\,[x := s] = t\,[x := s]\ u\,[x := s]$
$y \sharp (x,\ s) \Longrightarrow (\lambda y.\ t)\,[x := s] = \lambda y.\ t\,[x := s]$

Due to the constraint, the final equation is only applicable when y is fresh for x and s.

In principle it is always possible to rename variables in terms (or any finitely supported structure) apart from a given finite collection of variables. In order to relieve the user of doing so by hand, Nominal Isabelle [12] provides infrastructure for defining nominal functions, giving rise to strong induction principles that take care of appropriate renaming. (However, nominal functions do not come for free: after stating the defining equations, we are faced with proof obligations that ensure pattern-completeness, termination, equivariance, and well-definedness. With the help of some home-brewed Eisbach [6] methods we were able to handle those obligations automatically.) We first consider the Substitution Lemma, cf. [2, Lemma 2.1.16].

Lemma 2. $x \sharp (y,\ u) \Longrightarrow t\,[x := s]\,[y := u] = t\,[y := u]\,[x := s\,[y := u]]$

Proof. In principle the proof proceeds by induction on t. However, for the case of λ-abstractions we additionally want the bound variable to be fresh for s, u, x, and y. With Nominal Isabelle it is enough to indicate that the variables of those terms should be avoided in order to obtain appropriately renamed bound variables. We will not mention this fact again in future proofs.

- In the base case $t = z$ for some variable z. If $z = x$ then $t\,[x := s]\,[y := u] = s\,[y := u]$ and $t\,[y := u]\,[x := s\,[y := u]] = s\,[y := u]$, since then $z \neq y$ and thus $z\,[y := u] = z$. Otherwise $z \neq x$. Now if $z = y$, then $t\,[x := s]\,[y := u] = u$ and $t\,[y := u]\,[x := s\,[y := u]] = u$, since $x \sharp u$. If $z \neq y$ then both ends of the equation reduce to z and we are done.
- In case of an application, we conclude by definition and twice the IH.
- Now for the interesting case. Let $t = \lambda z.\ v$ such that $z \sharp (s, u, x, y)$. Then

$$\begin{aligned}(\lambda z.\ v)\,[x := s]\,[y := u] &= \lambda z.\ v\,[x := s]\,[y := u] && \text{since } z \sharp (s, u, x, y)\\ &= \lambda z.\ v\,[y := u]\,[x := s\,[y := u]] && \text{by IH}\\ &= (\lambda z.\ v)\,[y := u]\,[x := s\,[y := u]] && \text{since } z \sharp (s\,[y := u], u, x, y)\end{aligned}$$

  where in the last step $z \sharp s\,[y := u]$ follows from $z \sharp (s, u, y)$ by induction on s. ☐

Definition 3. We define β-reduction inductively by the *compatible* closure [2, Definition 3.1.4] of the β-rule (in its nominal version):

$x \sharp t \Longrightarrow (\lambda x.\ s)\ t \rightarrow_\beta s\,[x := t]$

The freshness constraint on the β-rule is needed to obtain an induction principle strong enough with respect to avoiding capture of bound variables. The following standard "congruence properties" (cf. [2, Lemma 3.1.6 and Proposition 3.1.16]) will be used freely in the remainder:

$s \rightarrow_\beta^* t \Longrightarrow u \rightarrow_\beta^* v \Longrightarrow s\ u \rightarrow_\beta^* t\ v$
$s \rightarrow_\beta^* t \Longrightarrow \lambda x.\ s \rightarrow_\beta^* \lambda x.\ t$
$s \rightarrow_\beta^* s' \Longrightarrow t \rightarrow_\beta^* t' \Longrightarrow t\,[x := s] \rightarrow_\beta^* t'\,[x := s']$

They are proven along the lines of their textbook proofs, the first two by induction on the length and the last one by (nominal) induction on t followed by a nested (nominal) induction on the definition of β-steps, using the Substitution Lemma. Furthermore we will make use of the easily proven fact that β-reduction is *coherent* with abstraction:

$\lambda x.\ s \rightarrow_\beta^* t \Longrightarrow \exists\, u.\ t = \lambda x.\ u \wedge s \rightarrow_\beta^* u$

3 Z

We present the Z-property for abstract rewriting, show that it implies confluence, and then instantiate it for the case of (nominal) λ-terms modulo α equipped with β-reduction.

Definition 4. A relation $\to$ on A has the *Z*-property if there is a map $\bullet : A \to A$ such that $a \to b \Longrightarrow b \to^* a^\bullet \wedge a^\bullet \to^* b^\bullet$.

If $\to$ has the Z-property then it indeed is monotonic, i.e., $a \to^* b$ implies $a^\bullet \to^* b^\bullet$, which is straightforward to show by induction on the length of the former.

Lemma 5. *A relation that has the Z-property is confluent.*

Proof. We show semi-confluence [1]. So assume $a \to^* c$ and $a \to d$. We show $d \downarrow c$ by case analysis on the reduction from a to c. If it is empty there is nothing to show. Otherwise there is a b with $a \to^* b$ and $b \to c$. Then by monotonicity we have $a^\bullet \to^* b^\bullet$. From $a \to d$ we have $d \to^* a^\bullet$ using the Z-property, so in total $d \to^* b^\bullet$. Since by applying the Z-property to $b \to c$ we also get $c \to^* b^\bullet$ we have $d \downarrow c$ as desired. ☐

There are two natural choices for functions on λ-terms that yield the Z-property for $\to_\beta$, namely the full-development function and the full-superdevelopment function. The former maps a term to the result of contracting all residuals of redexes in it [2, Definition 13.2.7] and the latter also contracts the upward-created redexes, cf. [9, Section 2.7]. While Dehornoy and van Oostrom developed both proofs [4], here we opt for the latter, which requires less case analysis.

Definition 6. We first define a variant of *App* with built-in β-reduction at the root:

$$x \sharp u \Longrightarrow (\lambda x.\, s') \cdot_\beta u = s'[x := u]$$
$$x \cdot_\beta u = x\ u$$
$$(s\ t) \cdot_\beta u = s\ t\ u$$

An easy case analysis on the first argument shows that this function satisfies the congruence-like property $s \to_\beta^* s' \Longrightarrow t \to_\beta^* t' \Longrightarrow s \cdot_\beta t \to_\beta^* s' \cdot_\beta t'$.

Definition 7. The full-superdevelopment function $\bullet$ on λ-terms is defined as follows:

$$x^\bullet = x$$
$$(\lambda x.\, t)^\bullet = \lambda x.\, t^\bullet$$
$$(s\ t)^\bullet = s^\bullet \cdot_\beta t^\bullet$$

Below, we freely use the fact that $s^\bullet\ t^\bullet \to_\beta^= (s\ t)^\bullet$, which is shown by considering whether or not $s^\bullet$ is an abstraction. The structure of the proof that the $\lambda\beta$-calculus has the Z-property follows that for self-distributivity in that it build on the Self- and Rhs-properties. The former expresses that each term *self*-expands to its full-superdevelopment, and the latter that applying $\bullet$ to the *right-hand side* of the β-rule, i.e., to the result of a substitution, "does more" than applying the map to its components first. Each is proven by structural induction.

Lemma 8 (Self)**.** *For all terms t we have $t \to_\beta^* t^\bullet$.*

Proof. By induction on t using an additional case analysis on $t_1^\bullet$ in the case that $t = t_1\ t_2$. ☐

Lemma 9 (Rhs)**.** *For all terms t, s and all variables x we have $t^\bullet\,[x := s^\bullet] \to_\beta^* t\,[x := s]^\bullet$.*

Proof. By induction on t. The cases $t = x$ and $t = \lambda y.\, t'$ are straightforward. If $t = t_1\ t_2$ we continue by case analysis on $t_1^\bullet$.

If $t_1^\bullet = \lambda y.\, u$ then $\lambda y.\, u\,[x := s^\bullet] = t_1^\bullet\,[x := s^\bullet] \to_\beta^* t_1\,[x := s]^\bullet$ by induction hypothesis. Then, using coherence of β-reduction with abstraction, we can obtain a term v with $t_1\,[x := s]^\bullet = \lambda y.\, v$ and $u\,[x := s^\bullet] \to_\beta^* v$. We then have $(t_1\ t_2)^\bullet\,[x := s^\bullet] = u\,[y := t_2^\bullet]\,[x := s^\bullet] = u\,[x := s^\bullet]\,[y := t_2^\bullet\,[x := s^\bullet]]$, using the substitution lemma in the last step. Together with $u\,[x := s^\bullet] \to_\beta^* v$ and the induction hypothesis for t_2 this yields $(t_1\ t_2)^\bullet\,[x := s^\bullet] \to_\beta^* v\,[y := t_2\,[x := s]^\bullet]$. Since we also have $(t_1\ t_2)\,[x := s]^\bullet = (t_1\,[x := s]\ t_2\,[x := s])^\bullet = v\,[y := t_2\,[x := s]^\bullet]$ we can conclude this case.

If $t_1^\bullet$ is not an abstraction. then from the induction hypothesis we have $(t_1\ t_2)^\bullet\,[x := s^\bullet] = t_1^\bullet\,[x := s^\bullet]\ t_2^\bullet\,[x := s^\bullet] \to_\beta^* t_1\,[x := s]^\bullet\ t_2\,[x := s]^\bullet \to_\beta^= (t_1\ t_2)\,[x := s]^\bullet$. ☐

Lemma 10 (Z)**.** *The full-superdevelopment function $\bullet$ yields the Z-property for $\to_\beta$, i.e., we have $s \to_\beta t \Longrightarrow t \to_\beta^* s^\bullet \wedge s^\bullet \to_\beta^* t^\bullet$ for all terms s and t.*

Proof. Assume $s \to_\beta t$. We continue by induction on the derivation of $\to_\beta$.

If $s \to_\beta t$ is a root step then $s = (\lambda x.\, s')\ t'$ and $t = s'\,[x := t']$ for some s' and t'. Then $s^\bullet = s'^\bullet\,[x := t'^\bullet]$ and thus $t \to_\beta^* s^\bullet$ using Lemma 8 twice, so $s^\bullet \to_\beta^* t^\bullet$ by Lemma 9.

The case where the step happens below an abstraction follows from the induction hypothesis.

If the step happens in the left argument of an application then $s = s'\ u$ and $t = t'\ u$. From the induction hypothesis and Lemma 8 we have $t'\ u \to_\beta^* s'^\bullet\ u^\bullet \to_\beta^= (s\ u)^\bullet$. That also $(s'\ u)^\bullet \to_\beta^* (t'\ u)^\bullet$ follows directly from the induction hypothesis. The case where the step happens in the right argument of an application is symmetric. ☐

4 Perspective

This note originated from the bold and vague claim of Dehornoy and van Oostrom [4] that the confluence proof for the $\lambda\beta$-calculus by establishing the Z-property for the full-superdevelopment map, is *the shortest*. We present a brief qualitative and quantitative analysis of this claim.

Three major methods in the literature for showing confluence of the $\lambda\beta$-calculus are:

$$\text{complete developments} \models \diamond \Longrightarrow \text{complete, full-developments} \models \angle \Longleftarrow \text{full-developments} \models \mathrm{Z}$$

From left to right, that complete developments have the diamond ($\diamond$) property is due to Tait and Martin–Löf [2, Section 3.2], that complete developments have the angle ($\angle$) property with respect to the full-development function is due to Takahashi [10] (cf. [11, Proposition 1.1.11]), and that full-developments have the Z-property is due to [4]. From the fact that the second method needs the concepts of both the others, it stands to reason that its formalization is not the shortest, as confirmed by a formalization of Nipkow [7] and our quantitative analysis below.

Our proof varies on the above picture along yet another dimension, replacing developments (due to Church and Rosser, cf. [2, Definition 11.2.11]) by superdevelopments (due to Aczel, cf. [9, Section 2.7]). Where full-developments give a "tight" upper bound on the single-step reducts of a given term, full-superdevelopments do not, and one may hope for a simplification of the analysis because of it. This is confirmed by our quantitative analysis below. One may vary along this dimension as well: *any* map $\bullet$ having the Z-property suffices as we show now.

Definition 11. A relation $\to$ on A has the *angle* property for a map $\bullet$ from A to A, and relation $\twoheadrightarrow\!\!\circ$ on A, if $\to \subseteq \twoheadrightarrow\!\!\circ \subseteq \to^*$ and $a \twoheadrightarrow\!\!\circ\, b$ implies $b \twoheadrightarrow\!\!\circ\, a^\bullet$.

Lemma 12. *A relation $\to$ has the Z-property for map $\bullet$ if and only if it has the angle property for map $\bullet$ and some relation $\twoheadrightarrow\!\!\circ$.*

Proof. First assume that $\mathrel{\circ\!\!\!\!\rightarrow}$ has the angle property for map $\bullet$ and relation $\mathrel{\circ\!\!\!\!\rightarrow}$. To show that $\rightarrow$ has Z assume $a \rightarrow b$. Then by assumption we also have $a \mathrel{\circ\!\!\!\!\rightarrow} b$ and hence $b \mathrel{\circ\!\!\!\!\rightarrow} a^\bullet$ and $a^\bullet \mathrel{\circ\!\!\!\!\rightarrow} b^\bullet$, by applying the angle property twice, which together with $\mathrel{\circ\!\!\!\!\rightarrow} \subseteq \rightarrow^*$ yields Z.

Now assume $\rightarrow$ has the Z-property. We define the $\bullet$-*development*[3] relation by $a \mathrel{\circ\!\!\!\!\rightarrow} b$ if $a \rightarrow^* b$ and $b \rightarrow^* a^\bullet$. Then $\rightarrow \subseteq \mathrel{\circ\!\!\!\!\rightarrow} \subseteq \rightarrow^*$ follows from the definition of $\mathrel{\circ\!\!\!\!\rightarrow}$ and the Z-property. The angle itself directly follows from the definition of $\mathrel{\circ\!\!\!\!\rightarrow}$ and monotonicity of $\bullet$. ☐

We turn to the quantitative analysis of the claim of [4]. Formalizing confluence of the $\lambda\beta$-calculus has a long history for which we refer the reader to [7]. We compare our formalization in Isabelle to two other such, Nipkow's formalization in Isabelle/HOL [7] (as currently distributed with Isabelle) and Urban and Arnaud's formalization in Nominal Isabelle.[4] There are two major differences of the present proof to Nipkow's formalization. On the one hand Nipkow uses de Brujin indices to represent λ-terms. This considerably increases the size of the formal theories – almost 200 lines of the roughly 550 line development are devoted to setting up terms and the required manipulations on indices. Our development is 300 lines (60 of which are used for our ad hoc Eisbach methods). The second difference is the actual technique used to show confluence: Nipkow proceeds by establishing the diamond property for complete developments. Urban and Arnaud proceed by establishing the angle property for multisteps with respect to the full-development function. This results in a 100 line increase compared to our formalization.

References

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

[2] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 2nd revised edition, 1984.

[3] P. Dehornoy. *Braids and Self-Distributivity*, volume 192 of *Progress in Mathematics*. Springer, 2000. doi:10.1007/978-3-0348-8442-6.

[4] P. Dehornoy and V. van Oostrom. Z, proving confluence by monotonic single-step upperbound functions. In *LMRC*, 2008. www.phil.uu.nl/~oostrom/publication/talk/lmrc060508.pdf.

