Certifying Confluence of Almost Orthogonal CTRSs via Exact Tree Automata Completion*

Christian Sternagel[1] and Thomas Sternagel[2]

1 **University of Innsbruck, Innsbruck, Austria**
`christian.sternagel@uibk.ac.at`
2 **University of Innsbruck, Innsbruck, Austria**
`thomas.sternagel@uibk.ac.at`

Abstract

Suzuki et al. showed that properly oriented, right-stable, orthogonal, and oriented conditional term rewrite systems with extra variables in right-hand sides are confluent. We present our Isabelle/HOL formalization of this result, including two generalizations. On the one hand, we relax proper orientedness and orthogonality to extended proper orientedness and almost orthogonality modulo infeasibility, as suggested by Suzuki et al. On the other hand, we further loosen the requirements of the latter, enabling more powerful methods for proving infeasibility of conditional critical pairs. Furthermore, we formalized a construction by Jacquemard that employs exact tree automata completion for non-reachability analysis and apply it to certify infeasibility of conditional critical pairs. Combining these two results and extending the conditional confluence checker ConCon accordingly, we are able to automatically prove and certify confluence of an important class of conditional term rewrite systems.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs, F.4.1 Mathematical Logic, F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases certification, conditional term rewriting, confluence, infeasible critical pairs, non-reachability

Digital Object Identifier 10.4230/LIPIcs.FSCD.2016.29

1 Introduction

Today there are a number of tools in existence which allow us to conveniently check various properties of standard term rewrite systems (TRSs). To not just rely on the trustworthiness and programming-prowess of the tool-authors, these tools are progressively accompanied by certifiers, that is, computer-verified programs which rigorously assure correctness of a tool's output with respect to a given input. The prevalent procedure for the development of certifiers comprises the following two phases: First, employ a proof assistant (in our case Isabelle/HOL [9]) 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 [17]).

Just for clarification, by *formalizing the underlying theory*, we mean that we take known proofs and definitions from the literature as well as our own results, scrutinize them, fill in the gaps, and provide such a level of detail that we arrive at a mechanized proof that is accepted by a proof assistant. Against common belief, such mechanized proofs are not necessarily

* The research described in this paper is supported by FWF (Austrian Science Fund) project P27502.
[1] `http://cl-informatik.uibk.ac.at/software/ceta/`

inscrutable to humans. On the contrary, especially Isabelle/HOL's *proof documents* are highly structured and, provided some practice, are sometimes easier to follow than their paper-originals (for the best reading experience we recommend Isabelle/jEdit [19] for browsing). More often than not, at least some minor inaccuracies and sometimes even proper errors in published proofs are exposed during the process of formalization (e.g., in previous work [14] one of the authors detected a missing left-linearity assumption in some of his yet earlier work [11]). So the benefits of formalization are threefold:

1. By scrutinizing known results, we gain a better understanding and clarify inaccuracies and sometimes even correct errors.
2. We obtain computer-checkable theories which may be used to generate certifiers for the underlying results as well as to build new results on top of them.
3. Using such certifiers we are able to increase the reliability of tools that build on the formalized theory and also expose errors in the tools themselves.

As mentioned earlier the formalization of standard term rewriting is ongoing work since almost a decade, with many widely used results.

Ultimately we strife to establish the same state of the art for conditional term rewrite systems (CTRSs). We already embarked on this journey in [12] and further this enterprise by generalizing and extending our previous work.

The developments we describe in this article are part of the IsaFoR library and are freely available for inspection, see theories `Conditional_Rewriting/Level_Confluence.thy` and `Tree_Automata/Exact_Tree_Automata_Completion.thy` and their `*_Impl.thy` variants.

Contribution. The following three tasks are the original contributions of this work. We already formalized the result that right-stable, properly oriented, almost orthogonal, oriented 3-CTRSs are confluent by Suzuki et al. [16] in previous work [12]. (1) Here, we extend the syntactical part of the criterion to be applicable to *extended properly oriented* CTRSs. Moreover, we revisit the definition of *almost orthogonality* and relax it in a way that we now may employ new infeasibility criteria (Sections 3 and 4.3). Moreover, we shortly revisit non-reachability and non-joinability via tcap (Section 4.1) with an eye towards certification. (2) Additionally, we formalized the known result that reachability is decidable for linear and growing TRSs by Jacquemard [8] (Section 4.2). We use this to check for infeasibility of conditional critical pairs. (3) Finally, we incorporated the above findings in the certifier CeTA (Section 4.3) as well as the conditional confluence checker ConCon [15] (Section 5), so we are able to certify a large portion of the confluence proofs which are generated by ConCon.

Related Work. Felgenhauer and Thiemann [3] formalize so called state-compatible automata and thereby also show that it is decidable if a regular tree language is closed under rewriting. This is also part of IsaFoR and loosely related to our work. In the yearly confluence competition[2] confluence checkers for various flavors of term rewriting (like CO3, CoScart, and ConCon for conditional rewriting [18]) compete in different categories. However, at the time of writing none of the other conditional confluence tools supports certification.

2 Preliminaries

We assume familiarity with the basic notions of (conditional) term rewriting [2, 10], but shortly recapitulate terminology and notation that we use in the remainder.

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

Given two arbitrary binary relations $\to_\alpha$ and $\to_\beta$, we write ${}_\alpha\!\leftarrow$, $\to_\alpha^+$, $\to_\alpha^*$ for the *inverse*, the *transitive closure*, and the *reflexive transitive closure* of $\to_\alpha$, respectively. Moreover, the relations ${}_\alpha^*\!\leftarrow \cdot \to_\beta^*$ and $\to_\beta^* \cdot {}_\alpha^*\!\leftarrow$ are called *meetability* and *joinability*. We say that $\to_\alpha$ and $\to_\beta$ *commute* whenever ${}_\alpha^*\!\leftarrow \cdot \to_\beta^* \subseteq \to_\beta^* \cdot {}_\alpha^*\!\leftarrow$ holds. The same property is called *confluence*, in case α and β coincide. Given a set B we define the set of ancestors with respect to $\to_\alpha$ by $(\to_\alpha)[B] = \{a \mid \exists b \in B.\, a \to_\alpha b\}$.

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 $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. A term t is called *ground* if $\mathcal{V}(t) = \varnothing$. The set of ground terms over $\mathcal{F}$ is denoted by $\mathcal{T}(\mathcal{F})$ and the set of ground instances of a term t over a fixed signature is denoted by $\Sigma(t)$. We say that terms s and t *unify*, written $s \sim t$, if $s\sigma = t\sigma$ for some substitution σ. For brevity, we speak about non-reachability, non-meetability, and non-joinability of two terms s and t, when we actually mean that the respective property holds for arbitrary substitution instances $s\sigma$ and $t\tau$. The tcap function [6] approximates the topmost part of a term, its "cap," that does not change under rewriting (we defer a formal definition until Section 4.1). It is well known that $\mathsf{tcap}(s) \not\sim t$ implies non-reachability of t from s.

For the purposes of this paper a *rewrite rule* (or just *rule*) is a pair of terms, written $\ell \to r$, whose left-hand side is not a variable (meaning that extra variables in right-hand sides are explicitly allowed). A *conditional rewrite rule* is additionally equipped with a list of pairs of terms, written $c = s_1 \approx t_1, \ldots, s_k \approx t_k$, and called its *conditions*. Let c_i denote the first i conditions of c and $c_{i,j}$ the list of conditions $s_i \approx t_i, \ldots, s_j \approx t_j$. A *(conditional) term rewrite system* $\mathcal{R}$ is a set of (conditional) rewrite rules. A CTRS which allows for extra variables in right-hand sides of rules is called a *3-CTRS* (formally, $\mathcal{V}(r) \subseteq \mathcal{V}(\ell, c)$ for all $\ell \to r \Leftarrow c \in \mathcal{R}$). We restrict our attention to *oriented* CTRSs, i.e., 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 $\mathcal{R}_{i+1} = \{\ell\sigma \to r\sigma \mid \ell \to r \Leftarrow c \in \mathcal{R}, \forall s \approx t \in c.\, s\sigma \to_{\mathcal{R}_i}^* t\sigma\}$. For brevity, we write $\to_n$ for the rewrite relation of $\mathcal{R}_n$ whenever $\mathcal{R}$ is clear from the context. Furthermore, we write $\sigma, n \vdash c$, whenever $s\sigma \to_n^* t\sigma$ for all $s \approx t$ in c. By dropping all conditions from a CTRS $\mathcal{R}$ we obtain its *underlying TRS*, denoted $\mathcal{R}_u$. Note that $\to_{\mathcal{R}} \subseteq \to_{\mathcal{R}_u}$.

