Certifying Confluence of Quasi-Decreasing Strongly Deterministic Conditional Term Rewrite Systems*

Christian Sternagel and Thomas Sternagel

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

Abstract. We formalize a confluence criterion for the class of quasi-decreasing strongly deterministic conditional term rewrite systems in Isabelle/HOL: confluence follows if all conditional critical pairs are joinable. However, quasi-decreasingness, strong determinism, and joinability of conditional critical pairs are all undecidable in general. Therefore, we also formalize sufficient criteria for those properties, which we incorporate into the general purpose certifier CeTA as well as the confluence checker ConCon for conditional term rewrite systems.

1 Introduction

In the area of equational reasoning *canonicity*—that is, termination together with confluence—plays an important role towards deciding equations with respect to equational theories and for avoiding redundant computations and nondeterminism. In the presence of powerful methods and tools for proving termination [1, 12, 17, 18, 31, 33], the remaining issue is to also establish confluence.

For plain term rewrite systems (TRSs), this issue was settled early on by Newman's Lemma [22], stating that *any terminating relation is confluent iff it is locally confluent*. Then, by the Critical Pair Lemma [15, 16], *local confluence reduces to joinability of all critical pairs*, which in turn, can be decided by exhaustive rewriting, due to termination.

However, for many applications plain TRSs are either inconvenient or not expressible enough, leading to several extensions of the base formalism. The one we are interested in here is *conditional term rewriting*. Two prominent areas where conditional rewriting is employed are the rewriting engines of modern proof assistants (like Isabelle's simplifier [23]) and functional(-logic) programming with where-clauses (like Haskell [21] and Curry [2]).

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

Example 1. As a first example, consider the following Haskell program, which computes the minimum of a given list of natural numbers. Below, we give a straightforward translation into a conditional term rewrite system $\mathcal{R}_{\mathsf{min}}$ with six rules that serves as our running example.

```
min (x:[])             = x
min (x:xs) | x < y     = x
           | otherwise = y
           where y = min xs
```

$$\mathsf{min}(\mathsf{cons}(x,\mathsf{nil})) \to x \tag{1}$$
$$\mathsf{min}(\mathsf{cons}(x,xs)) \to x \Leftarrow \mathsf{min}(xs) \approx y,\ x < y \approx \mathsf{true} \tag{2}$$
$$\mathsf{min}(\mathsf{cons}(x,xs)) \to y \Leftarrow \mathsf{min}(xs) \approx y,\ x < y \approx \mathsf{false} \tag{3}$$
$$x < 0 \to \mathsf{false} \tag{4}$$
$$0 < \mathsf{s}(y) \to \mathsf{true} \tag{5}$$
$$\mathsf{s}(x) < \mathsf{s}(y) \to x < y \tag{6}$$

Issue. Alas, even in the presence of termination, confluence is in general still undecidable for conditional term rewrite systems (CTRSs). While Avenhaus and Loría-Sáenz [4] gave a critical pair criterion for quasi-reductive and strongly deterministic CTRSs: *joinability of all conditional critical pairs (CCPs) implies confluence*; joinability of CCPs is undecidable in general, due to the inherent complexities of conditional rewriting. This lead to the development of sufficient criteria that are implemented in confluence tools for CTRSs like ConCon [27].

Such tools ultimately aim at automatic (program) verification. But they are programs themselves, and rather complex ones at that. So why should we trust them? This consideration lead to the introduction of certification in the area of term rewriting [8,9,30]. Here, the output of an automated tool—the certificate—is checked by a formally verified certifier that is code generated from a formalization inside a proof assistant. This approach was already quite successful for termination and confluence of TRSs, where state-of-the-art certifiers cover more than 80% of all generated certificates in the respective tool competitions [3,11].

For confluence of CTRSs, not so many techniques are known and even less are formalized and certifiable.

Contribution and Summary. In Section 3, we formalize the CCP criterion of Avenhaus and Loría-Sáenz [4, Theorem 4.2] (AL for short) and, based on our earlier work [28], strengthen it from quasi-reductivity to quasi-decreasingness.