[5] B. Felgenhauer, J. Nagele, V. van Oostrom, and C. Sternagel. The Z property. *AFP*, June 2016. https://www.isa-afp.org/entries/Rewriting_Z.shtml, Formal proof development.

[6] D. Matichuk, T. Murray, and M. Wenzel. Eisbach: A proof method language for Isabelle. *J. Autom. Reasoning*, 56(3):261–282, 2016. doi:10.1007/s10817-015-9360-2.

[7] T. Nipkow. More Church-Rosser proofs. *J. Autom. Reasoning*, 26(1):51–66, 2001. doi:10.1023/A:1006496715975.

[8] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. doi:10.1007/3-540-45949-9.

[9] F. van Raamsdonk. *Confluence and Normalisation for Higher-Order Rewriting*. PhD thesis, Vrije Universiteit Amsterdam, 1996.

[10] M. Takahashi. Parallel reductions in λ-calculus. *Inform. Comput.*, 118(1):120–127, 1995. doi:10.1006/inco.1995.1057.

[11] Terese. *Term Rewriting Systems*, volume 55 of *CTTCS*. CUP, 2003.

[12] C. Urban and C. Kaliszyk. General bindings and alpha-equivalence in Nominal Isabelle. *LMCS*, 8(2):14:1–14:35, 2012. doi:10.2168/LMCS-8(2:14)2012.

[3] For the full-development map $\bullet$ such *syntax-free* $\bullet$-developments may differ from the usual ones, e.g. for $(\lambda y.\, I)\ ((\lambda x.\, x\ x)\ I)$. We conjecture that on terminating, non-erasing and non-collapsing λ-terms they coincide.

[4] http://www.inf.kcl.ac.uk/staff/urbanc/cgi-bin/repos.cgi/nominal2/file/d79e936e30ea/Nominal/Ex/CR.thy

Formalized Confluence of Quasi-Decreasing, Strongly Deterministic Conditional TRSs*

Thomas Sternagel and Christian Sternagel

University of Innsbruck, Austria
{thomas,christian}.sternagel@uibk.ac.at

Abstract

We present an Isabelle/HOL formalization of a characterization of confluence for quasi-reductive strongly deterministic conditional term rewrite systems, due to Avenhaus and Loría-Sáenz.

1 Introduction

Already in 1994 Avenhaus and Loría-Sáenz [1] proved a critical pair criterion for deterministic conditional term rewrite systems with extra variables in right-hand sides, provided their rewrite relation is decidable and terminating. We use this criterion in our conditional confluence checker ConCon [6]. In the following we provide a description of our formalization of the conditional critical pair criterion where we strengthened the original result from quasi-reductivity to quasi-decreasingness. This is a first step towards certifying the confluence criterion that a quasi-decreasing and strongly deterministic CTRS is confluent if all of its critical pairs are joinable. The formalization described in this paper is part of a greater effort to formalize all methods employed by ConCon to be able to certify its output.

Contribution. We have formalized Theorem 4.1 from Avenhaus and Loría-Sáenz [1] in Isabelle/HOL [4] as well as strengthened the original theorem from quasi-reductivity to quasi-decreasingness. It is now part of the formal library IsaFoR [7] (the Isabelle Formalization of Rewriting) and freely available online at:

http://cl2-informatik.uibk.ac.at/rewriting/mercurial.cgi/IsaFoR/file/dbc03280d673/thys/Conditional_Rewriting/ALS94.thy

2 Preliminaries

We assume familiarity with the basic notions of (conditional) term rewriting [2, 5], but shortly recapitulate terminology and notation that we use in the remainder. Given an arbitrary binary relation $\to_\alpha$, we write ${}_\alpha\!\leftarrow$, $\to_\alpha^+$, $\to_\alpha^*$ for the *inverse*, the *transitive closure*, and the *reflexive transitive closure* of $\to_\alpha$, respectively. We use $\mathcal{V}(\cdot)$ to denote the set of variables occurring in a given syntactic object, like a term, a pair of terms, a list of terms, etc. The set of terms $\mathcal{T}(\mathcal{F},\mathcal{V})$ over a given signature of function symbols $\mathcal{F}$ and set of variables $\mathcal{V}$ is defined inductively: $x \in \mathcal{T}(\mathcal{F},\mathcal{V})$ for all variables $x \in \mathcal{V}$, and for every n-ary function symbol $f \in \mathcal{F}$ and terms $t_1,\ldots,t_n \in \mathcal{T}(\mathcal{F},\mathcal{V})$ also $f(t_1,\ldots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{V})$. We say that terms s and t *unify*, written $s \sim t$, if $s\sigma = t\sigma$ for some substitution σ. A substitution σ is *normalized with respect to* $\mathcal{R}$ if $\sigma(x)$ is a normal form with respect to $\to_\mathcal{R}$ for all $x \in \mathcal{V}$. We call a bijective variable substitution $\pi : \mathcal{V} \to \mathcal{V}$ a *variable renaming* or *(variable) permutation*, and denote its inverse by π^-. A term t is *strongly*

*The research described in this paper is supported by FWF (Austrian Science Fund) project P27502.

irreducible with respect to $\mathcal{R}$ if $t\sigma$ is a normal form with respect to $\to_{\mathcal{R}}$ for all normalized substitutions σ. A *strongly deterministic oriented 3-CTRS (SDTRS)* $\mathcal{R}$ is a set of conditional rewrite rules of the shape $\ell \to r \Leftarrow c$ where ℓ and r are terms and c is a possibly empty sequence of pairs of terms $s_1 \approx t_1, \ldots, s_n \approx t_n$. For all rules in $\mathcal{R}$ we have that $\ell \notin \mathcal{V}$, $\mathcal{V}(r) \subseteq \mathcal{V}(\ell, c)$, $\mathcal{V}(s_i) \subseteq \mathcal{V}(\ell, t_1, \ldots, t_{i-1})$ for all $1 \leqslant i \leqslant n$, and t_i is strongly irreducible with respect to $\mathcal{R}$ for all $1 \leqslant i \leqslant n$. We sometimes label rules like $\rho : \ell \to r \Leftarrow c$. For a rule $\rho : \ell \to r \Leftarrow c$ of an SDTRS $\mathcal{R}$ the set of *extra variables* is defined as $\mathcal{EV}(\rho) = \mathcal{V}(c) - \mathcal{V}(\ell)$. The rewrite relation $\to_{\mathcal{R}}$ is the smallest relation $\to$ satisfying $t[\ell\sigma]_p \to t[r\sigma]_p$ whenever $\ell \to r \Leftarrow c$ is a rule in $\mathcal{R}$ and $s\sigma \to_{\mathcal{R}}^* t\sigma$ for all $s \approx t \in c$. Two variable-disjoint variants of rules $\ell_1 \to r_1 \Leftarrow c_1$ and $\ell_2 \to r_2 \Leftarrow c_2$ in $\mathcal{R}$ such that $\ell_1|_p \notin \mathcal{V}$ and $\ell_1|_p\mu = \ell_2\mu$ with most general unifier (mgu) μ, constitute a *conditional overlap*. A conditional overlap that does not result from overlapping two variants of the same rule at the root, gives rise to a *conditional critical pair* (CCP) $r_1\mu \approx r_1[r_2]_p\mu \Leftarrow c_1\mu, c_2\mu$. A CCP $u \approx v \Leftarrow c$ is *joinable* if $u\sigma \downarrow_{\mathcal{R}} v\sigma$ for all substitutions σ such that $s\sigma \to_{\mathcal{R}}^* t\sigma$ for all $s \approx t \in c$. We denote the proper subterm relation by $\rhd$ and define $\succ_{\mathsf{st}} = (\succ \cup \rhd)^+$ for some reduction order $\succ$. Let $\succ$ be a reduction order on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ then an SDTRS $\mathcal{R}$ is *quasi-reductive with respect to* $\succ$ if for every substitution σ and every rule $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n$ in $\mathcal{R}$ we have $s_j\sigma \succeq t_j\sigma$ for $1 \leqslant j \leqslant i$ implies $\ell\sigma \succ_{\mathsf{st}} s_{i+1}\sigma$, and $s_j\sigma \succeq t_j\sigma$ for $1 \leqslant j \leqslant n$ implies $\ell\sigma \succ r\sigma$.[1] On the other hand, an SDTRS $\mathcal{R}$ over signature $\mathcal{F}$ is *quasi-decreasing* if there is a well-founded order $\succ$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $\succ = \succ_{\mathsf{st}}$, $\to_{\mathcal{R}} \subseteq \succ$, and for all rules $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n$ in $\mathcal{R}$, all substitutions $\sigma \colon \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$, and $0 \leqslant i < n$, if $s_j\sigma \to_{\mathcal{R}}^* t_j\sigma$ for all $1 \leqslant j \leqslant i$ then $\ell\sigma \succ s_{\mathrm{i+1}}\sigma$. Quasi-reductivity implies quasi-decreasingness (cf. [5, proof of Lemma 7.2.40]).

3 Confluence of Quasi-Decreasing SDTRSs

The main result from Avenhaus and Loría-Sáenz is the following theorem:

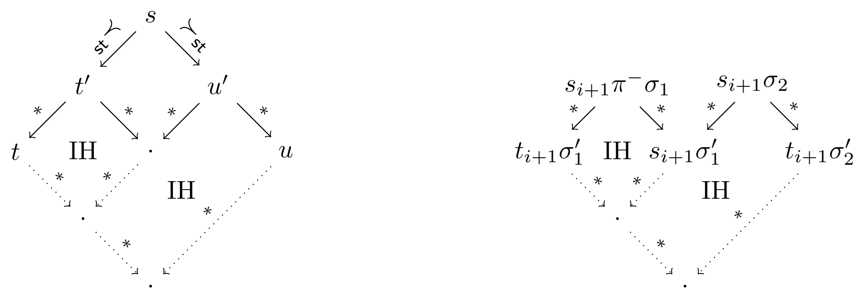
Theorem 1 (Avenhaus and Loría-Sáenz [1, Theorem 4.1])**.** *Let $\mathcal{R}$ be an SDTRS that is quasi-reductive with respect to $\succ$. $\mathcal{R}$ is confluent if and only if all conditional critical pairs are joinable.*

That all critical pairs of any CTRS $\mathcal{R}$ (no need for strong determinism or quasi-reductivity) are joinable if $\mathcal{R}$ is confluent is straight-forward so we will concentrate on the other direction. Our formalization is quite close to the original proof. The good news is: we could not find any errors (besides typos) in the original proof but as is often the case with formalizations there are places where the paper proof is too vague or does not spell out the technical details in favor of readability. A luxury we cannot afford. For example we heavily rely on an earlier formalization of permutations [3] in order to formalize variants of rules up to renaming. Even the change from quasi-reductivity to quasi-decreasingness did not pose a problem.

In the following we will give a description of the main theorem of our formalization and its proof.

Theorem 2. *Let $\mathcal{R}$ be an SDTRS that is quasi-decreasing with respect to $\succ$ and where all conditional critical pairs are joinable, then $\mathcal{R}$ is confluent.*

[1] This is the definition from [1] which differs from the one in [5, Definition 7.2.36] in two respects. First $\succ$ is a reduction order (hence also closed under substitutions; this is needed in the proof of [1, Theorem 4.2]) whereas in Ohlebusch $\succ$ is a well-founded partial order that is closed under contexts. Moreover Ohlebusch allows a signature extension for the substitutions σ which is not part of this definition.



(a) (b)

Figure 1

Proof. Assume that all critical pairs are joinable. We will look at an arbitrary peak $t \; {}_{\mathcal{R}}^{*}\!\!\leftarrow s \rightarrow_{\mathcal{R}}^{*} u$ and prove that $t \downarrow_{\mathcal{R}} u$ by well-founded induction on the relation $\succ_{\mathsf{st}}$. If $s = t$ or $s = u$ then t and u are trivially joinable and we are done. So we may assume that the peak contains at least one step in each direction: $t \; {}_{\mathcal{R}}^{*}\!\!\leftarrow t' \; {}_{\mathcal{R}}\!\!\leftarrow s \rightarrow_{\mathcal{R}} u' \rightarrow_{\mathcal{R}}^{*} u$.

We will proceed to prove that $t' \downarrow_{\mathcal{R}} u'$ then $t \downarrow_{\mathcal{R}} u$ follows by two applications of the induction hypothesis as shown in Figure 1a. Assume that $s = C[\ell_1\sigma_1]_p \rightarrow_{\mathcal{R}} C[r_1\sigma_1]_p = t'$ and $s = D[\ell_2\sigma_2]_q \rightarrow_{\mathcal{R}} D[r_2\sigma_2]_q = u'$ for rules $\rho_1 : \ell_1 \rightarrow r_1 \Leftarrow c_1$ and $\rho_2 : \ell_2 \rightarrow r_2 \Leftarrow c_2$ in $\mathcal{R}$, contexts C and D, positions p and q, and substitutions σ_1 and σ_2 such that $u\sigma_1 \rightarrow_{\mathcal{R}}^{*} v\sigma_1$ for all $u \approx v \in c_1$ and $u\sigma_2 \rightarrow_{\mathcal{R}}^{*} v\sigma_2$ for all $u \approx v \in c_2$. There are three possibilities: $p \parallel q$, $p \leqslant q$, or $q \leqslant p$. In the first case $t' \downarrow_{\mathcal{R}} u'$ holds because the two redexes do not interfere. The other two cases are symmetric and we only consider $p \leqslant q$ here. If $s \rhd s|_p = \ell_1\sigma_1$ then $s \succ_{\mathsf{st}} \ell_1\sigma_1$ (by definition of $\succ_{\mathsf{st}}$) and there is a position r such that $q = pr$ and so we have the peak $r_1\sigma_1 \; {}_{\mathcal{R}}^{*}\!\!\leftarrow \ell_1\sigma_1 \rightarrow_{\mathcal{R}}^{*} \ell_1\sigma_1[r_2\sigma_2]_r$ which is joinable by induction hypothesis. But then the peak $t' = s[r_1\sigma_1]_p \; {}_{\mathcal{R}}^{*}\!\!\leftarrow s[\ell_1\sigma_1]_p \rightarrow_{\mathcal{R}}^{*} s[\ell_1\sigma_1[r_2\sigma_2]_r]_q = u'$ is also joinable (by closure under contexts) and we are done. So we may assume that $p = \epsilon$ and thus $s = \ell_1\sigma_1$. Now, either q is a function position in ℓ_1 or there is a variable position q' in ℓ_1 such that $q' \leqslant q$. In the first case we either have a CCP which is joinable by assumption or we have a root-overlap of variants of the same rule. Then $\rho_1\pi = \rho_2$ for some permutation π. Moreover, $s = \ell_1\sigma_1 = \ell_2\sigma_2$ and we have

$$x\pi^{-}\sigma_1 = x\sigma_2 \text{ for all variables } x \text{ in } \mathcal{V}(\ell_2). \tag{1}$$

We will prove $x\pi^{-}\sigma_1 \downarrow_{\mathcal{R}} x\sigma_2$ for all x in $\mathcal{V}(\rho_2)$. Since $t' = r_1\sigma_1 = r_2\pi^{-}\sigma_1$ and $u' = r_2\sigma_2$ this shows $t' \downarrow_{\mathcal{R}} u'$. Because $\mathcal{R}$ is terminating (by quasi-decreasingness) we may define two normalized substitutions σ'_i such that

$$x\pi^{-}\sigma_1 \xrightarrow[\mathcal{R}]{*} x\sigma'_1 \text{ and } x\sigma_2 \xrightarrow[\mathcal{R}]{*} x\sigma'_2 \text{ for all variables } x. \tag{2}$$

We prove $x\sigma'_1 = x\sigma'_2$ for $x \in \mathcal{EV}(\rho_2)$ by an inner induction on the length of $c_2 = s_1 \approx t_1, \ldots, s_n \approx t_n$. If ρ_2 has no conditions this holds vacuously because there are no extra variables. In the step case the inner induction hypothesis is that $x\sigma'_1 = x\sigma'_2$ for $x \in \mathcal{V}(s_1, t_1, \ldots, s_i, t_i) - \mathcal{V}(\ell_2)$ and we have to show that $x\sigma'_1 = x\sigma'_2$ for $x \in \mathcal{V}(s_1, t_1, \ldots, s_{i+1}, t_{i+1}) - \mathcal{V}(\ell_2)$. If $x \in \mathcal{V}(s_1, t_1, \ldots, s_i, t_i, s_{i+1})$ we are done by the inner induction hypothesis and strong determinism of $\mathcal{R}$. So assume $x \in \mathcal{V}(t_{i+1})$. From strong determinism of $\mathcal{R}$, (1), (2), and the induction hypothesis we have that $y\sigma'_1 = y\sigma'_2$ for all $y \in \mathcal{V}(s_{i+1})$ and thus $s_{i+1}\sigma'_1 = s_{i+1}\sigma'_2$. With this we can find a join between $t_{i+1}\sigma'_1$ and $t_{i+1}\sigma'_2$ by applying the induction hypothesis twice as shown in Figure 1b. Since t_{i+1} is strongly irreducible and σ_1' and σ_2' are normalized, this yields $t_{i+1}\sigma_1' = t_{i+1}\sigma_2'$ and thus $x\sigma_1' = x\sigma_2'$.

