Certified Non-Confluence with ConCon 1.5*

Thomas Sternagel and Christian Sternagel

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

We present three methods to check CTRSs for non-confluence: (1) an ad hoc method for 4-CTRSs, (2) a specialized method for unconditional critical pairs, and finally, (3) a method that employs conditional narrowing to find non-confluence witnesses. We shortly describe our implementation of these methods in ConCon [8], then look into their certification with CeTA [11], and finally conclude with experiments on the confluence problems database (Cops).[1]

1 Preliminaries

We assume familiarity with the basic notions of (conditional) term rewriting [1,4], but shortly recapitulate terminology and notation that we use in the remainder. Given an arbitrary binary relation $\rightarrow$, we write $\leftarrow$, $\rightarrow^+$, $\rightarrow^*$ for *inverse*, *transitive closure*, and *reflexive transitive closure*, respectively. We use $\mathcal{V}(\cdot)$ to denote the set of variables occurring in a given syntactic object, like a term, a pair of terms, a list of terms, etc. The set of terms $\mathcal{T}(\mathcal{F},\mathcal{V})$ over a given signature of function symbols $\mathcal{F}$ and set of variables $\mathcal{V}$ is defined inductively: $x \in \mathcal{T}(\mathcal{F},\mathcal{V})$ for all variables $x \in \mathcal{V}$, and for every n-ary function symbol $f \in \mathcal{F}$ and terms $t_1,\ldots,t_n \in \mathcal{T}(\mathcal{F},\mathcal{V})$ also $f(t_1,\ldots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{V})$. We say that terms s and t *unify* if $s\sigma = t\sigma$ for some substitution σ. The topmost part of a term that does not change under rewriting (sometimes called its "cap") can be approximated for example by the tcap function [2]. Informally, $\mathsf{tcap}(x)$ for a variable x results in a fresh variable, while $\mathsf{tcap}(t)$ for a non-variable term $t = f(t_1,\ldots,t_n)$ is obtained by recursively computing $u = f(\mathsf{tcap}(t_1),\ldots,\mathsf{tcap}(t_n))$ and then asserting $\mathsf{tcap}(t) = u$ in case u does not unify with any left-hand side of rules in the TRS $\mathcal{R}$, and a fresh variable, otherwise. We call a bijective variable substitution $\pi : \mathcal{V} \rightarrow \mathcal{V}$ a *(variable) renaming*. A *CTRS* $\mathcal{R}$ is a set of conditional rewrite rules of the shape $\ell \rightarrow r \Leftarrow c$ where ℓ and r are terms and c is a (possibly empty) sequence of pairs of terms $s_1 \approx t_1,\ldots,s_k \approx t_k$. For all rules in $\mathcal{R}$ we have that $\ell \notin \mathcal{V}$. If additionally $\mathcal{V}(r) \subseteq \mathcal{V}(\ell,c)$ for all rules we call $\mathcal{R}$ a 3-CTRS otherwise a 4-CTRS. We restrict our attention to *oriented* CTRSs where conditions are interpreted as reachability requirements. The rewrite relation induced by an oriented CTRS $\mathcal{R}$ is structured into *levels*. For each level i, a TRS $\mathcal{R}_i$ is defined recursively as follows: $\mathcal{R}_0 = \varnothing$, and $\mathcal{R}_{i+1} = \{\ell\sigma \rightarrow r\sigma \mid \ell \rightarrow r \Leftarrow c \in \mathcal{R}, \forall s \approx t \in c.\, s\sigma \rightarrow^*_{\mathcal{R}_i} t\sigma\}$. The rewrite relation of $\mathcal{R}$ is defined as $\rightarrow_{\mathcal{R}} = \bigcup_{i \geqslant 0} \rightarrow_{\mathcal{R}_i}$. By dropping all conditions from a CTRS $\mathcal{R}$ we obtain its *underlying TRS*, denoted $\mathcal{R}_{\mathsf{u}}$. Note that $\rightarrow_{\mathcal{R}} \subseteq \rightarrow_{\mathcal{R}_{\mathsf{u}}}$. We sometimes label rules like $\rho\colon \ell \rightarrow r \Leftarrow c$. Two variable-disjoint variants of rules $\ell_1 \rightarrow r_1 \Leftarrow c_1$ and $\ell_2 \rightarrow r_2 \Leftarrow c_2$ in $\mathcal{R}$ such that $\ell_1|_p \notin \mathcal{V}$ and $\ell_1|_p\mu = \ell_2\mu$ with most general unifier (mgu) μ, constitute a *conditional overlap*. A conditional overlap that does not result from overlapping two variants of the same rule at the root, gives rise to a *conditional critical pair* (CCP) $\ell_1[r_2]_p\mu \approx r_1\mu \Leftarrow c_1\mu, c_2\mu$. A CCP $u \approx v \Leftarrow c$ is *joinable* if $u\sigma \rightarrow^*_{\mathcal{R}} \cdot {}_{\mathcal{R}}^{*}\!\leftarrow v\sigma$ for all substitutions σ such that $s\sigma \rightarrow^*_{\mathcal{R}} t\sigma$ for all $s \approx t \in c$. Moreover, a CCP $u \approx v \Leftarrow c$ is *infeasible* if there is no substitutions σ such that $s\sigma \rightarrow^*_{\mathcal{R}} t\sigma$ for all $s \approx t \in c$.