Moreover, to certify confluence of quasi-decreasing and strongly deterministic CTRSs, we formalize the variant of AL replacing joinability of all CCPs by the requirement that every CCP is either unfeasible[1] or context-joinable (Section 4). Both unfeasibility and context-joinability rely on the notion of contextual rewriting, which we formalize together with the crucial lemma that *contextual rewriting implies conditional rewriting for satisfying substitutions*, a result that was stated without proof by Avenhaus and Loría-Sáenz [4, Lemma 4.2]. Unfeasibility further employs strong irreducibility, which like strong determinism is an undecidable property. Thus, we formalize these two properties together with the two sufficient and decidable criteria of absolute irreducibility and absolute determinism.

[1] This is a technical term (see Definition 3) introduced by Avenhaus and Loría-Sáenz [4] and should not be confused with *infeasibility*.

Along the way, we identify and fix some problems in proofs and definitions (of absolute irreducibility, contextual rewriting, and unfeasibility) and provide a (not entirely obvious) proof for [4, Lemma 4.2]. We further adapt the original proof of AL to the new definitions and extend it by infeasibility.

In Section 5, we point out some challenges concerning certification. Then, in Section 6, we give an overview of all the check functions that are new in CeTA. In Section 7, we evaluate our contribution through experiments on the confluence problems database (Cops) [10]. Finally, we conclude in Section 8.

This work substantially contributes to the greater effort of making ConCon 100% certifiable by formalizing all of its methods. Our formalization is part of the formal IsaFoR library and supported by version 2.29 of its accompanying certifier CeTA [30]. Both IsaFoR and CeTA are freely available online at

http://cl-informatik.uibk.ac.at/isafor/

2 Preliminaries

We assume familiarity with the basic notions of (conditional) term rewriting [5,24], but shortly recapitulate terminology and notation that we use in the remainder. Given an arbitrary binary relation $\to_\alpha$, we write ${}_\alpha\!\leftarrow$, $\to_\alpha^+$, and $\to_\alpha^*$ for its *inverse*, its *transitive closure*, and its *reflexive transitive closure*, respectively. We use $\mathcal{V}(\cdot)$ to denote the set of variables occurring in a given list of syntactic objects, like terms, rules, etc. Given a term t, we write $\mathcal{P}\mathsf{os}(t)$ for the *set of positions* in t and $t|_p$ with $p \in \mathcal{P}\mathsf{os}(t)$ for the subterm of t at position p. We write $s[t]_p$ for the result of replacing $s|_p$ by t in s. We say that terms s and t *unify*, written $s \sim t$, if $s\sigma = t\sigma$ for some substitution σ. A substitution σ is $\mathcal{R}$-*normalized* if $\sigma(x)$ is an $\mathcal{R}$-normal form for all variables x. We call a bijective variable substitution π a *renaming* or *permutation*, and denote its inverse by π^-. For two substitutions σ, τ and a set of variables V we write $\sigma = \tau\ [V]$ if $\sigma(x) = \tau(x)$ for all $x \in V$. We write $\sigma\tau$ for the composition of σ and τ where $(\sigma\tau)(x) = \sigma(x)\tau$. A term t is *strongly* $\mathcal{R}$-*irreducible* if $t\sigma$ is an $\mathcal{R}$-normal form for all $\mathcal{R}$-normalized substitutions σ. A *strongly deterministic oriented 3-CTRS (SDTRS)* $\mathcal{R}$ is a set of conditional rewrite rules of the shape $\ell \to r \Leftarrow c$ where ℓ and r are terms and c is a possibly empty sequence of pairs of terms (called *conditions*) $s_1 \approx t_1, \ldots, s_n \approx t_n$, satisfying: ℓ is not a variable (CTRS), $\mathcal{V}(r) \subseteq \mathcal{V}(\ell, c)$ (3-CTRS), $\mathcal{V}(s_i) \subseteq \mathcal{V}(\ell, t_1, \ldots, t_{i-1})$ for all $1 \leqslant i \leqslant n$ (DTRS), and t_i is strongly $\mathcal{R}$-irreducible for all $1 \leqslant i \leqslant n$ (SDTRS). We sometimes label rules like $\rho : \ell \to r \Leftarrow c$. For a rule $\rho : \ell \to r \Leftarrow c$ of an SDTRS $\mathcal{R}$ the set of *extra variables* is defined as $\mathcal{EV}(\rho) = \mathcal{V}(c) - \mathcal{V}(\ell)$. Given an SDTRS $\mathcal{R}$, extended TRSs $\mathcal{R}_n$ are inductively defined for each level $n \geqslant 0$