We are left with the case that there is a variable position q' in ℓ_1 such that $q = q'r'$ for some position r'. Let x be the variable $\ell_1|_{q'}$. Then $x\sigma_1|_{r'} = \ell_2\sigma_2$, which implies $x\sigma_1 \rightarrow_{\mathcal{R}}^* x\sigma_1[r_2\sigma_2]_{r'}$. Now let τ be the substitution such that $\tau(x) = x\sigma_1[r_2\sigma_2]_{r'}$ and $\tau(y) = \sigma_1(y)$ for all $y \neq x$, and τ' some normalization, i.e., $y\tau \rightarrow_{\mathcal{R}}^* y\tau'$ for all y. Moreover, note that

$$y\sigma_1 \xrightarrow[\mathcal{R}]{*} y\tau \text{ for all } y. \tag{3}$$

We have $u' = \ell_1\sigma_1[r_2\sigma_2]_q = \ell_1\sigma_1[x\tau]_{q'} \rightarrow_{\mathcal{R}}^* \ell_1\tau$, and thus $u' \rightarrow_{\mathcal{R}}^* \ell_1\tau'$. From (3) we have $r_1\sigma_1 \rightarrow_{\mathcal{R}}^* r_1\tau$ and thus $t' = r_1\sigma_1 \rightarrow_{\mathcal{R}}^* r_1\tau'$. Finally, we will show that $\ell_1\tau' \rightarrow_{\mathcal{R}} r_1\tau'$, concluding the proof of $t' \downarrow_{\mathcal{R}} u'$. To this end, let $s_i \approx t_i \in c_1$. By (3) and the definition of τ' we obtain $s_i\sigma_1 \rightarrow_{\mathcal{R}}^* t_i\sigma_1 \rightarrow_{\mathcal{R}}^* t_i\tau'$ and $s_i\sigma_1 \rightarrow_{\mathcal{R}}^* s_i\tau'$. But then, by induction hypothesis, $s_i\tau' \downarrow_{\mathcal{R}} t_i\tau'$, and furthermore, since t_i is strongly irreducible, $s_i\tau' \rightarrow_{\mathcal{R}}^* t_i\tau'$. $\square$

4 Conclusion

Our formalization amounts to approximately 1800 lines of Isabelle. At some points we actually had to use variants of rules where the original proof assumes two rules to be identical. Apart from that the formalization was rather straight-forward. Also the modification from quasi-reductivity to quasi-decreasingness did not pose a problem.

Future Work. Formalizing the conditional critical pair criterion was only the first step. There are two challenges for automation: Checking if a term is strongly irreducible, and checking if a conditional critical pair is joinable. Both of these are undecidable in general. Avenhaus and Loría-Sáenz employ *absolute determinism* [1, Definition 4.2] to tackle strong irreducibility as well as *contextual rewriting* to handle joinability of conditional critical pairs. Then we have a computable overapproximation. We already started to extend our formalization to facilitate absolute determinism as well as contextual rewriting. It remains to provide check functions for CeTA [7] and also the proper certifiable output for ConCon.

Acknowledgments. We thank the Austrian Science Fund (FWF project P27502) for supporting our work. Moreover we would like to thank the anonymous reviewers for useful suggestions.

References

[1] Jürgen Avenhaus and Carlos Loría-Sáenz. On conditional rewrite systems with extra variables and deterministic logic programs. In *Proceedings of the 5th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 822 of *Lecture Notes in Computer Science*, pages 215–229. Springer, 1994. doi:10.1007/3-540-58216-9_40.

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

[3] Nao Hirokawa, Aart Middeldorp, and Christian Sternagel. A new and formalized proof of abstract completion. In *Proceedings of the 5th International Conference on Interactive Theorem Proving*, volume 8558 of *Lecture Notes in Computer Science*, pages 292–307. Springer, 2014. doi:10.1007/978-3-319-08970-6_19.

[4] Tobias Nipkow, Lawrence Charles Paulson, and Makarius Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002. doi:10.1007/3-540-45949-9.

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

[6] Thomas Sternagel and Aart Middeldorp. Conditional confluence (system description). In *Proceedings of the Joint 25th International Conference on Rewriting Techniques and Applications and 12th International Conference on Typed Lambda Calculi and Applications*, volume 8560 of *Lecture Notes in Computer Science*, pages 456–465. Springer, 2014. doi:10.1007/978-3-319-08918-8_31.

[7] René Thiemann and Christian Sternagel. Certification of termination proofs using CeTA. In *Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics*, volume 5674 of *Lecture Notes in Computer Science*, pages 452–468. Springer, 2009. doi:10.1007/978-3-642-03359-9_31.

Notes on Confluence of Ultra-Weakly-Left-Linear SDCTRSs via a Structure-Preserving Transformation

Naoki Nishida

Nagoya University, Nagoya, Japan
nishida@is.nagoya-u.ac.jp

Abstract

A structure-preserving transformation proposed by Şerbănuţă and Roşu for strongly or syntactically deterministic conditional term rewriting systems (SDCTRSs) that are ultra-left-linear has been shown to be applicable to weakly-left-linear (WLL) and ultra-WLL SDCTRSs without any change, and sound for such DCTRSs even if they are not SDCTRSs. In this paper, we show a confluent, WLL, and ultra-WLL DCTRS that is not an SDCTRS such that the transformed TRS is not confluent. We also show that for a WLL and ultra-WLL SDCTRS, if the transformed TRS is confluent, then so is the SDCTRS.

1 Introduction

Conditional term rewriting is known to be much more complicated than unconditional term rewriting in the sense of analyzing properties (cf. [9]). 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 in general 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. There are two approaches to transformations of CTRSs into TRSs: *unravelings* [6, 7] proposed by Marchiori (see, e.g., [2, 9]), and a transformation [16] proposed by Viry (see, e.g., [13, 2]).

The latest transformation based on Viry's approach is a *computationally equivalent* transformation proposed by Şerbănuţă and Roşu [13, 14] (the SR transformation, for short), which is one of *structure-preserving* transformations [4]. This transformation has been proposed for *normal* CTRSs in [13]—started with this class to simplify the discussion—and then been extended for *strongly or syntactically deterministic* CTRSs (SDCTRSs) that are *ultra-left-linear* (*semilinear* [14]). Here, for a syntactic property P, a CTRS is said to be *ultra-P* if its unraveled TRS via Ohlebusch's unraveling [12] has the property P. The transformation converts a confluent, operationally terminating, and ultra-left-linear SDCTRS into a TRS that is *computationally equivalent* to the CTRS. This means that such a converted TRS can be used to exactly simulate any derivation of the original CTRS to a normal form.

Recently, it has been shown in [8, a revised version] that the SR transformation for ultra-left-linear SDCTRSs is applicable to *weakly*-left-linear (WLL) and ultra-WLL SDCTRSs without any change, and sound for such DCTRSs even if they are not SDCTRSs. From this result, one may think that our target DCTRSs do not have to be strongly or syntactically deterministic. However, we must take this property into account when we consider confluence.

In this paper, we show a confluent, WLL, and ultra-WLL DCTRS that is not an SDCTRS such that the transformed TRS is not confluent, i.e., confluence is not preserved by the transformation. We also show that for a WLL and ultra-WLL SDCTRSs, if the transformed TRS is confluent, then so is the SDCTRS. None of the results in this paper is new compared to those in [14] and [10]. The contribution of this paper is to confirm that some results for ultra-left-linear SDCTRSs also hold for WLL and ultra-WLL ones.

2 Preliminaries

For the page limitation, we omit basic notions and notations for term rewriting [1, 12], and we assume that the reader is familiar with them. This paper follows the previous work in [8]. In this section, we briefly introduce very important notions to understand results in this paper.

An (oriented) *conditional rewrite rule* over a signature $\mathcal{F}$ is a triple (ℓ, r, c), denoted by $\ell \to r \Leftarrow c$, such that the *left-hand side* ℓ is a non-variable term in $T(\mathcal{F}, \mathcal{V})$, the *right-hand side* r is a term in $T(\mathcal{F}, \mathcal{V})$, and the *conditional part* c is a sequence $s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k$ of term pairs ($k \geq 0$) where all of $s_1, t_1, \ldots, s_k, t_k$ are terms in $T(\mathcal{F}, \mathcal{V})$. In particular, a conditional rewrite rule is called *unconditional* if the conditional part is the empty sequence (i.e., $k = 0$), and we may abbreviate it to $\ell \to r$. We sometimes attach a unique label ρ to the conditional rewrite rule $\ell \to r \Leftarrow c$ by denoting $\rho : \ell \to r \Leftarrow c$, and we use the label to refer to the rewrite rule. An *(oriented) conditional term rewriting system* (CTRS) over a signature $\mathcal{F}$ is a set of conditional rules over $\mathcal{F}$. A term t is called *strongly irreducible* (w.r.t. $\mathcal{R}$) if $t\sigma$ is a normal form w.r.t. $\mathcal{R}$ for every normalized substitution σ. The sets of *defined symbols* and *constructors* of $\mathcal{R}$ are denoted by $\mathcal{D}_\mathcal{R}$ and $\mathcal{C}_\mathcal{R}$, respectively: $\mathcal{D}_\mathcal{R} = \{root(\ell) \mid \ell \to r \Leftarrow c \in \mathcal{R}\}$ and $\mathcal{C}_\mathcal{R} = \mathcal{F} \setminus \mathcal{D}_\mathcal{R}$.

A CTRS $\mathcal{R}$ is called *deterministic* (DCTRS, for short) if for every rule $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k \in \mathcal{R}$, $\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(\ell, t_1, \ldots, t_{i-1})$ for all $1 \leq i \leq k$. In this paper, we deal with *3-DCTRSs*, i.e., for $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k \in \mathcal{R}$, $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell, s_1, t_1, \ldots, s_k, t_k)$. A DCTRS $\mathcal{R}$ is called *strongly deterministic* if for every rule $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k$, every term t_i is strongly irreducible w.r.t. $\mathcal{R}$, and called *syntactically deterministic* if for every rule $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k$, every term t_i is a constructor term or a ground normal form of the *underlying unconditional system* $\{\ell \to r \mid \ell \to r \Leftarrow c \in \mathcal{R}\}$. We simply call a strongly or syntactically deterministic CTRS an *SDCTRS*. Note that every *normal* CTRS is an SDCTRS. The number of occurrences of a variable x in a term sequence $t_1, \ldots, t_n$ is denoted by $|t_1, \ldots, t_n|_x$. A conditional rewrite rule $\rho : \ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k$ is called *weakly-left-linear* (WLL) [3] if $|\ell, t_1, \ldots, t_k|_x = 1$ for any variable $x \in \mathcal{V}ar(r, s_1, \ldots, s_k)$. Note that not all left-linear (LL, for short) DCTRSs are WLL, e.g., $\mathsf{f}(x) \to x \Leftarrow \mathsf{g}(x) \twoheadrightarrow x$ is LL but not WLL. Regarding the *simultaneous unraveling* $\mathbb{U}$ [12], a DCTRS $\mathcal{R}$ is called *ultra-left-linear w.r.t.* $\mathbb{U}$ ($\mathbb{U}$-LL) if for every rule $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k$, the sequence $\ell, t_1, \ldots, t_k$ is linear. In addition, $\mathcal{R}$ is called *ultra-weakly-left-linear w.r.t.* $\mathbb{U}$ ($\mathbb{U}$-WLL) [8] if all unconditional rules in $\mathcal{R}$ are WLL and every conditional rule $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k$ ($k > 0$) in $\mathcal{R}$ satisfies that the sequence $\ell, t_1, \ldots, t_{k-1}$ is linear and $|\ell, t_1, \ldots, t_k|_x \leq 1$ for any variable $x \in \mathcal{V}ar(r)$.

3 The SR Transformation

In this section, we briefly introduce the SR transformation [14] and its properties. We often denote a term sequence $t_i, t_{i+1}, \ldots, t_j$ by $\overrightarrow{t_{i..j}}$. Moreover, for the application of a mapping τ to $\overrightarrow{t_{i..j}}$, we denote the sequence $\tau(t_i), \ldots, \tau(t_j)$ by $\overrightarrow{\tau(t_{i..j})}$, e.g., for a substitution θ, we denote the sequence $t_i\theta, \ldots, t_j\theta$ by $\overrightarrow{\theta(t_{i..j})}$. For a finite set $X = \{o_1, o_2, \ldots, o_n\}$ of objects, a sequence $o_1, o_2, \ldots, o_n$ under some arbitrary but fixed total order on the objects is denoted by $\overrightarrow{X}$. In the following, we use the terminology "conditional" for a rewrite rule that has at least one condition, and distinguish "conditional rules" and "unconditional rules".

