IWC 2015

4th International Workshop on Confluence

Proceedings

Editors: Ashish Tiwari & Takahito Aoto

August 2, 2015, Berlin, Germany

Preface

This report contains the proceedings of the *4th International Workshop on Confluence (IWC 2015)*, which was held in Berlin on August 2, 2015. The workshop was affiliated with the 25th Jubilee Edition of the International Conference on Automated Deduction (CADE-25). The first edition of IWC took place in Nagoya (2012), the second in Eindhoven (2013) and the third in Vienna, Austria (2014).

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 IWC workshop series 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 2015 received 7 submissions. Each submission was reviewed by 3 program committee members. After deliberations the program committee decided to accept all submissions, which are contained in this report. Apart from these contributed talks, the workshop had two invited talks. The first invited speaker was *Koji Nakazawa* and the talk was titled *Lambda Calculi and Confluence from A to Z*. The second invited speaker was *Stefan Hetzl* and the talk was titled *Herbrand-Confluence in the Classical Sequent Calculus*. The abstracts of the invited talks are also included in the report. Furthermore, this report contains short descriptions of the 11 tools that participated in the 4th Confluence Competition (CoCo 2015). This competition ran live during the workshop and the results are available at `http://coco.nue.riec.tohoku.ac.jp/2015/`.

Several persons helped to make IWC 2015 a success. We would like to thank all authors for their contributions, the members of the Program Committee for their excellent work during the reviewing process and the invited speakers for kindly accepting the invitation. We also thank the members of the CADE-25 organizing committee for hosting IWC 2015 in Berlin. Finally, we gratefully acknowledge the support of EasyChair.

Sendai & Menlo Park, July 2015 *Takahito Aoto & Ashish Tiwari*

Program Committee

Takahito Aoto	RIEC, Tohoku University	(co-chair)
Mauricio Ayala Rincon	Universidade de Brasilia	
Karl Gmeiner	UAS Technikum Wien	
Samuel Mimram	École Polytechnique	
Haruhiko Sato	Hokkaido University	
Christian Sternagel	Universtity of Innsbruck	
Ashish Tiwari	SRI International - Menlo Park, CA	(co-chair)

Table of Contents

Author Index

Lambda Calculi and Confluence from A to Z

Koji Nakazawa

Kyoto University, Japan
knak@kuis.kyoto-u.ac.jp

Abstract

The proofs of the confluence of the λ-calculus have been improved and simplified in the long history. The first proof was given by Churh and Rosser [2] with the notion of residuals, and then refined by Tait and Martin-Löf with the parallel reduction, and further refined by Takahashi [4] with the complete development. More recently, Dehornoy and van Oostrom [3] introduced another elegant idea for confluence of abstract rewriting systems, called the *Z theorem*: if we find a mapping on terms with some property, called the *Z property*, then the rewriting system is confluent. It gives a simple proof of confluence of the λ-calculus since the ordinary complete development for the β-reduction has the Z property. In this talk, first we discuss on the Z theorem, and then show some applications of the Z theorem to some variants of λ-calculi, including the λ- and the $\lambda\mu$-calculus with disjunction and permutative conversions. For confluence of such calculi independent of strong normalization, Ando [1] shows that naïve application of the existing technique with the parallel reduction and the complete development does not work. We propose an extension of the Z theorem, called the *compositional Z*, and show that it can be used to avoid the difficulties.

References

[1] Ando, Y. Church-rosser property of a simple reduction for full first-order classical natural deduction. *Annals of Pure and Applied Logic*, 119:225–237, 2003.

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

[3] Dehornoy, P. and van Oostrom, V. Z, proving confluence by monotonic single-step upperbound functions. In *Logical Models of Reasoning and Computation (LMRC-08)*, 2008. Slides available at http://www.phil.uu.nl/~oostrom/publication/talk/lmrc060508.pdf.

[4] Takahashi, M. Parallel reductions in λ-calculus. *Information and Computation*, 118:120–127, 1995.

Herbrand-confluence in the classical sequent calculus

Stefan Hetzl

Institute of Discrete Mathematics and Geometry
Vienna University of Technology
Vienna, Austria
stefan.hetzl@tuwien.ac.at

This talk will be about cut-elimination in the sequent calculus for classical first-order logic. The cut-elimination theorem is a cornerstone of proof theory. It was shown in [4] by G. Gentzen by a stepwise transformation of proofs, in contemporary terminology: a proof rewriting system, and by specifying a terminating strategy for it.

This rewriting system is well-known to be non-confluent, see e.g. [5]. In fact, the number of different normal forms is non-elementary [2], i.e., it can not be bounded by a fixed number of iterations of the exponential function.

On the other hand, the interest in computational interpretations of (cut-elimination in) classical logic has led to a number of restrictions of this proof rewriting system which are confluent, for example the quite general system $\mathbf{LK}^{\mathrm{tq}}$ [3].

In this talk I will speak about another approach to obtaining confluence results for classical logic. Instead of looking for a restriction that satisfies syntactic confluence we prove confluence up to an equivalence relation on normal forms. The equivalence relation we are considering is for two cut-free proofs to have the same Herbrand-disjunction. This is an important relation since the mathematically relevant parts of a cut-free proof are already contained in its Herbrand-disjunction. A reduction is then called Herbrand-confluent if all its cut-free proofs have the same Herbrand-disjunction.

For proofs with up to Π_2-cuts the general reduction system has been shown to be Herbrand-confluent [6, 1]. For more complex cut-formulas this question is still open. The central technical tool for obtaining this result is to associate a tree grammar to a proof with cuts in such a way that cut-elimination corresponds to the computation of the (finite) tree language of that grammar and this language corresponds to the Herbrand-disjunction of the cut-free proof.

References

[1] Bahareh Afshari, Stefan Hetzl, and Graham E. Leigh. On Herbrand-confluence for first-order logic. submitted.

[2] Matthias Baaz and Stefan Hetzl. On the non-confluence of cut-elimination. *Journal of Symbolic Logic*, 76(1):313–340, 2011.

[3] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. A New Deconstructive Logic: Linear Logic. *Journal of Symbolic Logic*, 62(3):755–807, 1997.

[4] Gerhard Gentzen. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift*, 39(2):176–210, 1934.

[5] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. Cambridge University Press, 1989.

[6] Stefan Hetzl and Lutz Straßburger. Herbrand-Confluence for Cut-Elimination in Classical First-Order Logic. In Patrick Cégielski and Arnaud Durand, editors, *Computer Science Logic (CSL) 2012*, volume 16 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 320–334. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012.

Point-Decreasing Diagrams Revisited*

Bertram Felgenhauer[1]

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

Abstract

In this note we revisit Bognar's point version of decreasing diagrams. We show that it is an instance of van Oostrom's decreasing diagrams. Furthermore we demonstrate that the point version of decreasing diagrams is complete for confluence of *finite* abstract rewrite systems, contradicting a counterexample by Bognar.

1 Introduction

The decreasing diagrams technique [4] is a powerful confluence criterion for abstract rewrite systems (ARSs), based on labeling the rewrite steps of the system. The criterion is complete for confluence of countable ARSs. In [1], Bognar introduced a point version of decreasing diagrams, where labels are assigned to the objects instead of the rewrite steps. In order to prove this result, Bognar modified van Oostrom's proof [4], which is based on lexicographic path measures.

In this note, we revisit Bognar's point-decreasing diagrams. We give a new proof of the result based on van Oostrom's decreasing diagrams. We also show that point-decreasing diagrams are complete for confluence of *finite* ARSs.

This note is based on [3, Section 3.5].

2 Preliminaries

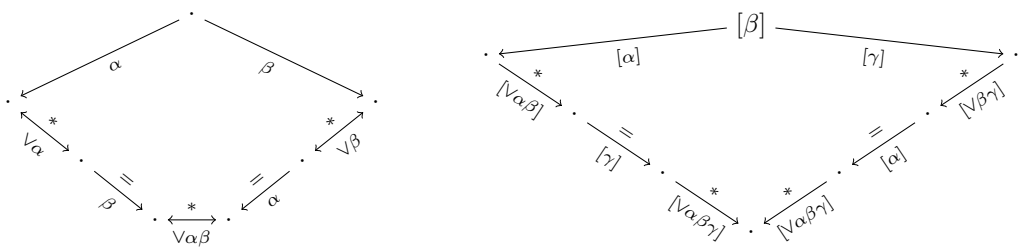
We use standard notation for abstract rewriting. An ARS $\langle \mathcal{A}, \to \rangle$ consists of a set of objects $\mathcal{A}$ and a (rewrite) relation $\to$ on $\mathcal{A}$. Let $\to^0 = \equiv$ and $\to^{n+1} = \to^n \cdot \to$. We denote the inverse, reflexive closure, symmetric closure and reflexive transitive closure of $\to$ by $\leftarrow$, $\to^=$, $\leftrightarrow$ and $\to^*$, respectively. We also consider labeled ARSs $\langle \mathcal{A}, (\to_\alpha)_{\alpha \in L} \rangle$, where L is a set of labels equipped with a well-founded order $>$ and $(\to_\alpha)_{\alpha \in L}$ is a family of relations on $\mathcal{A}$. For $M \subseteq L$ we let $\to_M = \bigcup_{\alpha \in M} \to_\alpha$. We define $\vee\alpha = \{\beta \mid \alpha > \beta\}$ and $\vee\alpha\beta = \vee\alpha \cup \vee\beta$. Recall van Oostrom's decreasing diagrams result:

Theorem 1. *Let $\langle \mathcal{A}, \to_\alpha \rangle_{\alpha \in L}$ be a labeled ARS. If for all $\alpha, \beta \in L$*

$$\underset{\alpha}{\leftarrow} \cdot \underset{\beta}{\rightarrow} \subseteq \overset{*}{\underset{\vee\alpha}{\leftrightarrow}} \cdot \overset{=}{\underset{\beta}{\rightarrow}} \cdot \overset{*}{\underset{\vee\alpha\beta}{\leftrightarrow}} \cdot \overset{=}{\underset{\alpha}{\leftarrow}} \cdot \overset{*}{\underset{\vee\beta}{\leftrightarrow}}$$

then $\to_L$ is confluent. (See also Figure 1(a).)

We will refer to van Oostrom's decreasing diagrams as the step version of decreasing diagrams, to distinguish it from Bognar's point version.



(a) Locally step-decreasing diagram. (b) Locally point-decreasing diagram.

Figure 1: Decreasing diagrams.

3 Point-Decreasing Diagrams

In this section we consider the point version of decreasing diagrams proposed by Bognar in [1], and show how it follows from step-decreasing diagrams. For point-decreasing diagrams, the objects (i.e., the points) rather than the steps of an abstract rewrite system are labeled.

Formally, we consider *point-labeled ARSs* $\langle \mathcal{A}, \to, \ell \rangle$ which consist of an ARSs $\langle \mathcal{A}, \to \rangle$ together with a function labeling the objects $\ell : \mathcal{A} \to W$, where W is a set of labels equipped with a well-founded order $>$. We annotate steps by the labels of their targets in square brackets, that is, we write $s \to_{[\ell(t)]} t$. We also allow sets inside the square brackets, in which case the target of the step may have any label from this set: $\to_{[M]} = \bigcup_{\alpha \in M} \to_{[\alpha]}$.

Bognar's version of local decreasingness [1, Corollary 8] can be stated as follows.

Theorem 2. *Let $\langle \mathcal{A}, \to, \ell \rangle$ be a point-labeled abstract rewrite system. Then $\to$ is confluent if every local peak $t \ {}_{[\alpha]}\!\leftarrow s \to_{[\gamma]} u$ with $\beta = \ell(s)$ has a joining valley*

$$t \xrightarrow[[\vee\alpha\beta]]{*} \cdot \xrightarrow[[\gamma]]{=} \cdot \xrightarrow[[\vee\alpha\beta\gamma]]{*} \cdot \xleftarrow[[\vee\alpha\beta\gamma]]{*} \cdot \xleftarrow[[\alpha]]{=} \cdot \xleftarrow[[\vee\beta\gamma]]{*} u \tag{PD}$$

(resulting in a locally point-decreasing diagram*, Figure 1(b)).*

Proof. Because any well-founded order can be extended to a well-order, and because locally point-decreasing diagrams are preserved when the order $>$ on W is extended, we may assume w.l.o.g. that $>$ is a well-order. We label steps by pairs from $W \times \{\bot, \top\}$, ordered lexicographically, using the order $\top > \bot$ on the second component. Note that this gives a well-order on $W \times \{\bot, \top\}$. Each step $s \to t$ is labeled by $\max(\langle \ell(s), \bot \rangle, \langle \ell(t), \top \rangle)$. In particular, the peak $t \ {}_{[\alpha]}\!\leftarrow s \to_{[\gamma]} u$ with $\beta = \ell(s)$ is labeled by $A = \max(\langle \beta, \bot \rangle, \langle \alpha, \top \rangle)$ to the left and $B = \max(\langle \beta, \bot \rangle, \langle \gamma, \top \rangle)$ to the right. We claim that using this labeling, the point-decreasing diagrams (PD) become decreasing diagrams. Consider a step $v \to w$ of the valley, with label $C = \max(\langle \ell(v), \bot \rangle, \langle \ell(w), \top \rangle)$. There are three cases.

1. Let $v \to w$ be from the $t \to^*_{[\vee\alpha\beta]} \cdot$ subderivation of the valley. The source v of such a step satisfies $\alpha \geqslant \ell(v)$ (hence $\langle \alpha, \top \rangle > \langle \ell(v), \bot \rangle$) or $\beta > \ell(v)$ (hence $\langle \beta, \bot \rangle > \langle \ell(v), \bot \rangle$), while the target w satisfies $\alpha > \ell(w)$ (hence $\langle \alpha, \top \rangle > \langle \ell(w), \top \rangle$)) or $\beta > \ell(w)$ (hence $\langle \beta, \bot \rangle > \langle \ell(w), \top \rangle$). Therefore, $A > C$.

*This research was supported by the Austrian Science Fund (FWF) projects P22467 and P27528.

2. Assume that $v \to w$ corresponds to the optional step $\cdot \to^{=}_{[\gamma]} \cdot$ of the valley. Then $\alpha \geqslant \ell(v)$ or $\beta > \ell(v)$, and $\ell(w) = \gamma$. We have $\langle \gamma, \top\rangle = \langle \ell(w), \top\rangle$, $\langle \alpha, \top\rangle > \langle \ell(v), \bot\rangle$ and $\langle \beta, \bot\rangle > \langle \ell(v), \bot\rangle$. Consequently, $B = C$ or $A > C$.

3. Let $v \to w$ be from the $\cdot \to^{*}_{[\vee\alpha\beta\gamma]} \cdot$ part of the valley. Then $\alpha \geqslant \ell(v)$, $\beta \geqslant \ell(v)$ or $\gamma > \ell(v)$, and $\alpha > \ell(w)$, $\beta > \ell(w)$ or $\gamma > \ell(w)$. Consequently, $\langle \alpha, \top\rangle > \langle \ell(v), \bot\rangle$, $\langle \beta, \top\rangle > \langle \ell(v), \bot\rangle$ or $\langle \gamma, \bot\rangle > \langle \ell(v), \bot\rangle$, and $\langle \alpha, \top\rangle > \langle \ell(w), \top\rangle$, $\langle \beta, \top\rangle > \langle \ell(w), \top\rangle$ or $\langle \gamma, \bot\rangle > \langle \ell(w), \top\rangle$. Consequently, $A > C$ or $B > C$ follows.

A symmetric argument applies to steps $w \leftarrow v$ on the left side of the valley. Therefore,

$$t \xrightarrow[\vee A]{*} \cdot \xrightarrow[B]{=} \cdot \xrightarrow[\vee AB]{*} \cdot \xleftarrow[\vee AB]{*} \cdot \xleftarrow[A]{=} \cdot \xleftarrow[\vee B]{*} u$$

This is a step-decreasing diagram. Since we started from an arbitrary peak, and because the order on the set of labels $L \times \{\bot, \top\}$ is well-founded, we conclude that the ARS $\to$ is decreasing by Theorem 1. □

Remark 3. In fact the proof of Theorem 2 shows that any point-decreasingly confluent system is also step-decreasing using the same joining sequences for local peaks. This detail is important for applications of decreasing diagrams, where one usually attempts to find suitable labelings for a given set of joining sequences. Our proof also provides some insight into how van Oostrom's original proof for step-decreasing diagrams relates to Bognar's adaptation of that proof for point-decreasing diagrams.

Remark 4. Condition (PD) only considers local peaks, which differs from Bognar's definition [1], which defines decreasing diagrams for peaks and valleys of arbitrary size, based on van Oostrom's *lexicographic path measure* [4]. Furthermore [1] assumes that $>$ is a well-order, whereas we allow an arbitrary well-founded order; note however, that this extra degree of freedom does not any power because any well-founded order can be extended to a well-order. If one assumes $>$ to be a well-order, then condition (PD) is equivalent to Bognar's decreasing diagrams for *local* peaks. This can be seen as follows.

We use notation from [1]. Any locally decreasing diagram can be written as

$$j \xleftarrow[[j]]{} i \xrightarrow[[k]]{} k \qquad\qquad j \xrightarrow[[\tau']]{*} l \xleftarrow[[\sigma']]{*} k$$

where σ' and τ' are strings of labels satisfying

$$|i; j; \tau'| \preceq_{\#} [i] \cup_{\#\ i\leqslant} |j| \cup_{\#\ i\leqslant} |k| \tag{DCR1}$$

and the symmetric property (DCR2) which is obtained by swapping the roles of j, τ' and k, σ'. Condition (DCR1) can be simplified as follows.

$$[i] \cup_{\#\ i\leqslant} |j| \cup_{\#\ \max(i,j)\leqslant} |\tau'| \preceq_{\#} [i] \cup_{\#\ i\leqslant} |j| \cup_{\#\ i\leqslant} |k|$$
$$\max(i,j)\leqslant |\tau'| \preceq_{\#}\ _{i\leqslant} |k|$$

The right-hand side is an empty multiset if $i > k$, in which case τ' must consist of labels all smaller than $\max(i, j)$ (including labels smaller than or equal to k). If $k \geqslant i$, then the right-hand side is the singleton multiset $[k]$, and τ' must consist of some labels smaller than $\max(i, j)$, optionally followed by k, followed by further labels smaller than $\max(i, j, k)$. Condition (PD) arises from these observations and the fact that a comparison by $\max(i, j)$ (or $\max(i, j, k)$) can be performed by comparing to each of i, j (or i, j, k) and taking the disjunction of the comparison results.

