# Certification of Confluence Proofs using CeTA*

Julian Nagele and René Thiemann

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

## 1 Introduction

CeTA was originally developed as a tool for certifying termination proofs [5] which have to be provided as certificates in the CPF-format. Its soundness is proven within IsaFoR, the Isabelle Formalization of Rewriting.[1] In the meantime, CeTA can also be used as a tool for certifying confluence and non-confluence proofs. In the following system description, we give a short overview on what kind of proofs are supported, and which information has to be put into the certificates. As we will see, only a small amount of information is required, so that in the future we hope that CSI [8] will not stay the only confluence tool which can produce certificates.

## 2 Terminating Term Rewrite Systems (TRSs)

It is well known that confluence of terminating TRSs is decidable by checking joinability of all critical pairs. The latter can be decided by reducing both terms of a critical pair to an arbitrary normal form and then compare whether these are equal. This technique is also supported in CeTA, where in the certificate one just has to provide the termination proof, and CeTA automatically constructs all critical pairs and checks their joinability by rewriting to normal forms.

Alternatively to this automatic mode via normal forms, there is also an automatic mode which does a breadth-first search for joinability, or one can completely provide the joining sequences within the certificate. Although the latter results in more verbose certificates which are harder to produce, it might actually be faster to check. For example, for $\mathcal{R} = \mathcal{R}_{\mathsf{ack}} \cup \{\mathsf{f}(x) \to x, \mathsf{a} \to \mathsf{ack}(1000,1000), \mathsf{a} \to \mathsf{f}(\mathsf{ack}(1000,1000))\}$ where $\mathcal{R}_{\mathsf{ack}}$ is a convergent TRS for the Ackermann-function, all critical pairs are joinable, but rewriting them to normal form definitely won't work within a reasonable amount of time.

## 3 Certificates for Confluence

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

For the latter, the certificate only consists of the statement that the TRS is weakly orthogonal which is a syntactic criterion that can easily be checked by CeTA.

For the former criterion, the interesting part is to ensure that a given TRS $\mathcal{R}$ is strongly closed, i.e., each (reversed) critical pair $(s,t)$ must be joinable to a common reduct $u$ such that $s \to_{\mathcal{R}}^{*} u$ and $t \to_{\mathcal{R}}^{=} u$. Clearly, rewriting to normal forms is of little use here, so we just offer a breadth-first search within CeTA. In the certificate one just has to provide a bound on the length of the joining derivations within the certificate. The reason for requiring the explicit bound is that all functions within Isabelle have to be total. In contrast to Section 2, at this point $\mathcal{R}$ is not necessarily terminating, and thus an unbounded breadth-first search might be

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

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

nonterminating, whereas an explicit bound on the depth easily ensures totality.

At this point, let us recall our notions of TRSs and critical pairs: a TRS $\mathcal{R}$ is just a set of rules which does not necessarily satisfy the following standard variable conditions.

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

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

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

where it is assumed that the variables in $\ell \to r$ and $\ell' \to r'$ have been renamed apart. We do not exclude root overlaps of a rule with itself, which gives rise to several trivial critical pairs $(r\sigma, r\sigma)$. Therefore, most techniques within IsaFoR that rely on critical pairs immediately try to removal all trivial critical pairs, i.e., they consider $\{(s,t) \in CP(\mathcal{R}) \mid s \neq t\}$ instead of $CP(\mathcal{R})$. So, in practice these additional critical pairs do not play any role. However, for TRSs that do not satisfy the variable conditions, it is essential to include them. For example, for the TRS $\mathcal{R}_1 = \{\mathsf{a} \to y\}$ we have $CP(\mathcal{R}) = \{(x, y)\}$, whereas without root-overlaps with the same rule there wouldn't be any critical pair and we might wrongly conclude confluence via orthogonality.

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

# 4 Disproving Confluence via Non-Joinable Forks

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

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

## 4.1 Different Normal Forms (NF)

Obviously, if $t_i \in NF(\mathcal{R}_i)$ for $i = 1, 2$ and $t_1 \neq t_2$, then $NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1, t_2)$. The certificate for this kind of non-joinability proof does not require any information. Note that for this technique only $VC_{lhs}(\mathcal{R}_i)$ has to be satisfied for both $i$—otherwise, there would not be any normal form.

## 4.2 Tcap and Unification

The function $tcap_{\mathcal{R}}$ can approximate an upper part of a term where no rewriting with $\mathcal{R}$ is possible, and thus, remains unchanged by rewriting. Hence, it suffices to check that $tcap_{\mathcal{R}_1}(t_1)$ is not unifiable with $tcap_{\mathcal{R}_2}(t_2)$ to ensure $NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1, t_2)$.