Before transforming a CTRS $\mathcal{R}$, we first extend the signature of $\mathcal{R}$ as follows: We keep the constructors of $\mathcal{R}$, whereas we replace each n-ary constructor c by $\bar{\mathsf{c}}$ having the arity n; The arity n of defined symbol f is extended to $n+m$ where f has m conditional rules in $\mathcal{R}$, replacing f by $\bar{\mathsf{f}}$ having the arity $n+m$; A fresh constant $\perp$ and a fresh unary symbol $\langle \cdot \rangle$ are introduced;

For every conditional rule $\rho : \ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k$ in $\mathcal{R}$, we introduce k fresh symbols $[\,]^\rho_1, [\,]^\rho_2, \ldots, [\,]^\rho_k$ with the arities $1, 1 + |\mathcal{V}ar(t_1)|, 1 + |\mathcal{V}ar(t_1, t_2)|, \ldots, 1 + |\mathcal{V}ar(t_1, \ldots, t_{k-1})|$. We assume that for every defined symbol f, the conditional rules for f are ranked by some arbitrary but fixed order. We denote the extended signature by $\overline{\mathcal{F}}$: $\overline{\mathcal{F}} = \{\overline{\mathsf{c}} \mid \mathsf{c} \in \mathcal{C}_\mathcal{R}\} \cup \{\overline{\mathsf{f}} \mid \mathsf{f} \in \mathcal{D}_\mathcal{R}\} \cup \{\bot, \langle\cdot\rangle\} \cup \{[\,]^\rho_j \mid \rho : \ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k \in \mathcal{R},\ 1 \le j \le k\}$. We introduce a mapping $\mathtt{ext}$ to extend the arguments of defined symbols in a term as follows: $\mathtt{ext}(x) = x$ for $x \in \mathcal{V}$; $\mathtt{ext}(\mathsf{c}(\overrightarrow{t_{1..n}})) = \overline{\mathsf{c}}(\overrightarrow{\mathtt{ext}(t_{1..n})})$ for $\mathsf{c}/n \in \mathcal{C}_\mathcal{R}$; $\mathtt{ext}(\mathsf{f}(\overrightarrow{t_{1..n}})) = \overline{\mathsf{f}}(\overrightarrow{\mathtt{ext}(t_{1..n})}, \overrightarrow{z_{1..m}})$ for $\mathsf{f}/n \in \mathcal{D}_\mathcal{R}$, where f has m conditional rules in $\mathcal{R}$ and $z_1, \ldots, z_m$ are fresh variables. To put $\bot$ into the extended arguments, we define a mapping $(\cdot)^\bot$ that puts $\bot$ to all the extended arguments of defined symbols, as follows: $(x)^\bot = x$ for $x \in \mathcal{V}$; $(\overline{\mathsf{c}}(\overrightarrow{t_{1..n}}))^\bot = \overline{\mathsf{c}}(\overrightarrow{(t_{1..n})^\bot})$ for $\mathsf{c}/n \in \mathcal{C}_\mathcal{R}$; $(\overline{\mathsf{f}}(\overrightarrow{t_{1..n}}, \overrightarrow{u_{1..m}}))^\bot = \overline{\mathsf{f}}(\overrightarrow{(t_{1..n})^\bot}, \bot, \ldots, \bot)$ for $\mathsf{f}/n \in \mathcal{D}_\mathcal{R}$; $(\langle t\rangle)^\bot = \langle (t)^\bot \rangle$; $(\bot)^\bot = \bot$; $([\ldots]^\rho_j)^\bot = \bot$. Now we define a mapping $\bar{\cdot}$ from $T(\mathcal{F}, \mathcal{V})$ to $T(\overline{\mathcal{F}}, \mathcal{V})$ as $\overline{t} = (\mathtt{ext}(t))^\bot$. On the other hand, the partial inverse mapping $\widehat{\cdot}$ for $\bar{\cdot}$ is defined as follows: $\widehat{x} = x$ for $x \in \mathcal{V}$; $\widehat{\overline{\mathsf{c}}(\overrightarrow{t_{1..n}})} = \mathsf{c}(\widehat{t_1}, \ldots, \widehat{t_n})$ for $\mathsf{c}/n \in \mathcal{C}_\mathcal{R}$; $\widehat{\overline{\mathsf{f}}(\overrightarrow{t_{1..n}}, \ldots)} = \mathsf{f}(\widehat{t_1}, \ldots, \widehat{t_n})$ for $\mathsf{f}/n \in \mathcal{D}_\mathcal{R}$; $\widehat{\langle t\rangle} = \widehat{t}$. Note that in applying $(\cdot)^\bot$ or $\widehat{\cdot}$ to *reachable terms* defined later, the case of applying $(\cdot)^\bot$ to $\bot$ or $[\ldots]^\rho_j$ never happens.

The SR transformation [14] for SDCTRSs is defined not only for $\mathbb{U}$-LL SDCTRSs but also for WLL and $\mathbb{U}$-WLL SDCTRSs as follows [8].

Definition 1 ($\mathbb{SR}$ [14, 8])**.** *Let $\mathcal{R}$ be a WLL and $\mathbb{U}$-WLL SDCTRS and the extended signature $\overline{\mathcal{F}}$ mentioned above. Then, the i-th conditional f-rule $\rho : \mathsf{f}(\overrightarrow{w_{1..n}}) \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k$ is transformed into a set of $k+1$ unconditional rules as follows:*

$$\mathbb{SR}(\rho) = \left\{ \begin{array}{l} \overline{\mathsf{f}}(\overrightarrow{w'_{1..n}}, \overrightarrow{z_{1..i-1}}, \bot, \overrightarrow{z_{i+1..m}}) \to \overline{\mathsf{f}}(\overrightarrow{w'_{1..n}}, \overrightarrow{z_{1..i-1}}, [\langle \overline{s_1}\rangle, \overrightarrow{V_1}]^\rho_1, \overrightarrow{z_{i+1..m}}), \\ \overline{\mathsf{f}}(\overrightarrow{w'_{1..n}}, \overrightarrow{z_{1..i-1}}, [\langle \mathtt{ext}(t_1)\rangle, \overrightarrow{V_1}]^\rho_1, \overrightarrow{z_{i+1..m}}) \to \overline{\mathsf{f}}(\overrightarrow{w'_{1..n}}, \overrightarrow{z_{1..i-1}}, [\langle \overline{s_2}\rangle, \overrightarrow{V_2}]^\rho_2, \overrightarrow{z_{i+1..m}}), \\ \vdots \\ \overline{\mathsf{f}}(\overrightarrow{w'_{1..n}}, \overrightarrow{z_{1..i-1}}, [\langle \mathtt{ext}(t_k)\rangle, \overrightarrow{V_k}]^\rho_k, \overrightarrow{z_{i+1..m}}) \to \langle \overline{r}\rangle \end{array} \right\}$$

where $\overrightarrow{w'_{1..n}} = \overrightarrow{\mathtt{ext}(w_{1..n})}$, $V_j = \mathcal{V}ar(\overrightarrow{t_{1..j-1}})$ for all $1 \le j \le k$, and $z_1, \ldots, z_{i-1}, z_i, \ldots, z_m$ are fresh variables. An unconditional rule in $\mathcal{R}$ is converted as follows: $\mathbb{SR}(\ell \to r) = \{\ \mathtt{ext}(\ell) \to \langle \overline{r}\rangle\ \}$. The set of auxiliary rules is defined as follows:

$$\begin{array}{rl} \mathcal{R}_{aux} = & \{\ \langle\langle x\rangle\rangle \to \langle x\rangle\ \} \cup \{\ \overline{\mathsf{c}}(\overrightarrow{x_{1..i-1}}, \langle x_i\rangle, \overrightarrow{x_{i+1..n}}) \to \langle \overline{\mathsf{c}}(\overrightarrow{x_{1..n}})\rangle \mid \mathsf{c}/n \in \mathcal{C}_\mathcal{R},\ 1 \le i \le n\ \} \\ & \cup \{\ \overline{\mathsf{f}}(\overrightarrow{x_{1..i-1}}, \langle x_i\rangle, \overrightarrow{x_{i+1..n}}, \overrightarrow{z_{1..m}}) \to \langle \overline{\mathsf{f}}(\overrightarrow{x_{1..n}}, \bot, \ldots, \bot)\rangle \mid \mathsf{f}/n \in \mathcal{D}_\mathcal{R},\ 1 \le i \le n\ \} \end{array}$$

where $x_1, \ldots, x_n, z_1, \ldots, z_m$ are distinct variables. The transformation $\mathbb{SR}$ is defined as follows: $\mathbb{SR}(\mathcal{R}) = \bigcup_{\rho \in \mathcal{R}} \mathbb{SR}(\rho) \cup \mathcal{R}_{aux}$. We say that $\mathbb{SR}$ (and also $\mathbb{SR}(\mathcal{R})$) is sound *for $\mathcal{R}$ if, for any term $s \in T(\mathcal{F}, \mathcal{V})$ and for any term $t \in T(\overline{\mathcal{F}}, \mathcal{V})$, $\langle \overline{s}\rangle \to^*_{\mathbb{SR}(\mathcal{R})} t$ implies $s \to^*_\mathcal{R} \widehat{t}$.*

Note that to define the transformation itself, $\mathcal{R}$ does not need to be strongly or syntactically deterministic. 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 $\langle \overline{s}\rangle \to^*_{\mathbb{SR}(\mathcal{R})} t$. It is clear that for any reachable term $t \in T(\overline{\mathcal{F}}, \mathcal{V})$, any term $t' \in T(\overline{\mathcal{F}}, \mathcal{V})$ with $t \to^*_{\mathbb{SR}(\mathcal{R})} t'$ is reachable, and also that any reachable term is not rooted by either $\bot$ or tuple symbols $[\,]^\rho_j$. In the following, for the extended signature $\overline{\mathcal{F}}$, we only consider reachable terms because it suffices to consider them in discussing soundness and confluence below.

Theorem 2 ([14, Section 6])**.** *Let $\mathcal{R}$ be a $\mathbb{U}$-LL SDCTRS.*

(a) For all ground terms $s, t \in T(\mathcal{F})$, if $s \to_{\mathcal{R}}^{*} t$, then $\langle \overline{s} \rangle \to_{\mathbb{SR}(\mathcal{R})}^{*} \langle \overline{t} \rangle$.

(b) If $\mathbb{SR}(\mathcal{R})$ is ground confluent (on reachable terms), then $\mathcal{R}$ is ground confluent.

In Theorem 2, we do not have to take care of groundness as in [10].

Theorem 3 ([8, a revised version])**.** *$\mathbb{SR}$ is sound for WLL and $\mathbb{U}$-WLL SDCTRSs.*

4 Confluence Criterion for WLL and $\mathbb{U}$-WLL SDCTRSs

In the proof of Theorem 3, DCTRSs do not have to be strongly or syntactically deterministic, i.e., $\mathbb{SR}$ is sound for WLL and $\mathbb{U}$-WLL DCTRSs. On the other hand, confluence of WLL and $\mathbb{U}$-WLL DCTRSs is not always preserved by $\mathbb{SR}$. In this section, we show a confluent, WLL, and $\mathbb{U}$-WLL DCTRS that is not an SDCTRS such that confluence is not preserved by $\mathbb{SR}$. We also show that for a WLL and $\mathbb{U}$-WLL SDCTRS $\mathcal{R}$, confluence of $\mathbb{SR}(\mathcal{R})$ ensures that of $\mathcal{R}$.

Example 4. *Consider the following confluent, WLL, and $\mathbb{U}$-WLL DCTRS:*

$$\mathcal{R}_1 = \{\, \mathsf{a} \to \mathsf{a},\ \ \mathsf{a} \to \mathsf{b},\ \ \mathsf{g}(\mathsf{g}(x,x),x) \to \mathsf{b},\ \ \rho_1 : \mathsf{f}(x) \to \mathsf{f}(\mathsf{a}) \Leftarrow x \twoheadrightarrow \mathsf{a},\ \ \rho_2 : \mathsf{f}(x) \to \mathsf{b} \Leftarrow x \twoheadrightarrow \mathsf{b} \,\}$$

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

$$\mathbb{SR}(\mathcal{R}_1) = \left\{ \begin{array}{rlr} \overline{\mathsf{a}} \to \langle \overline{\mathsf{a}} \rangle, \qquad \overline{\mathsf{a}} \to \langle \overline{\mathsf{b}} \rangle, & & \overline{\mathsf{g}}(\overline{\mathsf{g}}(x,x),x) \to \langle \overline{\mathsf{b}} \rangle, \\ \overline{\mathsf{f}}(x,\bot,z_2) \to \overline{\mathsf{f}}(x,[\langle x \rangle]_1^{\rho_1},z_2), & & \overline{\mathsf{f}}(x,z_1,\bot) \to \overline{\mathsf{f}}(x,z_1,[\langle x \rangle]_1^{\rho_2}), \\ \overline{\mathsf{f}}(x,[\langle \overline{\mathsf{a}} \rangle]_1^{\rho_1},z_2) \to \langle \overline{\mathsf{f}}(\overline{\mathsf{a}},\bot,\bot) \rangle, & & \overline{\mathsf{f}}(x,z_1,[\langle \overline{\mathsf{b}} \rangle]_1^{\rho_2}) \to \langle \overline{\mathsf{b}} \rangle, \\ \langle \langle x \rangle \rangle \to \langle x \rangle, & & \overline{\mathsf{f}}(\langle x \rangle,z_1,z_2) \to \langle \overline{\mathsf{f}}(x,\bot,\bot) \rangle, \\ \overline{\mathsf{g}}(\langle x \rangle,y) \to \langle \overline{\mathsf{g}}(x,y) \rangle, & & \overline{\mathsf{g}}(x,\langle y \rangle) \to \langle \overline{\mathsf{g}}(x,y) \rangle \end{array} \right\}$$

$\mathcal{R}_1$ is not an SDCTRS due to the condition $x \twoheadrightarrow \mathsf{a}$, and $\mathbb{SR}(\mathcal{R}_1)$ is not confluent because of a non-joinable critical peak $\overline{\mathsf{f}}(x,[\langle\langle \overline{\mathsf{b}} \rangle\rangle]_1^{\rho_1},z_2) \leftarrow_{\mathbb{SR}(\mathcal{R}_1)} \overline{\mathsf{f}}(x,[\langle \overline{\mathsf{a}} \rangle]_1^{\rho_1},z_2) \to_{\mathbb{SR}(\mathcal{R}_1)} \langle \overline{\mathsf{f}}(\overline{\mathsf{a}},\bot,\bot) \rangle$. Note that there is a non-joinable ground instance of this peak, and thus, $\mathbb{SR}(\mathcal{R}_1)$ is not ground confluent. Note also that both confluence of $\mathcal{R}_1$ and non-confluence of $\mathbb{SR}(\mathcal{R}_1)$ have been proved by the confluence prover ConCon *[15]. On the other hand, the unraveled TRS obtained by applying the improved version [5] of the simultaneous unraveling to $\mathcal{R}_1$ is confluent, and hence we can prove confluence of $\mathcal{R}_1$ by means of the improved unraveling.*

Finally, we show a confluence criterion for WLL and $\mathbb{U}$-WLL SDCTRSs via $\mathbb{SR}$. For normal CTRSs, the following result has been shown in [10].

Theorem 5 ([10, Theorem 3])**.** *For a normal CTRS $\mathcal{R}$, if $\mathbb{SR}(\mathcal{R})$ is sound for $\mathcal{R}$ and confluent (on reachable terms), then $\mathcal{R}$ is confluent.*

The proof of Theorem 5 in [10] does not depend on the definition of $\mathbb{SR}$, and thus, Theorem 5 holds for DCTRSs. As a trivial consequence of Theorems 3 and 5, Theorem 2 (b) holds for WLL and $\mathbb{U}$-WLL SDCTRSs since the proof does not depend on the $\mathbb{U}$-LL property.