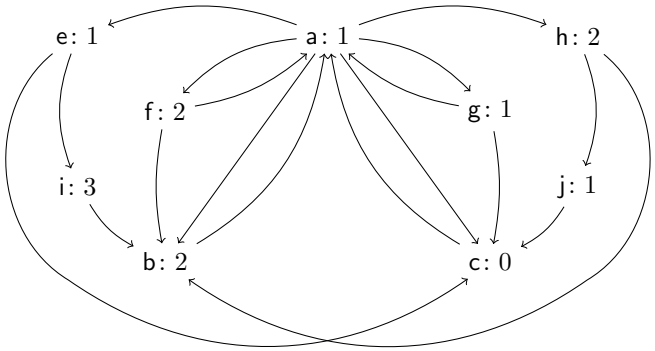


Figure 2: Labeling the "Maja the Bee"-example from [2].

Finally, we consider the question of completeness of the point version of decreasing diagrams.

Theorem 5. *Point-decreasing diagrams are complete for confluence of finite ARSs.*

Proof. Let $\langle \mathcal{A}, \to \rangle$ be a confluent, finite ARS. The relation $\leftrightarrow^*$ is an equivalence relation that partitions $\mathcal{A}$ into equivalence classes, the *components* of $\mathcal{A}$. Because $\mathcal{A}$ is finite, there are only finitely many components of $\mathcal{A}$ and each component contains finitely many objects from $\mathcal{A}$. Let $C \subseteq A$ be a component of $\mathcal{A}$. For all $s, t \in C$ we have $s \leftrightarrow^* t$. Therefore, by finiteness of C and confluence, we can choose an object $f_C \in A$ that is reachable from all elements of C. Furthermore, because $\to^* \subseteq \leftrightarrow^*$, $f_C \in C$. Let

$$F = \{f_C \mid C \text{ is a component of } \mathcal{A}\}$$

By this construction, every object $s \in \mathcal{A}$ reaches exactly one element of F, namely f_{C_s}, where C_s denotes the component which contains s. Let

$$\ell(a) = \min\{n \in \mathbb{N} \mid a \xrightarrow{n} f_{C_a}\}$$

Note that for any $s \in \mathcal{A}$ with $n = \ell(s)$, we have

$$s \xrightarrow[[\vee n]]{n} f_{C_s}$$

We claim that the labeling function ℓ makes $\langle \mathcal{A}, \to \rangle$ point-decreasing. To see why, it suffices to consider a local peak $t \,{}_{[n]}\!\leftarrow s \to_{[m]} u$, and note that

$$t \xrightarrow[[\vee n]]{*} f_{C_t} = f_{C_u} \xleftarrow[[\vee m]]{*} u \qquad \square$$

Remark 6. In the proof of Theorem 5, we use natural numbers as labels, ordered by the usual order, which is a well-order. Therefore, it applies to Bognar's original definition of point-decreasing diagrams as well.

Example 7. We consider the "Maja the Bee" example by Bognar and Klop [2], which has been presented as a counterexample to the completeness of the point-version of decreasing diagrams. The example is reproduced in Figure 2. There is only a single component $C = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{e}, \mathsf{f}, \mathsf{g}, \mathsf{h}\}$, and we pick $f_C = \mathsf{c}$, which is reachable from all objects in C. (In fact, C is strongly connected,

and we could pick any element of C.) The resulting labels are displayed in Figure 2. Consider the local peak $\mathsf{e} \,{}_{[1]}\!\leftarrow \mathsf{a} \rightarrow_{[2]} \mathsf{f}$. We obtain the joining valley $\mathsf{e} \rightarrow_{[0]} \mathsf{c} \,{}_{[0]}\!\leftarrow \mathsf{a} \,{}_{[1]}\!\leftarrow \mathsf{f}$, which passes through a.

This particular peak is of interest because in [2], it is argued that any conversion between e and f that passes through a cannot result in a point-decreasing diagram. Evidently, that is not the case with our labeling, due to the fact that $\ell(\mathsf{f}) > \ell(\mathsf{a})$.

4 Conclusion

We have presented a new proof of Bognar's point version of decreasing diagrams based on van Oostrom's decreasing diagrams. Furthermore, we showed that point-decreasing diagrams are complete for confluence of finite ARSs.

The question whether the point version of decreasing diagrams is complete for countable ARSs remains open.

References

[1] M. Bognar. A point version of decreasing diagrams. In *Proc. Accolade 1996*, Dutch Graduate School in Logic, pages 1–14, 1997.

[2] M. Bognar and J.W. Klop. A note on some abstract confluence criteria. Technical Report IR-411, Vrije Universiteit Amsterdam, 1996.

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

[4] V. van Oostrom. Confluence by decreasing diagrams. *Theoretical Computer Science*, 126(2):259–280, 1994.

Point-Step-Decreasing Diagrams*

Bertram Felgenhauer[1]

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

Abstract

Inspired by Bognar's point-decreasing diagrams, we present a generalisation of decreasing diagrams in which both the objects and the steps of conversions are labeled. We argue that this extension is more powerful than decreasing diagrams. However, it remains to be seen whether this power can be exploited in practice.

1 Introduction

There are two different approaches to proving confluence of abstract rewrite systems (ARSs) by decreasing diagrams. The first approach, by van Oostrom [4], labels the rewrite steps of an abstract rewrite system (ARS). In contrast, Bognar [1] proposed a variant that labels the points (i.e., objects) of an ARS instead. In this note we use the monotonic proof order from [2] to derive a point-step version of decreasing diagrams, which combines these two ideas.

This note is based on [3, Section 3.6].

2 Preliminaries

We use standard notation for abstract rewriting. An ARS $\langle \mathcal{A}, \to \rangle$ consists of a set of objects $\mathcal{A}$ and a (rewrite) relation $\to$ on $\mathcal{A}$. We denote the inverse, reflexive closure, symmetric closure and reflexive transitive closure of $\to$ by $\leftarrow$, $\to^=$, $\leftrightarrow$ and $\to^*$, respectively. An ARS $\to$ is Church-Rosser (equivalently, confluent) if $\leftrightarrow^* \subseteq \to^* \cdot {}^*\!\leftarrow$. Let L be an alphabet equipped with a well-founded order $\succ$. If $(\to_\kappa)_{\kappa \in L}$ is a family of relations on $\mathcal{A}$ then $\langle \mathcal{A}, (\to_\kappa)_{\kappa \in L} \rangle$ is a labeled ARS. For $M \subseteq L$ we let $\to_M = \bigcup_{\kappa \in M} \to_\kappa$. We define $\vee\kappa = \{\mu \mid \kappa \succ \mu\}$ and $\vee\kappa\mu = \vee\kappa \cup \vee\mu$.

We recall Greek strings [2], which we will use to represent conversions. For each $\kappa \in L$, there are three Greek letters, accented by acute ($\acute{\kappa}$), grave ($\grave{\kappa}$) or macron ($\bar{\kappa}$) accents. The notation $\hat{\kappa}$ represents a Greek letter with arbitrary accent. Greek strings are strings over Greek letters. We use the following notation for certain regular languages on Greek letters: $\kappa\!\succ$ represents any Greek letter whose label is less than κ, $[s]$ denotes an optional string, and $\{s\}$ denotes the Kleene star of s.

Definition 1. The order $\overline{\gg}_\bullet$ on Greek strings over L is defined inductively as follows:

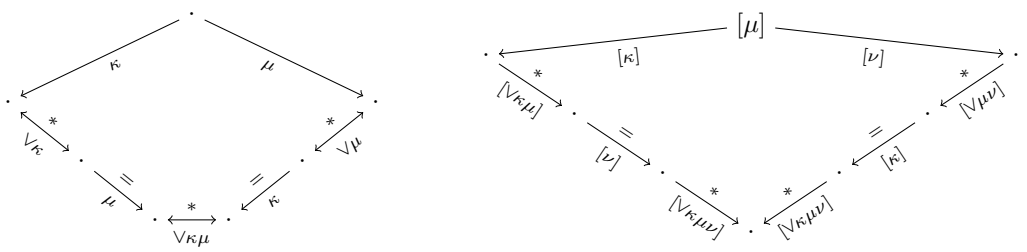
$$s \mathrel{\overline{\gg}_\bullet} t \qquad \text{iff} \qquad \langle s \rangle^g \; ((\succ, \overline{\gg}_\bullet)_{lex})_{mul} \; \langle t \rangle^g$$

where $>$ compares Greek letters by forgetting the accents, $(\cdot,\cdot)_{lex}$ and $(\cdot)_{mul}$ denote the lexicographic product and the multiset extension of relations, and the map

$$\langle s \rangle^g = [(\acute{\kappa}, q) \mid s = p\acute{\kappa}q] \cup [(\grave{\kappa}, p) \mid s = p\grave{\kappa}q] \cup [(\bar{\kappa}, \epsilon) \mid s = p\bar{\kappa}q]$$

collects acute letters together with their suffixes, grave letters together with their prefixes, and macron letters together with empty strings into a multiset. We also define $\overline{\gg}_\bullet^\Lambda = ((\succ, \overline{\gg}_\bullet)_{lex})_{mul}$.

*This research was supported by the Austrian Science Fund (FWF) projects P22467 and P27528.



(a) Locally decreasing diagram. (b) Locally point-decreasing diagram.

Figure 1: Decreasing diagrams.

Example 2. Let $L = \{1, 2, 3\}$ equipped with the order $3 \succ 2 \succ 1$. We claim that $\bar{2}\grave{3}\bar{1} \mathrel{\overline{\gg}_\bullet} \grave{3}\bar{2}\acute{2}\bar{1}$. In the resulting multiset comparison, it's easy to see that $(\grave{3}, \bar{2})$ is larger than every element of the right-hand side multiset:

$$\langle \bar{2}\grave{3}\bar{1}\rangle^g = [(\bar{2}, \epsilon), (\grave{3}, \bar{2}), (\bar{1}, \epsilon)] \mathrel{\overline{\gg}^\Lambda_\bullet} [(\grave{3}, \epsilon), (\bar{2}, \epsilon), (\acute{2}, \bar{1}), (\bar{1}, \epsilon)] = \langle \grave{3}\bar{2}\acute{2}\bar{1}\rangle^g$$

In [2] it is shown that $\mathrel{\overline{\gg}_\bullet}$ is a well-founded order on Greek strings that is monotonic, i.e., $q \mathrel{\overline{\gg}_\bullet} r$ implies $pq \mathrel{\overline{\gg}_\bullet} pr$ and $qp \mathrel{\overline{\gg}_\bullet} rp$ for all Greek strings p, q and r.

We recall van Oostrom's and Bognar's decreasing diagrams results. (In [3, Section 3.5] it is shown that Bognar's result follows from van Oostrom's, but this is not immediately obvious.)

Theorem 3 (decreasing diagrams [4])**.** *Let $\langle \mathcal{A}, \to_\kappa)_{\kappa\in L}$ be a labeled ARS. If for all $\kappa, \mu \in L$*

$$\underset{\kappa}{\leftarrow} \cdot \underset{\mu}{\rightarrow} \subseteq \overset{*}{\underset{\vee\kappa}{\leftrightarrow}} \cdot \overset{=}{\underset{\mu}{\rightarrow}} \cdot \overset{*}{\underset{\vee\kappa\mu}{\leftrightarrow}} \cdot \overset{=}{\underset{\kappa}{\leftarrow}} \cdot \overset{*}{\underset{\vee\mu}{\leftrightarrow}}$$

then $\to_L$ is confluent. (See also Figure 1(a).)

Theorem 4 (point-decreasing diagrams [1])**.** *Let $\langle \mathcal{A}, \to\rangle$ be an abstract rewrite system and $\ell : \mathcal{A} \to L$ be a function labeling the objects. We annotate steps by the labels of their targets in square brackets, that is, we write $s \to_{[\ell(t)]} t$. For $M \subset L$ we define $\to_{[M]} = \bigcup_{\kappa\in M} \to_{[\kappa]}$. If every local peak $t \,{}_{[\kappa]}\!\leftarrow s \to_{[\nu]} u$ with $\mu = \ell(s)$ has a joining valley*

$$t \overset{*}{\underset{[\vee\kappa\mu]}{\longrightarrow}} \cdot \overset{=}{\underset{[\nu]}{\rightarrow}} \cdot \overset{*}{\underset{[\vee\kappa\mu\nu]}{\longrightarrow}} \cdot \overset{*}{\underset{[\vee\kappa\mu\nu]}{\longleftarrow}} \cdot \overset{=}{\underset{[\kappa]}{\leftarrow}} \cdot \overset{*}{\underset{[\vee\mu\nu]}{\longleftarrow}} u$$

then $\to$ is confluent. (See also Figure 1(b).)

3 Point-Step Decreasing Diagrams

In this section we present a unified result that encompasses both standard [4] and point-decreasing diagrams [1] results, and is, to our knowledge, strictly more general than either one. The key idea is to use a representation of conversions as Greek strings that alternates between steps and objects, mapping objects to macron letters and steps to accented letters.

Definition 5. Let $\langle \mathcal{A}, (\to_\kappa)_{\kappa\in L}\rangle$ be a labeled ARS. Furthermore let $\ell : \mathcal{A} \to L$ be a function labeling the objects. Then each conversion

$$s_0 \underset{\kappa_1}{\longleftrightarrow} \cdots \underset{\kappa_n}{\longleftrightarrow} s_n$$

S	$\hat{\kappa} \mathrel{\overline{\gg}_\bullet} \{\kappa\succ\}$
M1	$\acute{\kappa}\bar{\mu} \mathrel{\overline{\gg}_\bullet} \{\kappa\mu\succ\}[\acute{\kappa}]\{\mu\succ\}$
M2	$\acute{\kappa}\bar{\mu} \mathrel{\overline{\gg}_\bullet} \{\kappa\succ\}\bar{\mu}\{\kappa\succ\}[\acute{\kappa}](\{\kappa\succ\} \cap \{\mu\succ\})$
P1	$\acute{\kappa}\bar{\mu}\grave{\nu} \mathrel{\overline{\gg}_\bullet} \{\kappa\mu\succ\}[\acute{\kappa}]\{\mu\succ\}[\grave{\nu}]\{\mu\nu\succ\}$
P2	$\acute{\kappa}\bar{\mu}\grave{\nu} \mathrel{\overline{\gg}_\bullet} \{\kappa\mu\succ\}[\grave{\nu}]\{\kappa\mu\nu\succ\}[\acute{\kappa}]\{\mu\nu\succ\}$
P3	$\acute{\kappa}\bar{\mu}\grave{\nu} \mathrel{\overline{\gg}_\bullet} \{\kappa\succ\}\bar{\mu}\{\kappa\succ\}[\acute{\kappa}](\{\kappa\succ\} \cap \{\mu\succ\})[\grave{\nu}]\{\nu\succ\}$
P4	$\acute{\kappa}\bar{\mu}\grave{\nu} \mathrel{\overline{\gg}_\bullet} \{\kappa\succ\}\bar{\mu}\{\kappa\succ\}[\grave{\nu}]\{\kappa\nu\succ\}[\acute{\kappa}]\{\nu\succ\}$
P5	$\acute{\kappa}\bar{\mu}\grave{\nu} \mathrel{\overline{\gg}_\bullet} \{\kappa\succ\}[\grave{\nu}]\{\kappa\nu\succ\}\bar{\mu}\{\kappa\nu\succ\}[\acute{\kappa}]\{\nu\succ\}$

Table 1: Comparing short Greek strings.

has a *point-step interpretation* $\hat{\kappa}_1\bar{\ell}(s_1)\dots\bar{\ell}(s_{n-1})\hat{\kappa}_n$, where

$$\hat{\kappa}_i = \begin{cases} \acute{\kappa}_i & \text{if } s_{i-1} \mathrel{{}_{\kappa_i}\!\leftarrow} s_i \\ \grave{\kappa}_i & \text{if } s_{i-1} \rightarrow_{\kappa_i} s_i \end{cases}$$

Note that the initial and final objects are omitted from the interpretation.

In order to obtain a point-step decreasing diagrams result, we map each conversion to its point-step interpretation, and then compare the resulting interpretations before and after pasting a local diagram by $\mathrel{\overline{\gg}_\bullet}$. Pasting a local diagram corresponds to replacing a substring $\acute{\kappa}\bar{\mu}\grave{\nu}$ (i.e., the interpretation of a local peak without its endpoints) by some other Greek string corresponding to the joining conversion (again, without endpoints). We omit the endpoints because their labels do not change.

Lemma 6. *All comparisons from Table 1 are true.*

Proof. Cases S, M1 and M2 are covered by [2, Lemma 31]. Consider P1. Let $s = \acute{\kappa}\bar{\mu}\grave{\nu}$ and $t \in \{\kappa\mu{>}\}[\acute{\kappa}]\{\mu{>}\}[\grave{\nu}]\{\mu\nu{>}\}$. Then

$$\langle s\rangle^g = [(\acute{\kappa}, \bar{\mu}\grave{\nu}), (\bar{\mu}, \epsilon), (\grave{\nu}, \acute{\kappa}\bar{\mu})]$$

We claim that for all elements of $\langle t\rangle^g$ there is a larger element in $\langle s\rangle^g$, which establishes $\langle s\rangle^g \mathrel{\overline{\gg}^\Lambda_\bullet} \langle t\rangle^g$. Indeed for pairs corresponding to $\{\kappa\mu{>}\}$, $\{\mu{>}\}$ or $\{\mu\nu{>}\}$, the first component is smaller than κ, μ or ν, and therefore the claim follows. The element corresponding to $[\grave{\mu}]$ (if any) is in $(\grave{\mu}, \{\kappa\mu{>}\}[\acute{\kappa}]\{\mu{>}\})$, and that is smaller than $(\grave{\nu}, \acute{\kappa}\bar{\mu})$ by property M1. For the element corresponding to $[\acute{\kappa}]$, a symmetric argument applies. Therefore, P1 holds.