Two variable-disjoint variants of rules $\ell_1 \to r_1 \Leftarrow c_1$ and $\ell_2 \to r_2 \Leftarrow c_2$ in $\mathcal{R}$ such that $\ell_1|_p \notin \mathcal{V}$ and $\ell_1|_p\mu = \ell_2\mu$ with most general unifier (mgu) μ, constitute a *conditional overlap*. A conditional overlap that does not result from overlapping two variants of the same rule at the root, gives rise to a *conditional critical pair* (CCP) $r_1\mu \approx r_1[r_2]_p\mu \Leftarrow c_1\mu, c_2\mu$. A CCP $s \approx t \Leftarrow c$ is said to be *infeasible* if its conditions cannot be satisfied by any substitution σ. We sometimes use rules, overlaps, critical pairs, etc. without the addendum "conditional."

We consider bottom-up non-deterministic finite tree automata (TA) $\mathcal{A} = \langle \mathcal{F}, Q, Q_f, \Delta \rangle$ where $\mathcal{F}$ is a signature, Q a set of states disjoint from the signature, $Q_f \subseteq Q$ the set of final states, and Δ a set of transitions of the shape $f(q_1, \ldots, q_n) \to q$ with $f \in \mathcal{F}$ and $q_1, \ldots, q_n, q \in Q$ or $q \to p$ with $q, p \in Q$. The language of a TA $\mathcal{A}$ is given by $L(\mathcal{A}) = \{t \in \mathcal{T}(\mathcal{F}) \mid \exists q \in Q_f.\, t \to_{\mathcal{A}}^* q\}$. We say that a set of ground terms E is *regular* if there is a TA $\mathcal{A}$ such that $L(\mathcal{A}) = E$. A substitution from variables to states is called a *state substitution*. A TRS $\mathcal{R}$ is called *growing* if for all $\ell \to r \in \mathcal{R}$ the variables in $\mathcal{V}(\ell) \cap \mathcal{V}(r)$ occur at depth at most 1 in ℓ (cf. [8]). Given a TRS $\mathcal{R}$ the *linear growing approximation* [8] is defined as any linear growing TRS obtained from $\mathcal{R}$ by linearizing the left-hand sides,

renaming the variables in the right-hand sides that occur at a depth greater than one in the corresponding left-hand side, and finally also linearizing the right-hand sides.

3 An Enhanced Criterion for Level-Confluence of CTRSs

Level-confluence of a CTRS $\mathcal{R}$ is the property that $\mathcal{R}_n$ is confluent for each level n. Clearly, level-confluence implies confluence (take the maximum of the two levels employed in a peak).

The following purely syntactic criterion for level-confluence of (possibly nonterminating) oriented CTRSs with extra variables in right-hand sides was given by Suzuki et al.

▶ **Lemma 1** (Suzuki et al. [16, Corollary 4.7]). *Orthogonal, properly oriented, right-stable, and oriented 3-CTRSs are level-confluent.* ◀

In earlier work [12] we formalized Lemma 1 in Isabelle/HOL and extended it from orthogonal to almost orthogonal CTRSs with infeasible critical pairs. In the following we present a relaxation of *almost orthogonality modulo infeasibility* that allows us to conclude non-meetability from non-joinability when showing infeasibility of conditional overlaps.

▶ **Definition 2** (Almost Orthogonality modulo Infeasibility). A left-linear CTRS $\mathcal{R}$ is *almost orthogonal (modulo infeasibility)* if each overlap between rules $\ell_1 \to r_1 \Leftarrow c_1$ and $\ell_2 \to r_2 \Leftarrow c_2$ with mgu μ at position p either

1. results from overlapping two variants of the same rule at the root, or
2. is trivial (i.e., $p = \epsilon$ and $r_1\mu = r_2\mu$), or
3. is infeasible in the following sense: for arbitrary m and n, whenever levels m and n commute, then it is impossible to satisfy the conditions stemming from the first rule on level m and at the same time the conditions stemming from the second rule on level n. More formally: $\forall m\, n.\, ({}^*_m\!\leftarrow \cdot \to^*_n \subseteq \to^*_n \cdot {}^*_m\!\leftarrow \implies \nexists \sigma.\, m, \sigma \vdash c_1\mu \wedge n, \sigma \vdash c_2\mu)$.

Note that without **2** and **3**, Definition 2 corresponds to plain orthogonality. Also note that by dropping **3**, Definition 2 reduces to the definition of *almost orthogonality* given by Hanus [7]. In our original definition of *almost orthogonality modulo infeasibility* [12], **3** is the stronger requirement that the conditions of the resulting critical pair are infeasible (i.e., $\nexists \sigma\, n.\, n, \sigma \vdash c_1\mu, c_2\mu$). In the following, whenever we talk about *almost orthogonality* we mean Definition 2.

Observe that the level-commutation[3] assumption of the third alternative in Definition 2 allows us to reduce non-meetability to non-joinability. That this is useful in practice is shown by the following example.

▶ **Example 3** (Non-Meetability via tcap). Consider the CTRS consisting of the two rules $\{\mathsf{f}(x) \to \mathsf{a} \Leftarrow x \approx \mathsf{a}, \mathsf{f}(x) \to \mathsf{b} \Leftarrow x \approx \mathsf{b}\}$ which has the critical pair $\mathsf{a} \approx \mathsf{b} \Leftarrow x \approx \mathsf{a}, x \approx \mathsf{b}$. Since $\mathsf{tcap}(\mathsf{cs}(x, x)) = \mathsf{cs}(y, z) \sim \mathsf{cs}(\mathsf{a}, \mathsf{b})$, where cs is an auxiliary function symbol, we cannot conclude infeasibility via non-reachability analysis using tcap. However, $\mathsf{tcap}(\mathsf{a}) = \mathsf{a} \not\sim \mathsf{b} = \mathsf{tcap}(\mathsf{b})$ shows non-joinability of a and b. By **3** this shows non-meetability of a and b and thereby infeasibility of the critical pair.

In general it is beneficial to test for non-meetability via non-joinability of conditions with identical left-hand sides, see also Lemma 21.

Before we can state the main result of this section we have to define two syntactic properties of conditional rewrite rules.

[3] While this is called *shallow confluence* in the literature, we believe that *level-commutation* is a better, since more descriptive, name.

▶ **Definition 4** (Right-Stable, Extended Properly Oriented). A conditional rule $\ell \to r \Leftarrow c$ with k conditions $c = s_1 \approx t_1, \ldots, s_k \approx t_k$ is called

1. *right-stable* whenever we have $\mathcal{V}(t_i) \cap \mathcal{V}(\ell, c_{i-1}, s_i) = \varnothing$ and t_i is either a linear constructor term or a ground $\mathcal{R}_u$-normal form, for all $1 \leqslant i \leqslant k$; and
2. *extended properly oriented* when either $\mathcal{V}(r) \subseteq \mathcal{V}(\ell)$ or there is some $0 \leqslant m \leqslant k$ such that $\mathcal{V}(s_i) \subseteq \mathcal{V}(\ell, t_1, \ldots, t_{i-1})$ for all $1 \leqslant i \leqslant m$ and $\mathcal{V}(r) \cap \mathcal{V}(s_i \approx t_i) \subseteq \mathcal{V}(\ell, t_1, \ldots, t_m)$ for all $m < i \leqslant k$.

Observe the following property of extended properly oriented rules of 3-CTRSs

$$\mathcal{V}(r) \subseteq \mathcal{V}(\ell, c_m) \cup (\mathcal{V}(r) \cap \mathcal{V}(c_{m+1,k})) \tag{$\star$}$$

which we will use later and which can be shown by induction on $k - m$.