Theorem 6. *For a WLL and $\mathbb{U}$-WLL SDCTRS $\mathcal{R}$, if $\mathbb{SR}(\mathcal{R})$ is (ground) confluent (on reachable terms), then $\mathcal{R}$ is (ground) confluent.*

Proof. This proof is exactly the same as that of Theorem 5 in [10] for normal CTRSs. Let s, t_1, and t_2 be (ground) terms in $T(\mathcal{F},\mathcal{V})$ such that $t_1 \leftarrow_{\mathcal{R}}^{*} s \to_{\mathcal{R}}^{*} t_2$. It follows from Theorem 2 (a) that $\langle \overline{t_1} \rangle \leftarrow_{\mathbb{SR}(\mathcal{R})}^{*} \langle \overline{s} \rangle \to_{\mathbb{SR}(\mathcal{R})}^{*} \langle \overline{t_2} \rangle$. It follows from (ground) confluence of $\mathbb{SR}(\mathcal{R})$ that there exists a (ground) 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 $\mathbb{SR}(\mathcal{R})$ (i.e., Theorem 3) that $t_1 \to_{\mathcal{R}}^{*} \widehat{u} \leftarrow_{\mathcal{R}}^{*} t_2$. Therefore, $\mathcal{R}$ is (ground) confluent. $\square$

Since Theorem 3 holds for WLL and $\mathbb{U}$-WLL DCTRSs that are not SDCTRSs, Theorem 6 holds not only for SDCTRSs but also for DCTRSs that are not SDCTRSs.

Every *join 1*-CTRS $\mathcal{R}$ over a signature $\mathcal{F}$ can be transformed into a normal 1-CTRS $\mathcal{R}'$ ($= n(\mathcal{R})$) such that $\to_{\mathcal{R}} = \to_{\mathcal{R}'}$ over $T(\mathcal{F},\mathcal{V})$ [12, Definition 7.1.6 and Proposition 7.1.7], and by definition, it holds that if $\mathcal{R}$ is WLL (i.e., $\mathcal{R}'$ is WLL), then $\mathcal{R}'$ is $\mathbb{U}$-WLL. Therefore, from Theorem 6, it holds that if $\mathcal{R}$ is WLL and $\mathbb{SR}(\mathcal{R}')$ is confluent, then $\mathcal{R}$ is confluent.

It has not been shown yet that $\mathbb{SR}$ preserves confluence of WLL and $\mathbb{U}$-WLL SDCTRSs. One of future work is to prove this conjecture.

Acknowledgments We thank the anonymous reviewers very much for their useful comments and suggestions to improve this paper.

References

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

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

[3] K. Gmeiner, B. Gramlich, and F. Schernhammer. On soundness conditions for unraveling deterministic conditional rewrite systems. In *Proc. RTA 2012*, volume 15 of *LIPIcs*, pp. 193–208. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012.

[4] K. Gmeiner and N. Nishida. Notes on structure-preserving transformations of conditional term rewrite systems. In *Proc. WPTE 2014*, volume 40 of *OASIcs*, pp. 3–14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014.

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

[6] M. Marchiori. Unravelings and ultra-properties. In *Proc. ALP 1996*, volume 1139 of *LNCS*, pp. 107–121. Springer, 1996.

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

[8] R. Nakayama, N. Nishida, and M. Sakai. Sound structure-preserving transformation for ultra-weakly-left-linear deterministic conditional term rewriting systems. In *Informal Proc. WPTE 2016*, pp. 61–75, 2016. A revised version is available from `http://www.apal.i.is.nagoya-u.ac.jp/~nishida/wpte16/`.

[9] 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.

[10] N. Nishida, M. Yanagisawa, and K. Gmeiner. On proving confluence of conditional term rewriting systems via the computationally equivalent transformation. In *Proc. IWC 2014*, pp. 24–28, 2014.

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

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

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

[14] 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.

[15] T. Sternagel and A. Middeldorp. Conditional Confluence (System Description). In *Proc. RTA-TLCA 2014*, volume 8560 of *LNCS*, pp. 456–465. Springer, 2014.

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

Conditions for confluence of innermost terminating term rewriting systems

Sayaka Ishizuki, Michio Oyamaguchi, and Masahiko Sakai

Nagoya University, Nagoya, Japan

Abstract

We present a counterexample for the open problem whether innermost joinability of all critical pairs ensures confluence of innermost terminating term rewriting systems. We then show that innermost joinability of all normal instances of the critical pairs is a necessary and sufficient condition. We also show a decidable sufficient condition for confluence of innermost terminating systems.

1 Introduction

B. Gramlich [2] has shown that innermost terminating, locally confluent overlay term rewriting systems (TRSs) are terminating and confluent. But, in the case of non-overlay TRSs the same condition does not necessarily ensure confluence or termination. E. Ohlebusch [4] has posed an open problem concerning this subject: *Is an innermost terminating TRS confluent when its every critical pair is innermost joinable?*

In this article, we give a negative answer to this open problem. We then give a necessary and sufficient condition for confluence of innermost terminating TRSs. That is, an innermost terminating TRS is confluent if and only if all normal substitution instances of its every critical pair are innermost joinable. We also show a decidable sufficient condition for confluence of innermost terminating TRSs by strengthening the condition.

2 Preliminaries

We follow [1] for fundamental notations and definitions. $\mathrm{Pos}(t)$ represents the set of *positions* of a term t and $\mathrm{Pos}_F(t)$ represents the set of positions of function symbols of t. For positions p and p', we write $p \geq p'$ when $p = p'.q$ with some position q, and they are in *parallel positions* if neither $p \geq p'$ nor $p \leq p'$ holds.

For a *substitution* σ, its domain is defined as $\mathrm{Dom}(\sigma) = \{x \in \mathcal{V} \mid x\sigma \neq x\}$. $\sigma \leq \sigma'$ means that $\sigma\theta = \sigma'$ for some substitution θ. When $\mathrm{Dom}(\sigma) \cap \mathrm{Dom}(\sigma') = \emptyset$, the union of two substitutions $\sigma \cup \sigma'$ is naturally defined as: $x(\sigma \cup \sigma')$ is $x\sigma'$ if $x \in \mathrm{Dom}(\sigma')$, and $x\sigma$ otherwise.

We write the rewrite relation of a *term rewriting system* (TRS) $\mathcal{R}$ by $\to_{\mathcal{R}}$, where $\mathcal{R}$ can be omitted as $\to$. We write $s \leftrightarrow t$ if either $s \to t$ or $s \leftarrow t$. We say that terms s and s' are *joinable*, written as $s \downarrow s'$, if $s \xrightarrow{*} t$ and $s' \xrightarrow{*} t$ for some term t. A rewrite relation $\to$ is *locally confluent* if $(\leftarrow \cdot \to) \subseteq \downarrow$, *confluent* if $(\xleftarrow{*} \cdot \xrightarrow{*}) \subseteq \downarrow$, and *Church-Rosser* if $\xleftrightarrow{*} \subseteq \downarrow$. Confluence and Church-Rosser properties are equivalent. A rewrite relation $\to$ is *terminating* if it admits no infinite sequence $t_0 \to t_1 \to \cdots$. A substitution is *normal* if all the substituted terms are in normal form. A *rewrite step* $t[l\sigma]_p \to t[r\sigma]_p$ is *innermost*, if any proper subterm of $l\sigma$ is in normal form. We write $\to_{\mathrm{i}}$ if the step is innermost.

A substitution τ is a *unifier* of terms s and t if $s\tau = t\tau$. Let τ be a unifier of terms s and t. If $\tau \leq \tau'$ for any unifier τ' of s and t, then we say τ a *most general unifier* (*mgu* for short) of s and t. Let $l_1 \to r_1$ and $l_2 \to r_2$ be rules in a rewrite system whose variables have been renamed as $\mathrm{Var}(l_1) \cap \mathrm{Var}(l_2) = \emptyset$. If for $p \in \mathrm{Pos}_F(l_1)$ there exists an mgu τ of $l_1|_p$ and l_2, then $\langle l_1\tau[r_2\tau]_p, r_1\tau\rangle$ is a *critical pair.* If $p = \varepsilon$, the pair is *overlay.* A TRS is called *overlay* if every critical pair is overlay. We write $\mathrm{CP}_{\mathcal{R}}$ to indicate the set of critical pairs of TRS $\mathcal{R}$.

For terms s, t, and parallel positions $p_0, \ldots, p_n \in \mathrm{Pos}(s)$, if $s = s[s_0]_{p_0} \cdots [s_n]_{p_n}$, $t = s[t_0]_{p_0} \cdots [t_n]_{p_n}$, and $s_i \leftrightarrow t_i$ $(0 \le i \le n)$, then we write $s \mathrel{+\!\!\!\!\leftrightarrow} t$, and call it a *parallel step.*

Example 1. *The following $\mathcal{R}_1$ is innermost terminating, locally confluent, and not overlay, but it is neither confluent nor terminating.*

$$\begin{aligned} \mathcal{R}_1 &= \{f(c) \to g(c),\ g(c) \to f(c),\ c \to d\} \\ \mathrm{CP}_{\mathcal{R}_1} &= \{\langle f(d), g(c)\rangle,\ \langle g(d), f(c)\rangle\} \end{aligned}$$

3 A counterexample to the conjecture

The following open problem is a variant of the famous result on the confluence for terminating TRSs by Knuth and Bendix [3].

Conjecture 2 ([4])**.** *Let $\mathcal{R}$ be an innermost terminating TRS. If $u \downarrow_{\mathrm{i}} v$ for every critical pair $\langle u, v\rangle$ of $\mathcal{R}$, then $\mathcal{R}$ is confluent.*

This open problem is negatively solved by the following example.

Example 3.

$$\begin{aligned} \mathcal{R}_2 = \{\ & g(x) \to h(k(x)),\ g(x) \to x,\ h(k(x)) \to f(x), \\ & f(x) \to x,\ k(c) \to c,\ f(c) \to g(c)\} \\ \mathrm{CP}_{\mathcal{R}_2} = \{\ & \langle x, h(k(x))\rangle,\ \langle h(c), f(c)\rangle,\ \langle c, g(c)\rangle\ \} \end{aligned}$$

$\mathcal{R}_2$ is innermost terminating and every critical pair of $\mathcal{R}_2$ is innermost joinable. $\mathcal{R}_2$ is, however, not confluent, since $c \overset{*}{\leftrightarrow} h(c)$ but c and $h(c)$ are not joinable.

4 A necessary and sufficient condition for confluence of innermost terminating TRSs

This section shows that an innermost terminating $\mathcal{R}$ is confluent if every critical pair $\langle u, v\rangle$ of $\mathcal{R}$ is *innermost joinable for normal instances* (IJN); $u\sigma \downarrow_{\mathrm{i}} v\sigma$ for any normal substitution σ. Henceforth, we assume that $\mathcal{R}$ is innermost terminating.

First, we give a lemma that decomposes confluence to two properties.

Lemma 4. $\mathrm{CR}(\to)$ *if and only if* $\mathrm{CR}(\to_{\mathrm{i}})$ *and* $\to \subseteq \downarrow_{\mathrm{i}}$

Proof. Only-if-part. Suppose $s \to t$ is a non-innermost step. Since $\mathcal{R}$ is innermost terminating, we can write $s' \overset{*}{\leftarrow}_{\mathrm{i}} s \to t \overset{*}{\to}_{\mathrm{i}} t'$ for normal forms s' and t'. From $\mathrm{CR}(\to)$, $s' = t'$ hence $s \downarrow_{\mathrm{i}} t$. Similarly supposing $s \overset{*}{\leftrightarrow}_{\mathrm{i}} t$, we obtain $s' \overset{*}{\leftarrow}_{\mathrm{i}} s \overset{*}{\leftrightarrow}_{\mathrm{i}} t \overset{*}{\to}_{\mathrm{i}} t'$ for normal forms s' and t', and $s' = t'$. Thus $s \overset{*}{\to}_{\mathrm{i}} \cdot \overset{*}{\leftarrow}_{\mathrm{i}} t$.

If-part. We show that $s \overset{n}{\leftrightarrow} t$ implies $s \downarrow_{\mathrm{i}} t$ by induction on n. Since the case $n = 0$ is trivial, we consider $n > 0$. We can write $s \leftrightarrow s' \overset{n-1}{\leftrightarrow} t$ for some term s'. We obtain $s \downarrow_{\mathrm{i}} s'$ by $\to \subseteq \downarrow_{\mathrm{i}}$. and $s' \downarrow_{\mathrm{i}} t$ by the induction hypothesis. Since $\mathrm{CR}(\to_{\mathrm{i}})$, it follows that $s \downarrow_{\mathrm{i}} t$. Therefore $\mathrm{CR}(\to)$ holds. $\square$

We define the following condition PIJN, which looks weaker than but is equivalent to IJN for innermost terminating TRSs.

Definition 5. *A critical pair* $\langle u, v\rangle$ *is* pseudo-innermost-joinable for normal instances (PIJN) *if* $u\sigma \xrightarrow{*}_{\mathrm{i}} \cdot \overset{=}{\leftrightarrow} \cdot \xleftarrow{*}_{\mathrm{i}} v\sigma$ *for every normal substitution* σ*. If all critical pairs of* $\mathcal{R}$ *are PIJN,* $\mathcal{R}$ *is* PIJN [1].

Lemma 6. *Let* $l \to r$ *be a rule in* $\mathcal{R}$*, and* σ *be a normal substitution. If* $l\sigma \to r\sigma$ *is a non-innermost step, then there exist a critical pair* $\langle u, v\rangle$ *and a normal substitution* θ *such that*

$$l\sigma \to_{\mathrm{i}} u\theta \quad \textit{and} \quad r\sigma = v\theta.$$

Proof. Since $l\sigma \to r\sigma$ is a non-innermost step and σ is a normal substitution, there exists an innermost step $l\sigma = l\sigma[l'\sigma']_p \to_{\mathrm{i}} l\sigma[r'\sigma']_p$ for some rule $l' \to r' \in \mathcal{R}$, substitution σ', and rewriting position $p > \varepsilon$ which is in $\mathrm{Pos}_F(l)$. This means that there exists an mgu τ of $l|_p$ and l' hence $\langle l\tau[r'\tau]_p, r\tau\rangle$ is a critical pair. Suppose σ'' is a unifier of $l|_p$ and l' such that $\sigma'' = \sigma \cup \sigma'$ where $\mathrm{Dom}(\sigma) \cap \mathrm{Dom}(\sigma') = \emptyset$. Since τ is an mgu and σ'' is a unifier of $l|_p$ and l', there exists a substitution θ' such that $\sigma'' = \tau\theta'$. This implies that

$$\begin{aligned} &l\sigma = l\sigma[l'\sigma']_p = l\tau\theta'[l'\tau\theta']_p = (l\tau[l'\tau]_p)\theta', \quad \text{and} \\ &l\sigma = l\tau\theta'. \end{aligned}$$

Therefore,

$$\begin{aligned} &l\sigma = (l\tau[l'\tau]_p)\theta' \to (l\tau[r'\tau]_p)\theta', \quad \text{and} \\ &l\sigma = (l\tau)\theta' \to (r\tau)\theta' = r\sigma. \end{aligned}$$

Now we show that $x\theta'$ is in normal form for any variable $x \in \mathrm{Var}(l\tau[l'\tau]_p) \cup \mathrm{Var}(r\tau)$. Since $\mathrm{Var}(r) \subseteq \mathrm{Var}(l)$, it is enough to show that $y\theta'$ is in normal form for any variable $y \in \mathrm{Var}(l\tau[l'\tau]_p) \cup \mathrm{Var}(l\tau) = \mathrm{Var}(l'\tau) \cup \mathrm{Var}(l\tau)$. $\mathrm{Dom}(l'\tau) = \mathrm{Dom}(l|_p\tau) \subseteq \mathrm{Dom}(l\tau)$ hence $\mathrm{Dom}(l'\tau) \subseteq \mathrm{Dom}(l\tau)$. Thus we only need to see if $y\theta'$ is in normal form where $y \in \mathrm{Dom}(l\tau)$, and in this case $y\theta'$ is indeed in normal form since $l\sigma = l\tau\theta'$ and σ is a normal substitution. From this fact, a substitution θ such that $\theta = \theta'|_{\mathrm{Var}(l\tau[r'\tau]_p)\cup\mathrm{Var}(r\tau)}$ is a normal substitution. ☐

Lemma 7. *Let* $\mathcal{R}$ *be PIJN. If* $s \to t$*, then* $s \downarrow_{\mathrm{i}} t$*.*

Proof. We show that if $s \to t$ then $s \downarrow_{\mathrm{i}} t$, by Noetherian induction on $\{s, t\}$ with respect to the multiset extension of $\xrightarrow{+}_{\mathrm{i}}$. Here we write $>^{\mathrm{i}}_{mul}$ for the multiset extension.