For P2…P5 the same basic approach works: Apply $\langle\cdot\rangle^g$ to both sides, and then establish the resulting multiset comparison by $\mathrel{\overline{\gg}^\Lambda_\bullet}$ using $>$ on the first component and properties M1 and M2 on the second component of the resulting pairs. □

Example 7. Let (M) denote an object whose label is in $M \subseteq L$, and let $\leftrightarrow^*_M$ stand for a conversion with all steps and intermediate objects (not including the initial and final objects of the conversion) having labels in M. Property P2 corresponds to the following conversion joining a local peak $t \mathrel{{}_\kappa\!\leftarrow} s \rightarrow_\nu u$, where $\mu = \ell(s)$:

$$t \underset{\vee\kappa\mu}{\overset{*}{\longleftrightarrow}} (\vee\kappa\mu) \underset{\nu}{\overset{=}{\rightarrow}} (\vee\kappa\mu\nu) \underset{\vee\kappa\mu\nu}{\overset{*}{\longleftrightarrow}} (\vee\kappa\mu\nu) \underset{\kappa}{\overset{=}{\leftarrow}} (\vee\mu\nu) \underset{\vee\mu\nu}{\longleftrightarrow} u$$

Theorem 8. *Let $\langle\mathcal{A}, (\rightarrow_\kappa)_{\kappa\in L}\rangle$ be a labeled ARS, and $\ell : \mathcal{A} \to L$ be a labeling function. Assume that every local peak $t \mathrel{{}_\kappa\!\leftarrow} s \rightarrow_\nu u$, with $\mu = \ell(s)$ has a joining conversion $s \leftrightarrow^* t$ whose interpretation matches a right-hand side of P1, P2, P3, P4 or P5 from Table 1. Then $\rightarrow$ is confluent.*

Proof. Recall that $\overline{\gg}_\bullet$ is well-founded and monotonic. We show that every conversion $t \leftrightarrow^* u$ has an equivalent valley proof $t \to^* \cdot {}^*\!\leftarrow u$ by well-founded induction on $t \leftrightarrow^* u$, measured by the interpretation of the conversion according to Definition 5 and ordered according to $\overline{\gg}_\bullet$. If $t \leftrightarrow^* u$ is a valley proof then we are done. Otherwise, there must be a local peak, say

$$t \overset{*}{\leftrightarrow} t' \underset{\kappa}{\leftarrow} s' \underset{\nu}{\rightarrow} u' \overset{*}{\leftrightarrow} u \tag{P}$$

Let $\mu = \ell(s')$. Then the interpretation of (P) can be written as $p\acute{\kappa}\bar{\mu}\grave{\nu}r$, where p is the interpretation of $t \leftrightarrow^* t'$ followed by $\bar{\ell}(t')$ (if the conversion is non-empty) and r is the interpretation of $u' \leftrightarrow^* u$ preceded by $\bar{\ell}(u')$ (if the conversion is non-empty) By assumption, the local peak $t' \;{}_\kappa\!\leftarrow s' \rightarrow_\nu u'$ has a joining conversion $t' \leftrightarrow^* u'$ whose interpretation q satisfies $\acute{\kappa}\bar{\mu}\grave{\nu} \;\overline{\gg}_\bullet\; q$ by Lemma 6. By monotonicity, this implies $p\acute{\kappa}\bar{\mu}\grave{\nu}r \;\overline{\gg}_\bullet\; pqr$. Consequently, we can apply the induction hypothesis to the resulting conversion $t \leftrightarrow^* t' \leftrightarrow^* u' \leftrightarrow^* u$, whose interpretation is pqr (or smaller if $t' \leftrightarrow^* u'$ is an empty conversion), to conclude. □

Remark 9. Theorem 8 entails both decreasing diagrams and point-decreasing diagrams. To obtain decreasing diagrams, simply label all objects by a fresh label $\bot$ that is minimal with respect to $\succ$. Then case D of Table 1 (which encodes a decreasing diagram) corresponds to case P2, noting that each of the sets $\{\kappa\mu\succ\}$, $\{\kappa\mu\nu\succ\}$ and $\{\mu\nu\succ\}$ contains $\grave{\bot}$.

In order to obtain point-decreasing diagrams, let ℓ_p be the labeling function for establishing point-decreasingness. We label steps $s \to t$ by $(\ell_p(t), \top)$ and objects s by $(\ell_p(s), \bot)$, with the tuples ordered lexicographically by $\succ$ on the original L and $\top \succ \bot$. Then if we consider the joining conversion from Theorem 4, we easily see that it corresponds to case P2 of Table 6.

Corollary 10. *Let $\langle \mathcal{A}, \to\rangle$ be an ARS, and $\ell : \mathcal{A} \to L$ a labeling function. If for every peak $t \leftarrow s \to u$, either*

$$t \to v \leftarrow u \qquad \text{or} \qquad t \leftrightarrow v_1 \cdots v_n \leftrightarrow u$$

such that $\ell(s) = \ell(v)$ or $\ell(s) \succ \ell(v_i)$ for $1 \le i \le n$, then $\to$ is confluent.

Proof. Let $L_\bot = L \cup \{\bot\}$, where $\bot$ is a fresh label. We extend $\succ$ to $L_\bot$ by letting $\alpha \succ \bot$ for all $\alpha \in L$. Let $(\to_\alpha)_{\alpha \in L_\bot}$ be the labeled ARS given by $\to_\bot = \to$ and $\to_\alpha = \varnothing$. Then since both

$$\acute{\bot}\bar{\ell}(s)\grave{\bot} \;\overline{\gg}_\bullet\; \grave{\bot}\bar{\ell}(v)\acute{\bot}$$

by P5 from Table 1 and

$$\acute{\bot}\bar{\ell}(s)\grave{\bot} \;\overline{\gg}_\bullet\; \hat{\bot}\bar{\ell}(v_1)\dots\bar{\ell}(v_n)\hat{\bot}$$

by P2, we conclude that the local diagrams are point-step decreasing and therefore $\to$ is confluent. □

Remark 11. Corollary 10 is interesting because to our knowledge, neither decreasing diagrams nor point-decreasing diagrams can prove it directly, without changing the joining conversions for the local peaks. (In particular, while step-decreasing diagrams are complete for confluence of countable ARSs, the proof picks particular joining valleys for local peaks.) This indicates that point-step decreasing diagrams strictly generalize decreasing diagrams and point-decreasing diagrams for practical purposes.

For the point-decreasing diagrams, the main obstacle presents itself as follows: In order to obtain a point-decreasing diagram (with labeling function ℓ') for joining $t \leftarrow s \to u$ as $t \leftrightarrow v_1 \dots v_n \leftrightarrow u$ for $n > 1$, since we cannot assume anything about the labels of t and u, we will have to ensure that $\ell'(s) > \ell'(v_i)$ whenever $\ell(s) > \ell(v_i)$. This suggests using $\ell' = \ell$.

But then the case of joining $t \to v \leftarrow u$ with $\ell(s) = \ell(v)$ does not result in a point-decreasing diagram unless $\ell(t) \succ \ell(s)$ or $\ell(u) \succ \ell(s)$ holds. So the proof attempt fails.

For decreasing diagrams, the picture is less clear. Let us assume that the order $\succ$ on labels is total. The simplest labeling that makes the joining conversions $t \leftrightarrow v_1 \dots v_n \leftrightarrow u$ decreasing labels each step $s \to t$ by $\max(\ell(s), \ell(t))$. Then for the $t \to v \leftarrow u$ join, we have to join the peak $t \,{}_\kappa\!\leftarrow s \to_\mu u$ (with $\kappa = \max(\ell(s), \ell(t))$ and $\mu = \max(\ell(s), \ell(u))$) by $t \to_\kappa v \,{}_\mu\!\leftarrow u$. However, this is only a decreasing diagram if $\kappa = \mu$, and we cannot ensure this in general.

Remark 12. The results presented here extend to Church-Rosser modulo. Recall that an ARS $\to$ is Church-Rosser modulo $\mapsto$ if $\leftrightarrow^* \subseteq \to^* \cdot \mapsto^* \cdot {}^*\!\leftarrow$, where $\mapsto$ is a symmetric relation. The idea is that equality steps $\mapsto_\kappa$ are mapped to macron letters $\bar{\kappa}$ when interpreting conversions. In addition to interpretations of peaks $\acute{\kappa}\bar{\mu}\grave{\nu}$ one also needs to consider interpretations of cliffs $\acute{\kappa}\bar{\mu}\bar{\nu}$ in Lemma 6. Details can be found in [3, Chapter 3.6].

4 Conclusion

We have presented an extension of decreasing diagrams in which both steps and objects of an ARS are labeled. As argued in Remark 11, the extension appears to be more powerful than standard decreasing diagrams. Note that while decreasing diagrams are complete for confluence of countable ARSs, this fact does not often help us with finding concrete labelings that establish confluence, since cofinal derivations are hard to describe. On the other hand, Table 1 is quite intimidating, so it remains to be seen whether this result will be useful in practice. Perhaps the labeling framework from [5] could be extended to label terms as well.

References

[1] M. Bognar. A point version of decreasing diagrams. In *Proc. Accolade 1996*, Dutch Graduate School in Logic, pages 1–14, 1997.

[2] B. Felgenhauer and V. van Oostrom. Proof orders for decreasing diagrams. In *Proc. 24th RTA*, number 21 in LIPIcs, pages 174–189, 2013.

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

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

[5] H. Zankl, B. Felgenhauer, and A. Middeldorp. Labelings for decreasing diagrams. *JAR*, 54(2):101–133, 2015.

Infeasible Conditional Critical Pairs*

Thomas Sternagel and Aart Middeldorp

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

1 Introduction

This paper is concerned with automatically proving (non-)confluence of conditional term rewrite systems (CTRSs). Although confluence of CTRSs has been investigated for decades, the first tools (CO3[1] and ConCon [8]) appeared in 2014. Several of the techniques implemented in these tools would benefit if certain conditional critical pairs (CCPs) are determined to be *infeasible*, which means that their conditional parts are not satisfiable. For instance, the CCP $\mathsf{f}(\mathsf{c}) \approx \mathsf{b} \Leftarrow x \approx \mathsf{a}, x \approx \mathsf{c}$ of the oriented CTRS $\mathcal{R}$

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

is infeasible since no term rewrites to both a and c. Hence $\mathcal{R}$ is orthogonal and thus confluent.

In this paper we present an overview of infeasibility methods for oriented 3-CTRSs, one of the most popular types of conditional rewriting. In such systems extra variables in conditions and right-hand sides of rewrite rules are allowed. Moreover, satisfiability of the conditions amounts to reachability. As a consequence of the latter, establishing infeasibility is similar to the problem of eliminating arrows in dependency graph approximations, a problem which has been investigated extensively in the literature. The difference is that we deal with CTRSs and the terms we test may share variables.

In the sequel we summarize the methods that we have analyzed and adapted for infeasibility. The methods have been implemented in our confluence tool ConCon and experimental data will be presented.

2 Preliminaries

We assume familiarity with (conditional) term rewriting and related topics [7]. We very briefly recall important concepts that are used in the sequel.

Given two variants $\ell_1 \to r_1 \Leftarrow c_1$ and $\ell_2 \to r_2 \Leftarrow c_2$ without common variables of rules in a CTRS $\mathcal{R}$, a position $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$, and an mgu σ of ℓ_1 and $\ell_2|_p$, the conditional equation $\ell_2\sigma[r_1\sigma]_p \approx r_2\sigma \Leftarrow c_1\sigma, c_2\sigma$ is a CCP of $\mathcal{R}$. As usual, we exclude the case that $p = \epsilon$ and the rules are variants of the same rule. A CCP $s \approx t \Leftarrow s_1 \approx t_1, \ldots, s_k \approx t_k$ is *infeasible* if there is no substitution σ such that $s_i\sigma \to_{\mathcal{R}}^* t_i\sigma$ for all $1 \leqslant i \leqslant k$. We write $\mathsf{cs}(\vec{s})$ and $\mathsf{cs}(\vec{t})$ for the terms $\mathsf{cs}(s_1, \ldots, s_k)$ and $\mathsf{cs}(t_1, \ldots, t_k)$. Here cs is a fresh function symbol of arity k.

The set of ground instances of a term t is denoted by $\Sigma(t)$ and we write $s \nabla t$ to denote that the terms s and t are unifiable. The sets of $\mathcal{R}$-ancestors and $\mathcal{R}$-descendants of a set of terms T are defined as $(\to_{\mathcal{R}}^*)[T] = \{s \mid s \to_{\mathcal{R}}^* t \text{ for some } t \in T\}$ and $[T](\to_{\mathcal{R}}^*) = \{t \mid s \to_{\mathcal{R}}^* t \text{ for some } s \in T\}$, respectively. A TRS $\mathcal{R}$ is called *growing* if variables in $\mathcal{V}\mathsf{ar}(\ell) \cap \mathcal{V}\mathsf{ar}(r)$ occur at depth at most one in ℓ, for all $\ell \to r \in \mathcal{R}$. The *growing approximation* [6] of a TRS $\mathcal{R}$ is denoted by $\mathsf{g}(\mathcal{R})$. We

*The research described in this paper is supported by FWF (Austrian Science Fund) project I963.

[1] http://www.trs.cm.is.nagoya-u.ac.jp/co3/

call $\mathcal{R}$ *regularity preserving* if $[T](\to_{\mathcal{R}}^*)$ is regular whenever T is regular. We write $\mathsf{ren}(t)$ for a linearization of the term t using fresh variables. As usual $\mathcal{R}^{-1}$ denotes the inverse of a TRS $\mathcal{R}$.

To apply methods developed for unconditional TRSs we need some transformation from CTRSs to TRSs. In its simplest form this means to just forget about the conditions, giving rise to the *underlying* TRS $\mathcal{R}_{\mathsf{u}} = \{\ell \to r \mid \ell \to r \Leftarrow c \in \mathcal{R}\}$ of a CTRS $\mathcal{R}$. More sophisticated transformations modify the signature. For that reason a transformation T comes equipped with an encoding function $\sharp\colon \mathcal{T}(\mathcal{F},\mathcal{V}) \to \mathcal{T}(\mathcal{F}',\mathcal{V})$ and a partial decoding function $\flat\colon \mathcal{T}(\mathcal{F}',\mathcal{V}) \to \mathcal{T}(\mathcal{F},\mathcal{V})$. Here $\mathcal{F}$ is the signature of the CTRS $\mathcal{R}$ under consideration, $\mathcal{F}'$ is the signature of the transformed TRS $\mathsf{T}(\mathcal{R})$, and we require that $\flat(\sharp(t)) = t$ for all terms $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$. In case of $\mathcal{R}_{\mathsf{u}}$ both $\sharp$ and $\flat$ are just the identity. To be useful for infeasibility checking a transformation $(\mathsf{T},\sharp,\flat)$ has to be *complete*, i.e., if $s \to_{\mathcal{R}}^* t$ then $\sharp(s) \to_{\mathsf{T}(\mathcal{R})}^* \sharp(t)$ for all terms $s,t \in \mathcal{T}(\mathcal{F},\mathcal{V})$. The transformation from $\mathcal{R}$ to $\mathcal{R}_{\mathsf{u}}$ obviously has this property. In our experiments we also employed other transformations which are known to be complete for certain kinds of CTRSs, including the (optimized) unraveling $\mathsf{U}_{\mathsf{opt}}$ [4,7] as well as the structure-preserving transformation SR [10].

3 Unification

In [8] we already reported on the use of the tcap function in ConCon for checking infeasibility. Given a TRS $\mathcal{R}$, it is defined as $\mathsf{tcap}_{\mathcal{R}}(t) = u$ if $t = f(t_1,\dots,t_n)$ and $u \mathrel{\not\triangledown} \ell$ for all $\ell \to r \in \mathcal{R}$, and $\mathsf{tcap}_{\mathcal{R}}(t) = y$ otherwise, where $u = f(\mathsf{tcap}_{\mathcal{R}}(t_1),\dots,\mathsf{tcap}_{\mathcal{R}}(t_n))$ and y is a fresh variable.

Lemma 3.1. *Let $\mathcal{R}$ be a TRS. For terms s and t, if $\mathsf{tcap}_{\mathcal{R}}(s) \mathrel{\not\triangledown} t$ then $s\sigma \not\to_{\mathcal{R}}^* t\tau$ for all substitutions σ and τ.*

In [8] we used the above lemma to test the conditions in CCPs separately. Here we combine the conditions to obtain a strictly more powerful criterion for infeasibility. Another difference is that the criterion is now parameterized by an arbitrary complete transformation.

Corollary 3.2 (TCAP)**.** *Let $\mathcal{R}$ be an oriented 3-CTRS $\mathcal{R}$ and $(\mathsf{T},\sharp,\flat)$ a complete transformation. A CCP $s \approx t \Leftarrow s_1 \approx t_1,\dots,s_k \approx t_k$ of $\mathcal{R}$ is infeasible if $\mathsf{tcap}_{\mathsf{T}(\mathcal{R})}(\mathsf{cs}(\sharp(\vec{s}))) \mathrel{\not\triangledown} \mathsf{cs}(\sharp(\vec{t}))$.*

Example 3.3. Consider the CTRS $\mathcal{R}$ consisting of the two rules

$$\mathsf{f}(x) \to \mathsf{a} \;\Leftarrow\; \mathsf{a} \approx x \qquad\qquad \mathsf{f}(x) \to \mathsf{b} \;\Leftarrow\; \mathsf{b} \approx x$$

The two CCPs $\mathsf{a} \approx \mathsf{b} \Leftarrow \mathsf{a} \approx x$, $\mathsf{b} \approx x$ and $\mathsf{b} \approx \mathsf{a} \Leftarrow \mathsf{b} \approx x$, $\mathsf{a} \approx x$ are infeasible by Corollary 3.2 because $\mathsf{cs}(\mathsf{a},\mathsf{b})$ and $\mathsf{cs}(\mathsf{b},\mathsf{a})$ do not unify with $\mathsf{cs}(x,x)$.

4 Tree Automata Techniques

Tree automata techniques are another method that were used to approximate dependency graphs in termination analysis. The following result is from [6].

Lemma 4.1. *Let $\mathcal{R}$ be a regularity preserving TRS. For terms s and t, if $[\Sigma(\mathsf{ren}(s))](\to_{\mathcal{R}}^*) \cap \Sigma(t) = \varnothing$ or $\Sigma(s) \cap (\to_{\mathcal{R}}^*)[\Sigma(\mathsf{ren}(t))] = \varnothing$ then $s\sigma \not\to_{\mathcal{R}}^* t\tau$ for all substitutions σ and τ.*

From this result we obtain the following general infeasibility criterion.

Corollary 4.2 (TAC). *Let $\mathcal{R}$ be an oriented 3-CTRS $\mathcal{R}$ and $(\mathsf{T}, \sharp, \flat)$ a complete transformation. A CCP $s \approx t \Leftarrow s_1 \approx t_1, \ldots, s_k \approx t_k$ of $\mathcal{R}$ is* infeasible *if one of the following intersections is empty:*[2]

$$[\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{s}))))](\xrightarrow[\mathsf{T}(\mathcal{R})]{*}) \cap \Sigma(\mathsf{cs}(\sharp(\vec{t}))) \qquad [\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{t}))))](\xrightarrow[\mathsf{T}(\mathcal{R})^{-1}]{*}) \cap \Sigma(\mathsf{cs}(\sharp(\vec{s})))$$

To obtain an effective criterion for infeasibility based on Corollary 4.2, we need to construct a tree automaton that over-approximates the ground terms in the sets of descendants or ancestors. There are basically three ways to achieve this.