Since $tcap_{\mathcal{R}_i}$ replaces variables by fresh ones, it is more precise to consider the terms $t_1\sigma$

and $t_2\sigma$ instead of $t_1$ and $t_2$, where $\sigma$ substitutes each variable by a different fresh constants [8]. Obviously, $NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1\sigma, t_2\sigma)$ implies $NJ_{\mathcal{R}_1,\mathcal{R}_2}(t_1, t_2)$ for every substitution $\sigma$, and therefore CeTA always constructs a suitable substitution $\sigma$ and then applies $tcap_{\mathcal{R}_i}$ and the unification algorithm. The certificate does not demand any additional information for this technique.

## 4.3 Usable Rules for Reachability

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

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

Whereas the crucial property was easily formalized within IsaFoR following the original proof, it was actually more complicated to provide an implementation of usable rules which turns the inductive definition of $\mathcal{U}_0$ into executable code.

The implementation is a standard working list algorithm which moves non-usable rules into the set of usable rules until no new usable rules occur. The major problem was that cyclic reasoning like $\mathsf{a} \to \mathsf{b}$ is usable, since $\mathsf{b} \to \mathsf{a}$ is usable, since $\mathsf{a} \to \mathsf{b}$ is usable, ... is perfectly fine in the inductive definition, but it cannot be represented in the working list algorithm. To this end, the crucial idea was to internally built a proof object why some rule is usable which explicitly contains all involved rules. Afterwards, by a minimality argument one can remove all cycles and then soundness of the algorithm is easily proven. Note that we did not had this problem in previous work on usable rules [3] where we explicitly demand that the set of usable rules is provided in the certificate. However, due to our implementation of usable rules, we no longer require the set of usable rules in the certificate.

## 4.4 Discrimination Pairs

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

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

*Proof.* We perform a proof by contradiction, so assume $t_1 \to^*_{\mathcal{R}_1} u$ and $t_2 \to^*_{\mathcal{R}_2} u$, hence $t_2 \to^*_{\mathcal{R}_1^{-1} \cup \mathcal{R}_2} t_1$. Then by the preconditions we obtain $t_2 \succsim^* t_1 \succ t_2$. By iteratively using $\succsim \circ \succ \subseteq \succ$ we achieve $t_2 \succ t_2$ in contradiction to irreflexivity of $\succ$. □

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

## 4.5 Argument Filters

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

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

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

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

### 4.6 Interpretations

Let $\mathcal{F}$ be some signature. Let $\mathcal{A}$ be some weakly monotone $\mathcal{F}$-algebra $(A, (f^{\mathcal{A}})_{f\in\mathcal{F}}, \geq)$, i.e., $f^{\mathcal{A}} : A^n \to A$ for each $n$-ary symbol $f \in \mathcal{F}$, $\geq$ is a partial order, and for each $a, b, f$, $a \geq b$ implies $f^{\mathcal{A}}(\dots, a, \dots) \geq f^{\mathcal{A}}(\dots, b, \dots)$. $\mathcal{A}$ is a quasi-model for $\mathcal{R}$ iff $[\![\ell]\!]_{\mathcal{A},\alpha} \geq [\![r]\!]_{\mathcal{A},\alpha}$ for each $\ell \to r \in \mathcal{R}$ and valuation $\alpha : \mathcal{V} \to A$. Let $a$ be some arbitrary but fixed element of $A$. This allows us to define $\alpha_a(x) = a$ as some default valuation.

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

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

This proof was easy to formalize as it could reuse the formalization on semantic labeling [4], which also includes algorithms to check the quasi-model conditions. Here, CeTA is currently restricted to algebras on finite domains.

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

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

### 4.7 Tree Automata

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

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

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

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

## 5 Modularity of Confluence

In [6] it was proven that confluence is a modular property for disjoint unions of TRSs. Whereas a certificate for applying this proof technique is trivial by just providing the decomposition, we

cannot certify these proofs since currently a formalization of this modularity result is missing.

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

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

*Proof.* By assuming $\neg CR(\mathcal{R})$ there are $s, t, u$ such that $s \to^*_{\mathcal{R}} t$, $s \to^*_{\mathcal{R}} u$, and $NJ_{\mathcal{R},\mathcal{R}}(t, u)$. Since $\mathcal{F}(\mathcal{R}) \cap \mathcal{F}(\mathcal{S}) = \emptyset$, w.l.o.g. we assume $\mathcal{F}(s) \cap \mathcal{F}(\mathcal{S}) = \emptyset$.[3] By $VC_{\supseteq}(\mathcal{R})$ we conclude that also $(\mathcal{F}(t) \cup \mathcal{F}(u)) \cap \mathcal{F}(\mathcal{S}) = \emptyset$ must hold. Assume that $t$ and $u$ are joinable by $\mathcal{R} \cup \mathcal{S}$. By looking at the function symbols and using $VC_{lhs}(\mathcal{S})$ we conclude that the joining sequences cannot use any rule from $\mathcal{S}$. Hence, $t$ and $u$ are joinable by $\mathcal{R}$, a contradiction to $NJ_{\mathcal{R},\mathcal{R}}(t, u)$. □

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

The relaxation on the variable conditions sometimes is helpful:

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

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

# References

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

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

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

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

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

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

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

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

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