If $s \to t$ is an innermost step, it is trivial. Suppose $s \to t$ is not innermost. Then, for a substitution σ and a rule $l \to r \in \mathcal{R}$, terms s and t are represented by $s[l\sigma]_p$ and $s[r\sigma]_p$, respectively.

If $x\sigma$ is not in normal form for some $x \in \mathrm{Var}(l)$, there exists an innermost derivation $s = s[l\sigma]_p \xrightarrow{+}_{\mathrm{i}} s[l\sigma']_p = s'$ for some substitution σ', hence the derivation $t = s[r\sigma]_p \xrightarrow{*}_{\mathrm{i}} s[r\sigma']_p = t'$ is also possible. Since $s' \to t'$ and $\{s, t\} >^{\mathrm{i}}_{mul} \{s', t'\}$, we have $s' \downarrow_{\mathrm{i}} t'$ by induction hypothesis. Thus $s \downarrow_{\mathrm{i}} t$.

Otherwise, by Lemma 6, there exist a critical pair $\langle u, v\rangle$ and a normal substitution θ such that $s \to_{\mathrm{i}} s[u\theta]_p$ and $t = s[v\theta]_p$. We use s' to represent $s[u\theta]_p$. Since θ is a normal substitution, $s' \xrightarrow{*}_{\mathrm{i}} s'' \overset{=}{\leftrightarrow} t' \xleftarrow{*}_{\mathrm{i}} t$ holds for some terms s'' and t' from the PIJN property. In the case of $s'' = t'$, we have done. Otherwise, $s'' \to t'$ or $t' \to s''$ hold. Since $\{s, t\} >^{\mathrm{i}}_{mul} \{s'', t'\}$, by induction hypothesis we have $s'' \downarrow_{\mathrm{i}} t'$. Therefore $s \downarrow_{\mathrm{i}} t$. ☐

---

[1] In the condition, we can restrict critical pairs to prime critical pairs [5]; a critical pair $\langle u, v\rangle$ is *prime* if $(u, v) \in (\leftarrow_{\mathrm{i}} \cdot \to)$ Moreover, $\overset{=}{\leftrightarrow}$ can be relaxed to parallel step $\nleftrightarrow$.

Lemma 8. *Let $\mathcal{R}$ be PIJN. If $u \leftarrow_{\mathrm{i}} \cdot \rightarrow_{\mathrm{i}} v$, then $u \downarrow_{\mathrm{i}} v$.*

Proof. If the rewriting steps to u and v occur at the same position by different rules, u and v can be represented by $u[u'\theta]_p$ and $u[v'\theta]_p$ respectively for some $\langle u', v'\rangle \in \mathrm{CP}_{\mathcal{R}}$ and substitution θ. Since u and v are obtained by innermost rewriting, θ is in normal form. (The proof is similar to that of Lemma 6.) Hence, we have $u = u[u'\theta]_p \xrightarrow{*}_{\mathrm{i}} \cdot \overset{=}{\leftrightarrow} \cdot \xleftarrow{*}_{\mathrm{i}} u[v'\theta]_p = v$, so that by Lemma 7, $u \downarrow_{\mathrm{i}} v$ holds.

Otherwise, the rewriting steps to u and v occur at parallel positions. Therefore, there exists a term t such that $u \rightarrow_{\mathrm{i}} t \leftarrow_{\mathrm{i}} v$. □

Lemma 9. *Let $\mathcal{R}$ be PIJN. Then* $\mathrm{CR}(\rightarrow_{\mathrm{i}})$.

Proof. By the precondition, $\rightarrow_{\mathrm{i}}$ is terminating and we know that $\rightarrow_{\mathrm{i}}$ is locally confluent by Lemma 8. By Newman's lemma, $\rightarrow_{\mathrm{i}}$ is confluent. Hence, $\mathrm{CR}(\rightarrow_{\mathrm{i}})$. □

Combining Lemmas 4, 7, and 9, the following theorem follows.

Theorem 10. *Let a TRS $\mathcal{R}$ be innermost terminating. Then, $\mathcal{R}$ is confluent if and only if $\mathcal{R}$ is PIJN, i.e., all critical pairs are pseudo-innermost-joinable for normal instances.*

Corollary 11. *Let a TRS $\mathcal{R}$ be innermost terminating. $\mathcal{R}$ is PIJN if and only if all normal instances $\langle u\sigma, v\sigma\rangle$ of every critical pairs $\langle u, v\rangle$ of $\mathcal{R}$ are innermost joinable.*

Proof. The if-part is obvious. The only-if-part is shown by Theorem 10. □

This corollary shows a condition for confluence of innermost terminating TRSs, which has a similar form to the condition suggested in the open problem.

5 A sufficient condition

At present, the problem deciding whether $\mathcal{R}$ is PIJN remains open. This section introduces a decidable sufficient condition for confluence of innermost terminating TRSs.

If $s = s[l\sigma]_p \rightarrow_{\mathrm{i}} s[r\sigma]_p = t$ and $l\sigma$ is a ground term, we write $s \rightarrow_{\mathrm{gi}} t$.

Definition 12. *A critical pair $\langle u, v\rangle$ is* pseudo-innermost-ground-joinable (PIJ-ground) *if $u \xrightarrow{*}_{\mathrm{gi}} \cdot \overset{=}{\leftrightarrow} \cdot \xleftarrow{*}_{\mathrm{gi}} v$. If all critical pairs of $\mathcal{R}$ are PIJ-ground, $\mathcal{R}$ is* PIJ-ground.

Since a PIJ-ground TRS is obviously PIJN, we have the following corollary.

Corollary 13. *Let a TRS $\mathcal{R}$ be innermost terminating. Then, $\mathcal{R}$ is confluent if $\mathcal{R}$ is PIJ-ground, i.e., all critical pairs are pseudo-innermost-ground-joinable.*

Example 14. *The following $\mathcal{R}_3$ is innermost terminating, and PIJ-ground, so that it is confluent by the corollary.*

$$\begin{aligned} \mathcal{R}_3 = \{\ & g(x) \rightarrow h(k(x), x),\ g(x) \rightarrow x,\ h(k(x), x) \rightarrow x, \\ & k(c) \rightarrow c,\ h(k(c), c) \rightarrow g(c),\ h(c, c) \rightarrow c\ \} \\ \mathrm{CP}_{\mathcal{R}_3} = \{\ & \langle x, h(k(x), x)\rangle,\ \langle h(c, c), c\rangle,\ \langle h(c, c), g(c)\rangle,\ \langle c, g(c)\rangle \} \end{aligned}$$

Note that we have tried to show confluence of TRS $\mathcal{R}_3$ by confluence checker ACP [6] and Saigawa [7], and both of them failed.

6 Conclusion

We have given a negative answer to the open problem posed by E. Ohlebusch [4], and shown some conditions necessary and sufficient for confluence of innermost terminating TRSs. Using one of the conditions, we have given a decidable sufficient condition for the confluence. At present, the problem of deciding whether an innermost terminating TRS is confluent, remains open.

Acknowledgements

The authors are grateful to Yoshihito Toyama, Aart Middeldorp, Nao Hirokawa, Takahito Aoto, and the anonymous reviewers for their valuable comments.

References

[1] N. Dershowitz, J.-P. Jouannaud, Rewrite Systems, in J. van Leeuwen, ed., *Handbook of Theoretical Computer Science* , MIT Press, pp. 243–320, 1990.

[2] B. Gramlich, Abstract Relations between Restricted Termination and Confluence Properties of Rewrite Systems, *Fundamenta Informaticae* 24, pp. 3–23, 1995.

[3] D. E. Knuth and P. B. Bendix, Simple Word Problems in Universal Algebras, in J. Leech, ed., *Computational problems in abstract algebra*, Pergamon Press, pp. 263–297, 1970.

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

[5] D. Kapur, D.R. Musser, and P. Narendran, Only prime superpositions need be considered in the Knuth-Bendix completion procedure, *Journal of Symbolic Computation* 6 (1), pp. 19–36, 1988.

[6] T. Aoto, J. Yoshida, Y. Toyama, Proving confluence of term rewriting systems automatically, In *Proc. of RTA 2009*, *LNCS*, 5595, pp. 93–102, 2009.

[7] N. Hirokawa and D. Klein, Saigawa: A confluence tool, In *Proc. of 1st IWC*, 49, 2012.

ACP: System Description for CoCo 2016

Takahito Aoto[1] and Yoshihito Toyama[2]

[1] Faculty of Engineering, Niigata University
aoto@ie.niigata-u.ac.jp
[2] RIEC, Tohoku University
toyama@nue.riec.tohoku.ac.jp

ACP is an automated confluence prover for term rewriting systems (TRSs) that has been developed in Toyama–Aoto group in RIEC, Tohoku University. ACP integrates multiple direct criteria for guaranteeing confluence of TRSs. It incorporates divide–and–conquer criteria by which confluence or non-confluence of TRSs can be inferred from those of their components. Several methods for disproving confluence are also employed. A list of implemented criteria and methods can be found on the website of ACP [1]. For a TRS to which direct confluence criteria do not apply, the prover decomposes it into components using divide–and–conquer criteria, and tries to apply direct confluence criteria to each component. Then the prover combines these results to infer the (non-)confluence of the whole system.

ACP is written in Standard ML of New Jersey (SML/NJ) and is provided as a heap image that can be loaded into SML/NJ runtime systems. It uses a SAT prover such as MiniSAT and an SMT prover YICES as external provers. It internally contains an automated (relative) termination prover for TRSs but external (relative) termination provers can be substituted optionally. The input TRS is specified in the (old) TPDB format. Users can specify criteria to be used so that each criterion or any combination of them can be tested. Several levels of verbosity are available for the output so that users can investigate details of the employed approximations for each criterion or can get only the final result of prover's attempt. For some criteria, it supports generation of proofs in CPF format that can be certified by certifiers. The source code and a list of implemented criteria are found on the webpage [1].

The internal structure of the prover is kept simple and is mostly inherited from the version 0.11a, which has been described in [2]. No new (non-)confluence criterion has been incorporated from the one submitted for CoCo 2015.

References

[1] ACP (Automated Confluence Prover). http://www.nue.riec.tohoku.ac.jp/tools/acp/.

[2] T. Aoto, J. Yoshida, and Y. Toyama. Proving confluence of term rewriting system automatically. In *Proc. of 20th RTA*, volume 5595 of *LNCS*, pages 93–102. Springer-Verlag, 2009.

ACPH: System Description for CoCo 2016

Kouta Onozawa[1], Kentaro Kikuchi[1], Takahito Aoto[2], and Yoshihito Toyama[1]

[1] RIEC, Tohoku University
{ onozawa, kentaro, toyama }@nue.riec.tohoku.ac.jp
[2] Faculty of Engineering, Niigata University
aoto@ie.niigata-u.ac.jp

Higher-order rewriting systems (HRSs) is a formalism of rewriting with variable binding and higher-order functions [2]. Higher-order rewriting deals with simply-typed lambda-terms with constants, which are identified modulo $\beta\eta$-equality. HRSs are a set of rewrite rules whose left-hand sides are restricted to patterns.

ACPH (Automated Confluence Prover for HRSs) is a tool for proving confluence of HRSs. If the tool succeeds to prove that an input HRS is confluent, it outputs YES. If the tool succeeds to prove that an input HRS is not confluent, it outputs NO. If the tool can not determine whether an input HRS is confluent or not, it outputs MAYBE. The tool uses following criteria for proving confluence and non-confluence of HRSs [1].

- If a HRS $\mathcal{R}$ is weakly orthogonal (left-linear and all critical pairs are trivial), then $\mathcal{R}$ is confluent.
- If a HRS $\mathcal{R}$ is terminating, then all critical pairs are joinable iff $\mathcal{R}$ is confluent.

The algorithms used in the program are based on those described in [1, 2]. For proving termination of HRSs, a higher-order termination tool WANDA[3] is used. ACPH program is written in Standard ML of New Jersey, and ACPH is provided as a heap image that can be loaded into SML/NJ runtime systems. It can be used from the command line by typing the following command:

```
$ sml @SMLload=acph.x86-linux <filename>
```

A bug that has been indentified at CoCo 2015 has been fixed in the submitted version.

References

[1] Tobias Nipkow, Functional unification of higher-order patterns, *Proceedings of eighth annual IEEE symposium on logic in computer science*, pp.64-74, 1993.

[2] Richard Mayr, Tobias Nipkow, Higher-order rewrite systems and their confluence, *Theoretical computer science 192*, pp. 3-29, 1998.

[3] WANDA: A Higher-Order Termination Tool, http://wandahot.sourceforge.net/index.html

AGCP: System Description for CoCo 2016

Takahito Aoto[1] and Yoshihito Toyama[2]

[1] Faculty of Engineering, Niigata University
aoto@ie.niigata-u.ac.jp
[2] RIEC, Tohoku University
toyama@nue.riec.tohoku.ac.jp

A many-sorted term rewriting system is said to be *ground confluent* if all ground terms are confluent. AGCP (Automated Groud Confluence Prover) [1] is a tool for proving ground confluence of many-sorted term rewriting systems. AGCP is written in Standard ML of New Jersey (SML/NJ). The tool is registered to the category of ground confluence of many-sorted term rewriting systems that has been adapted as one of the demonstration categories in CoCo 2016.

AGCP proves ground confluence of many-sorted term rewriting systems based on two ingredients. One ingredient is to divide the ground confluence problem of a many-sorted term rewriting system $\mathcal{R}$ into that of $\mathcal{S} \subseteq \mathcal{R}$ and the inductive validity problem of equations $u \approx v$ w.r.t. $\mathcal{S}$ for each $u \to r \in \mathcal{R} \setminus \mathcal{S}$. Here, an equation $u \approx v$ is inductively valid w.r.t. $\mathcal{S}$ if all its ground instances $u\sigma \approx v\sigma$ is valid w.r.t. $\mathcal{S}$, i.e. $u\sigma \overset{*}{\leftrightarrow}_{\mathcal{S}} v\sigma$. Another ingredient is to prove ground confluence of a many-sorted term rewriting system via the *bounded ground convertibility* of the critical pairs. Here, an equation $u \approx v$ is said to be bounded ground convertibile w.r.t. a quasi-order $\gtrsim$ if $u\theta_g \xleftrightarrow[\gtrsim]{*}_{\mathcal{R}} v\theta_g$ for any its ground instance $u\sigma_g \approx v\sigma_g$, where $x \xleftrightarrow[\gtrsim]{*} y$ iff there exists $x = x_0 \leftrightarrow \cdots \leftrightarrow x_n = y$ such that $x \gtrsim x_i$ or $y \gtrsim x_i$ for every x_i.