*This work is supported by FWF (Austrian Science Fund) project P27502.

[1] http://cops.uibk.ac.at/?q=ctrs+oriented

2 Finding Witnesses for Non-Confluence of CTRSs

To prove non-confluence of a CTRS we have to find a witness, that is, two diverging rewrite sequences starting at the same term whose end points are not joinable.

The first criterion only works for CTRSs that contain at least one unconditional rule of type 4, that is, with extra-variables in the right-hand side.

Lemma 1. *Given a 4-CTRS $\mathcal{R}$ and an unconditional rule $\rho\colon \ell \to r$ in $\mathcal{R}$ where $\mathcal{V}(r) \nsubseteq \mathcal{V}(\ell)$ and r is a normal form with respect to $\mathcal{R}_\mathsf{u}$ then $\mathcal{R}$ is non-confluent.*

Proof. Since $\mathcal{V}(r) \nsubseteq \mathcal{V}(\ell)$ we can always find two renamings μ_1 and μ_2 restricted to $\mathcal{V}(r) \setminus \mathcal{V}(\ell)$ such that $r\mu_1 \ {}_\rho\!\leftarrow \ell\mu_1 = \ell\mu_2 \to_\rho r\mu_2$ and $r\mu_1 \neq r\mu_2$. As r is a normal form with respect to $\mathcal{R}_\mathsf{u}$ also $r\mu_1$ and $r\mu_2$ are (different) normal forms with respect to $\mathcal{R}_\mathsf{u}$ (and hence also with respect to $\mathcal{R}$). Because we found a non-joinable peak $\mathcal{R}$ is non-confluent. □

Example 2. Consider the second (unconditional) rule of the 4-CTRS $\mathcal{R}_{320} = \{\mathsf{e} \to \mathsf{f}(x) \Leftarrow \mathsf{l} \approx \mathsf{d},\ \mathsf{A} \to \mathsf{h}(x,x)\}$ from Cops. Its right-hand side $\mathsf{h}(x,x)$ is a normal form with respect to the underlying TRS of $\mathcal{R}_{320}$ and the only variable occurring in it does not appear in its left-hand side A. So by Lemma 1 $\mathcal{R}_{320}$ is non-confluent.

A natural candidate for diverging situations are the critical peaks of a CTRS. We will base our next criterion on the analysis of *unconditional* critical pairs (CPs) of CTRSs. This restriction is necessary to guarantee the existence of the actual peak. If we would also allow conditional CPs, we first would have to check for infeasibility, since infeasibility is undecidable in general these checks are potentially very costly (see for example [9]).