▶ **Theorem 5.** *Almost orthogonal, extended properly oriented, right-stable, and oriented 3-CTRSs are confluent.*

Before proving this statement we need an auxiliary definition, where we adopt the convention that the number of holes of a multihole context is denoted by the corresponding lower-case letter, e.g., c for C, d for D, e for E etc.

▶ **Definition 6.** We say that there is an *extended parallel rewrite step at level* n from s to t, written $s \mathrel{\#\!\!\!\to}_n t$, whenever we have a multihole context C, and sequences of terms $s_1, \ldots, s_c$ and $t_1, \ldots, t_c$, such that $s = C[s_1, \ldots, s_c]$, $t = C[t_1, \ldots, t_c]$, 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 $s_i \to_{n-1}^* t_i$.

Proof of Theorem 5. Let $\mathcal{R}$ be a CTRS satisfying all required properties. Instead of directly proving the above statement, we prove the commuting diamond property, ${}_m\!\mathrel{\leftarrow\!\!\!\#} \cdot \mathrel{\#\!\!\!\to}_n \subseteq \mathrel{\#\!\!\!\to}_n \cdot {}_m\!\mathrel{\leftarrow\!\!\!\#}$, for all m and n and a suitable relation $\to_n \subseteq \mathrel{\#\!\!\!\to}_n \subseteq \to_n^*$ which is called *extended parallel rewriting*. This yields commutation of $\to_m^*$ and $\to_n^*$ for all m and n, and thereby level-confluence, which in turn ensures confluence.

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 $t \;{}_m\!\mathrel{\leftarrow\!\!\!\#}\; s \mathrel{\#\!\!\!\to}_n u$. If any of m and n equals 0, we are done (since $\mathrel{\#\!\!\!\to}_0$ 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 $1 \leqslant i \leqslant d$.

It is relatively easy to define the greatest lower bound $C \sqcap D$ of two contexts C and D by a recursive function (that simultaneously traverses the two contexts in a top-down manner and replaces subcontexts that differ by a hole) and prove that we obtain a lower semilattice. Now we identify the common part E of C and D, employing the semilattice properties of multihole contexts, 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, it suffices to show that for each $1 \leqslant i \leqslant e$ there is a term v such that $t'_i \mathrel{\#\!\!\!\to}_n v \;{}_m\!\mathrel{\leftarrow\!\!\!\#}\; v'_i$, in order to conclude the proof. We concentrate on the case $C_i = \Box$ (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, 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 $\ell \to r \Leftarrow c$ with conditions $c = s_1 \approx t_1, \ldots, s_j \approx t_j$ were involved, i.e., $u'_i = \ell\sigma_1 = \ell\sigma_2$ for some substitutions σ_1 and σ_2 that both satisfy all conditions in c. Without extra variables in r, this is the end of the story (since then $r\sigma_1 = r\sigma_2$); but we also want to cover the case where $\mathcal{V}(r) \not\subseteq \mathcal{V}(\ell)$, and thus have to reason why this does not cause any trouble. Together with the fact that $\ell \to r \Leftarrow c$ is extended properly oriented we obtain a $0 \leqslant k \leqslant j$ such that (1) $\mathcal{V}(s_i) \subseteq \mathcal{V}(\ell, t_1, \ldots, t_{i-1})$ for all $1 \leqslant i \leqslant k$ and (2) $\mathcal{V}(r) \cap \mathcal{V}(s_i \approx t_i) \subseteq \mathcal{V}(\ell, t_1, \ldots, t_k)$ for all $k < i \leqslant j$ by Definition 4.**2**. Then we prove by an inner induction on $i \leqslant j$ that there is a substitution σ such that

   a. $\sigma(x) = \sigma_1(x) = \sigma_2(x)$ for all x in $\mathcal{V}(\ell)$, and

   b. $\sigma_1(x) \twoheadrightarrow^*_{n'} \sigma(x)$ and $\sigma_2(x) \twoheadrightarrow^*_{m'} \sigma(x)$ for all x in $\mathcal{V}(\ell, c_{\min\{i,k\}}) \cup (\mathcal{V}(r) \cap \mathcal{V}(c_{k+1,i}))$.

   In the base case σ_1 satisfies the requirements. So suppose $i > 0$ and assume by IH that both properties hold for $i - 1$ and some substitution σ. If $i > k$ we are done by (2). Otherwise $i \leqslant k$. Now consider the condition $s_i \approx t_i$. By (1) we have $\mathcal{V}(s_i) \subseteq \mathcal{V}(\ell, c_{i-1})$. Using the IH for **1b** we obtain $s_i\sigma_1 \twoheadrightarrow^*_{n'} s_i\sigma$ and $s_i\sigma_2 \twoheadrightarrow^*_{m'} s_i\sigma$. Moreover $s_i\sigma_1 \twoheadrightarrow^*_{m'} t_i\sigma_1$ and $s_i\sigma_2 \twoheadrightarrow^*_{n'} t_i\sigma_2$ since σ_1 and σ_2 satisfy c, and thus by the outer IH we obtain s' such that $t_i\sigma_1 \twoheadrightarrow^*_{n'} s'$ and $t_i\sigma_2 \twoheadrightarrow^*_{m'} s'$. Recall that by right-stability t_i is either a ground $\mathcal{R}_u$-normal form or a linear constructor term. In the former case $t_i\sigma_1 = t_i\sigma_2 = s'$ and hence σ satisfies **1a** and **1b**. In the latter case right-stability allows us to combine the restriction of σ_1 to $\mathcal{V}(t_i)$ and the restriction of σ to $\mathcal{V}(\ell, c_{i-1})$ into a substitution satisfying **1a** and **1b**. This concludes the inner induction. Since $\mathcal{R}$ is a 3-CTRS, using $(\star)$ together with **1b** shows $r\sigma_1 \twoheadrightarrow^*_{n'} r\sigma$ and $r\sigma_2 \twoheadrightarrow^*_{m'} r\sigma$. Since $\twoheadrightarrow^*_{n'} \subseteq \twoheadrightarrow_n$ and $\twoheadrightarrow^*_{m'} \subseteq \twoheadrightarrow_m$ we can take $v = r\sigma$ to conclude this case.
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 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. [16] – but surprisingly hard to formalize, even more so when having to deal with infeasibility). We again close this case by induction on the number of conditions making use of right-stability of $\mathcal{R}$. ◀

Clearly, applicability of Theorem 5 relies on having powerful techniques for proving infeasibility at our disposal. Those are the topic of the next section.

4 Infeasibility

In the context of oriented conditional term rewriting we may employ non-reachability criteria in order to conclude infeasibility of conditions. The two prevalent methods to check for non-reachability use unification and tree automata techniques, respectively. In the following two sections we describe both of these.

4.1 Non-reachability by unification

Probably the fastest available method of checking for non-reachability is to try to unify the tcap of the source term with the target term; which is the de facto standard for approximating dependency graphs for termination proofs [6]. Typical "pen and paper" definitions rely on replacing subterms by "fresh variables" making them somewhat hard to formalize (as already remarked in [17]). Instead of inventing fresh variables out of thin air, the IsaFoR-version of tcap replaces every variable occurrence by the symbol $\Box$. The resulting terms behave like ground multihole contexts – we call them *ground contexts* – and they are intended to represent the set of all terms resulting from replacing all "holes" by arbitrary terms. This is made formal by the *substitution instance class* $[\![t]\!]$ of a ground context t: $[\![\Box]\!] = \mathcal{T}(\mathcal{F}, \mathcal{V})$ and $[\![f(t_1, \ldots, t_n)]\!] = \{f(s_1, \ldots, s_n) \mid s_i \in [\![t_i]\!]\}$. Note that for variable-disjoint terms s and t, unifiability coincides with $s\sigma = t\tau$ for some substitutions σ and τ. Thus asking whether a term t unifies with a variable-disjoint term represented by the ground context s is equivalent to checking whether $t\sigma \in [\![s]\!]$ for some substitution σ. The latter is called *ground context matching* and shown to be decidable by an efficient algorithm by Thiemann and Sternagel [17]. Thus we can define an efficient executable version of tcap by

$$\mathsf{tcap}_{\mathcal{R}}(t) = \begin{cases} \Box & \text{if } t \text{ is a variable} \\ \Box & \text{if } t = f(t_1, \ldots, t_n) \text{ and } \ell\sigma \in [\![u]\!] \text{ for some } \sigma \text{ and } \ell \to r \in \mathcal{R} \\ u & \text{otherwise} \end{cases}$$

where $u = f(\mathsf{tcap}_{\mathcal{R}}(t_1), \ldots, \mathsf{tcap}_{\mathcal{R}}(t_n))$.