Rewriting induction [2] is a well-known method for proving inductive validity of many-sorted term rewriting systems. In [1], an extension of rewriting induction to prove bounded ground convertibility of the equations has been reported. Namely, for a reduction quasi-order $\gtrsim$ and a quasi-reducible many-sorted term rewriting system $\mathcal{R}$ such that $\mathcal{R} \subseteq \succ$, the extension proves bounded ground convertibility of the input equations w.r.t. $\gtrsim$. The extension not only allows to deal with non-orientable equations but also with many-sorted TRSs having non-free constructors. AGCP uses this extension of the rewriting induction to prove not only inductive validity of equations but also the bounded ground convertibility of the critial pairs.

References

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

[2] U. S. Reddy. Term rewriting induction. In *Proc. of CADE-10*, volume 449 of *LNAI*, pages 162–177. Springer-Verlag, 1990.

CoCo 2016 Participant: CeTA 2.28*

Julian Nagele, Christian Sternagel, and Thomas Sternagel

Department of Computer Science, University of Innsbruck, Austria

Automatic provers have become popular in many areas like theorem proving, SMT, etc. Since such provers are complex pieces of software, they might contain errors that lead to wrong answers, i.e., incorrect proofs. Therefore, certification of the generated proofs is of major importance.

The tool CeTA [7] is a certifier that can be used to certify confluence and non-confluence proofs of term rewrite systems (TRSs) and conditional term rewrite systems (CTRSs). Its soundness is proven as part of IsaFoR, the *Isabelle Formalization of Rewriting*. The following techniques are currently supported in CeTA—for further details we refer to the certification problem format (CPF) and to the sources of IsaFoR and CeTA (`http://cl-informatik.uibk.ac.at/software/ceta/`).

Term rewrite systems. Since CeTA was originally conceived for termination analysis, our first method is Newman's lemma in combination with the critical pair theorem. For possibly non-terminating TRSs, CeTA can ensure that weakly orthogonal, strongly closed, and almost parallel closed TRSs are confluent [4], as well as check applications of the rule labeling heuristic [5] and addition and removal of redundant rules [3]. To certify non-confluence one can provide a divergence and a certificate for non-joinability. Here CeTA supports: distinct normal forms, *tcap*, usable rules, discrimination pairs, argument filters and interpretations [1], and reachability analysis using tree automata techniques [2].

Conditional term rewrite systems. Since last year CeTA also supports confluence criteria for conditional rewriting. CeTA can certify that almost orthogonal, extended properly oriented, right-stable 3-CTRSs are confluent, including support for infeasible critical pairs, where the supported justification is a certificate for non-reachability using either *tcap* or tree automata [6]. The second supported technique for CTRSs is unraveling [8], transforming the system into a TRS where then the aforementioned techniques can be certified.

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, confluence, and termination analysis with state-compatible automata. *I&C*, 2016. Available Online.

[3] J. Nagele, B. Felgenhauer, and A. Middeldorp. Improving automatic confluence analysis of rewrite systems by redundant rules. In *RTA*, volume 36 of *LIPIcs*, pages 257–268, 2015.

[4] J. Nagele and A. Middeldorp. Certification of classical confluence results for left-linear term rewrite systems. In *ITP*, volume 9807 of *LNCS*, pages 290–306, 2016.

[5] J. Nagele and H. Zankl. Certified rule labeling. In *RTA*, volume 36 of *LIPIcs*, pages 269–284, 2015.

[6] C. Sternagel and T. Sternagel. Certifying confluence of almost orthogonal CTRSs via exact tree automata completion. In *FSCD*, volume 52 of *LIPIcs*, pages 29:1–29:16, 2016.

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

[8] S. Winkler and R. Thiemann. Formalizing soundness and completeness of unravelings. In *FroCoS*, volume 9322 of *LNCS (LNAI)*, 2015.

*Supported by Austrian Science Fund (FWF), projects P27502, and P27528.

CO3 (Version 1.3)

Naoki Nishida[1], Takayuki Kuroda[1], and Karl Gmeiner[2]

[1] Nagoya University, Nagoya, Japan
{nishida@, kuroda@apal.i.}is.nagoya-u.ac.jp
[2] UAS Technikum Wien, Vienna, Austria
gmeiner@technikum-wien.at

CO3, a converter for proving confluence of conditional TRSs, is a tool for proving confluence of conditional term rewriting systems (CTRS) by using a transformational approach. The tool is based on the result in [6, 1, 4]: the tool first transforms a given *weakly-left-linear (WLL) and ultra-WLL 3-DCTRS* into an unconditional term rewriting system (TRS) by using the *SR transformation* $\mathbb{SR}$ [8, 9, 3] or the *unraveling* $\mathbb{U}$ [2, 7], and then verify confluence of the transformed TRS. This tool is basically a converter of CTRSs to TRSs. The main expected use of this tool is the collaboration with other tools for proving confluence of TRSs, and thus this tool has very simple and lightweight functions to verify properties such as confluence and termination of TRSs. The tool is available from http://www.trs.cm.is.nagoya-u.ac.jp/co3/.

The main technique for proving confluence of CTRSs is based on the following theorem: a weakly left-linear normal 1-CTRS $\mathcal{R}$ is confluent if one of $\mathbb{SR}(\mathcal{R})$ and $\mathbb{U}(\mathcal{R})$ is confluent [6]. The other important features can be seen in a system description of the previous version [5].

The new feature is to adapt the main technique to WLL and ultra-WLL 3-DCTRSs. More precisely, the implementation of the SR transformation and the unraveling are adapted to 3-DCTRSs [3, 1], and the following theorems are introduced: a WLL 3-DCTRS $\mathcal{R}$ is confluent if $\mathbb{U}(\mathcal{R})$ is confluent [1]; a WLL and ultra-WLL 3-DCTRS $\mathcal{R}$ is confluent if $\mathbb{SR}(\mathcal{R})$ is confluent [4].

References

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

[2] M. Marchiori. Unravelings and ultra-properties. In *Proc. ALP 1996*, volume 1139 of *LNCS*, pp. 107–121. Springer, 1996.

[3] R. Nakayama, N. Nishida, and M. Sakai. Sound structure-preserving transformation for ultra-weakly-left-linear deterministic conditional term rewriting systems. In *Informal Proc. WPTE 2016*, pp. 61–75, 2016.

[4] N. Nishida. Notes on confluence of ultra-WLL SDCTRSs via a structure-preserving transformation. In *Proc. IWC 2016*, 2016. to appear.

[5] N. Nishida, T. Kuroda, M. Yanagisawa, and K. Gmeiner. CO3: a COnverter for proving COfluence of COnditional TRSs (Version 1.2). In *Proc. IWC 2015*, p. 42, 2015.

[6] N. Nishida, M. Yanagisawa, and K. Gmeiner. On proving confluence of conditional term rewriting systems via the computationally equivalent transformation. In *Proc. IWC 2014*, pp. 24–28, 2014.

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

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

[9] 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.

CoLL-Saigawa: A Joint Confluence Tool*

Nao Hirokawa and Kiraku Shintani

JAIST, Japan

CoLL-Saigawa is a tool for automatically proving or disproving confluence of (ordinary) term rewrite systems (TRSs). The tool, written in OCaml, is freely available from:

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

The typical usage is: `collsaigawa <file>`. Here the input file is written in the standard WST format. The tool outputs `YES` if confluence of the input TRS is proved, `NO` if non-confluence is shown, and `MAYBE` if the tool does not reach any conclusion.

CoLL-Saigawa is a joint confluence tool of CoLL v1.1 [8] and Saigawa v1.8 [4]. If an input TRS is left-linear, CoLL proves confluence. Otherwise, Saigawa analyzes confluence. CoLL is a confluence tool specialized for left-linear TRSs. It proves confluence by using Hindley's commutation theorem [3] together with the three commutation criteria: Development closeness [2, 9], rule labeling with weight function [10, 1], and Church-Rosser modulo A/C [6]. Saigawa can deal with non-left-linear TRSs. The tool employs the four confluence criteria: The criteria based on critical pair systems [5, Theorem 3] and on extended critical pairs [7, Theorem 2], rule labeling [10], and Church-Rosser modulo AC [6]. Saigawa uses $\mathsf{T_TT_2}$ and MU-TERM to check (relative) termination.[1] A suitable rule labeling is searched by using MiniSmt.[2]

This version of CoLL-Saigawa is still at the experimental stage. Full integration of the two tools is planned for the next version.

References

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

[2] 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.

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

[4] N. Hirokawa. Saigawa: A confluence tool. In *3rd Confluence Competition (CoCo 2014)*, pages 1–1, 2014.

[5] N. Hirokawa and A. Middeldorp. Commutation via relative termination. In *Proc. 2nd IWC*, pages 29–33, 2013.

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

[7] D. Klein and N. Hirokawa. Confluence of non-left-linear TRSs via relative termination. In *Proc. 18th LPAR*, volume 7180 of *LNCS*, pages 258–273, 2012.

[8] K. Shintani and N. Hirokawa. CoLL: A confluence tool for left-linear term rewrite systems. In *Proc. 25th CADE*, LNAI, 2015. To appear.

[9] V. van Oostrom. Developing developments. *Theoretical Computer Science*, 175(1):159–181, 1997.

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

*This work is partly supported by the JSPS Core-to-Core Program (A. Advanced Research Networks).

[1] http://colo6-c703.uibk.ac.at/ttt2/ and http://zenon.dsic.upv.es/muterm/

[2] http://cl-informatik.uibk.ac.at/software/minismt/

CoCo 2016 Participant: ConCon*

Thomas Sternagel and Aart Middeldorp

Institute of Computer Science, University of Innsbruck, Austria
{thomas.sternagel, aart.middeldorp}@uibk.ac.at

ConCon is a fully automatic confluence checker for *oriented* first-order conditional term rewrite systems (CTRSs). The tool implements three known confluence criteria:

(A) A quasi-decreasing strongly irreducible deterministic 3-CTRS $\mathcal{R}$ is confluent if and only if all critical pairs are joinable [1].

(B) Almost orthogonal extended properly oriented right-stable 3-CTRSs are confluent [6].

(C) A weakly left-linear deterministic CTRS $\mathcal{R}$ is confluent if $\mathbb{U}(\mathcal{R})$ is confluent [2].

We refer to [4] for a more detailed description of the above results. ConCon is written in Scala 2.11 and available under the LGPL license. It can be downloaded from:

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

A web interface can also be found there. For some of the methods ConCon issues calls to the external unconditional confluence and termination checkers CSI and $\mathsf{T_TT_2}$ as well as the theorem prover Waldmeister.

To make criteria (A) and (B) more useful, we implemented a variety of methods to check for infeasibility of conditional critical pairs, ranging from a simple technique based on the tcap function, via tree automata completion, to equational reasoning. These are described in [5]. ConCon can generate certifiable output for method (C),[1] which is made possible due the formalization efforts described in [7] as well as certifiable output for method (B) along with most of the infeasibility methods due to the formalization described in [3]. We are currently working on certifiable output for method (A).

References

[1] J. Avenhaus and C. Loría-Sáenz. On Conditional Rewrite Systems with Extra Variables and Deterministic Logic Programs. In *Proc. 5th LPAR*, volume 822 of *LNAI*, pages 215–229, 1994.

[2] K. Gmeiner, N. Nishida, and B. Gramlich. Proving Confluence of Conditional Term Rewriting Systems via Unravelings. In *Proc. 2nd IWC*, pages 35–39, 2013.

[3] C. Sternagel and T. Sternagel. Certifying Confluence of Almost Orthogonal CTRSs via Exact Tree Automata Completion. In *Proc. 1st FSCD*, volume 52 of *LIPIcs*, pages 29:1–29:16, 2016.

[4] T. Sternagel and A. Middeldorp. Conditional Confluence (System Description). In *Proc. Joint 25th RTA and 12th TLCA*, volume 8560 of *LNCS*, pages 456–465, 2014.

[5] T. Sternagel and A. Middeldorp. Infeasible Conditional Critical Pairs. In *Proc. 4th IWC*, pages 13–17, 2015.

[6] T. Suzuki, A. Middeldorp, and T. Ida. Level-Confluence of Conditional Rewrite Systems with Extra Variables in Right-hand Sides. In *Proc. 6th RTA*, volume 914 of *LNCS*, pages 179–193, 1995.

[7] R. Thiemann and S. Winkler. Formalizing soundness and completeness of unravelings. In *Proc. 10th FroCoS*, LNAI, 2015. To appear.

*Supported by FWF (Austrian Science Fund) project P27502.

[1] We are grateful to Sarah Winkler for this extension.

CoScart: Confluence Prover in Scala

Karl Gmeiner[1]

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

1 Overview

CoScart is a tool to prove confluence of first-order term rewrite systems and deterministic conditional term rewrite systems automatically. It originates from the project *KaRT*, a collection of Java classes for term rewriting focussing on comparin transformations of conditional term rewrite systems and program transformations for functional programming languages. A first version of KaRT was used to conduct the experiments in [2]. To speed up and simplify development, in particular with focus on implementing CoScart, the whole project was ported to Scala, a functional, object-oriented programming language that compiles to Java Bytecode.

CoScart also comes with an automated termination prover and thus is a stand-alone-tool that does not rely on any other software.

2 Technical Details

The rewrite engine of Scart stores DAGs of terms that are collected in a linked list. This way rewriting is very efficient.

In order to use the Knuth-Bendix method, Scart contains an automatic termination prover (*TeScart*) for first-order TRSs that uses the dependency pairs method in combination with argument filterings with the *some more*-heuristics of [1].

A web interface is planned. New features compared to last year use the latest result of [4] that shows that confluence can be proved via transformations of CTRSs without considering soundness.

Since CoScart is currently a one-man project, there are no sophisticated user interfaces yet, but a web interface is planned.

CoScart proves confluence of (deterministic conditional) TRSs using the following methods: Transformation of [3] from DCTRSs into TRSs, modularity of confluence, Knuth-Bendix, and development-closed critical pairs of left-linear TRSs.

Scart is available at `https://github.com/searles/RewriteTool/`.

References

[1] N. Hirokawa and A. Middeldorp. Automating the Dependency Pair Method. In *Proc. CADE 2003*, LNAI vol. 2741, pp. 32–46, Springer-Verlag, 2003.

[2] K. Gmeiner and B. Gramlich. Transformations of Conditional Rewrite Systems Revisited. In *Proc. WADT 2008*, LNCS vol. 5486, pp. 166-186, Springer-Verlag, 2009.

[3] K. Gmeiner and N. Nishida. Notes on Structure-Preserving Transformations of Conditional Term Rewrite Systems. In *Proc. WPTE 2014*, OASIcs vol. 40, pp. 3–14, 2014.

[4] K. Gmeiner. Confluence of Conditional Term Rewrite Systems via Transformations. In *Proc. WPTE 2016*, 2016.