The first method is to replace $\mathsf{T}(\mathcal{R})$ and $\mathsf{T}(\mathcal{R})^{-1}$ by their *growing approximations*. Adopting a construction of Jacquemard [5], we obtain an exact tree automaton representation of the over-approximation (due to the growing approximation) of the sets of ancestors. Since the growing approximation of $\mathsf{T}(\mathcal{R})^{-1}$ is different from the inverse of the growing approximation of $\mathsf{T}(\mathcal{R})$, we test the emptiness of the following two intersections:

$$\Sigma(\mathsf{cs}(\sharp(\vec{s}))) \cap (\xrightarrow[\mathsf{g}(\mathsf{T}(\mathcal{R}))]{*})[\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{t}))))] \qquad \Sigma(\mathsf{cs}(\sharp(\vec{t}))) \cap (\xrightarrow[\mathsf{g}(\mathsf{T}(\mathcal{R})^{-1})]{*})[\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{s}))))]$$

This method will be referred to as eTAC in the sequel.

In the second method we attempt to construct a tree automaton for sets of descendants by a process known as tree automata completion, an idea which goes back to Genet [3]. This process is parameterized by an abstraction function which limits the number of newly generated states during completion, thereby providing a trade-off between the termination behavior (and thus runtime) of the process and its accuracy. It takes the tree automaton which represents a regular set (like $\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{s}))))$) and a left-linear TRS (like $\mathsf{T}(\mathcal{R})$) as input. Compatibility violations between the tree automaton and the TRS are resolved in an iterative process. Termination depends on the employed abstraction function. We test both $\mathsf{T}(\mathcal{R})$ and $\mathsf{T}(\mathcal{R})^{-1}$:

$$[\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{s}))))](\xrightarrow[\mathsf{T}(\mathcal{R})]{*}) \cap \Sigma(\mathsf{cs}(\sharp(\vec{t}))) \qquad [\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{t}))))](\xrightarrow[\mathsf{T}(\mathcal{R})^{-1}]{*}) \cap \Sigma(\mathsf{cs}(\sharp(\vec{s})))$$

This method will be referred to as uTAC in the sequel.

In [1] Feuillade and Genet present a version of tree automata completion which operates directly on CTRSs. They showed that this direct approach results in smaller tree automata (thereby reducing the possibility of divergence of the completion process) compared to tree automata completion applied to $\mathcal{R}_{\mathsf{u}}$. We adapted the procedure, which is defined in [1] for join 1-CTRSs with at most one condition per rule, to oriented 3-CTRSs with an arbitrary number of conditions. Here we check the following two intersections for emptiness:

$$[\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{s}))))](\xrightarrow[\mathcal{R}]{*}) \cap \Sigma(\mathsf{cs}(\sharp(\vec{t}))) \qquad [\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{t}))))](\xrightarrow[\mathcal{R}^{-1}]{*}) \cap \Sigma(\mathsf{cs}(\sharp(\vec{s})))$$

This third method will be referred to as cTAC in the sequel.

5 Equational Reasoning

The final method, referred to as ER in Section 6, that we present for infeasibility was also first used for computing dependency graphs [9]. It employs Waldmeister [2], a powerful automatic

[2] Of course we can also use ancestors instead of descendants, provided we interchange $\mathsf{cs}(\sharp(\vec{s}))$ and $\mathsf{cs}(\sharp(\vec{t}))$ in the intersections. Depending on the concrete algorithm, ancestors or descendants might be preferable.

	TCAP			eTAC			ER			all
	$\mathcal{R}_\mathsf{u}$	U_opt	SR	$\mathcal{R}_\mathsf{u}$	U_opt	SR	$\mathcal{R}_\mathsf{u}$	U_opt	SR	
confluent	2	2	0	5	4	0	5	0	0	8
infeasible	16	16	2	46	40	0	17	0	0	61
maybe	220	220	234	190	196	236	219	236	236	175
timeout	2	2	2	2	6	17	2	2	2	1
avg. time	5.66	5.75	5.70	7.12	12.84	26.68	5.82	5.76	6.91	12.30

Table 1: Results for 46 oriented 3-CTRSs with at least one CCP (236 CCPs in total).

theorem prover for equational logic with uninterpreted function symbols. Waldmeister uses a variant of ordered completion to determine for a given set of equations $\mathcal{R}$ and a goal equation (called conclusion) $s \approx t$ whether there exist substitutions σ and τ such that $s\sigma \leftrightarrow^*_\mathcal{R} t\tau$. If Waldmeister refutes the conclusion then surely there are no substitutions σ and τ such that $s\sigma \to^*_\mathcal{R} t\tau$.

Example 5.1. Consider system 361 from Cops:[3]

$$\begin{aligned}
0 \leqslant x &\to \mathsf{true} & \mathsf{s}(x) > 0 &\to \mathsf{true} & x - 0 &\to x \\
\mathsf{s}(x) \leqslant \mathsf{s}(y) &\to x \leqslant y & \mathsf{s}(x) > \mathsf{s}(y) &\to x > y & 0 - x &\to 0 \\
x \div y &\to \langle 0, y\rangle & \Leftarrow\ y > x &\approx \mathsf{true} & \mathsf{s}(x) - \mathsf{s}(y) &\to x - y \\
x \div y &\to \langle \mathsf{s}(q), r\rangle & \Leftarrow\ y \leqslant x &\approx \mathsf{true},\ (x - y) \div y \approx \langle q, r\rangle
\end{aligned}$$

This CTRS has two trivial unconditional CPs and one (modulo symmetry) CCP

$$\langle 0, x\rangle \approx \langle \mathsf{s}(y), z\rangle \Leftarrow x \leqslant w \approx \mathsf{true},\ (w - x) \div x \approx \langle y, z\rangle,\ x > w \approx \mathsf{true}$$

which is infeasible because of the contradictory conditions $x \leqslant w \approx \mathsf{true}$ and $x > w \approx \mathsf{true}$. This is confirmed by Waldmeister in conjunction with the $\mathcal{R} \mapsto \mathcal{R}_\mathsf{u}$ transformation.[4]

6 Experiments

Our test bed consists of 46 oriented 3-CTRSs from the Cops problem collection which have *at least one* CCP. These 46 CTRSs have 236 CCPs. Our experiments have been conducted on a 64 bit GNU/Linux machine. The time limit was set to 60 seconds. In Table 1 we compare combinations of the various infeasibility methods with different transformations.

The row labeled 'confluent' lists the number of systems which are confluent but could not be shown confluent *without any* infeasibility methods. The next line lists the number of CCPs which could be shown to be infeasible with each method. The rows labeled 'maybe' and 'timeout' give the number of CCPs for which infeasibility could not be shown (within the time limit).

Without any infeasibility checking ConCon could show 11 CTRSs confluent and 1 CTRS non-confluent (with 2 timeouts) with an average time of 5.8 s. Using TCAP yields another 2 confluent systems but these CTRSs are also handled by eTAC and ER. The more involved tree automata methods uTAC and cTAC could not improve upon eTAC and are not listed in the table.

[3] http://cops.uibk.ac.at

[4] The eTAC method together with the unraveling U_opt also shows infeasibility of this CCP.

The eTAC and ER methods are incomparable in power, the first one succeeding on systems 288, 292, 330, 336, 361, and 409, the second one on systems 336, 361, 406, 407, and 409. The unraveling $\mathsf{U}_{\mathsf{opt}}$ could improve upon $\mathcal{R}_{\mathsf{u}}$ in one instance (system 361) while SR could not. All in all, ConCon can show 19 of the 46 systems confluent using the infeasibility methods, which handle 61 of the 236 CCPs (with eTAC handling the most).

We conclude with an example where the methods of this paper are not helpful.

Example 6.1. Consider system 327 from Cops:

$$\begin{array}{rrr}
\mathsf{gcd}(x,x) \to x & x < 0 \to \mathsf{false} & 0 - \mathsf{s}(y) \to 0 \\
\mathsf{gcd}(\mathsf{s}(x),0) \to \mathsf{s}(x) & 0 < \mathsf{s}(y) \to \mathsf{true} & x - 0 \to x \\
\mathsf{gcd}(0,\mathsf{s}(y)) \to \mathsf{s}(y) & \mathsf{s}(x) < \mathsf{s}(y) \to x < y & \mathsf{s}(x) - \mathsf{s}(y) \to x - y \\
\mathsf{gcd}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{gcd}(x-y,\mathsf{s}(y)) \Leftarrow y < x \approx \mathsf{true} & & \\
\mathsf{gcd}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{gcd}(\mathsf{s}(y),x-y) \Leftarrow x < y \approx \mathsf{true} & &
\end{array}$$

This CTRS has six CCPs of which we show two (the conditions of the others are similar):

$$\begin{array}{l}
\mathsf{gcd}(\mathsf{s}(x),y-x) \approx \mathsf{gcd}(x-y,\mathsf{s}(y)) \Leftarrow y < x \approx \mathsf{true},\ x < y \approx \mathsf{true} \\
\mathsf{gcd}(x-x,\mathsf{s}(x)) \approx \mathsf{s}(x) \Leftarrow x < x \approx \mathsf{true}
\end{array}$$

These CCPS are obviously infeasible, but this cannot be shown by the methods of this paper. For instance, when using $\mathcal{R}_{\mathsf{u}}$ we open the door for inconsistencies:

$$\mathsf{s}(0) \mathrel{{}^*\!\leftarrow} \mathsf{gcd}(\mathsf{s}(0),\mathsf{s}(0)) \mathrel{{}^*\!\leftarrow} \mathsf{gcd}(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0)) \to^* \mathsf{gcd}(0,\mathsf{s}(\mathsf{s}(0))) \to^* \mathsf{s}(\mathsf{s}(0))$$

and thus $\mathsf{gcd}(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0)) < \mathsf{gcd}(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0)) \to^* \mathsf{s}(0) < \mathsf{s}(\mathsf{s}(0)) \to^* \mathsf{true}$. Consequently, we may substitute $\mathsf{gcd}(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0))$ for both x and y to satisfy the conditions of the CCPs.

References

[1] G. Feuillade and T. Genet. Reachability in conditional term rewriting systems. In *Proc. 4th FTP*, volume 86 of *ENTCS*, pages 133–146, 2003.

[2] J.-M. Gaillourdet, Th. Hillenbrand, B. Löchner, and H. Spies. The new Waldmeister loop at work. In *Proc. 19th CADE*, volume 2741 of *LNCS*, pages 317–321. Springer, 2003.

[3] T. Genet. Decidable approximations of sets of descendants and sets of normal forms. In *Proc. 9th RTA*, volume 1379 of *LNCS*, pages 151–165, 1998.

[4] K. Gmeiner, N. Nishida, and B. Gramlich. Proving confluence of conditional term rewriting systems via unravelings. In *Proc. 2nd IWC*, pages 35–39, 2013.

[5] F. Jacquemard. Decidable approximations of term rewriting systems. In *Proc. 7th RTA*, volume 1103 of *LNCS*, pages 362–376, 1996.

[6] A. Middeldorp. Approximating dependency graphs using tree automata techniques. In *Proc. 1st IJCAR*, volume 2083 of *LNCS*, pages 593–610. Springer, 2001.

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

[8] 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.

[9] H. Zankl and A. Middeldorp. Equational reasoning for termination of rewriting. In *Proc. 10th WST*, pages 112–115, 2009.

[10] T. Şerbănuţă and G Roşu. Computationally equivalent elimination of conditions. In *Proc. 6th RTA*, volume 4098 of *LNCS*, pages 19–34. Springer, 2006.

Operational Confluence of Conditional Term Rewrite Systems*

Karl Gmeiner

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

Abstract

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

1 Introduction

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

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

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

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

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

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

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

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

Now, consider another CTRS:

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

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

2 Preliminaries

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

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

3 Operational Confluence

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

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

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

We introduce the following notion of closeable proof trees:

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

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

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

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

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

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

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

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

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

4 Operational Confluence of CTRSs

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5 Conclusion and Related Work

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

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

5.1 Relation to ultra-properties

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

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

References

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

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

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

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

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

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

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

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

Commutation and Signature Extensions*

Nao Hirokawa

JAIST, Japan

Abstract

We show that (local) commutation is preserved under signature extensions for left-linear term rewrite systems.

1 Introduction

Toyama's Theorem [14] on the modularity of confluence tells us that confluence of a term rewrite system (TRS) is preserved under signature extensions. Formally, if a TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is confluent then for every signature $\mathcal{G}$ with $\mathcal{F} \subseteq \mathcal{G}$ the TRS $\mathcal{R}$ over $\mathcal{G}$ is also confluent. In the setting of unsorted and unconditional first-order rewriting, signature extensions preserve most of the major rewriting properties: Just to name a few, local confluence, the unique normal form property, and (relative) termination (see [8, 11]).

What about commutation? Commutation is a generalization of confluence, and it has been successfully used for analyzing various properties like confluence and normalization (see e.g. [2, 5, 6, 15]). Because preservation under signature extensions is regarded as a trivial case of modularity [8, 14, 13, 15], it is a desirable property in these applications.

In this note we investigate how signature extensions affect commutation. After recalling the basic notions of term rewriting, we investigate preservation of local commutation in Section 3. In Section 4 we show that commutation is not preserved in general but left-linearity recovers the preservation.

2 Preliminaries

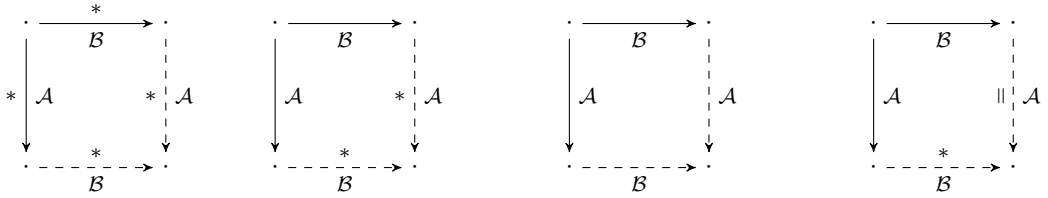
We assume familiarity with abstract rewriting and term rewriting [3].

Term rewrite systems. Let $\mathcal{V}$ be a countable set of variables and $\mathcal{F}$ a signature. The set of all terms built from $\mathcal{V}$ and $\mathcal{F}$ is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$. A pair (ℓ, r) of terms over $\mathcal{F}$ is called a rewrite rule, denoted by $\ell \to r$, if ℓ is not a variable and $\mathcal{V}\mathsf{ar}(r) \subseteq \mathcal{V}\mathsf{ar}(\ell)$. A TRS $\mathcal{R}$ over $\mathcal{F}$ is a set of rewrite rules over $\mathcal{F}$. The rewrite relation $\to_{\mathcal{R}}$ of $\mathcal{R}$ is defined as the smallest relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ containing $\mathcal{R}$ such that $\to_{\mathcal{R}}$ is closed under contexts and substitutions. Let $\mathcal{R}$ be a TRS over $\mathcal{F}$. We define the set $\mathcal{F}_{\mathcal{R}}$ as follows:

$$\mathcal{F}_{\mathcal{R}} = \bigcup_{\ell \to r \in \mathcal{R}} \mathcal{F}\mathsf{un}(\ell) \cup \mathcal{F}\mathsf{un}(r)$$

Here $\mathcal{F}\mathsf{un}(t)$ stands for the set of all function symbols occurring in a term t.

*Supported by JSPS KAKENHI Grant Number 25730004 and JSPS Core to Core Program.



(a) *commutation* (b) *local commutation* (c) *commuting diamond* (d) *strong commutation*

Figure 1: Various commutation properties.

Commutation. Let $\to_{\mathcal{A}}$ and $\to_{\mathcal{B}}$ be relations on the same set. The relations $\to_{\mathcal{A}}$ and $\to_{\mathcal{B}}$ *commute* if ${}^{*}_{\mathcal{A}}\!\leftarrow \cdot \to^{*}_{\mathcal{B}} \subseteq \to^{*}_{\mathcal{B}} \cdot {}^{*}_{\mathcal{A}}\!\leftarrow$, and *locally commute* if ${}_{\mathcal{A}}\!\leftarrow \cdot \to_{\mathcal{B}} \subseteq \to^{*}_{\mathcal{B}} \cdot {}^{*}_{\mathcal{A}}\!\leftarrow$. These inclusions are depicted in Figure 1(a,b). We say that two TRSs (locally) commute if their rewrite relations (locally) commute, respectively. We recall two basic commutation criteria. We say that relations $\to_{\mathcal{A}}$ and $\to_{\mathcal{B}}$ have the *commuting diamond property* if ${}_{\mathcal{A}}\!\leftarrow \cdot \to_{\mathcal{B}} \subseteq \to_{\mathcal{B}} \cdot {}_{\mathcal{A}}\!\leftarrow$ holds, and have the *strong commutation property* if ${}_{\mathcal{A}}\!\leftarrow \cdot \to_{\mathcal{B}} \subseteq \to^{*}_{\mathcal{B}} \cdot \overline{}^{}_{\mathcal{A}}\!\leftarrow$ holds, see Figure 1(c,d).

Lemma 1 ([5])**.** *Strong commutation implies commutation.* □

Lemma 2. *The commuting diamond property implies commutation.* □

Critical pairs. Conditions for commutation are often based on the notion of critical pairs. Let $\ell_1 \to r_1$ be a rule in a TRS $\mathcal{R}$ and $\ell_2 \to r_2$ a variant of a rule in a TRS $\mathcal{S}$ with $\mathcal{V}\mathsf{ar}(\ell_1) \cap \mathcal{V}\mathsf{ar}(\ell_2) = \varnothing$. When $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$ and σ is a most general unifier of ℓ_1 and $\ell_2|_p$, the pair $(\ell_2\sigma[r_1\sigma]_p, r_2\sigma)$ is called a *critical pair* of $\mathcal{R}$ on $\mathcal{S}$, and written $\ell_2\sigma[r_1\sigma]_p \; {}_{\mathcal{R}}\!\leftarrow \rtimes \to_{\mathcal{S}} r_2\sigma$.

3 Local Commutation

Local confluence is preserved under signature extensions. This can be easily shown by using the Critical Pair Lemma [7]. Local commutation of left-linear TRSs is also characterized by critical pairs.

Lemma 3. *Left-linear TRSs $\mathcal{R}$ and $\mathcal{S}$ locally commute if and only if the inclusion*

$$({}_{\mathcal{R}}\!\leftarrow \rtimes \to_{\mathcal{S}}) \cup ({}_{\mathcal{R}}\!\leftarrow \ltimes \to_{\mathcal{S}}) \subseteq \to^{*}_{\mathcal{S}} \cdot {}^{*}_{\mathcal{R}}\!\leftarrow$$

holds. □

Unlike the case of local confluence, left-linearity is essential for the if-direction in Lemma 3 (see e.g. [4]).