▶ **Lemma 7** (tcap is sound). *If* $s\sigma \to_{\mathcal{R}}^* t$ *then* $t \in [\![\mathit{tcap}(s)]\!]$. ◀

Then checking non-reachability of t from s amounts to deciding whether $\nexists\tau.\, t\tau \in [\![\mathsf{tcap}(s)]\!]$, for which we use the more succinct notation $\mathsf{tcap}(s) \not\sim t$ almost everywhere else in this paper.

While the above definition of tcap and the corresponding soundness lemma were already present in IsaFoR, the following easy extension also allows us to test for non-joinability.

▶ **Lemma 8.** *If* $s\sigma \to_{\mathcal{R}}^* \cdot {}_{\mathcal{R}}^*\!\leftarrow t\tau$ *then* $[\![\mathit{tcap}(s)]\!] \cap [\![\mathit{tcap}(t)]\!] \neq \varnothing$.

Proof. We have $s\sigma \to_{\mathcal{R}}^* u$ and $t\tau \to_{\mathcal{R}}^* u$ for some u. By Lemma 7 we have $u \in \mathsf{tcap}(s)$ and $u \in \mathsf{tcap}(t)$. ◀

Fortunately the same techniques that are used to obtain an algorithm for ground context matching can be reused for *ground context unifiability*, i.e., checking $[\![\mathsf{tcap}(s)]\!] \cap [\![\mathsf{tcap}(t)]\!] \neq \varnothing$ (elsewhere in this paper we use the notation $\mathsf{tcap}(s) \not\sim \mathsf{tcap}(t)$).

4.2 Non-reachability by exact tree automata completion

What is generally known as tree automata completion today was introduced by Genet in 1998 [4, 5]. But already in 1996 Jacquemard [8] used a similar concept to show decidability of reachability for linear and growing TRSs. His proof was based on the construction of a tree automaton that accepts the set of ground terms which are normalizable with respect to a given linear and growing TRS $\mathcal{R}$. If we replace the automaton recognizing $\mathcal{R}$-normal forms in Jacquemard's construction by an arbitrary automaton $\mathcal{A}$ we arrive at a tree automaton that accepts the $\mathcal{R}$-ancestors of the language of $\mathcal{A}$.

We need some basic definitions and auxiliary lemmas before we present the construction of this *ancestor automaton* in detail.

▶ **Definition 9** (Ground-Instance Transitions, Δ_t)**.** Let $[t]$ denote a term $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ where all variable-occurrences have been replaced by a fresh symbol $\Box$. Using such terms as states we define the set Δ_t that contains all transitions which are needed to recognize all ground-instances of a term $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ in state $[t]$.

$$\Delta_t = \begin{cases} \{f([t_1],\dots,[t_n]) \to [t]\} \cup \bigcup_{1 \leqslant i \leqslant n} \Delta_{t_i} & \text{if } t = f(t_1,\dots,t_n) \\ \{f(\Box,\dots,\Box) \to \Box \mid f \in \mathcal{F}\} & \text{otherwise} \end{cases}$$

Note that if t is not linear this actually gives an overapproximation.

The next lemma holds by definition of Δ_t.

▶ **Lemma 10.** *For any subterm s of any term t if there is a sequence $u \to^+_{\Delta_t} [s]$ then u is a ground-instance of s, and vice versa if t is linear.* ◀

We now use Δ_t to define an automaton for the ground-instances of t.

▶ **Definition 11** (Ground-Instance Automaton, $\mathcal{A}_{\Sigma(t)}$)**.** Let Q_t denote the set of states occurring in Δ_t then we call the tree automaton $\mathcal{A}_{\Sigma(t)} = \langle \mathcal{F}, Q_t, \{[t]\}, \Delta_t \rangle$ the *ground-instance automaton for t*.

▶ **Lemma 12.** *The language of $\mathcal{A}_{\Sigma(t)}$ is an overapproximation of the set of ground-instances of t in general and an exact characterization if t is linear.* ◀

Using the concept of ground-instance automaton we are now able to define a tree automaton which accepts all $\mathcal{R}$-ancestors of a given regular set of ground terms using exact tree automata completion (ETAC).

▶ **Definition 13** (Ancestor Automaton, $\mathsf{anc}_\mathcal{R}(\mathcal{A})$)**.** Given a tree automaton $\mathcal{A} = \langle \mathcal{F}, Q_\mathcal{A}, Q_f, \Delta \rangle$ whose states are all accessible, and a linear and growing TRS $\mathcal{R}$ the construction proceeds as follows.

First we extend the set of transitions of $\mathcal{A}$ in such a way that we can match left-hand sides of rules in $\mathcal{R}$. This yields the set of transitions $\Delta_0 = \Delta \cup \bigcup_{\ell \to r \in \mathcal{R}} \Delta_\ell$. Let $\mathcal{A}_0 = \langle \mathcal{F}, Q, Q_f, \Delta_0 \rangle$ where Q denotes the set of states in Δ_0. We have to ensure (for example by using the disjoint union of states) that for any state q which is used in both Δ and some Δ_ℓ, the terms which can reach it are the same ($\{t \mid t \to^+_\Delta q\} = \{t \mid t \to^+_{\Delta_\ell} q\}$). Then the language does not change, that is, $L(\mathcal{A}_0) = L(\mathcal{A})$.

Finally, we saturate Δ_0 by inference rule (†) in order to extend the language by $\mathcal{R}$-ancestors, that is, if we can reach a state q from an instance of a right-hand side of a rule in $\mathcal{R}$ we add a transition which ensures q is reachable from the corresponding left-hand side.[4]

$$\frac{f(\ell_1,\dots,\ell_n) \to r \in \mathcal{R} \quad r\theta \to^*_{\Delta_k} q}{f(q_1,\dots,q_n) \to q \in \Delta_{k+1}} \tag{†}$$

Here $\theta\colon \mathcal{V}(r) \to Q$ is a state substitution and $q_i = \ell_i\theta$ if ℓ_i is a variable in r and $q_i = [\ell_i]$ otherwise. Note that this inductive definition possibly adds many new transitions from Δ_k to Δ_{k+1}.

Since $\mathcal{R}$ is finite, the number of states is finite, and we do not introduce new states using (†), this process terminates after finitely many steps resulting in the set of transitions Δ_m. Also note that Δ_k is monotone with respect to k, i.e., $\Delta_k \subseteq \Delta_{k+1}$ for all $k \geqslant 0$. We call $\mathsf{anc}_\mathcal{R}(\mathcal{A}) = \langle \mathcal{F}, Q, Q_f, \Delta_m \rangle$ the $\mathcal{R}$-ancestors automaton for $\mathcal{A}$. It is easy to show that $L(\mathcal{A}_0) \subseteq L(\mathsf{anc}_\mathcal{R}(\mathcal{A}))$.

[4] This is symmetric to resolving compatibility violations in the tree automata completion by Genet [4, 5].

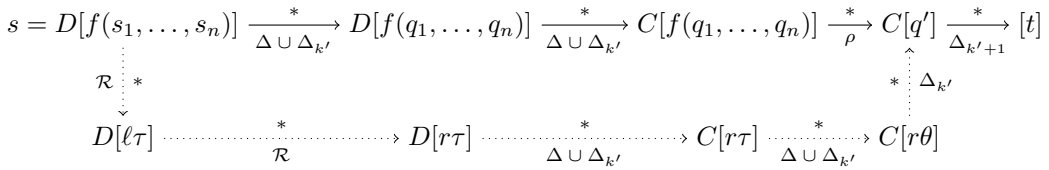

Figure 1 Bypassing ρ to close the induction step.