CRC: A Church-Rosser Checker Tool for Conditional Order-Sorted Equational Maude Specifications

Francisco Durán

Universidad de Málaga, Spain

The (ground) Church-Rosser and termination properties are essential for an equational specification to have good executability conditions, and also for having a complete agreement between the specification's initial algebra, mathematical semantics, and its operational semantics by rewriting. For order-sorted specifications, being Church-Rosser and terminating means not only confluence, but also a *descent* property ensuring that the normal form will have the least possible sort among those of all other equivalent terms.

The Maude Church-Rosser Checker tool (CRC) checks whether a (possibly conditional) order-sorted equational specification modulo equational axioms satisfies the Church-Rosser property. CRC is particularly well-suited for checking Maude specifications [1] with an initial algebra semantics to be ground-Church-Rosser, although it can be used to check the Church-Rosser property of conditional order-sorted specifications that do not have an initial algebra semantics. If the specification cannot be shown to be Church-Rosser by the tool, proof obligations are generated and are given back to the user, which can be used as a guide in the attempt to establish the ground-Church-Rosser property. Specifically, the tool gives as output a set of critical pairs and a set of membership assertions that must be shown, respectively, ground-joinable, and ground-rewritable to a term with the required sort.

The CRC tool and the Maude Termination Tool [3] are both integrated in the Maude Formal Environment [5], and can effectively deal with Maude equational specifications that are order-sorted, conditional, possibly with extra variables in their conditions, and whose equations can be applied modulo any combination of associativity, commutativity and identity axioms. Besides its generality, the main features of the tool are: (i) the capacity to discharge unjoinable critical pairs by proving them to be either unfeasible or context-joinable; and (ii) the capacity to deal with any combination of associativity and/or commutativity and/or identity axioms. CRC can be used on any Maude module, including structured modules, parameterized modules, etc.

CRC is available at `http://maude.lcc.uma.es/CRChC`. Its foundations, design and methodological guidelines can be found in [4]. The check of specifications with any combination of associativity/commutativity/identity axioms has not been available until the release of Maude 2.7.1, which includes built-in support for unification modulo these combinations of theories [2].

References

[1] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude - A High-Performance Logical Framework*, LNCS 4350. Springer, 2007.

[2] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. Maude 2.7.1 Manual. Available in `http://maude.cs.uiuc.edu`, July 2016.

[3] F. Durán, S. Lucas, C. Marché, J. Meseguer, and X. Urbain. Proving operational termination of membership equational programs. *Higher-Order and Symbolic Computation*, 21(1-2):59–88, 2008.

[4] F. Durán and J. Meseguer. On the church-rosser and coherence properties of conditional order-sorted rewrite theories. *J. Log. Algebr. Program.*, 81(7-8):816–850, 2012.

[5] F. Durán, C. Rocha, and J. M. Álvarez. Tool interoperability in the Maude Formal Environment. In A. Corradini, B. Klin, and C. Cîrstea, ed., *CALCO 2011.* LNCS 6859: 400–406. Springer, 2011.

CoCo 2016 Participant: CSI 0.6*

Bertram Felgenhauer, Aart Middeldorp, and Julian Nagele

Department of Computer Science, University of Innsbruck, Austria

CSI is an automatic tool for (dis)proving confluence of first-order term rewrite systems (TRSs). Its name is derived from the Confluence of the rivers Sill and Inn in Innsbruck. The tool is available from

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

under a LGPLv3 license, where a web interface is provided as well. CSI is based on the termination prover $\mathsf{T_TT_2}$. An overview of CSI's implementation and core features can be found in [10].

CSI is equipped with a strategy language for directing the proof search, allowing to configure it flexibly. It features a modular implementation of the decreasing diagrams technique, decomposing TRSs into smaller TRSs based on ordered sorts [4], a cubic time decision procedure for confluence of ground TRSs [1], and non-confluence checks based on tcap and tree automata [10]. Furthermore it adds and removes redundant rules [6]. For many techniques, CSI supports proof output in cpf format that can be verified independently by certifiers like CeTA [9].

The 2016 version of CSI additionally supports labeling of multisteps [2] as well as critical-pair-closing systems [8]. Furthermore, we added basic support for uniqueness of normal forms with respect to conversions and reductions, including decision procedures for ground TRSs [3] and the non-ω-overlapping criterion of [5]. We also provide cpf output for parallel closedness [7].

References

[1] B. Felgenhauer. Deciding confluence of ground term rewrite systems in cubic time. In *Proc. 23rd RTA*, volume 15 of *LIPIcs*, pages 165–175, 2012.

[2] B. Felgenhauer. Labeling multi-steps for confluence of left-linear term rewrite systems. In *Proc. 4th IWC*, pages 33–37, 2015.

[3] B. Felgenhauer. Efficiently deciding uniqueness of normal forms and unique normalization for ground TRSs. In *Proc. 5th IWC*, 2016. This volume.

[4] B. Felgenhauer, A. Middeldorp, H. Zankl, and V. van Oostrom. Layer systems for proving confluence. *ACM TOCL*, 16(2:14):1–32, 2015.

[5] S. Kahrs and C. Smith. Non-ω-overlapping TRSs are UN. In *Proc. 1st FSCD*, volume 52 of *LIPIcs*, pages 22:1–22:17, 2016.

[6] J. Nagele, B. Felgenhauer, and A. Middeldorp. Improving automatic confluence analysis of rewrite systems by redundant rules. In *Proc. 26th RTA*, volume 36 of *LIPIcs*, pages 257–268, 2015.

[7] J. Nagele and A. Middeldorp. Certification of classical confluence results for left-linear term rewrite systems. In *Proc. 7th ITP*, volume 9807 of *LNCS*, pages 290–306, 2016.

[8] M. Oyamaguchi and N. Hirokawa. Confluence and critical-pair-closing systems. In *Proc. 3rd IWC*, pages 29–33, 2014.

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

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

*Supported by the Austrian Science Fund (FWF) P27528.

CoCo 2016 Participant: CSI^ho 0.2*

Julian Nagele

Department of Computer Science, University of Innsbruck, Austria
julian.nagele@uibk.ac.at

Higher-order rewriting combines standard, first-order rewriting with notions and concepts from the λ-calculus, resulting in rewriting systems with higher-order functions and bound variables. CSI^ho is a tool for automatically proving confluence of such higher-order systems, specifically pattern rewrite systems (PRSs) as introduced by Nipkow [3, 5]. The restriction to pattern left-hand sides is essential for obtaining decidability of unification and thus makes it possible to compute critical pairs. To this end CSI^ho implements a version of Nipkow's algorithm for higher-order pattern unification [6].

CSI^ho is built on top of CSI [9], a powerful confluence prover for first-order term rewrite systems. It is available from http://cl-informatik.uibk.ac.at/software/csi/ho/. Using CSI as foundation, CSI^ho inherits many of its attractions, in particular a strategy language, which allows for flexible configuration of the proof search. CSI^ho supports the following techniques:

2015 Knuth and Bendix' criterion, that is, for terminating PRSs we decide confluence by checking joinability of critical pairs [5]. For showing termination CSI^ho uses a basic higher-order recursive path ordering and static dependency pairs with dependency graph decomposition and the subterm criterion. For potentially non-terminating PRSs it supports weak orthogonality [8] and van Oostrom's result on development closed critical pairs [7].

2016 As a first divide-and-conquer criterion CSI^ho includes modularity of confluence for left-linear PRSs—note that confluence of PRSs is not modular in general [1]. To improve CSI^ho on terminating systems, external termination tools like WANDA [2] can now be used as a termination back-end. The final novelty this year is the simple technique of adding and removing redundant rules [4], adapted for PRSs.

References

[1] C. Appel, V. van Oostrom, and J. G. Simonsen. Higher-order (non-)modularity. In *Proc. 21st RTA*, volume 6 of *LIPIcs*, pages 17–32, 2010.

[2] Cynthia Kop. *Higher Order Termination.* PhD thesis, Vrije Universiteit, Amsterdam, 2012.

[3] R. Mayr and T. Nipkow. Higher-order rewrite systems and their confluence. *TCS*, 192(1):3–29, 1998.

[4] J. Nagele, B. Felgenhauer, and A. Middeldorp. Improving automatic confluence analysis of rewrite systems by redundant rules. In *Proc. 26th RTA*, volume 36 of *LIPIcs*, pages 257–268, 2015.

[5] T. Nipkow. Higher-order critical pairs. In *Proc. 6th LICS*, pages 342–349, 1991.

[6] Tobias Nipkow. Functional unification of higher-order patterns. In *Proc. 8th LICS*, pages 64–74, 1993.

[7] V. van Oostrom. Developing developments. *TCS*, 175(1):159–181, 1997.

[8] V. van Oostrom and F. van Raamsdonk. Weak orthogonality implies confluence: The higher order case. In *Proc. 3rd LFCS*, volume 813 of *LNCS*, pages 379–392, 1994.

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

*Supported by Austrian Science Fund (FWF), project P27528.

CoCo 2016 Participant: FORT 1.0*

Franziska Rapp and Aart Middeldorp

Department of Computer Science, University of Innsbruck, Austria
{franziska.rapp|aart.middeldorp}@uibk.ac.at

FORT is a decision and synthesis tool for the first-order theory of rewriting for finite left-linear right-ground rewrite systems. It implements the decision procedure for this theory, which uses tree automata techniques and goes back to Dauchet and Tison [1]. In this theory confluence-related properties on ground terms are easily expressible. The basic functionality of FORT is described in [2] and in [3] we report on an extension to deal with non-ground terms.

FORT 1.0 is a completely new implementation in Java, for which the JAR file can be downloaded from

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

The tool participates in the demo categories GCR and UN at CoCo 2016. The former is about ground-confluence of *many-sorted* rewrite systems. Since the set of well-typed terms according to a many-sorted type discipline is accepted by a tree automaton, the modifications required in FORT were straightforward.

The most significant change in FORT 1.0 is the support for parallelism, using the multi-threading capabilities of Java. This greatly speeds up the synthesis of rewrite systems satisfying certain properties expressible in the first-order theory of rewriting. Furthermore, we exploit this functionality for the UN demo category. In this category tools report the strongest property among CR, NFP, UNC and UN that can be established, or the answer NO if UN can be disproved. For the given rewrite system FORT checks the four properties in parallel, reusing basic automata constructions that can be shared among the properties. As soon as it has the required information, it reports the optimal result. In case this information is not present shortly before the time limit, it kills all remaining threads and reports the strongest result that was established. This strategy can be illustrated quite well on COPS #215.[1] Within 260 milliseconds FORT has established NFP, while the thread checking for confluence is still running. Hence, we do not yet know the exact answer. Shortly before the 60 seconds time limit FORT reports NFP. However, this is not the optimal answer, since this system is actually confluent. As can be seen on this and many other examples, confluence is often harder to verify than the other three properties.

References

[1] M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *Proc. 5th IEEE Symposium on Logic in Computer Science*, pages 242–248, 1990. doi: 10.1109/LICS.1990.113750.

[2] F. Rapp and A. Middeldorp. Automating the first-order theory of left-linear right-ground term rewrite systems. In *Proc. 1st International Conference on Formal Structures for Computation and Deduction*, volume 52 of *Leibniz International Proceedings in Informatics*, pages 36:1–36:12, 2016. doi: 10.4230/LIPIcs.FSCD.2016.36.

[3] F. Rapp and A. Middeldorp. Confluence properties on open terms in the first-order theory of rewriting. In *Proc. 5th International Workshop on Confluence*, 2016. This volume.

*Supported by FWF (Austrian Science Fund) project P27528.

[1] http://cops.uibk.ac.at/?q=215

Nrbox: System Description for CoCo 2016

Takahito Aoto[1] and Kentaro Kikuchi[2]

[1] Faculty of Engineering, Niigata University
aoto@ie.niigata-u.ac.jp
[2] RIEC, Tohoku University
kentaro@nue.riec.tohoku.ac.jp

Nominal rewriting [4, 5] is a framework that extends first-order term rewriting by a binding mechanism. A distinctive feature of the nominal approach is that α-conversion and capture-avoiding substitution are not relegated to meta-level—they are explicitly dealt with at object-level. This makes nominal rewriting significantly different from classical frameworks of higher-order rewriting systems based on 'higher-order syntax'.

Nrbox (Nominal rewriting toolbox) is an automated confluence prover for *nominal rewrite systems (NRSs)*. Nrbox is written in Standard ML of New Jersey (SML/NJ). The tool registered to the category of confluence of nominal rewrite systems that has been adopted as one of the demonstration categories in CoCo 2016. Nrbox proves whether input NRSs are Church-Rosser modulo the α-equivalence (CR$\approx_\alpha$) based on the following results (we refer to [1] for the notions and notations):

Proposition 1 ([7])**.** *Orthogonal and abstract skeleton preserving NRSs are CR$\approx_\alpha$.*

Proposition 2 ([8])**.** *Linear uniform NRSs are CR$\approx_\alpha$ if $\Gamma \vdash u \to^{=} \circ \approx_\alpha \circ \leftarrow^{*} v$ and $\Gamma \vdash u \to^{*} \circ \approx_\alpha \circ \leftarrow^{=} v$ for any basic critical pair $\Gamma \vdash \langle u, v\rangle$.*

Proposition 3 ([8])**.** *Terminating uniform NRS are CR$\approx_\alpha$ iff all basic critial pairs are joinable.*

Proposition 4 ([6])**.** *Left-linear uniform NRSs are CR$\approx_\alpha$ if $\Gamma \vdash u \mathrel{-\!\!\!\Vdash\!\!\!\!\to} \circ \approx_\alpha v$ ($u \mathrel{-\!\!\!\Vdash\!\!\!\!\to} \circ \approx_\alpha \circ \leftarrow^{*} v$) for any inner (resp. outer) basic critical pair $\Gamma \vdash \langle u, v\rangle$.*

Termination of NRSs is proved by encoding the problem into the termination problem of first-order term rewriting, which is explained in [1]. For the comptation of BCPs (basic critical pairs), the equivariant unification algorithm [3] is required; our equivariant unification procedure is based on the algorithm explained in [2].

References

[1] T. Aoto and K. Kikuchi. Nominal confluence tool. In *Proc. of 8th IJCAR*, volume 9706 of *LNCS*, pages 173–182. Springer-Verlag, 2016.

[2] T. Aoto and K. Kikuchi. A rule-based equivariant unification procedure. In *Proc. of 8th HOR*, 2016.

[3] J. Cheney. Equivariant unification. *J. of Automated Reasoning*, 45:267–300, 2010.

[4] M. Fernández and M. J. Gabbay. Nominal rewriting. *Inform. and Comput.*, 205:917–965, 2007.

[5] M. Fernández, M. J. Gabbay, and I. Mackie. Nominal rewriting systems. In *Proc. 6th PPDP*, pages 108–119. ACM Press, 2004.

[6] K. Kikuchi, T. Aoto, and Y. Toyama. Parallel closure theorem for left-linear nominal rewriting systems. http://www.nue.riec.tohoku.ac.jp/user/kentaro/cr-nominal/pct.pdf.

[7] T. Suzuki, K. Kikuchi, T. Aoto, and Y. Toyama. Confluence of orthogonal nominal rewriting systems revisited. In *Proc. 26th RTA*, volume 36 of *LIPIcs*, pages 301–317, 2015.

[8] T. Suzuki, K. Kikuchi, T. Aoto, and Y. Toyama. Critical pair analysis in nominal rewriting. In *Proc. 7th SCSS*, volume 39 of *EPiC*, pages 156–168. EasyChair, 2016.