Example 4. Consider the TRSs $\mathcal{R}$ and $\mathcal{S}$ over the signature $\{\mathsf{f}^{(2)}, \mathsf{a}^{(0)}, \mathsf{b}^{(0)}, \mathsf{c}^{(0)}\}$:

$$\mathcal{R} = \{\, \mathsf{a} \to \mathsf{b} \,\} \qquad\qquad \mathcal{S} = \{\, \mathsf{f}(x,x) \to \mathsf{c} \,\}$$

We have $({}_{\mathcal{R}}\!\leftarrow \rtimes \to_{\mathcal{S}}) \cup ({}_{\mathcal{R}}\!\leftarrow \ltimes \to_{\mathcal{S}}) = \varnothing \subseteq \to^{*}_{\mathcal{S}} \cdot {}^{*}_{\mathcal{R}}\!\leftarrow$. However, the local peak

$$\mathsf{f}(\mathsf{b},\mathsf{a}) \; {}_{\mathcal{R}}\!\leftarrow \mathsf{f}(\mathsf{a},\mathsf{a}) \to_{\mathcal{S}} \mathsf{c}$$

cannot be joined, due to $\mathsf{f}(\mathsf{b},\mathsf{a}) \in \mathsf{NF}(\mathcal{S})$ and $\mathsf{c} \in \mathsf{NF}(\mathcal{R})$. Hence, local commutation of $\mathcal{R}$ and $\mathcal{S}$ does not hold.

The next results are immediate consequences of Lemma 3.

Theorem 5. *Left-linear TRSs $\mathcal{R}$ and $\mathcal{S}$ over the same signature locally commute if and only if the TRSs $\mathcal{R}$ and $\mathcal{S}$ over $\mathcal{F}_{\mathcal{R}\cup\mathcal{S}}$ locally commute.*

Proof. The if-direction follows from Lemma 3 and the observation that the existence of function symbols not in $\mathcal{F}_{\mathcal{R}\cup\mathcal{S}}$ does not affect to the joinability of critical pairs. The reverse direction is trivial. □

Corollary 6. *Local commutation of left-linear TRSs is preserved under signature extensions.* □

Is local commutation preserved under signature extensions in general? Very recently, Shintani and this author found a counterexample [9, 10].

Example 7 ([9, 10])**.** Consider the TRSs $\mathcal{R}$ and $\mathcal{S}$ over the signature $\mathcal{F} = \{\mathsf{f}^{(2)}, \mathsf{a}^{(0)}, \mathsf{b}^{(0)}\}$:

$$\mathcal{R} = \{\, \mathsf{a} \to \mathsf{b} \,\} \qquad \mathcal{S} = \{\, \mathsf{f}(x,x) \to \mathsf{b},\ \mathsf{f}(\mathsf{b},x) \to \mathsf{b},\ \mathsf{f}(x,\mathsf{b}) \to \mathsf{b} \,\}$$

Since $C[\mathsf{b}] \to_{\mathcal{S}}^{*} \mathsf{b}$ holds for all contexts C, the inclusion

$$\mathcal{R}{\leftarrow} \cdot {\to_{\mathcal{S}}} \subseteq {\to_{\mathcal{S}}^{*}} \cdot \bar{\bar{\mathcal{R}}}{\leftarrow} \tag{1}$$

holds. Therefore, $\mathcal{R}$ and $\mathcal{S}$ locally commute. Now we extend the underlying signature to $\mathcal{F} \cup \{\mathsf{g}^{(1)}\}$. Because $\mathsf{f}(\mathsf{g}(\mathsf{b}), \mathsf{g}(\mathsf{a}))$ and b are normal forms of $\mathcal{S}$ and $\mathcal{R}$ respectively, the local peak $\mathsf{f}(\mathsf{g}(\mathsf{b}), \mathsf{g}(\mathsf{a})) \,{}_{\mathcal{R}}{\leftarrow}\, \mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a})) \to_{\mathcal{S}} \mathsf{b}$ cannot be joined.

4 Commutation

In the last section we witnessed that signature extensions may not preserve local commutation. The next example shows that signature extensions may affect commutation too.

Example 8. Recall the TRSs $\mathcal{R}$ and $\mathcal{S}$ over $\mathcal{F}$ and the inclusion (1) in Example 7. They actually satisfy the strong commutation property. So according to Lemma 1, they are actually commuting TRSs. However, as seen in Example 7, if we extend the signature to $\mathcal{F} \cup \{\mathsf{g}^{(1)}\}$, local commutation no longer holds. Hence, commutation is not preserved.

Commutation of left-linear TRSs is preserved under signature extensions. The remaining part of this section is devoted to proving this claim. Let $\mathcal{R}$ and $\mathcal{S}$ be TRSs over a signature $\mathcal{F}$. To this end, we introduce auxiliary notions.

Definition 9. An *alien pair* is a pair of a non-variable term t over $\mathcal{F}_{\mathcal{R}\cup\mathcal{S}}$ and a substitution σ such that $\mathrm{root}(x\sigma) \in \mathcal{F} \setminus \mathcal{F}_{\mathcal{R}\cup\mathcal{S}}$ and $|t|_x = 1$ for all $x \in \mathcal{D}\mathsf{om}(\sigma)$.

Example 10. Consider the TRSs $\mathcal{R}$ and $\mathcal{S}$ over the extended signature in Example 4 again. While $(\mathsf{f}(x,y), \{x \mapsto \mathsf{g}(\mathsf{a}), y \mapsto \mathsf{g}(\mathsf{a})\})$ is an alien pair, the pairs $(\mathsf{f}(x,x), \{x \mapsto \mathsf{g}(\mathsf{a})\})$ and $(\mathsf{f}(x, \mathsf{g}(\mathsf{a})), \{x \mapsto \mathsf{g}(\mathsf{a})\})$ are not.

Every non-variable term can be split into a unique alien pair (modulo renaming). The next lemma formalizes this uniqueness property.

Lemma 11. *Let (s,σ) and (t,τ) be alien pairs. If $s\sigma = t\tau$ then $s\rho = t$ and $\sigma = \rho\tau$ for some renaming ρ.* □

We introduce a concept akin to the multi-step relation [12, Definition 4.7.11], which is a key for our proof.

Definition 12. The relation $\multimap\!\!\to_{\mathcal{R}}$ is defined as the smallest relation on $\mathcal{T}(\mathcal{F},\mathcal{V})$ that satisfies the following three statements.

1. $x \multimap\!\!\to_{\mathcal{R}} x$ for all variables x.
2. $s\sigma \multimap\!\!\to_{\mathcal{R}} t\tau$ if (s,σ) is an align pair, $s \to_{\mathcal{R}}^{*} t$, and $\sigma \multimap\!\!\to_{\mathcal{R}} \tau$.
3. $f(s_1,\dots,s_n) \multimap\!\!\to_{\mathcal{R}} f(t_1,\dots,t_n)$ if $f \notin \mathcal{F}_{\mathcal{R}\cup\mathcal{S}}$ and $s_i \multimap\!\!\to_{\mathcal{R}} t_i$ for all $1 \leqslant i \leqslant n$.

Here $\sigma \multimap\!\!\to_{\mathcal{R}} \tau$ stands for $x\sigma \multimap\!\!\to_{\mathcal{R}} x\tau$ for all variables x. The relation $\multimap\!\!\to_{\mathcal{S}}$ is similarly defined.

Lemma 13. *The inclusions* $\to_{\mathcal{R}}^{=} \subseteq \multimap\!\!\to_{\mathcal{R}} \subseteq \to_{\mathcal{R}}^{*}$ *hold for every left-linear TRS* $\mathcal{R}$.

Proof. One can verify $= \subseteq \multimap\!\!\to_{\mathcal{R}} \subseteq \to_{\mathcal{R}}^{*}$. We show $\to_{\mathcal{R}} \subseteq \multimap\!\!\to_{\mathcal{R}}$. Let $s \to_{\mathcal{R}} t$ be a rewrite step at a position p by a rule $\ell \to r \in \mathcal{R}$. We perform induction on s. Because s is reducible, s is a non-variable term. If $\mathrm{root}(s) \notin \mathcal{F}_{\mathcal{R}\cup\mathcal{S}}$ then $s \multimap\!\!\to_{\mathcal{R}} t$ follows from the induction hypotheses, $= \subseteq \multimap\!\!\to_{\mathcal{R}}$, and Definition 12(3). Otherwise, there is an alien pair (s',σ) such that $s = s'\sigma$. Since ℓ contains no symbol in $\mathcal{F} \setminus \mathcal{F}_{\mathcal{R}}$, one of the following holds.

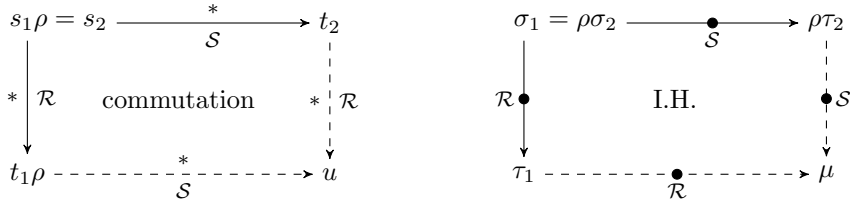
- If $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(s')$ then $s = s'\sigma = (s'[\ell\tau]_p)\sigma \multimap\!\!\to_{\mathcal{R}} (s'[r\tau]_p)\sigma = t$ holds for the substitution $\tau = \{x \mapsto s'|_{pp_x} \mid x \in \mathcal{V}\mathsf{ar}(\ell)\}$. Here p_x stands for a position of a variable x in ℓ. Note that p_x is uniquely determined as ℓ is linear.
- If $p = qq'$ for some positions q, q' with $s'|_q \in \mathcal{V}$ then $s = s'\sigma \multimap\!\!\to_{\mathcal{R}} s'\{s'|_q \mapsto t|_q\} = t$. □

Left-linearity is essential for the first inclusion of Lemma 13. To see it, recall the non-left-linear TRS $\mathcal{S} = \{\mathsf{f}(x,x) \to \mathsf{c}\}$ from Examples 4 and 10. We have $\mathsf{f}(\mathsf{g}(\mathsf{a}),\mathsf{g}(\mathsf{a})) \to_{\mathcal{S}} \mathsf{c}$, but $\mathsf{f}(\mathsf{g}(\mathsf{a}),\mathsf{g}(\mathsf{a})) \multimap\!\!\to_{\mathcal{S}} \mathsf{c}$ does not holds.

Lemma 14. *The relations* $\multimap\!\!\to_{\mathcal{R}}$ *and* $\multimap\!\!\to_{\mathcal{S}}$ *have the commuting diamond property.*

Proof. We perform structural induction on the term of the peak ${}_{\mathcal{R}}\!\leftarrow\!\!\multimap \cdot \multimap\!\!\to_{\mathcal{S}}$.

- If $x \,{}_{\mathcal{R}}\!\leftarrow\!\!\multimap x \multimap\!\!\to_{\mathcal{S}} x$ with $x \in \mathcal{V}$ then trivially $x \multimap\!\!\to_{\mathcal{S}} x \,{}_{\mathcal{R}}\!\leftarrow\!\!\multimap x$.
- If $f(t_1,\dots,t_n) \,{}_{\mathcal{R}}\!\leftarrow\!\!\multimap f(s_1,\dots,s_n) \multimap\!\!\to_{\mathcal{S}} f(u_1,\dots,u_n)$ with $t_i \,{}_{\mathcal{R}}\!\leftarrow\!\!\multimap s_i \multimap\!\!\to_{\mathcal{S}} u_i$ for all $1 \leqslant i \leqslant n$ then the induction hypotheses yield $t_i \multimap\!\!\to_{\mathcal{R}} v_i \,{}_{\mathcal{R}}\!\leftarrow\!\!\multimap u_i$ for all $1 \leqslant i \leqslant n$. Therefore, $f(t_1,\dots,t_n) \multimap\!\!\to_{\mathcal{S}} f(s_1,\dots,s_n) \,{}_{\mathcal{R}}\!\leftarrow\!\!\multimap f(u_1,\dots,u_n)$ holds.
- Suppose $t_1\tau_1 \,{}_{\mathcal{R}}\!\leftarrow\!\!\multimap s_1\sigma_1 = s_2\sigma_2 \multimap\!\!\to_{\mathcal{S}} t_2\tau_2$ with $s_1 \to_{\mathcal{R}}^{*} t_1$, $s_2 \to_{\mathcal{S}}^{*} t_2$, $\sigma_1 \multimap\!\!\to_{\mathcal{R}} \tau_1$, and $\sigma_2 \multimap\!\!\to_{\mathcal{R}} \tau_2$. By Lemma 11 there is a renaming ρ with $s_1\rho = s_2$ and $\sigma_1 = \rho\sigma_2$. The diagrams



  hold for some term u and substitution μ. Because we have $t_1\rho\rho^{-1} \to_{\mathcal{R}}^{*} u\rho^{-1}$ and $\rho^{-1}\rho\tau_2 \multimap\!\!\to_{\mathcal{R}} \rho^{-1}\mu$, the relations $t_1\tau_1 = t_1\rho\rho^{-1}\tau_1 \multimap\!\!\to_{\mathcal{R}} u\rho^{-1}\mu \,{}_{\mathcal{S}}\!\leftarrow\!\!\multimap t_2\rho^{-1}\rho\tau_2 = t_2\tau_2$ are concluded. □

Theorem 15. *Left-linear TRSs $\mathcal{R}$ and $\mathcal{S}$ over the same signature commute if and only if the TRSs $\mathcal{R}$ and $\mathcal{S}$ over $\mathcal{F}_{\mathcal{R}\cup\mathcal{S}}$ commute.*

Proof. Straightforward from Lemmata 2, 13, and 14. □

Corollary 16. *Commutation of left-linear TRSs is preserved under signature extensions.* □

5 Conclusion

In this note we showed that signature extensions may break (local) commutation, and this undesired phenomenon happens only if a non-left-linear rule is present. Future work includes investigations of sufficient conditions for persistency of commutation [1]. Recall the TRSs $\mathcal{R}$ and $\mathcal{S}$ from Example 4 and assume the next many-sorted signature: $\{\mathsf{f}^{\alpha\times\alpha\to\alpha}, \mathsf{a}^{\beta}, \mathsf{b}^{\beta}, \mathsf{c}^{\alpha}\}$. It is easy to see that there is no local peak, so $\mathcal{R}$ and $\mathcal{S}$ commute. This means that commutation is not a persistent property. We conjecture that commutation of left-linear TRSs are persistent.

Acknowledgements. I am grateful to the anonymous reviewers for valuable comments.

References

[1] T. Aoto and Y. Toyama. Persistency of confluence. *Journal of Universal Computer Science*, 3(11):1134–1147, 1997.

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

[4] G. Burel, G. Dowek, and J. Jiang. A completion method to decide reachability in rewrite systems. 2015.

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

[6] N. Hirokawa, A. Middeldorp, and G. Moser. Leftmost outermost revisited. In *Proc. 25th RTA*, Leibniz International Proceedings in Informatics, 2015. To appear.

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

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

[9] K. Shintani. Confluence analysis for term rewriting via commutation. Master's thesis, JAIST, 2015.

[10] K. Shintani and N. Hirokawa. CoLL: A confluence tool for left-linear term rewrite systems. In *Proc. 25th CADE*, LNAI, 2015. To appear.

[11] C. Sternagel and R. Thiemann. Signature extensions preserve termination – an alternative proof via dependency pairs. In *Proc. 19th CSL*, volume 6247 of *LNCS*, pages 514–528, 2010.

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

[13] Y. Toyama. Counterexamples to the termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25(3):141–143, 1987.

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

[15] Y. Toyama. Commutativity of term rewriting systems. In K. Fuchi and L. Kott, editors, *Programming of Future Generation Computers II*, pages 393–407. North-Holland, 1988.

Level-Confluence of 3-CTRSs in Isabelle/HOL*

Christian Sternagel and Thomas Sternagel

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

Abstract

We present an Isabelle/HOL formalization of an earlier result by Suzuki, Middeldorp, and Ida; namely that a certain class of conditional rewrite systems is level-confluent. Our formalization is basically along the lines of the original proof, from which we deviate mostly in the level of detail as well as concerning some basic definitions.

1 Introduction

In the realm of standard term rewriting, many properties of term rewrite systems (TRSs) can be conveniently checked "at the push of a button" due to a wealth of existing automated tools. To maximize the reliability of this approach, such automated tools are progressively complemented by certifiers, that is, verified programs that rigorously ensure that the output of an automated tool for a given input is correct. At the time of writing the prevalent methodology for certifier development consists of the following two phases: First, employ a proof assistant (in our case Isabelle/HOL [5]) in order to formalize the underlying theory, resulting in a *formal library* (in our case IsaFoR,[1] an *Isabelle/HOL Formalization of Rewriting*). Then, verify a program using this library, resulting in the actual certifier (in our case CeTA [9]).

Our ultimate goal is to establish the same state of the art also for conditional term rewrite systems (CTRSs). As a starting point, we present our Isabelle/HOL formalization of the following result:

Lemma 1 (Suzuki et al. [8, Corollary 4.7])**.** *Orthogonal, properly oriented, right-stable 3-CTRSs are level-confluent.* □

Which is actually a corollary of a more general result, whose statement – together with a high-level overview of its proof – we defer until after we have established some necessary preliminaries.

The development we describe in this note is now part of the IsaFoR library and is freely available for inspection at:

http://cl2-informatik.uibk.ac.at/rewriting/mercurial.cgi/IsaFoR/file/a2cd778de34a/IsaFoR/Conditional_Rewriting/Level_Confluence.thy

Throughout the remainder we will from time to time refer to the Isabelle/HOL sources of our development (by active hyperlink).

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

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

2 Preliminaries

We assume familiarity with (conditional) term rewriting [1, 6]. In the sequel we consider oriented 3-CTRSs where extra variables in conditions and right-hand sides of rewrite rules are allowed, i.e., for all rules $\ell \to r \Leftarrow c$ in the CTRS we only demand $\mathcal{V}\mathsf{ar}(r) \subseteq \mathcal{V}\mathsf{ar}(\ell, c)$. For such systems extended TRSs $\mathcal{R}_n$ are inductively defined for each level $n \geqslant 0$ as follows

$$\begin{aligned} \mathcal{R}_0 &= \varnothing \\ \mathcal{R}_{n+1} &= \{\ell\sigma \to r\sigma \mid \ell \to r \Leftarrow c \in \mathcal{R} \text{ and } s\sigma \xrightarrow[\mathcal{R}_n]{*} t\sigma \text{ for all } s \approx t \in c\} \end{aligned}$$

where $\to_{\mathcal{R}_n}$ denotes the standard (unconditional) rewrite relation of the TRS $\mathcal{R}_n$. We write $s \to_{\mathcal{R}} t$ if we have $s \to_{\mathcal{R}_n} t$ for some $n \geqslant 0$. Moreover, for brevity, the latter is written $s \to_n t$ whenever the corresponding CTRS is clear from the context. Given two variable disjoint variants $\ell_1 \to r_1 \Leftarrow c_1$ and $\ell_2 \to r_2 \Leftarrow c_2$ of rules in a CTRS $\mathcal{R}$, a function position p in ℓ_1, and a most general unifier (mgu) μ of $\ell_1|_p$ and ℓ_2; we call the triple $(\ell_1 \to r_1 \Leftarrow c_1, \ell_2 \to r_2 \Leftarrow c_2, p)$ a *conditional overlap* of $\mathcal{R}$. A conditional overlap $(\ell_1 \to r_1 \Leftarrow c_1, \ell_2 \to r_2 \Leftarrow c_2, p)$ with mgu μ is *infeasible* (that is, cannot occur during actual rewriting) if there is no substitution σ such that $s\sigma \to_{\mathcal{R}}^* t\sigma$ for all $s \approx t$ in $c_1\mu, c_2\mu$.