▶ **Theorem 14.** *Given a tree automaton $\mathcal{A}$ as well as a linear and growing TRS $\mathcal{R}$ the language of $\mathsf{anc}_{\mathcal{R}}(\mathcal{A})$ is exactly the set of $\mathcal{R}$-ancestors of $L(\mathcal{A})$.*

Proof. First we prove that $(\to_{\mathcal{R}}^*)[L(\mathcal{A})] \subseteq L(\mathsf{anc}_{\mathcal{R}}(\mathcal{A}))$. Pick a term $s \in (\to_{\mathcal{R}}^*)[L(\mathcal{A})]$. But that means that there is a rewrite sequence $s \to_{\mathcal{R}}^k t$ of length $k \geqslant 0$ for some $t \in L(\mathcal{A})$. We proceed by induction on k. If $k = 0$ then $s = t$ and hence $s \in L(\mathsf{anc}_{\mathcal{R}}(\mathcal{A}))$. Now assume $k = k' + 1$ for some $k' \geqslant 0$ then there is a rewrite sequence $s = C[f(\ell_1, \ldots, \ell_n)\sigma] \to_{\mathcal{R}} C[r\sigma] \to_{\mathcal{R}}^{k'} t$ for some context C, rewrite rule $f(\ell_1, \ldots, \ell_n) \to r \in \mathcal{R}$, and substitution σ. By induction hypothesis $C[r\sigma] \in L(\mathsf{anc}_{\mathcal{R}}(\mathcal{A}))$. But that means that there is a state substitution $\theta\colon \mathcal{V}(r) \to Q$, a state $q \in Q$, and a final state $q_f \in Q_f$ such that $C[r\sigma] \to_{\Delta_m}^* C[r\theta] \to_{\Delta_m}^* C[q] \to_{\Delta_m}^* q_f$. From the construction using rule (†) we know that there is a transition $f(q_1, \ldots, q_n) \to q \in \Delta_m$ such that $q_i = \ell_i\theta$ if $\ell_i \in \mathcal{V}(r)$ and $q_i = [\ell_i]$ otherwise. If $\ell_i \in \mathcal{V}(r)$ then $\ell_i\sigma \to_{\Delta_m}^+ \ell_i\theta$ and otherwise $\ell_i\sigma \to_{\Delta_m}^+ [\ell_i]$ for all $1 \leqslant i \leqslant n$. Hence in both cases $\ell_i\sigma \to_{\Delta_m}^+ q_i$. But then we can construct the sequence $s = C[f(\ell_1\sigma, \ldots, \ell_n\sigma)] \to_{\Delta_m}^* C[f(q_1, \ldots, q_n)] \to_{\Delta_m} C[q] \to_{\Delta_m}^* q_f$ and hence $s \in L(\mathsf{anc}_{\mathcal{R}}(\mathcal{A}))$.

For the other direction we prove the following two properties for all sequences $s \to_{\Delta_m}^+ q$:

1. If $q = [t]$ for some subterm of a left-hand side of a rule in $\mathcal{R}$ then $s \in (\to_{\mathcal{R}}^*)[\Sigma(t)]$.
2. If $q \in Q_f$ then $s \in (\to_{\mathcal{R}}^*)[L(\mathcal{A})]$.

The proof for both properties works along the same lines. We sketch the one for the first property here. From the construction using rule (†) we know that $s \to_{\Delta_k}^+ [t]$ for some $k \geqslant 0$. We proceed by induction on k. If $k = 0$ then $s \to_{\Delta_0}^+ [t]$. By construction of $\mathcal{A}_0$ and Lemma 10 we have $s \in \Sigma(t)$ and hence also $s \in (\to_{\mathcal{R}}^*)[\Sigma(t)]$. Now assume that $k = k' + 1$ for some $k' \geqslant 0$. By induction hypothesis (IH_0) $s \to_{\Delta_{k'}}^+ [t]$ implies $s \in (\to_{\mathcal{R}}^*)[\Sigma(t)]$ for all terms s and t. Consider the set $\Delta_{k'+1} \setminus \Delta_{k'}$ of transitions which were newly added in $\Delta_{k'+1}$. We continue by a second induction on the size of $\Delta_{k'+1} \setminus \Delta_{k'}$. If it is empty we have a $\Delta_{k'}$-sequence and may simply close the proof with an application of IH_0. Otherwise we have some set Δ and transition $\rho\colon f(q_1, \ldots, q_n) \to q'$ that was created from some rule $\ell \to r \in \mathcal{R}$ with $\ell = f(\ell_1, \ldots, \ell_n)$ and the sequence $r\theta \to_{\Delta'_k}^* q'$ by an application of rule (†) such that $\{\rho\} \uplus \Delta \subseteq \Delta_{k'+1} \setminus \Delta_{k'}$. The second induction hypothesis (IH_1) is if $s \to_{\Delta \cup \Delta_{k'}}^+ [t]$ then $s \in (\to_{\mathcal{R}}^*)[\Sigma(t)]$. Let m denote the number of steps that use ρ. We continue by a third induction on m. If $m = 0$ the sequence from s to $[t]$ only used transitions in $\Delta \cup \Delta_{k'}$ and using IH_1 we are done. Otherwise there is some $m' \geqslant 0$ such that $m = m' + 1$ and the induction hypothesis (IH_2) is that for all terms s, t if $s \to_{\Delta \cup \Delta_{k'}}^+ [t]$ using ρ only m' times then $s \in (\to_{\mathcal{R}}^*)[\Sigma(t)]$. Now we look at the first step using ρ in the sequence, i.e., $s = D[f(s_1, \ldots, s_n)] \to_{\Delta \cup \Delta_{k'}}^* C[f(q_1, \ldots, q_n)] \to_\rho C[q'] \to_{\Delta_{k'+1}}^* [t]$. Note that from this we get $D[u] \to_{\Delta \cup \Delta_{k'}}^* C[u]$ for all terms u.

Next we define a substitution τ such that $s \to_{\mathcal{R}}^* D[\ell\tau] \to_{\mathcal{R}} D[r\tau] \to_{\Delta \cup \Delta_{k'}}^* C[r\tau] \to_{\Delta \cup \Delta_{k'}}^* C[q']$. This allows us to bypass the ρ-step and so we arrive at a $\Delta_{k'+1}$-sequence from $D[r\tau]$ to $[t]$ containing one less ρ-step as shown in Figure 1. The construction of τ proceeds as follows: We fix $1 \leqslant i \leqslant n$. If ℓ_i is a variable in r define τ_i to be $\{\ell_i \mapsto s_i\}$. Otherwise we

know from the definition of inference rule (†) that $q_i = [\ell_i]$ and $s_i \to^+_{\Delta\cup\Delta_{k'}} [\ell_i]$. From that we have that $s_i \in (\to^*_{\mathcal{R}})[\Sigma(\ell_i)]$ but that means that there is some substitution τ_i such that $s_i \to^*_{\mathcal{R}} \ell_i\tau_i$. Moreover let $\tau' = \{x \mapsto u_x \mid x \in \mathcal{V}(r) \setminus \mathcal{V}(\ell)\}$ where u_x is an arbitrary but fixed ground term such that $u_x \to^*_{\Delta_0} x\theta$.[5] Now let τ be the disjoint union of $\tau_1, \ldots, \tau_n, \tau'$. This substitution is well-defined because ℓ is linear. By construction of τ we get $s \to^*_{\mathcal{R}} D[\ell\tau]$.