Lemma 3. *Given a CTRS $\mathcal{R}$ and an unconditional CP $s \approx t$ of it. If s and t are not joinable with respect to $\mathcal{R}_\mathsf{u}$ then $\mathcal{R}$ is non-confluent.*

Proof. The CP $s \approx t$ originates from a critical overlap between two unconditional rules $\rho_1\colon \ell_1 \to r_1$ and $\rho_2\colon \ell_2 \to r_2$ for some mgu μ of $\ell_1|_p$ and ℓ_2 such that $s = \ell_1\mu[r_2\mu]_p \leftarrow \ell_1\mu[\ell_2\mu]_p \to r_1\mu = t$. Since s and t are not joinable with respect to $\mathcal{R}_\mathsf{u}$ they are of course also not joinable with respect to $\mathcal{R}$ and we have found a non-joinable peak. So $\mathcal{R}$ is non-confluent. □

Example 4. Consider the 3-CTRS $\mathcal{R}_{271} = \{\mathsf{p}(\mathsf{q}(x)) \to \mathsf{p}(\mathsf{r}(x)),\ \mathsf{q}(\mathsf{h}(x)) \to \mathsf{r}(x),\ \mathsf{r}(x) \to \mathsf{r}(\mathsf{h}(x)) \Leftarrow \mathsf{s}(x) \approx \mathsf{0},\ \mathsf{s}(x) \to \mathsf{1}\}$ from Cops. First of all we can immediately drop the third rule because we can never satisfy its condition and so it does not influence the rewrite relation of $\mathcal{R}_{271}$. This results in the TRS $\mathcal{R}'_{271}$. Now the left- and right-hand sides of the unconditional CP $\mathsf{p}(\mathsf{r}(z)) \approx \mathsf{p}(\mathsf{r}(\mathsf{h}(z)))$ are not joinable because they are two different normal forms with respect to the underlying TRS of $\mathcal{R}'_{271}$. Hence $\mathcal{R}_{271}$ is not confluent by Lemma 3.

While the above lemmas are easy to check and we have fast methods to do so they are also rather ad hoc. A more general but potentially very expensive way to search for non-joinable forks is to use conditional narrowing [3].

Definition 5 (Conditional narrowing)**.** Given a CTRS $\mathcal{R}$ we say that s *(conditionally) narrows* to t, written $s \rightsquigarrow_\sigma t$ if there is a variant of a rule $\rho\colon \ell \to r \Leftarrow c \in \mathcal{R}$, such that $\mathcal{V}(s) \cap \mathcal{V}(\rho) = \varnothing$ and $u \rightsquigarrow^*_\sigma v$ for all $u \approx v \in c$, a position $p \in \mathcal{P}\mathsf{os}_\mathcal{F}(s)$, a unifier[2] σ of $s|_p$ and ℓ, and $t = s[r]_p\sigma$. For a narrowing sequence $s_1 \rightsquigarrow_{\sigma_1} s_2 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_{n-1}} s_n$ of length n we write $s_1 \rightsquigarrow^n_\sigma s_n$ where $\sigma = \sigma_1\sigma_2\cdots\sigma_{n-1}$. If we are not interested in the length we also write $s \rightsquigarrow^*_\sigma t$.

The following property of narrowing carries over from the unconditional case:

[2] In our implementation we start from an mgu of $s|_p$ and ℓ and extend it while trying to satisfy the conditions.

Property 6. *If $s \rightsquigarrow_{\sigma} t$ then $s\sigma \to t\sigma$ with the same rule that was employed in the narrowing step. Moreover, if $s_1 \rightsquigarrow_{\sigma_1} s_2 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_{n-1}} s_n$ then $s_1\sigma_1\sigma_2\cdots\sigma_{n-1} \to s_2\sigma_2\cdots\sigma_{n-1} \to \cdots \to s_n$. Again employing the same rule for each rewrite step as in the corresponding narrowing step.*

Using conditional narrowing we can now formulate a more general non-confluence criterion.