A note on permutations. At the highly formal level of Isabelle/HOL (which we tend to avoid in the following exposition) we employ an existing formalization of *permutation types* (that is, types that contain variables which may be renamed w.r.t. a given permutation) to tackle variable renamings, renaming rules apart, and checking whether two rules are variants of each other. This abstract view on renamings (as opposed to explicit renaming functions on strings) proved to be useful in previous applications [3, 4].

We call a CTRS *almost orthogonal* [2] *(modulo infeasibility)* if it is left-linear and all its conditional overlaps are either infeasible or take place at root position ($\ell_1\mu = \ell_2\mu$) and are either between variants of the same rule or also result in syntactically equal right-hand sides ($r_1\mu = r_2\mu$). A CTRS $\mathcal{R}$ is called *properly oriented* if for all rules $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_k \approx t_k \in \mathcal{R}$ where $\mathcal{V}\mathsf{ar}(r) \not\subseteq \mathcal{V}\mathsf{ar}(\ell)$ and $1 \leqslant i \leqslant k$ we have $\mathcal{V}\mathsf{ar}(s_i) \subseteq \mathcal{V}\mathsf{ar}(\ell, t_1, \ldots, t_{i-1})$. It is called *right-stable* if for every rule $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_k \approx t_k \in \mathcal{R}$ and $1 \leqslant i \leqslant k$ we have $\mathcal{V}\mathsf{ar}(\ell, s_1, \ldots, s_i, t_1, \ldots, t_{i-1}) \cap \mathcal{V}\mathsf{ar}(t_i) = \varnothing$ and t_i is either a linear constructor term or a ground $\mathcal{R}_{\mathsf{u}}$-normal form.

We say that two binary relations $\to_\alpha$ and $\to_\beta$ have the *commuting diamond property* [1], whenever ${}_\alpha\!\leftarrow \cdot \to_\beta \subseteq \to_\beta \cdot {}_\alpha\!\leftarrow$. Moreover, we adopt the notion of *extended parallel rewriting* from Suzuki et al. [8].

Definition 2. Let $\mathcal{R}$ be a CTRS. We say that there is an *extended parallel $\mathcal{R}$-rewrite step at level n* from s to t, written $s \mathrel{\rightarrowtail\!\!\!\!\!\twoheadrightarrow}_{\mathcal{R}_n} t$ (or $s \mathrel{\rightarrowtail\!\!\!\!\!\twoheadrightarrow}_n t$ for brevity), whenever we have a multihole context C, and sequences of terms $s_1, \ldots, s_k$ and $t_1, \ldots, t_k$, such that $s = C[s_1, \ldots, s_k]$, $t = C[t_1, \ldots, t_k]$, and for all $1 \leqslant i \leqslant k$ we have one of $(s_i, t_i) \in \mathcal{R}_n$ (that is, a root-step at level n) and $s_i \to_{n-1}^* t_i$.

Suzuki et al. [8], state this definition slightly differently, that is, instead of multihole contexts they try to rely exclusively on sets of positions:

> *We write $s \mathrel{\rightarrowtail\!\!\!\!\!\twoheadrightarrow}_n t$ if there exists a subset P of pairwise disjoint positions in s such that for all $p \in P$ either $(s|_p, t|_p) \in \mathcal{R}_n$ or $s|_p \to_{n-1}^* t|_p$.*

While it is quite clear what is meant, a slight problem (at least for a formal development inside a proof assistant) is the fact that this definition does not enforce t to be exactly the same as s

outside of the positions in P, that is, it does not require the multihole context around the $|P|$ rewrite sequences to stay the same. In order to express this properly, it seems unavoidable to employ multihole contexts (or something equivalent).

In the remainder we employ the convention that the number of holes of a multihole context, is denoted by the corresponding lower-case letter, e.g., c for a multihole context C, d for D, e for E, etc.

3 The Main Result

As remarked in the last two sections of Suzuki et al. [8], we actually consider *almost orthogonal* systems modulo *infeasibility*. We are now in a position to state the main theorem.

Theorem 3 (Suzuki et al. [8, Theorem 4.6])**.** *Let $\mathcal{R}$ be an almost orthogonal (modulo infeasibility), properly oriented, right-stable 3-CTRS. Then, for any two levels m and n, the extended parallel rewrite relations $\twoheadrightarrow_m$ and $\twoheadrightarrow_n$, have the commuting diamond property.*

As a special case of the above theorem, we obtain that for a fixed level n, the relation $\twoheadrightarrow_n$ has the diamond property. Moreover, it is well known that whenever a relation S with the diamond property, is between a relation R and its reflexive, transitive closure (that is, $R \subseteq S \subseteq R^*$), then R is confluent. Taken together, this yields level-confluence of $\to_{\mathcal{R}}$, since clearly $\to_n \subseteq \twoheadrightarrow_n \subseteq \to_n^*$.

We now give a high-level overview of the proof of Theorem 3. The general structure is similar to the one followed by Suzuki et al. [8], only that we employ multihole contexts instead of sets of positions. Therefore, we do not give all the details (if you are interested, see `Conditional_Rewriting/Level_Confluence`, starting from `comm_epar_n` in line 1499), but mostly comment on the parts that differ (if only slightly).

Proof (Sketch) of Theorem 3. We proceed by complete induction on $m + n$. By induction hypothesis (IH) we may assume the result for all $m' + n' < m + n$. Now consider the peak $t \;{}_m\!\twoheadleftarrow\; s \twoheadrightarrow_n u$. If any of m and n equals 0, we are done (since $\twoheadrightarrow_0$ is the identity relation). Thus we may assume $m = m' + 1$ and $n = n' + 1$ for some m' and n'. By the definition of extended parallel rewriting, we obtain multihole contexts C and D, and sequences of terms $s_1, \ldots, s_c$, $t_1, \ldots, t_c$, $u_1, \ldots, u_d$, $v_1, \ldots, v_d$, such that $s = C[s_1, \ldots, s_c]$ and $t = C[t_1, \ldots, t_c]$, as well as $s = D[u_1, \ldots, u_d]$ and $u = D[v_1, \ldots, v_d]$; and $(s_i, t_i) \in \mathcal{R}_m$ or $s_i \to_{m'}^* t_i$ for all $1 \leqslant i \leqslant c$, as well as $(u_i, v_i) \in \mathcal{R}_n$ or $u_i \to_{n'}^* v_i$ for all $1 \leqslant i \leqslant d$.

Now we identify the common part E of C and D, employing the semi-lattice properties of multihole contexts (see `Rewriting/Multihole_Context`), that is, $E = C \sqcap D$. Then $C = E[C_1, \ldots, C_e]$ and $D = E[D_1, \ldots, D_e]$ for some multihole contexts $C_1, \ldots, C_e$ and $D_1, \ldots, D_e$ such that for each $1 \leqslant i \leqslant e$ we have $C_i = \Box$ or $D_i = \Box$. This also means that there is a sequence of terms $s'_1, \ldots, s'_e$ such that $s = E[s'_1, \ldots, s'_e]$ and for all $1 \leqslant i \leqslant e$, we have $s'_i = C_i[s_{k_i}, \ldots, s_{k_i + c_i - 1}]$ for some subsequence $s_{k_i}, \ldots, s_{k_i + c_i - 1}$ of $s_1, \ldots, s_c$ (we denote similar terms for t, u, and v by t'_i, u'_i, and v'_i, respectively). Moreover, note that by construction $s'_i = u'_i$ for all $1 \leqslant i \leqslant e$. Since extended parallel rewriting is closed under multihole contexts (see `epar_n_mctxt`), it suffices to show that for each $1 \leqslant i \leqslant e$ there is a term v such that $t'_i \twoheadrightarrow_n v \;{}_m\!\twoheadleftarrow\; v'_i$, in order to conclude the proof. We concentrate on the case that $C_i = \Box$ (the case $D_i = \Box$ is completely symmetric). Moreover, note that when we have $s'_i \to_{m'}^* t'_i$, the proof concludes by IH (together with some basic properties of the involved relations), and thus we remain with $(s'_i, t'_i) \in \mathcal{R}_m$. At this point we distinguish the following cases:

1. $(D_i = \Box)$. Also here, the non-root case $u'_i \to^*_{n'} v'_i$ is covered by the IH. Thus, we may restrict to $(u'_i, v'_i) \in \mathcal{R}_n$, giving rise to a root overlap. Since $\mathcal{R}$ is almost orthogonal (modulo infeasibility), this means that either the resulting conditions are not satisfiable or the resulting terms are the same (in both of these cases we are done) or two variable disjoint variants of the same rule $\ell \to r \Leftarrow c$ were involved. Without extra variables in r, this is the end of the story; but since we also want to cover the case where $\mathcal{V}\mathsf{ar}(r) \not\subseteq \mathcal{V}\mathsf{ar}(l)$, we have to reason why this does not cause any trouble. This case is finished by a technical lemma (see `trs_n_peak`) that shows, by induction on the number of conditions in c, that we can join the two respective instances of the right-hand side r by extended parallel rewriting. (This is also where proper orientedness and right-stability of $\mathcal{R}$ is first used, that is, were we to relax this properties, we had to adapt the technical lemma.)

2. $(D_i \neq \Box)$. Then for some $1 \leqslant k \leqslant d$, we have $(u_j, v_j) \in \mathcal{R}_n$ or $u_j \to^*_{n'} v_j$ for all $k \leqslant j \leqslant k + d_i - 1$, that is, an extended parallel rewrite step of level n from $s'_i = u'_i = D_i[u_{k_i}, \ldots, u_{k_i+d_i-1}]$ to $D_i[v_{k_i}, \ldots, v_{k_i+d_i-1}] = v'_i$. Since $\mathcal{R}$ is almost orthogonal (modulo infeasibility) and, by $D_i \neq \Box$, root overlaps are excluded, the constituent parts of the extended parallel step from s'_i to v'_i take place exclusively inside the substitution of the root-step to the left (which is somewhat obvious – as also stated by Suzuki et al. [8] – but surprisingly hard to formalize, see `epar_n_varpeak'`, even more so when having to deal with infeasibility). We again close this case by induction on the number of conditions making use of proper orientedness and right-stability of $\mathcal{R}$, see `epar_n_varpeak` for details. □

4 Conclusions and Future Work

In the original paper [8] the proof of Theorem 3 begins after only three definitions (proper orientedness, right-stability, and extended parallel rewriting) and stretches across two pages, including two figures.

By contrast, in our formalization we need 8 definitions and 42 lemmas (mainly stating properties of extended parallel rewriting) before we can start with the main proof. Furthermore, we need two auxiliary technical lemmas to cover the induction proofs on the number of conditions which are nested inside the main case analysis. All in all, resulting in a theory file of about 1500 lines. This yields a *de Bruijn factor* of approximately 18, that is, for every line in the original "paper proof," our formal proof development contains 18 lines of Isar (the formal language of Isabelle/HOL).

In the latest version of our formalization we further relaxed the condition for conditional overlaps to be infeasible (making the result applicable to a larger class of systems) and proved that the main result still holds. More concretely, a conditional overlap $(\ell_1 \to r_1 \Leftarrow c_1, \ell_2 \to r_2 \Leftarrow c_2, p)$ with mgu μ is infeasible iff

$$\forall m\, n.\ \xleftarrow[m]{*} \cdot \xrightarrow[n]{*} \subseteq \xrightarrow[n]{*} \cdot \xleftarrow[m]{*} \implies \nexists \sigma.\ (\forall s \approx t \in c_1\mu.\ s\sigma \xrightarrow[m]{*} t\sigma) \wedge (\forall s \approx t \in c_2\mu.\ s\sigma \xrightarrow[n]{*} t\sigma).$$

That is, we may assume "level-commutation" (which is called shallow-confluence in the literature) when showing that the combined conditions of two rules are not satisfiable. This may be helpful, since it allows us to turn diverging sequences (as would for example result from two conditions with identical left-hand sides) into joining sequences.

Future Work. The ultimate goal of this formalization is of course to certify level-confluence proofs of conditional confluence tools, e.g. ConCon [7]. To this end we need executable check

functions for the syntactic properties a CTRS has to meet in order to apply the theorem. The check functions for proper orientedness as well as right-stability should be straightforward to implement. For orthogonality, however, there is a small obstacle to overcome. On the one hand, in our formalization we use the abstract notion of *permutation types* inside the definition of conditional critical pairs, only demanding that the set of variables is infinite. While this guarantees that we can always rename two finite sets of variables apart, we do not directly have an executable renaming function at our disposal. On the other hand, in the current version of IsaFoR the type of variables in (standard) critical pairs is fixed to strings, and their definition employs a concrete, executable renaming function. Therefore, it remains to establish a suitable connection between the executable implementation using strings and the abstract definition: for each critical pair in the abstract definition, there is some variant that we obtain by the executable implementation.

Moreover, Suzuki et al. [8] additionally remark (without proof) that the proof of Theorem 3 could easily be adapted to extended proper orientedness. To us, it is not immediately clear how to adapt our formalization. For the time being, we leave this enhancement as future work.

References

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

[2] Michael Hanus. On extra variables in (equational) logic programming. In *Proceedings of the 12th International Conference on Logic Programming*, pages 665–679. MIT Press, 1995.

[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] Nao Hirokawa, Aart Middeldorp, and Christian Sternagel. Normalization equivalence of rewrite systems. In *Proceedings of the 3rd International Workshop on Confluence*, 2014.

[5] 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.

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

[7] 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.

[8] Taro Suzuki, Aart Middeldorp, and Tetsuo Ida. Level-confluence of conditional rewrite systems with extra variables in right-hand sides. In *Proceedings of the 6th International Conference on Rewriting Techniques and Applications*, volume 914 of *Lecture Notes in Computer Science*, pages 179–193. Springer, 1995. doi:10.1007/3-540-59200-8_56.

[9] 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.

Labeling Multi-Steps for Confluence of Left-Linear Term Rewrite Systems*

Bertram Felgenhauer[1]

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

Abstract

We show how to use the commutation version of van Oostrom's decreasing diagrams for labeling left-linear term rewriting systems, based on Zankl et al.'s labeling framework. The resulting confluence criterion requires joining simultaneous critical pairs decreasingly, subsuming the criterion by Okui.

1 Introduction

This note is concerned with the confluence of term rewrite systems. Okui introduced simultaneous critical pairs in [3], which are critical pairs between a multi-step (to the left) and a single rewrite step (to the right). He showed that any left-linear term rewrite system (TRS) is confluent if all its simultaneous critical pairs are joinable using a rewrite sequence from the left and a single multi-step from the right. In this note we show how to combine this idea with van Oostrom's decreasing diagrams [4] by labeling the rewrite steps in a suitable way. We base our work on [6], where such ideas have already been used for single rewrite steps and critical pairs as well as parallel rewrite steps and parallel critical pairs.

This work is motivated by the fact that the parallel critical pair criterion of [6] comes with an awkward restriction on the variables involved in the parallel step in the joining peak, and furthermore by the hope that the criterion will become applicable to higher order rewrite systems.

2 Preliminaries

We assume familiarity with term rewriting. For an introduction see [1].

Redex patterns. Let $\mathcal{R}$ be a left-linear TRS. A *redex pattern* is a pair $\pi = \langle p_\pi, l_\pi \to r_\pi \rangle$ consisting of a position p_π and a rewrite rule $l_\pi \to r_\pi \in \mathcal{R}$. A redex pattern π *matches* a term t if $t|_{p_\pi}$ is an instance of l_π. If π matches t, then π and t uniquely determine a term t^π such that $t \to_{p_\pi, l_\pi \to r_\pi} t^\pi$. We denote this rewrite step as $t \to^\pi t^\pi$. For a position q, $q\pi$ denotes the redex pattern $\langle qp_\pi, l_\pi \to r_\pi \rangle$. Let π_1 and π_2 be redex patterns that match a common term. They are called *parallel* ($\pi_1 \parallel \pi_2$) if $p_{\pi_1} \parallel p_{\pi_2}$. If $p_{\pi_2} \leqslant p_{\pi_1}$ and $p_{\pi_1} \backslash p_{\pi_2} \in \mathcal{P}os_{\mathcal{F}}(l_{\pi_2})$ or $p_{\pi_1} \leqslant p_{\pi_2}$ and $p_{\pi_1} \backslash p_{\pi_1} \in \mathcal{P}os_{\mathcal{F}}(l_{\pi_1})$ then π_1 and π_2 *overlap*, otherwise they are *orthogonal* ($\pi_1 \perp \pi_2$). Note that $\pi_1 \parallel \pi_2$ implies $\pi_1 \perp \pi_2$. We write $P \perp Q$ if $\pi \perp \pi'$ for all $\pi \in P$ and $\pi' \in Q$ and similarly $P \parallel Q$ if $\pi \parallel \pi'$ for all $\pi \in P$ and $\pi' \in Q$. We say that a set of redex patterns is *compatible* if P is a set of pairwise orthogonal redex patterns and there is a term t such that all $\pi \in P$ match t. Given a compatible set of redex patterns P matching a term t there is a multi-step $t \multimap^P t^P$.

*Supported by the Austrian Science Fund (FWF) project P27528.

Residuals. We refer to [5] for a formal treatment of residuals. Recall the parallel moves lemma: If $P \cup Q$ is a compatible set of redex patterns and we have co-initial multi-steps $t \mathrel{-\!\!\!\circledcirc\!\!\!\to}^P t^P$, $t \mathrel{-\!\!\!\circledcirc\!\!\!\to}^Q t^Q$, then there are multi-steps $t^P \mathrel{-\!\!\!\circledcirc\!\!\!\to}^{Q/P} t^{P\cup Q}$ and $t^Q \mathrel{-\!\!\!\circledcirc\!\!\!\to}^{P/Q} t^{P\cup Q}$, where P/Q is the set of residuals of P after Q. Another important property of residuals for left-linear term rewrite systems is that residuals of orthogonal redex patterns remain orthogonal: If $P \cup Q \cup R$ is a compatible set of redex patterns and Q and R are disjoint (which implies that $Q \perp R$), then $Q/P \perp R/P$.