Consider a variable x occurring in r. If x also occurs in ℓ we have $x = \ell_i$ for some unique $1 \leqslant i \leqslant n$ because $\mathcal{R}$ is growing. But then by construction of τ_i we get $x\tau = \ell_i\tau_i = s_i$. Moreover from the definition of q_i in inference rule (†) we have $q_i = \ell_i\theta = x\theta$. But then $x\tau \to^+_{\Delta\cup\Delta_{k'}} x\theta$ from $s_i \to^+_{\Delta\cup\Delta_{k'}} q_i$. On the other hand, if x does not occur in ℓ then $x\tau = x\tau'$ and $x\tau' \to^*_{\Delta_0} x\theta$ by construction of τ'. So in both cases $r\tau \to^*_{\Delta\cup\Delta_{k'}} r\theta$. Together with $r\theta \to^*_{\Delta_{k'}} q'$ and $C[q'] \to^*_{\Delta_{k'+1}} [t]$ we may construct the sequence $D[r\tau] \to^+_{\Delta_{k'+1}} q_f$ which uses ρ only m' times. Using IH_2 we arrive at $D[r\tau] \in (\to^*_{\mathcal{R}})[\Sigma(t)]$. Together with $s \to^*_{\mathcal{R}} D[\ell\tau] \to^*_{\mathcal{R}} D[r\tau]$ this means that $s \in (\to^*_{\mathcal{R}})[\Sigma(t)]$ and we are done. ◀

▶ **Lemma 15** (Non-Reachability via anc). *Let $\mathcal{R}$ be a linear and growing TRS over signature $\mathcal{F}$. We may conclude non-reachability of t from s if the following language is empty:*

$$L(\mathcal{A}_{\Sigma(s)} \cap \mathsf{anc}_{\mathcal{R}}(\mathcal{A}_{\Sigma(t)}))$$ ◀

▶ **Example 16** (Infeasibility via anc). Consider the CTRS $\mathcal{R} = \{\mathsf{f}(\mathsf{a}, x) \to \mathsf{a}, \mathsf{f}(\mathsf{b}, x) \to \mathsf{b}, \mathsf{g}(\mathsf{a}, x) \to \mathsf{c} \Leftarrow \mathsf{f}(x, \mathsf{a}) \approx \mathsf{a}, \mathsf{g}(x, \mathsf{a}) \to \mathsf{d} \Leftarrow \mathsf{f}(x, \mathsf{b}) \approx \mathsf{b}, \mathsf{c} \to \mathsf{c}\}$. It has two critical pairs $\mathsf{c} \approx \mathsf{d} \Leftarrow \mathsf{f}(\mathsf{a}, \mathsf{b}) \approx \mathsf{b}, \mathsf{f}(\mathsf{a}, \mathsf{a}) = \mathsf{a}$ and the symmetric one. Since $\mathsf{tcap}(\mathsf{f}(\mathsf{a}, \mathsf{b})) = x \sim \mathsf{b}$ and $\mathsf{tcap}(\mathsf{f}(\mathsf{a}, \mathsf{a})) = x \sim \mathsf{a}$ unification is not sufficient to show infeasibility of these critical pairs. On the other hand, since the underlying TRS $\mathcal{R}_u$ is linear and growing, we may construct the tree automata $\mathcal{A}_{\Sigma(\mathsf{f}(\mathsf{a},\mathsf{b}))}$ and $\mathsf{anc}_{\mathcal{R}_u}(\mathcal{A}_{\Sigma(\mathsf{b})})$. Because the language of the intersection automaton is empty we have shown infeasibility of the condition $\mathsf{f}(\mathsf{a}, \mathsf{b}) \approx \mathsf{b}$ by Lemma 15. So both critical pairs are infeasible.

Moreover, as shown in the following example, anc may be employed to show infeasibility of conditions of a conditional rewrite rule, directly. Such rules can never be used to rewrite and so might as well be removed from a CTRS.

▶ **Example 17** (Infeasible Rules via anc). Consider the CTRS $\mathcal{R} = \{\mathsf{h}(x) \to \mathsf{a}, \mathsf{g}(x) \to x, \mathsf{g}(x) \to \mathsf{a} \Leftarrow \mathsf{h}(x) \approx \mathsf{b}, \mathsf{c} \to \mathsf{c}\}$. The condition of the third rule is infeasible because $\mathsf{h}(x)$ only rewrites to a and not to b. This cannot be shown by unification because $\mathsf{tcap}(\mathsf{h}(x)) = y \sim \mathsf{b} = \mathsf{tcap}(\mathsf{b})$. Fortunately the underlying TRS $\mathcal{R}_u$ is linear and growing and hence we can construct the tree automata $\mathcal{A}_{\Sigma(\mathsf{h}(x))}$ and $\mathsf{anc}_{\mathcal{R}_u}(\mathcal{A}_{\Sigma(\mathsf{b})})$. The language of the intersection automaton is empty and we have shown infeasibility of the condition $\mathsf{h}(x) \approx \mathsf{b}$ by Lemma 15.

Remember that in our setting the right-hand sides of conditions are always linear terms. Hence it is beneficial to start with the ground-instance automaton for the right-hand side of a condition (which in this case is exact) and compute the set of ancestors rather than taking the possibly non-linear left-hand side of a condition, overapproximating the ground-instances and only then computing the descendants of this set.

4.3 Certification

In this section we give an overview of all techniques that are newly supported by our certifier CeTA and what kind of information it requires from a certificate in CPF [14] (short

[5] Since all states in $\mathcal{A}_0$ are accessible we can always find such a term u_x.

for *certification problem format*). Before we come to the special infeasibility condition of Definition 2, we handle the common case where, given a list of conditions c, we are interested in proving $\sigma, n \not\vdash c$ for every substitution σ and level n.

▶ **Lemma 18** (Infeasibility Certificates). *Given $(\mathcal{R}, c)$ consisting of a CTRS $\mathcal{R}$ and a list of conditions $c = s_1 \approx t_1, \ldots, s_k \approx t_k$, infeasibility of c with respect to $\mathcal{R}$ can be certified in one of the following ways:*

1. *Provide two terms s and t with $s \approx t \in c$, and a non-reachability certificate for $(\mathcal{R}_u, s, t)$.*
2. *Provide a function symbol cs of arity n (called a* compound symbol*) together with a non-reachability certificate for $(\mathcal{R}_u, \mathsf{cs}(s_1, \ldots, s_k), \mathsf{cs}(t_1, \ldots, t_k))$.*
3. *For an arbitrary subset c' of c, provide an infeasibility certificate for $(\mathcal{R}, c')$.*
4. *Provide three terms s, t, and u such that $s \approx u$ and $t \approx u$ are equations in c together with a non-joinability certificate for $(\mathcal{R}_u, s, t)$.*

Proof. Note that **3** is obvious and **1** only exists for tool-author convenience but is subsumed by the combination of **2** and **3**. Moreover, **2** follows from the fact that $\mathsf{cs}(s_1, \ldots, s_k)\sigma \not\to^*_{\mathcal{R}_u} \mathsf{cs}(t_1, \ldots, t_k)\tau$ for all σ and τ, implies the existence of at least one $1 \leqslant i \leqslant k$ such that $s_i\sigma \not\to^*_{\mathcal{R}_u} t_i\tau$ for all σ and τ. Finally, for **4**, whenever $s\sigma$ and $t\tau$ are not joinable for arbitrary σ and τ, the existence of μ and n such that $\mu, n \vdash s \approx u, t \approx u$ is impossible. ◀

Note that in **2** we check for non-reachability between left-hand sides and their corresponding right-hand sides, while in **4** we check for non-joinability between two left-hand sides. Thus, while in general non-joinability is more difficult to show than non-reachability, **4** is not directly subsumed by **2**.

▶ **Lemma 19** (Non-Reachability Certificates). *Given $(\mathcal{R}, s, t)$ consisting of a TRS $\mathcal{R}$ and two terms s and t, $\mathcal{R}$-non-reachability of t from s can be certified in one of the following ways:*

1. *Indicate that $\mathsf{tcap}(s)$ does not unify with t.*
2. *Provide a TRS $\mathcal{R}'$ such that for each $\ell \to r \in \mathcal{R}$ there is $\ell' \to r' \in \mathcal{R}'$ and a substitution σ with $\ell = \ell'\sigma$ and $r = r'\sigma$, together with a non-reachability certificate for $(\mathcal{R}', s, t)$.*
3. *Provide a non-reachability certificate for $(\mathcal{R}^{-1}, t, s)$.*
4. *Make sure that $\mathcal{R}$ is linear and growing and provide a finite signature $\mathcal{F}$ and two constants a and $\Box$ such that $\mathsf{a} \in \mathcal{F}$ but $\Box \notin \mathcal{F}$, together with a tree automaton $\mathcal{A}$ that is an overapproximation of $\mathsf{anc}_{\mathcal{R}}(\mathcal{A}_{\Sigma(t)})$ and satisfies $L(\mathcal{A}_{\Sigma(s)} \cap \mathcal{A}) = \varnothing$.*

Proof. If $\mathsf{tcap}_{\mathcal{R}}(s) \not\sim t$, then **1** holds by Lemma 7. Further note that $\to_{\mathcal{R}} \subseteq \to_{\mathcal{R}'}$ and thus **2** is immediate. Moreover, **3** is obvious, leaving us with **4**. From a certification perspective this is the most interesting case. To begin with, there are two reasons why we do not want to repeat the full construction of anc inside CeTA. Firstly, we would unnecessarily repeat an operation with at least exponential worst-case complexity. Secondly, a fully-verified executable algorithm is not even part of our formalization, instead we heavily rely on inductive definitions.[6] In CeTA we check that $\mathcal{A}$ is an overapproximation of $\mathsf{anc}_{\mathcal{R}}(\mathcal{A}_{\Sigma(t)})$ as follows: firstly, we ensure that $\mathcal{A}$ does not contain epsilon transitions, that $[t]$ is included in the final states of $\mathcal{A}$, and that Δ_t as well as the matching rules with respect to the signature $\mathcal{F}$ are part of the transitions of $\mathcal{A}$; secondly, we check that $\mathcal{A}$ is closed with respect to inference rule (†). Since $\mathcal{A}_{\Sigma(s)}$ is an overapproximation of $\Sigma(s)$ and by the required conditions together with

[6] While turning the existing inductive definitions into executable recursive functions would definitely be possible, we stress that this is not necessary.