Lemma 7. *Given a CTRS $\mathcal{R}$, if we can find two narrowing sequences $u \rightsquigarrow_{\sigma}^{*} s$ and $v \rightsquigarrow_{\tau}^{*} t$ such that $u\sigma\mu = v\tau\mu$ for some mgu μ and $s\sigma\mu$ and $t\tau\mu$ are not $\mathcal{R}_{\mathsf{u}}$-joinable then $\mathcal{R}$ is non-confluent.*

Proof. Employing Property 6 we immediately get the two rewriting sequences $u\sigma \to_{\mathcal{R}}^{*} s\sigma$ and $v\tau \to_{\mathcal{R}}^{*} t\tau$. Since rewriting is closed under substitutions we have $s\sigma\mu \;{}_{\mathcal{R}}^{*}\!\!\leftarrow u\sigma\mu = v\tau\mu \to_{\mathcal{R}}^{*} t\tau\mu$. As the two endpoints of these forking sequences $s\sigma\mu$ and $t\tau\mu$ are not joinable by $\mathcal{R}_{\mathsf{u}}$ they are certainly also not joinable by $\mathcal{R}$. This establishes non-confluence of the CTRS $\mathcal{R}$. □

Example 8. Consider the 3-CTRS $\mathcal{R}_{262} = \{0 + y \to y,\ \mathsf{s}(x) + y \to x + \mathsf{s}(y),\ \mathsf{f}(x, y) \to z \Leftarrow x + y \approx z + z'\}$ from Cops. Starting from a variant of the left-hand side of the third rule $u = \mathsf{f}(x', y')$ we have a narrowing sequence $\mathsf{f}(x', y') \rightsquigarrow_{\sigma} x_1$ using the variant $\mathsf{f}(x_1, x_2) \to x_3 \Leftarrow x_1 + x_2 \approx x_3 + x_4$ of the third rule and the substitution $\sigma = \{x' \mapsto x_1, x_3 \mapsto x_1, x_4 \mapsto x_2\}$. We also have another narrowing sequence $\mathsf{f}(x', y') \rightsquigarrow_{\tau} x_3$ using the same variant of rule three and substitution $\tau = \{x \mapsto x_3 + x_4, x' \mapsto 0, y' \mapsto x_3 + x_4, x_1 \mapsto 0, x_2 \mapsto x_3 + x_4\}$ where for the condition $x_1 + x_2 \approx x_3 + x_4$ we have the narrowing sequence $x_1 + x_2 \rightsquigarrow_{\tau} x_3 + x_4$, using a variant of the first rule $0 + x \to x$. Finally, there is an mgu $\mu = \{x_1 \mapsto 0, x_2 \mapsto x_3 + x_4\}$ such that $u\sigma\mu = \mathsf{f}(0, x_3 + x_4) = u\tau\mu$. Moreover, $x_1\sigma\mu = 0$ and $x_3\tau\mu = x_3$ are two different normal forms. Hence $\mathcal{R}_{262}$ is non-confluent by Lemma 7.

3 Implementation

Starting from its first participation in the confluence competition (CoCo)[3] in 2014 ConCon 1.2.0.3 came equipped with some non-confluence heuristics. Back then it only used Lemmas 1 and 3 and had no support for certification of the output. In the next two years (ConCon 1.3.0 and 1.3.2) we focused on other developments [5, 6, 9, 10] and nothing changed for the non-confluence part. For this year's CoCo (2017) we have added Lemma 7 employing conditional narrowing to ConCon 1.5 and the output of all of the non-confluence methods is now certifiable by CeTA.

Our implementation of Lemma 1 takes an unconditional rule $\rho\colon \ell \to r$, a substitution $\sigma = \{x \mapsto y\}$ with $x \in \mathcal{V}(r) \setminus \mathcal{V}(\ell)$ and y fresh w.r.t. ρ and builds the non-joinable fork $r \;{}_{\rho}\!\!\leftarrow \ell \to_{\rho} r\sigma$.