Simultaneous Critical Pairs. Let $\mathcal{R}$ be a left-linear TRS, π be a redex pattern and P be a non-empty set of pairwise orthogonal redex patterns that overlap with π. By choosing variants of rules in $\mathcal{R}$, we can ensure that l_π and $l_{\pi'}$ ($\pi' \in P$) have no variables in common. Furthermore assume that $\epsilon = p_\pi$ or $\epsilon \in \{p_{\pi'} \mid \pi' \in P\}$. Let π_ϵ be one of these redex patterns at the root position. We set up a unification problem as follows. Let $p \dot{-} q = p \setminus q$ if $p \geqslant q$ and $p \dot{-} q = \epsilon$ if $p < q$. For each $\pi' \in P$ we consider the equation $l_\pi|_{p_{\pi'} \dot{-} p_\pi}\sigma =^? l_{\pi'}|_{p_\pi \dot{-} p_{\pi'}}\sigma$. If the unification problem consisting of all these equations has an mgu σ, then there is a peak $t \mathrel{{}^P\!\!\leftarrow\!\!\circledcirc} l_{\pi_\epsilon}\sigma \to^{\pi \dot{-} \pi_\epsilon} u$, and we call $t \leftarrow\!\!\circledcirc\!\!\rtimes\!\!\to u$ a *simultaneous critical pair*.

Lemma 1. *If $t \mathrel{{}^P\!\!\leftarrow\!\!\circledcirc} s \to^\pi u$ then $P \perp \pi$ or there are a simultaneous critical pair $t' \mathrel{{}^Q\!\!\leftarrow\!\!\circledcirc\!\!\rtimes\!\!\to} u'$, a context C with hole at position p and a substitution σ such that $t \mathrel{{}^{P/pQ}\!\!\leftarrow\!\!\circledcirc} C[t'\sigma] \mathrel{{}^{pQ}\!\!\leftarrow\!\!\circledcirc} s = C[s'\sigma] \to C[u'\sigma] = u$, where $pQ = \{p\pi \mid \pi \in Q\}$.*

Finally we recall van Oostrom's decreasing diagrams [4]. We will use a commutation version of extended decreasingness (cf. [2]). Let L be a set of labels equipped with a well-founded order $>$ and a compatible quasi-order $\geqslant$ (i.e., $\geqslant \cdot > \subseteq >$.)

Theorem 2. *Let $(\rightharpoonup_\alpha)_{\alpha\in L}$ and $(\rightharpoondown_\alpha)_{\alpha\in L}$ be labeled ARSs. Then $\rightharpoonup$ and $\rightharpoondown$ commute if for all $\alpha, \beta \in L$,*

$$\underset{\alpha}{\leftharpoondown} \cdot \underset{\beta}{\rightharpoondown} \subseteq \underset{\forall\alpha}{\overset{*}{\longleftarrow}} \cdot \underset{\forall\beta}{\overset{=}{\rightharpoondown}} \cdot \underset{\forall\alpha\beta}{\overset{*}{\longleftarrow}} \cdot \underset{\forall\alpha}{\overset{=}{\leftharpoondown}} \cdot \underset{\forall\beta}{\overset{*}{\longleftarrow}}$$

Sets of labels are ordered by the Hoare preorder of $(\geqslant, >)$, which we denote by $(\geqslant_H, >_H)$ and is defined by

$$\begin{aligned} \Gamma >_H \Delta &\iff \Gamma \neq \varnothing \wedge \forall \beta \in \Delta.\, \exists\, \alpha \in \Gamma.\, \alpha > \beta \\ \Gamma \geqslant_H \Delta &\iff \forall \beta \in \Delta.\, \exists\, \alpha \in \Gamma.\, \alpha \geqslant \beta \end{aligned}$$

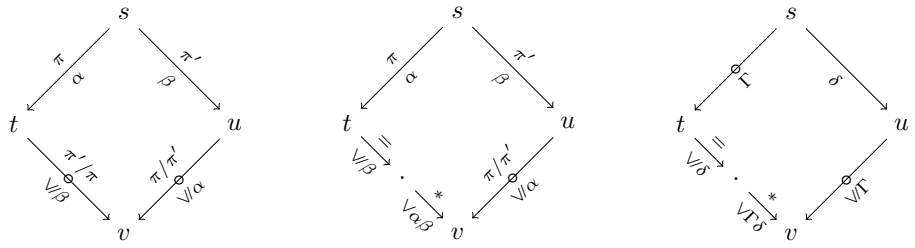
If $(\geqslant, >)$ is a pair of a well-founded order $>$ and a compatible quasi-order $\geqslant$, then so is $(\geqslant_H, >_H)$. For readability we drop the subscript H when attaching labels to rewrite steps as in $\mathrel{-\!\!\!\circledcirc\!\!\!\to}_{\forall\Gamma}$.

3 Labeling Development Steps

In this section we show that the weak LL-labelings of [6] can be fruitfully applied to development steps, leading to a criterion based on simultaneous critical pairs [3]. Our result subsumes Okui's main result [3]. The key idea for establishing confluence in [3] is to show that $\leftarrow\!\!\circledcirc$ and $\to$ commute. We do the same, using the commutation version of extended decreasing diagrams (Theorem 2).

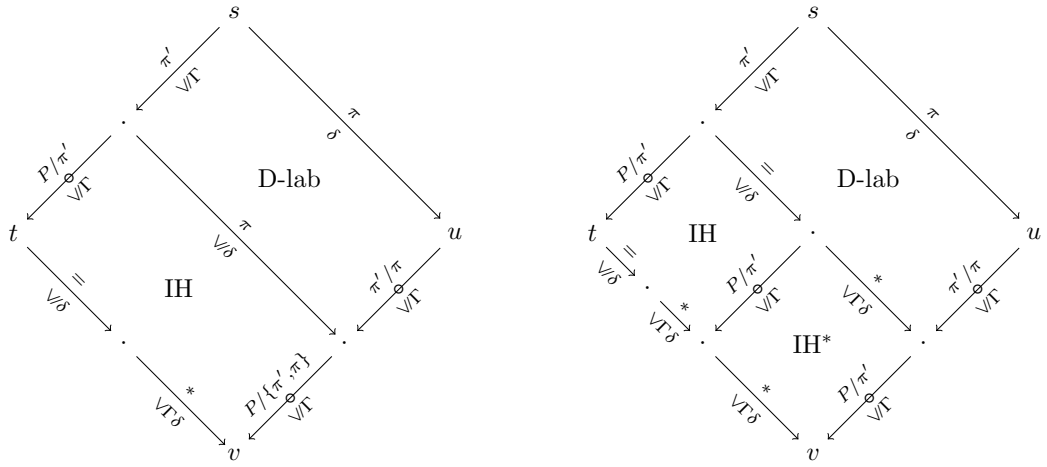
Definition 3. Let L be a set of labels equipped with a well-founded order $>$ and compatible quasi-order $\geqslant$. A function ℓ that maps rewrite steps $t \to^\pi t^\pi$ is a labeling function if for all contexts C with hole position p and substitutions σ,

1. $\ell(t \to^\pi t^\pi) > \ell(u \to^{\pi'} u^{\pi'})$ implies $\ell(C[t\sigma] \to^{p\pi} C[t^\pi\sigma]) > \ell(C[u\sigma] \to^{p\pi'} C[u^{\pi'}\sigma])$, and



(a) weak LL-labeling (b) D-labeling (c) Lemma 6

Figure 1: Labelings



(a) $p_{\pi'} \not< p_\pi$ (b) $p_{\pi'} < p_\pi$

Figure 2: Proof of Lemma 6

2. $\ell(t \to^\pi t^\pi) \geqslant \ell(u \to^{\pi'} u^{\pi'})$ implies $\ell(C[t\sigma] \to^{p\pi} C[t^\pi\sigma]) \geqslant \ell(C[u\sigma] \to^{p\pi'} C[u^{\pi'}\sigma])$.

We need to label development steps. We do so in essentially the same way as we labeled parallel steps in [6], i.e., we collect the labels of the constituent rewrite steps in a set.

Definition 4. Consider a development step $t \mathrel{-\!\!\!\circledcirc}^P t'$. For each $\pi \in P$, there is a rewrite step $t \to_\pi t^\pi$. We label $t \mathrel{-\!\!\!\circledcirc}^P t'$ by $\ell^\circ(t \mathrel{-\!\!\!\circledcirc}^P t')$, where

$$\ell^\circ(t \overset{P}{\mathrel{-\!\!\!\circledcirc}} t') = \{\ell(t \to^\pi t^\pi) \mid \pi \in P\}$$

This means that a development rewrite step is labeled by the set of the labels of the constituent steps. We indicate labels along with the step, writing $t \mathrel{-\!\!\!\circledcirc}_\Gamma t'$ if $\Gamma = \ell^\circ(t \mathrel{-\!\!\!\circledcirc} t')$.

Definition 5. A labeling ℓ is a *weak LL-labeling* if any orthogonal peak $t \,{}_\alpha\!\leftarrow s \to_\beta u$ is joined as in Figure 1(a), where we $\curlyvee\!/\gamma$ stands for $\curlyvee\!/\{\gamma\}$.

A pair of weak LL-labelings $\langle \ell', \ell \rangle$ is a *D-labeling* if any orthogonal peak $t \,{}_\alpha\!\leftarrow s \to_\beta u$ can be joined as in Figure 1(b), where leftward steps are labeled using ℓ' and rightward steps are labeled using ℓ, and the $t \to^{=}_{\curlyvee\!/\beta} \cdot \to^*_{\curlyvee\alpha\beta} v$ sequence is a complete development of $t \mathrel{-\!\!\!\circledcirc}_{\pi'/\pi} v$.

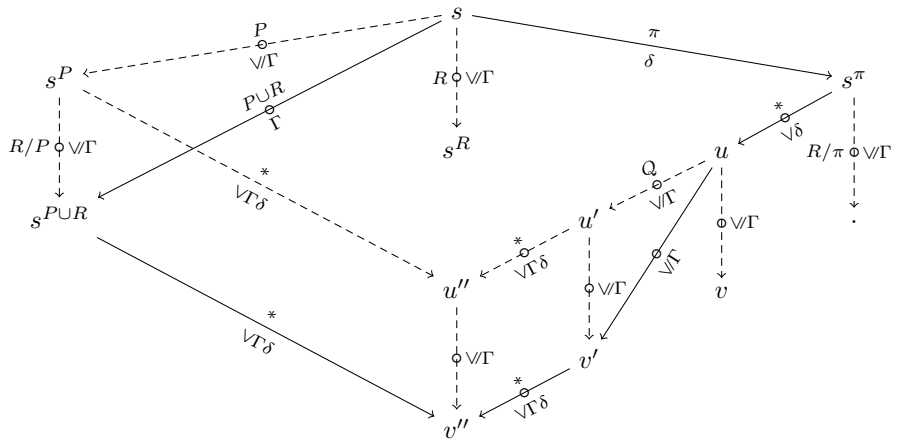


Figure 3: Proof of Theorem 7.

The D-labeling property ensures that orthogonal peaks can be joined decreasingly:

Lemma 6. *If $P \perp \pi$ and $t \overset{P}{\underset{\Gamma}{\leftarrow\!\circ\!\rightarrow}} s \to^{\pi}_{\delta} u$ then $t \to^{=}_{\vee\!\!/\delta} \cdot \to^{*}_{\vee\Gamma\delta} \cdot {}_{\vee\!\!/\Gamma}\!\leftarrow\!\circ\!\rightarrow \cdot u$. See also Figure 1(c).*

Proof. The proof is by induction on $\#P$. If P is empty, then there is nothing to prove; otherwise, let $\pi' \in P$ be an innermost redex pattern in P. The remainder of the proof is sketched in Figure 2. Note that if π/π' is not a singleton set then $p_{\pi'} < p_\pi$ and therefore by the choice of π', $P/\pi = P$. Therefore all applications of the induction hypothesis in the $\cdot {}_{\vee\!\!/\Gamma}\!\leftarrow\!\circ\!\rightarrow \cdot \to^{=}_{\vee\!\!/\delta} \cdot$ and $\cdot {}_{\vee\!\!/\Gamma}\!\leftarrow\!\circ\!\rightarrow \cdot \to^{*}_{\vee\Gamma\delta} \cdot$ peaks use the same set $P/\pi' = P \setminus \{\pi'\}$ (because π' is an innermost redex pattern of P) which satisifies $\#(P/\pi') < \#P$. □

Theorem 7. *Let $\langle \ell', \ell \rangle$ be a D-labeling for a left-linear TRS $\mathcal{R}$. Then $\mathcal{R}$ is confluent if every simultaneous critical pair $t \;{}_{\Gamma}\!\leftarrow\!\circ\!\rightarrow\; s \to_\delta u$ can be joined decreasingly as*

$$t \xrightarrow[\vee\Gamma\delta]{*} \cdot \underset{\vee\Gamma\delta}{\overset{*}{\leftarrow\!\circ\!\rightarrow}} \cdot \underset{\vee\!\!/\Gamma}{\leftarrow\!\circ\!\rightarrow} \cdot \underset{\vee\delta}{\overset{*}{\leftarrow\!\circ\!\rightarrow}} u$$

Proof. Consider a peak $t \overset{P_0}{\underset{\Gamma}{\leftarrow\!\circ\!\rightarrow}} s \to^{\pi}_{\delta} u$. We claim that this peak can be joined decreasingly. If $P_0 \perp \pi$ then the claim follows from Lemma 6. Otherwise, we can apply Lemma 1 to obtain a simultaneous critical pair $t' \overset{P'_0}{\leftarrow\!\circ\!\rightarrow} s' \to^{\pi'} u'$, a context C with hole at position p, and a substitution σ such that with $P = pP'_0$ and $R = P_0 - P$, $P \perp R$, $\pi = p\pi'$, and

$$s^P = C[t'\sigma] \underset{\vee\!\!/\Gamma}{\overset{P}{\leftarrow\!\circ\!\rightarrow}} s = C[s'\sigma] \xrightarrow[\delta]{\pi} s^\pi = C[u'\sigma] \tag{1}$$

By the assumption and the definition of weak LL-labelings, there are terms u, u', u'' such that

$$s^P \xrightarrow[\vee\Gamma\delta]{*} u'' \underset{\vee\Gamma\delta}{\overset{*}{\leftarrow}} u' \underset{\vee\!\!/\Gamma}{\overset{Q}{\leftarrow\!\circ\!\rightarrow}} u \underset{\vee\delta}{\overset{\pi_n}{\leftarrow}} \cdots \underset{\vee\delta}{\overset{\pi_1}{\leftarrow}} s^\pi$$

where $\pi_1, \ldots, \pi_n$ is a sequence of redex patterns. Furthermore the step $s \overset{R}{\underset{\vee\!\!/\Gamma}{\leftarrow\!\circ\!\rightarrow}} s^R$ has residuals as shown in Figure 3: $s^P \overset{R/P}{\underset{\vee\!\!/\Gamma}{\leftarrow\!\circ\!\rightarrow}} s^{P\cup R} = t$, $u \overset{R/(\pi;\overline{\pi_i})}{\underset{\vee\!\!/\Gamma}{\leftarrow\!\circ\!\rightarrow}} v$, $u' \;{}_{\vee\!\!/\Gamma}\!\leftarrow\!\circ\!\rightarrow\; v'$ and $u'' \;{}_{\vee\!\!/\Gamma}\!\leftarrow\!\circ\!\rightarrow\; v''$. Since R is orthogonal to the whole instance of the simultaneous critical pair (1), its residual $u \leftarrow\!\circ\!\rightarrow v$ is orthogonal to Q, and therefore we obtain a single development step $u \overset{Q\cup R/(\pi;\pi_1;\ldots;\pi_n)}{\underset{\vee\!\!/\Gamma}{\leftarrow\!\circ\!\rightarrow}} v'$, completing the decreasing diagram. □

Corollary 8 (Okui's confluence criterion)**.** *If all simultaneous critical pairs of a left-linear TRS $t \mathrel{\multimap} s \to u$ are joinable as $t \to^* v \mathrel{\multimap} u$ then $\mathcal{R}$ is confluent.*

Proof. Using $L = \{\bot, \top\}$ with $\bot < \top$, and the D-labeling $\langle \ell', \ell \rangle$ defined by $\ell'(\cdot) = \top$ and $\ell(\cdot) = \bot$, we see that the requirements of Theorem 7 are fulfilled. Therefore, confluence of $\mathcal{R}$ follows. □

Example 9. The TRS consisiting of the following rules demonstrates that Theorem 7 strictly subsumes Okui's criterion.

$$1\colon \mathsf{g}(\mathsf{b}, x) \to \mathsf{f}(x, x) \qquad 2\colon \mathsf{c} \to \mathsf{a} \qquad 3\colon \mathsf{c} \to \mathsf{b} \qquad 4\colon \mathsf{a} \to \mathsf{b} \qquad 5\colon \mathsf{f}(\mathsf{a}, \mathsf{a}) \to \mathsf{g}(\mathsf{c}, \mathsf{c})$$

We let $\ell'(s \to^\pi s^\pi)$ be the index of the used rule $l_\pi \to r_\pi$ and $\ell(\cdot) = 0$; this results in a D-labeling with the standard order on natural numbers. There are 5 simultaneous critical pairs, $\{\mathsf{f}(\mathsf{a},\mathsf{b}), \mathsf{f}(\mathsf{b},\mathsf{a}), \mathsf{f}(\mathsf{b},\mathsf{b})\} \mathrel{\multimap\!\!\!\!\to} \mathsf{g}(\mathsf{c},\mathsf{c})$ and $\mathsf{g}(\mathsf{c},\mathsf{c}) \mathrel{\multimap\!\!\!\!\to} \{\mathsf{f}(\mathsf{a},\mathsf{b}), \mathsf{f}(\mathsf{b},\mathsf{a})\}$. They are joinable with steps below $\ell'(\mathsf{a} \to \mathsf{b}) = 4$, because $\mathsf{g}(\mathsf{c},\mathsf{c}) \mathrel{\multimap}_{\not\vee 3} \mathsf{g}(\mathsf{b},\mathsf{c}) \mathrel{\multimap}_{\not\vee 3} \mathsf{f}(\mathsf{c},\mathsf{c}) \mathrel{\multimap}_{\not\vee 3} \{\mathsf{f}(\mathsf{a},\mathsf{b}), \mathsf{f}(\mathsf{b},\mathsf{a}), \mathsf{f}(\mathsf{b},\mathsf{b})\}$ (and we can use the same rewrite sequence to the left, with all labels equal to 0). Note that a single development step does not suffice, so Okui's criterion fails.