Theorem 14, $L(\mathcal{A})$ overapproximates $[\to_{\mathcal{R}}^*](\Sigma(t))$, we can conclude $\Sigma(s) \cap [\to_{\mathcal{R}}^*](\Sigma(t)) = \varnothing$. Thus there are no *ground* substitutions σ and τ such that $s\sigma, t\tau \in \mathcal{T}(\mathcal{F})$ and $s\sigma \to_{\mathcal{R}}^* t\tau$. In order to conclude that the same holds true for *arbitrary* substitutions (not necessarily restricted to $\mathcal{F}$), we rely on an earlier result [13] that implies that whenever $s\sigma \to_{\mathcal{R}}^* t\tau$ for arbitrary σ and τ and $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ then there are σ' and τ' such that $s\sigma', t\tau' \in \mathcal{T}(\mathcal{F})$ and $s\sigma' \to_{\mathcal{R}}^* t\tau'$. ◀

Note that **2** allows us to certify the linear growing approximation of a TRS without actually having to formalize it in Isabelle/HOL. More specifically, whenever $\mathcal{R}'$ is the result of applying the linear growing approximation to $\mathcal{R}$, then the corresponding certificate will pass **2** and $\mathcal{R}'$ will be linear and growing in the check for **4**; otherwise **4** will fail.

▶ **Lemma 20** (Non-Joinability Certificates). *Given $(\mathcal{R}, s, t)$ consisting of a TRS $\mathcal{R}$ and two term s and t, $\mathcal{R}$-non-joinability of s and t can be certified in one of the following ways.*

1. *Indicate that $\mathsf{tcap}(s)$ does not unify with $\mathsf{tcap}(t)$.*
2. *If at least one of the terms, say t, is a ground $\mathcal{R}$-normal form provide a non-reachability certificate for $(\mathcal{R}, s, t)$.*

Proof. We prove **1** by Lemma 8 and **2** by Lemma 7 since non-joinability reduces to non-reachability when one of the terms is an $\mathcal{R}$-normal form. ◀

▶ **Lemma 21** (Ao-Infeasibility Certificates). *Given $(\mathcal{R}, c_1, c_2)$ consisting of a CTRS $\mathcal{R}$ fulfilling all syntactic requirements of Theorem 5 and two lists of conditions c_1 and c_2, infeasibility with respect to almost orthogonality can be certified in one of the following ways:*

1. *Provide an infeasibility certificate for $(\mathcal{R}, c)$ where c is the concatenation of c_1 and c_2.*
2. *Provide three terms s, t and u such that $s \approx t$ is an equation in c_1 and $s \approx u$ an equation in c_2, together with a non-joinability certificate for $(\mathcal{R}_u, t, u)$.*

Proof. While **1** follows from Lemma 18, in **2** we make use of the level-commutation assumption of Definition 2 to deduce non-meetability of t and u from non-joinability of t and u. ◀

4.4 Comparison

For our main use case, Theorem 5, we are restricted to left-linear CTRSs (via almost orthogonality) and linear right-hand sides of conditions (via right-stability). The latter also holds for right-hand sides that are combined by a compound symbol (again by right-stability). We show that in this setting anc subsumes tcap (at least in theory and for the forward direction).

▶ **Lemma 22.** *Let $\mathcal{R}$ be a left-linear CTRS and t a linear term. If tcap can show non-reachability of t from s, then so can anc.*

Proof. We proof the contrapositive and assume that anc cannot show non-reachability. Moreover, let $\mathcal{R}'$ denote the result of applying the linear growing approximation to $\mathcal{R}_u$. Then there is some term u such that $u \in L(\mathcal{A}_{\Sigma(s)})$ and $u \in L(\mathsf{anc}_{\mathcal{R}'}(\mathcal{A}_{\Sigma(t)}))$. Since t is linear and $\mathcal{R}'$ is linear and growing the latter implies that $u \in (\to_{\mathcal{R}'}^*)[\Sigma(t)]$ and thus $u \to_{\mathcal{R}'}^* t\tau$ for some substitution τ. By Lemma 7, this means that $t\tau \in [\![\mathsf{tcap}_{\mathcal{R}'}(u)]\!]$. Since $u \in L(\mathcal{A}_{\Sigma(s)})$, it is clearly the case that $u \in \Sigma(\mathsf{ren}(s))$ and thus $u = \mathsf{ren}(s)\sigma$ for some substitution σ, where ren denotes an arbitrary linearization of s. Moreover $[\![\mathsf{tcap}_{\mathcal{R}'}(u)]\!] \subseteq [\![\mathsf{tcap}_{\mathcal{R}'}(\mathsf{ren}(s))]\!] = [\![\mathsf{tcap}_{\mathcal{R}'}(s)]\!]$. Together with $t\tau \in [\![\mathsf{tcap}_{\mathcal{R}'}(u)]\!]$ from above, we obtain $t\tau \in [\![\mathsf{tcap}_{\mathcal{R}'}(s)]\!]$. However, tcap does only consider the left-hand sides of rules, which are the same in $\mathcal{R}'$ and $\mathcal{R}_u$, thus also $t\tau \in [\![\mathsf{tcap}_{\mathcal{R}_u}(s)]\!]$ which implies $\mathsf{tcap}_{\mathcal{R}_u}(s) \sim t$. ◀

Table 1 82 right-stable, extended properly oriented, and oriented 3-CTRSs.

	ConCon			
	uncertified	certified 2	certified 2+3	certified+
confluent	47	23	32	35
non-confluent	15	-	-	-
don't know	20	59	50	47

If we also consider the reverse direction, that is, checking if $t \to^*_{\mathcal{R}_u^{-1}} s$ for some condition $s \approx t$ in Theorem 5, then tcap may well succeed where anc fails, as shown by the next example.

▶ **Example 23** (anc vs. tcap)**.** The oriented 3-CTRS $\mathcal{R} = \{\mathsf{g}(x) \to \mathsf{f}(x,x), \mathsf{g}(x) \to \mathsf{g}(x) \Leftarrow \mathsf{g}(x) \approx \mathsf{f}(\mathsf{a},\mathsf{b})\}$ is right-stable and extended properly oriented. It has two symmetric CCPs of the form $\mathsf{f}(x,x) \approx \mathsf{g}(x) \Leftarrow \mathsf{g}(x) \approx \mathsf{f}(\mathsf{a},\mathsf{b})$. The underlying TRS $\mathcal{R}_u$ is not linear and growing, so if we want to apply anc we have to apply the linear growing approximation, resulting in $\mathcal{R}' = \{\mathsf{g}(x) \to \mathsf{f}(x,y), \mathsf{g}(x) \to \mathsf{g}(x)\}$. But then anc is not able to show infeasibility since the language of $\mathcal{A}_{\Sigma(\mathsf{g}(x))} \cap \mathsf{anc}_{\mathcal{R}'}(\mathcal{A}_{\Sigma(\mathsf{f}(\mathsf{a},\mathsf{b}))})$ is not empty and also for the reverse direction $\mathcal{A}_{\Sigma(\mathsf{f}(\mathsf{a},\mathsf{b}))} \cap \mathsf{anc}_{\mathcal{R}'^{-1}}(\mathcal{A}_{\Sigma(\mathsf{g}(x))})$ we get a non-empty language. On the other hand using the reversed underlying system $\mathcal{R}_u^{-1} = \{\mathsf{f}(x,x) \to \mathsf{g}(x), \mathsf{g}(x) \to \mathsf{g}(x)\}$ we have that $\mathsf{tcap}_{\mathcal{R}_u^{-1}}(\mathsf{f}(\mathsf{a},\mathsf{b})) = \mathsf{f}(\mathsf{a},\mathsf{b}) \not\sim \mathsf{g}(x)$. So in this case tcap succeeds where anc fails.