For Lemma 3 we have three concrete implementations that consider an overlap from which an unconditional CP $s \approx t$ arises: The first of which just takes this overlap and then checks that s and t are two different normal forms with respect to $\mathcal{R}_{\mathsf{u}}$. The second employs the `tcap`-function to check for non-joinability, that is, it checks whether $\mathsf{tcap}(s)$ and $\mathsf{tcap}(t)$ are not unifiable. The third makes a special call to the TRS confluence checker CSI [12] providing the underlying TRS $\mathcal{R}_{\mathsf{u}}$ as well as the unconditional CP $s \approx t$ where all variables in s and t have been replaced by fresh constants. We issue the following command:

```
csi -s '(nonconfluence -nonjoinability -steps 0 -tree)[30]' -C RT
```

The strategy '`(nonconfluence -nonjoinability -steps 0 -tree)[30]`' tells CSI to check non-joinability of two terms using tree automata techniques. Here '`-steps 0`' means that CSI does not rewrite the input terms further before checking non-joinability (this would be unsound in our setting). The timeout is set to 30 seconds. To encode the two terms for which we want

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

to check non-joinability in the input we set CSI to read relative-rewriting input (‘-C RT’). We provide $\mathcal{R}_{\mathsf{u}}$ in the usual Cops-format and add one line for the CP $s \approx t$ where its “grounded” left- and right-hand sides are related by ‘->=’, that is, we encode it as a relative-rule. This is necessary to distinguish the unconditional CP from the rewrite rules.

Now, for an implementation of Lemma 7 we have to be careful to respect the freshness requirement of the variables in the used rule for every narrowing step with respect to all the previous terms and rules. The crucial point is to efficiently find the two narrowing sequences, to this end we first restrict the set of terms from which to start narrowing. As a heuristic we only consider the left-hand sides of rules of the CTRS under consideration. Next we also prune the search space for narrowing. Here we restrict the length of the narrowing sequences to at most three. In experiments on Cops allowing sequences of length four or more did not yield additional non-confluence proofs but slowed down the tool significantly to the point where we lost other proofs. Further, we also limit the recursion depth of conditional narrowing by restricting the level (see the definition of the conditional rewrite relation in the Preliminaries) to at most two. Again, we set this limit as tradeoff after thorough experiments on Cops. Finally, we use Property 6 to translate the forking narrowing sequences into forking conditional rewriting sequences. In this way we generate a lot of forking sequences so we only use fast methods, like non-unifiability of the tcap’s of the endpoints or that they are different normal forms, to check for non-joinability of the endpoints. Calls to CSI are to expensive in this context.

4 Certification

Certification is quite similar for all of the described methods. We have to provide a non-confluence witness, that is, a non-joinable fork. So besides the CTRS $\mathcal{R}$ under investigation we also need to provide the starting term s, the two endpoints of the fork t and u, as well as, certificates for $s \rightarrow_{\mathcal{R}}^{+} t$ and $s \rightarrow_{\mathcal{R}}^{+} u$, and a certificate that t and u are not joinable. For the forking rewrite sequences we reuse a recent formalization of ours to build the certificates (see [7]). We also want to stress that because of Property 6 we did not have to formalize conditional narrowing because going from narrowing to rewrite sequences is already done in ConCon and in the certificate only the rewrite sequences show up. For the non-joinability certificate of t and u there are three options: either we state that t and u are two different normal forms or that $\mathsf{tcap}(t)$ and $\mathsf{tcap}(u)$ are not unifiable; both of these checks are performed within CeTA; or, when the witness was found by an external call to CSI, we just include the generated non-joinability certificate.