4 Conclusion

We have derived a new application of decreasing diagrams to left-linear term rewrite systems, based on the commutation of single steps and development steps, and simultaneous critical pairs. Our criterion subsumes Okui's criterion. It should be noted that commutation is essential for obtaining a finite criterion: If one were to consider peaks composed of two development steps, one would end up with an infinite set of critical peaks in general. For example, the single rule TRS $\{\mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(x)\}$ has *critical* overlaps of arbitrary size, e.g., $\mathsf{f}^{n+1}(x) \mathrel{\multimap} \mathsf{f}^{2n+1}(x) \mathrel{\multimap} \mathsf{f}^{n+1}(x)$, where the left multi-step has redexes at positions of even length and the right multi-step has redexes at positions of odd length.

As future work, we plan to implement this criterion in CSI. We also hope to apply the criterion to higher-order systems, in particular pattern rewrite systems. In order to do so, we need to generalise simultaneous critical pairs to that setting.

References

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

[2] N. Hirokawa and A. Middeldorp. Decreasing diagrams and relative termination. In *Proc. 5th International Joint Conference on Automated Reasoning*, volume 6173 of *Lecture Notes in Artificial Intelligence*, pages 487–501, 2010.

[3] S. Okui. Simultaneous critical pairs and Church-Rosser property. In *Proc. 9th International Conference on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 2–16, 1998.

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

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

[6] H. Zankl, B. Felgenhauer, and A. Middeldorp. Labelings for decreasing diagrams. *Journal of Automated Reasoning*, 54(2):101–133, 2015.

ACP: System Description for CoCo 2015

Takahito Aoto[1] and Yoshihito Toyama[1]

RIEC, Tohoku University Sendai, Miyagi, Japan
{ aoto, 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. 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 2014.

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

Kouta Onozawa, Kentaro Kikuchi, Takahito Aoto and Yoshihito Toyama

RIEC, Tohoku University, Sendai, Japan
{ onozawa, kentaro, aoto, toyama }@nue.riec.tohoku.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>
```

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 2015

Takahito Aoto and Yoshihito Toyama

RIEC, Tohoku University, Sendai, Japan
{ aoto, 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) is a tool for proving ground confluence of many-sorted term rewriting systems. AGCP is written in Standard ML of New Jersey (SML/NJ). Several codes are incorporated from confluence prover ACP [4] and an inductive theorem prover developed in [1]. 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 2015.

AGCP proves ground confluence of many-sorted term rewriting systems based on two ingredients. One ingredient is to divide the ground confluence problem of a many-sorted term rewriting system $\mathcal{R}$ into that of $\mathcal{S} \subseteq \mathcal{R}$ and the inductive validity problem of equations $u \approx v$ w.r.t. $\mathcal{S}$ for each $u \to r \in \mathcal{R} \setminus \mathcal{S}$. Here, an equation $u \approx v$ is inductively valid w.r.t. $\mathcal{S}$ if all its ground instances $u\sigma \approx v\sigma$ is valid w.r.t. $\mathcal{S}$, i.e. $u\sigma \overset{*}{\leftrightarrow}_{\mathcal{S}} v\sigma$. Another ingredient is to prove ground confluence of a many-sorted term rewriting system via the *bounded ground convertibility* of the critical pairs. Here, an equation $u \approx v$ is said to be bounded ground convertibile w.r.t. a quasi-order $\succsim$ if $u\theta_g \underset{\succsim}{\overset{*}{\longleftrightarrow}}_{\mathcal{R}} v\theta_g$ for any its ground instance $u\sigma_g \approx v\sigma_g$, where $x \underset{\succsim}{\overset{*}{\longleftrightarrow}} y$ iff there exists $x = x_0 \leftrightarrow \cdots \leftrightarrow x_n = y$ such that $x \succsim x_i$ or $y \succsim x_i$ for every x_i.

Rewriting induction [5] is a well-known method for proving inductive validity of many-sorted term rewriting systems. In [3], an extension of rewriting induction to prove bounded ground convertibility of the equations has been reported. Namely, for a reduction quasi-order $\succsim$ and a quasi-reducible many-sorted term rewriting system $\mathcal{R}$ such that $\mathcal{R} \subseteq \succ$, the extension proves bounded ground convertibility of the input equations w.r.t. $\succsim$. The extension not only allows to deal with non-orientable equations but relaxes other limitations of basic rewriting induction; in particular, it can take more flexible positions to expand and deal with non-free constructors [2]. 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. Designing a rewriting induction prover with an increased capability of non-orientable equations. In *Proc. of 1st SCSS*, volume 08-08 of *RISC Technical Report*, pages 1–15, 2008.

[2] T. Aoto and Y. Toyama. Proving ground confluence of term rewriting systems by rewriting induction with non-orientable equations. Draft, 2015.

[3] T. Aoto and Y. Toyama. Proving ground confluence of term rewriting systems by rewriting induction with non-orientable equations (invited talk). IFIP WG 1.6, 2015.

[4] T. Aoto, Y. Yoshida, and Y. Toyama. Proving confluence of term rewriting systems automatically. In *Proc. of 20th RTA*, volume 5595 of *LNCS*, pages 93–102. Springer-Verlag, 2009.

[5] U. S. Reddy. Term rewriting induction. In *Proc. of CADE-10*, volume 449 of *LNAI*, pages 162–177. Springer-Verlag, 1990.

CoCo Participant: CeTA 2.21*

Julian Nagele, Christian Sternagel, Thomas Sternagel,
René Thiemann, Sarah Winkler and Harald Zankl

Institute of Computer Science, University of Innsbruck, Austria

Automatic provers have become popular in several areas like first-order theorem proving, SMT, etc. Since these provers are complex pieces of software, they might contain errors which might lead to wrong answers, i.e., incorrect proofs. Therefore, certification of the generated proofs is of major importance.

The tool CeTA [6] 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 and strongly closed TRSs are confluent, as well as check applications of the rule labeling heuristic [4] and addition and removal of redundant rules [3]. To disprove confluence one can provide a divergence $s \to^* t_1$, $s \to^* t_2$ and a certificate for non-joinability. Here CeTA supports: t_1 and t_2 are distinct normal forms, testing that $tcap(t_1\sigma)$ and $tcap(t_2\sigma)$ are not unifiable, usable rules, discrimination pairs, argument filters and interpretations [1], and reachability analysis using tree automata techniques [2].

Conditional term rewrite systems. This year a major novelty in CeTA's repertoire of confluence criteria is support for conditional rewriting. CeTA can now certify that almost orthogonal, properly oriented, right-stable 3-CTRSs are confluent [5], including support for infeasible critical pairs, where currently the supported justification is a certificate for non-reachability using *tcap*. The second supported technique for CTRSs is unraveling [7], 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 analysis with state-compatible automata. In *LATA*, volume 8370 of *LNCS*, pages 347–359, 2014.

[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 H. Zankl. Certified rule labeling. In *RTA*, volume 36 of *LIPIcs*, pages 269–284, 2015.

[5] C. Sternagel and T. Sternagel. Level-confluence of 3-CTRSs in Isabelle/HOL. In *IWC*, 2015. This volume.

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

[7] S. Winkler and R. Thiemann. Formalizing soundness and completeness of unravelings. In *FroCoS*, volume 9322 of *LNCS (LNAI)*, 2015. To appear.

*Supported by Austrian Science Fund (FWF), projects I963, P27502, P27528, and Y757.

CO3

a COnverter for proving COfluence of COnditional TRSs

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

[1] Nagoya University, Nagoya, Japan
{nishida@, kuroda@apal.i., makishi@trs.cm.}is.nagoya-u.ac.jp
[2] UAS Technikum Wien, Vienna, Austria
gmeiner@technikum-wien.at

CO3 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 [4]: the tool first transforms a given normal 1-CTRS into an unconditional term rewriting system (TRS) by using the *SR transformation* [6] or the *unraveling* [3, 5], 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/` via a web interface.

The tool supports *normal 1-CTRSs* without any strategy and theory (specified by `STRATEGY` and `THEORY`, resp.), the class of which includes *TRSs*. Due to a technical reason as shown below, the tool is working for *weakly left-linear* CTRSs which are not TRSs. To enter the competition, the scope of the tool was modified to *oriented 1-CTRSs*.

The main technique in this tool 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 [4], where the (optimized) SR transformation [6] and the sequential (optimized) *unraveling* are denoted by $\mathbb{SR}$ and $\mathbb{U}$, resp. For proving confluence and termination of TRSs, CO3 is using the following very fundamental (sufficient) conditions: (Confluence) "orthogonality" and "termination and joinability of all the critical pairs"; (Termination) "non-existence of SCCs in the *estimated dependency graph* [1]" and "the *dependency pair theorem* [1, Theorem 7] with the *reduction order* based on term-size and variable-occurrence [2, Example 5.2.2]".

The main new feature for CoCo 2015 is to drop *infeasible* rewrite rules. Implemented sufficient conditions for infeasibility are (1) "non-unifiability for the both sides of conditions under $REN(CAP(\cdot))$ in [1]", (2) "left-to-right unreachability of conditions at the root position", and (3) "trivial divergence of evaluating conditions".

References

[1] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theor. Comput. Sci.*, 236(1-2):133–178, 2000.

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

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

[4] 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.

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

[6] T.-F. Şerbănuţă and G. Roşu. Computationally equivalent elimination of conditions. In *Proc. RTA 2006*, vol. 4098 of *LNCS*, pp. 19–34. Springer, 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 T$_{\text{T}}$T$_2$ 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 supported by the JSPS KAKENHI Grant Number 25730004.

[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 2015 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 of $\mathcal{R}$ are joinable [1].

(B) Almost orthogonal 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 different implementations of tree automata completion, to equational reasoning. These are described in [5]. Another recent extension is 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) due to the formalization described in [3]. Future extensions will include support for *join* and *semi-equational* CTRSs.

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. Level-confluence of 3-CTRSs in Isabelle/HOL. In *Proc. 4th IWC*, 2015. This volume.

[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*, 2015. This volume.

[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 I963.

[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 *ScaRT* that consists of multiple classes for term rewriting. The main purpose of this project is to experimentally compare transformations of conditional term rewrite systems and implement program transformations for in particular functional programming languages.

Scart itself is a reimplementation of this project that was originally written in Java (under the working title "KaRT"). A first version 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 recently, a functional, object-oriented programming language.

2 Technical Details

Scart itself only supported higher-order rewrite systems originally. Support for first order terms was only added to simplify the development of CoScart. Therefore, a future support of HORSs is the highest priority for the future development.

Scart contains a rewrite engine that stores and rewrites DAGs of terms in a list. This turns out to be a very efficient way concerning time and memory.

In order to use the Knuth-Bendix method to prove confluence, Scart contains an automatic termination prover for first-order TRSs that uses the dependency pairs method in combination with argument filterings with the *some more*-heuristics of [1].

Since CoScart is currently a one-man project, there are no sophisticated iuser nterfaces yet. Scart is available at `https://github.com/searles/RewriteTool/`.

2.1 Implemented Methods

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.

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.

CoCo 2015 Participant: CSI 0.5.1*

Bertram Felgenhauer, Aart Middeldorp, Julian Nagele and Harald Zankl

Institute 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 open source and available from http://cl-informatik.uibk.ac.at/software/csi, where also a web interface is linked. CSI is based on the termination prover TTT2. The main features of CSI are listed below. Several of these are described in more detail in [8].

2012 CSI is equipped with a strategy language, which allows to configure it flexibly. It features a modular implementation of the decreasing diagrams technique, decomposing TRSs into smaller TRSs based on ordered sorts [2], a cubic time decision procedure for confluence of ground TRSs [1], and non-confluence checks based on tcap and tree automata [8]. CSI can produce proofs in cpf format that can be verified by certifiers like CeTA [7].

2013 The tree automata techniques for detecting non-confluence have been improved. We extended the modular decreasing diagrams implementation to optionally use parallel rewrite steps and parallel critical pairs [9].

2014 We implemented a redex based labeling and a refinement of rule labeling using persistence [9]. CSI produces certifiable output for tree automata based non-confluence [3].

2015 CSI now adds and removes rules before trying to establish (non-)confluence [4]. Furthermore, we incorporated confluence criterion for strongly non-overlapping systems by Sakai et al. [6]. Finally, CSI produces cpf output for the rule labeling heuristic, optionally combined with relative termination [5].

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, A. Middeldorp, H. Zankl, and V. van Oostrom. Layer systems for proving confluence. *ACM TOCL*, 16(2:14):1–32, 2015.

[3] B. Felgenhauer and R. Thiemann. Reachability analysis with state-compatible automata. In *Proc. 8th LATA*, volume 8370 of *LNCS*, pages 347–359, 2013.

[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] J. Nagele and H. Zankl. Certified rule labeling. In *Proc. 26th RTA*, pages 269–284, 2015.

[6] M. Sakai, M. Oyamaguchi, and M. Ogawa. Non-E-overlapping and weakly shallow TRSs are confluent (extended abstract). In *Proc. 3rd IWC*, pages 34–38, 2014.

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

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

[9] H. Zankl, B. Felgenhauer, and A. Middeldorp. Labelings for decreasing diagrams. *JAR*, 54(2):101–133, 2015.

*Supported by the Austrian Science Fund (FWF) P27528.

CoCo 2015 Participant: CSIˆho 0.1*

Julian Nagele

Institute 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. CSIˆho is a tool for automatically proving confluence of such higher-order systems, specifically pattern rewrite systems (PRSs) as introduced by Nipkow [2,3]. 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 CSIˆho implements a version of Nipkow's algorithm for higher-order pattern unification [4].

CSIˆho is built on top of CSI [8], a powerful confluence prover for first-order term rewrite systems, and is available from

http://cl-informatik.uibk.ac.at/software/csi/ho/

Using CSI as foundation, CSIˆho inherits many of its attractions, in particular a strategy language, which allows for flexible configuration. The following confluence criteria are currently supported in CSIˆho:

- Knuth and Bendix' criterion, that is, for terminating PRSs we decide confluence by checking joinability of critical pairs [3]. This is currently the only method CSIˆho implements for proving non-confluence. For showing termination the supported techniques are a basic higher-order recursive path ordering [7] and static dependency pairs with dependency graph decomposition and the subterm criterion [1].
- Weak orthogonality [6], i.e., left-linearity and $s = t$ for all critical pairs $s \leftarrow\!\rtimes\!\rightarrow t$.
- Van Oostrom's development closed critical pair criterion [5]. That is, we conclude confluence of a left-linear PRS if $\leftarrow\!\rtimes\!\rightarrow \subseteq \twoheadrightarrow$ and $\leftarrow\!\bowtie\!\rightarrow \subseteq \twoheadrightarrow \cdot {}^*\!\leftarrow$. Here we approximate $\rightarrow^*$ by $\twoheadrightarrow$.

References

[1] K. Kusakari, Y. Isogai, M. Sakai, and F. Blanqui. Static dependency pair method based on strong computability for higher-order rewrite systems. *IEICE TIS*, 92-D(10):2007–2015, 2009.

[2] R. Mayr and T. Nipkow. Higher-order rewrite systems and their confluence. *TCS*, 192(1):3–29, 1998.

[3] T. Nipkow. Higher-order critical pairs. In *Proc. 6th LICS*, pages 342–349, 1991.

[4] Tobias Nipkow. Functional unification of higher-order patterns. In *Proc. 8th LICS*, pages 64–74, 1993.

[5] V. van Oostrom. Developing developments. *TCS*, 175(1):159–181, 1997.

[6] 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.

[7] F. van Raamsdonk. On termination of higher-order rewriting. In *Proc. 12th RTA*, volume 2051 of *LNCS*, pages 261–275, 2001.

[8] 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.

NoCo: System Description for CoCo 2015

Takaki Suzuki, Kentaro Kikuchi and Takahito Aoto

RIEC, Tohoku University, Sendai, Japan
{ takaki, kentaro, aoto }@nue.riec.tohoku.ac.jp

Nominal rewriting [2, 3] is a framework that extends first-order term rewriting by a binding mechanism. Studies of nominal rewriting are preceded by extensive studies of a *nominal approach* to terms and unifications [4, 5, 7]. 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'.

NoCo (Nominal Confluence tool) is an automated confluence prover for *nominal rewrite systems (NRSs)*. The tool has been developed in Toyama–Aoto group in RIEC, Tohoku University and has been reported in [6]. NoCo 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 2015. Up to our knowledge, it is a first tool that deals with confluence of NRSs.

NoCo proves whether input NRSs are Church-Rosser modulo the α-equivalence (CR$\approx_\alpha$) based on Corollary 40 of [6]. Notions and notations in the following explanation are based on [6]. The corollary provides the following conditions for NRS $\mathcal{R}$ being CR$\approx_\alpha$: (1) $\mathcal{R}$ is orthogonal and (2) $\mathcal{R}$ is abstract skeleton preserving (ASP). It is straightforward to check (2), as the standardness is just a syntactical restriction and $\nabla \vdash a\#X$ is easily checked for any freshness constraint ∇, $a \in \mathcal{A}$ and $X \in \mathcal{X}$. For (1), one has to check (1-a) left-linearity and that (1-b) there's no proper overlaps. The checking of (1-a) is easy. For (1-b), one has to check whether $\nabla_1 \cup \nabla_2^{\pi_2} \cup \{l_1 \approx l_2^{\pi_2}|_p\}$ is unifiable for some permutation π_2, for given $\nabla_1, \nabla_2, l_1, l_2|_p$—this problem is different from nominal unification problems as π_2 is not fixed. Fortunately, this problem is known as an equivariant unification, and has been known to be decidable [1]. From the equivariant unification algorithm in [1], we obtain a constraint of π_2 for unifiability, if the problem is equivariantly unifiable. The system also reports concrete critical pairs generated from this constraint, if there is a proper overlap.

References

[1] J. Cheney. Equivariant unification. *Journal of Automated Reasoning*, 45:267–300, 2010.

[2] M. Fernández and M. J. Gabbay. Nominal rewriting. *Information and Computation*, 205:917–965, 2007.

[3] M. Fernández, M. J. Gabbay, and I. Mackie. Nominal rewriting systems. In *Proceedings of PPDP'04*, pages 108–119. ACM Press, 2004.

[4] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2002.

[5] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186:165–193, 2003.

[6] T. Suzuki, K. Kikuchi, T. Aoto, and Y. Toyama. Confluence of orthogonal nominal rewriting systems revisited. In *Proceedings of 26th RTA*, volume 36 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 301–317, 2015.

[7] C. Urban, A. M. Pitts, and M. J. Gabbay. Nominal unification. *Theoretical Computer Science*, 323:473–497, 2004.