5 Conclusion and Future Work

We have not only produced several thousand ($\sim$6600) lines of proof documents, but also refined and extended an earlier result which allows us to certify confluence proofs for a larger class of CTRSs. Moreover, a new method to check for non-reachability between terms has been added, which (at least theoretically) further expands this class. The certifier CeTA has been updated to handle all new certificates and the confluence tool ConCon has been extended to use the new results and generate certifiable output for them.

Our formalization exposed an error in the exact tree automata completion procedure implemented in ConCon which is now corrected.

Experiments We shortly examine ConCon's ability to provide certifiable proofs before and after the implementation of the presented results. Our testbed comprises 82 right-stable, extended properly oriented, and oriented 3-CTRSs taken from the confluence problems database (Cops).[7] Note that only 52 of these 82 systems have at least one CCP and hence are amenable to the improved infeasibility methods.

All in all ConCon implements three criteria for checking confluence of oriented CTRSs.

1. Strongly deterministic, quasi-decreasing 3-CTRSs are confluent if all CPs are joinable [1].
2. Theorem 5 from above.
3. A deterministic 3-CTRS is confluent if its unraveling is left-linear and confluent [20].

These are accompanied by some infeasibility-methods (including the ones presented). So far criteria **2** and **3** have been formalized.

[7] `http://cops.uibk.ac.at`

In Table 1 we summarize our findings. The column labeled *uncertified* contains the results of unleashing ConCon using all (possibly not yet certifiable) criteria for confluence and infeasibility it implements. The next two columns labeled *certified 2* and *certified 2+3*, respectively, show the numbers when only using certifiable methods already present in ConCon before the modification. In *certified 2* we only used the syntactic method from Suzuki et al. Column *certified 2+3* combines the latter with method **3** employing unravelings. Finally, the column labeled *certified+* gives the results for the current version of ConCon including the newly implemented certifiable methods. One interesting point is that method **2** completely subsumes **3** in *certified+*. What we can see from the table is the following: We were able to increase the applicability of **2** by 12 systems. But if we also take into account method **3** we only gain 3 certified proofs. All together we can certify 35 out of 47 confluence proofs. So far we have not worked on formalizing the used non-confluence criteria, hence none of the non-confluence proofs can be certified.

Concerning Examples 16 and 17 it is interesting to note that criterion **1** does not apply, because both systems are non-terminating and also criterion **3** does not apply, because the unraveled system is non-confluent in both cases. So **2** is the only of the three methods that can handle both examples.

Finally, we believe that the small gain of only 3 certified proofs in total, has to be taken with a grain of salt. On the one hand, since the number of CTRSs in Cops is rather low. On the other hand, and more importantly, since the class of CTRSs to which Theorem 5 potentially applies, closely corresponds to what is actually allowed in functional and logic programs.

Future Work For the moment we have restricted our attention to confluence of CTRSs in general and Theorem 5 in particular. However, we provide general techniques for the certification of non-reachability and non-joinability; those should be applicable also in other areas like certification of dependency graph approximations for termination of TRSs, non-confluence of TRSs, and more hypothetically the correctness of protocols.

As for the applicability of our method to the certification of dependency graph approximations, we conducted some preliminary experiments. Here, a *potential edge* consists of two pairs of terms (s, t) and (u, v) where the goal is to prove non-reachability of u from t (since then it does not turn into an actual edge, which might lead to an easier termination proof). Since at the time of writing the only certifiable way of handling dependency graphs in current termination tools is a tcap-based estimation, we started with all the 542428 potential edges obtained from the *termination problem database*[8] that cannot already be handled by tcap. Of those, 129599 are trivially shown to be actual edges via unification (i.e., no rewrite steps are necessary). Out of the remaining 412829 potential edges, we were able to show non-reachability for 10217/24291/43364 when using a timeout of 1/3/10 seconds per edge.

Another direction of future work will be to formalize criterion **1** and also extend our formalization to non-confluence proofs.

Acknowledgments. We thank Bertram Felgenhauer for producing at least 1.5 of our examples. We also thank the anonymous reviewers for their constructive and helpful comments. Further thanks goes to the Austrian Science Fund (FWF project P27502) for supporting our work. Note that the authors are given in alphabetical order.

[8] http://termination-portal.org/wiki/TPDB

References

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

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

3 Bertram Felgenhauer and René Thiemann. Reachability analysis with state-compatible automata. In *Proceedings of the 8th International Conference on Language and Automata Theory and Applications*, volume 8370 of *Lecture Notes in Computer Science*, pages 347–359. Springer, 2014. 10.1007/978-3-319-04921-2_28.

4 Thomas Genet. Decidable approximations of sets of descendants and sets of normal forms. In *Proceedings of the 9th International Conference on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 151–165. Springer, 1998. 10.1007/BFb0052368.

5 Thomas Genet and Valerié Viet Triem Tong. Reachability analysis of term rewriting systems with timbuk. In *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 2250 of *Lecture Notes in Computer Science*, pages 695–706. Springer, 2001. 10.1007/3-540-45653-8_48.

6 Jürgen Giesl, René Thiemann, and Peter Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proceedings of the 5th International Workshop on Frontiers of Combining Systems*, volume 3717 of *Lecture Notes in Computer Science*, pages 216–231. Springer, 2005. 10.1007/11559306_12.

7 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.

8 Florent Jacquemard. Decidable approximations of term rewriting systems. In *Proceedings of the 7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 362–376. Springer, 1996. 10.1007/3-540-61464-8_65.

9 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. 10.1007/3-540-45949-9.

10 Enno Ohlebusch. *Advanced Topics in Term Rewriting*. Springer, 2002.

11 Christian Sternagel and Aart Middeldorp. Root-labeling. In *Proceedings of the 19th International Conference on Rewriting Techniques and Applications*, volume 5117 of *Lecture Notes in Computer Science*, pages 336–350. Springer, 2008. 10.1007/978-3-540-70590-1_23.

12 Christian Sternagel and Thomas Sternagel. Level-confluence of 3-CTRSs in Isabelle/HOL. In Tiwari and Aoto [18]. `http://www.csl.sri.com/users/tiwari/iwc2015/iwc2015.pdf`.

13 Christian Sternagel and René Thiemann. Signature extensions preserve termination - an alternative proof via dependency pairs. In *Proceedings of the 19h EACSL Annual Conference on Computer Science Logic*, volume 6247 of *Lecture Notes in Computer Science*, pages 514–528. Springer, 2010. 10.1007/978-3-642-15205-4_39.

14 Christian Sternagel and René Thiemann. The Certification Problem Format. In *Proceedings of the 11th Workshop on User Interfaces for Theorem Provers*, pages 61–72, 2014. 10.4204/EPTCS.167.8.

15 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. 10.1007/978-3-319-08918-8_31.

16 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. 10.1007/3-540-59200-8_56.

17 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. 10.1007/978-3-642-03359-9_31.

18 Ashish Tiwari and Takahito Aoto, editors. *Proceedings of the 4th International Workshop on Confluence*, 2015. `http://www.csl.sri.com/users/tiwari/iwc2015/iwc2015.pdf`.

19 Makarius Wenzel. System description: Isabelle/jEdit in 2014. In *Proceedings of the 11th Workshop on User Interfaces for Theorem Provers*, pages 84–94, 2014. 10.4204/EPTCS.167.10.

20 Sarah Winkler and René Thiemann. Formalizing soundness and completeness of unravelings. In *Proceedings of the 10th International Workshop on Frontiers of Combining Systems*, volume 9322 of *Lecture Notes in Computer Science*, pages 239–255. Springer, 2015. 10.1007/978-3-319-24246-0_15.