5 Experiments

We tested the above non-confluence methods on all 129 oriented CTRSs from Cops (as of 2017-06-27). Most of those are 3-CTRSs but there are also four 4-CTRSs. On the whole ConCon 1.5 and CeTA are able to certify non-confluence of 42 systems (including all 4-CTRSs). Last year’s version of ConCon can show non-confluence of 30 of the same 129 CTRSs only using the implementation of Lemmas 1 and 3. By first removing infeasible rules, a feature which we have also recently implemented in ConCon, we gain another system (271, see Example 4). Another new feature is inlining of conditions (see [7]) which gives two more systems (351, 353). Finally, with the help of conditional narrowing (Lemma 7) we gain another 9 systems (272, 328, 330, 352, 391, 404, 410, 411, 524). In contrast ConCon 1.5 succeeds in showing confluence of 60 systems (where 7 use methods that are not certifiable yet, like using Waldmeister to show infeasibility of CCPs). So from ConCon’s perspective only 27 of the 129 oriented CTRSs of Cops

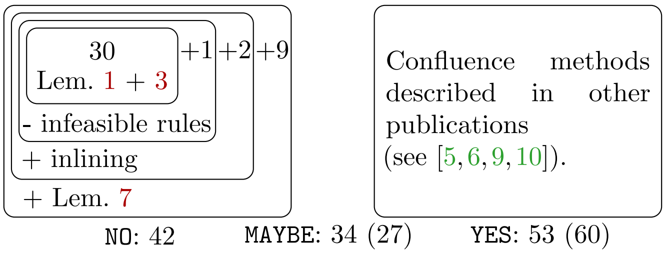

Figure 1: (Non)-confluence results on 129 oriented CTRSs.

are still open. These results are summarized in Figure 1.

Concerning future work we currently employ standard conditional narrowing in our implementation. Implementing basic conditional narrowing or LSE conditional narrowing should increase efficiency and maybe yield new NO-instances.

Acknowledgments. We thank Bertram Felgenhauer for providing the CSI-interface to check non-joinability of two terms. Also the first author wants to thank Vincent van Oostrom for valuable insights during the implementation of conditional narrowing.

References

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

[2] J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and Disproving Termination of Higher-Order Functions. In *Proc. 5th FROCOS*, volume 3717 of *LNCS*, pages 216–231. Springer, 2005. doi:10.1007/11559306_12.

[3] A. Middeldorp and E. Hamoen. Completeness Results for Basic Narrowing. *Appl. Algebra Eng. Commun. Comput.*, 5:213–253, 1994. doi:10.1007/BF01190830.

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

[5] C. Sternagel and T. Sternagel. Level-Confluence of 3-CTRSs in Isabelle/HOL. In *Proc. 4th IWC*, pages 28–32, 2015. arXiv:1602.07115.

[6] C. Sternagel and T. Sternagel. Certifying Confluence of Almost Orthogonal CTRSs via Exact Tree Automata Completion. In *Proc. 1st FSCD*, volume 51 of *LIPIcs*, pages 29:1–29:16. Dagstuhl, 2016. doi:10.4230/LIPIcs.FSCD.2016.29.

[7] C. Sternagel and T. Sternagel. Certifying Confluence of Quasi-Decreasing Strongly Deterministic Conditional Term Rewrite Systems. In *Proc. 26th CADE*, LNCS. Springer, 2017. To be published.

[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. Springer, 2014. doi:10.1007/978-3-319-08918-8_31.

[9] T. Sternagel and A. Middeldorp. Infeasible Conditional Critical Pairs. In *Proc. 4th IWC*, pages 13–17, 2015.

[10] T. Sternagel and C. Sternagel. Formalized Confluence of Quasi-Decreasing, Strongly Deterministic Conditional TRSs. In *Proc. 5th IWC*, pages 60–64, 2016. arXiv:1609.03341.

[11] R. Thiemann and C. Sternagel. Certification of Termination Proofs using CeTA. In *Proc. 22nd TPHOLs*, volume 5674 of *LNCS*, pages 452–468. Springer, 2009. doi:10.1007/978-3-642-03359-9_31.

[12] H. Zankl, B. Felgenhauer, and A. Middeldorp. CSI – A Confluence Tool. In *Proc. 23rd CADE*, volume 6803 of *LNAI*, pages 499–505. Springer, 2011. doi:10.1007/978-3-642-22438-6_38.