$$\mathcal{R}_0 = \varnothing$$
$$\mathcal{R}_{n+1} = \{\ell\sigma \to r\sigma \mid \ell \to r \Leftarrow c \in \mathcal{R} \text{ and } s\sigma \to_{\mathcal{R}_n}^* t\sigma \text{ for all } s \approx t \in c\}$$

where $\to_{\mathcal{R}_n}$ denotes the rewrite relation of the (unconditional) TRS $\mathcal{R}_n$, that is, the smallest relation $\to$ satisfying $t[\ell\sigma]_p \to t[r\sigma]_p$ whenever $\ell \to r$ is a rule in $\mathcal{R}_n$. We write $s \to_{\mathcal{R},n} t$ if we have $s \to_{\mathcal{R}_n} t$ and $s \to_{\mathcal{R}} t$ whenever $s \to_{\mathcal{R}_n} t$ for some

$n \geqslant 0$. We say that a substitution σ *satisfies* a sequence of conditions c if for all $s \approx t \in c$ we have $s\sigma \to_{\mathcal{R}}^* t\sigma$. 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 $\ell_1|_p$ is not a variable 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) $r_1\mu \approx \ell_1[r_2]_p\mu \Leftarrow c_1\mu, c_2\mu$.

Example 2. The CTRS $\mathcal{R}_{\mathsf{min}}$ from Example 1 has 6 CCPs, 3 modulo symmetry:

$$x \approx x \Leftarrow \mathsf{min}(\mathsf{nil}) \approx y,\ x < y \approx \mathsf{true} \qquad (1,2)$$
$$x \approx y \Leftarrow \mathsf{min}(\mathsf{nil}) \approx y,\ x < y \approx \mathsf{false} \qquad (1,3)$$
$$x \approx y \Leftarrow \mathsf{min}(xs) \approx z,\ x < z \approx \mathsf{true},\ \mathsf{min}(xs) \approx y,\ x < y \approx \mathsf{false} \qquad (2,3)$$

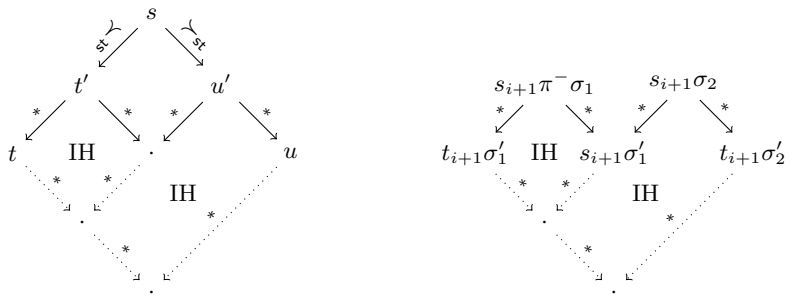
A CCP $u \approx v \Leftarrow c$ is said to be *infeasible* if its conditions are not satisfied by any substitution. Moreover, a CCP is *joinable* if $u\sigma \downarrow_{\mathcal{R}} v\sigma$ for all substitutions σ that satisfy c. The topmost part of a term that does not change under rewriting (sometimes called its "cap") can be approximated for example by the tcap function [13]. 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 $\mathcal{R}$, and a fresh variable, otherwise. It is well known that $\mathsf{tcap}(s) \not\sim t$ implies non-reachability of t from s. We denote the proper superterm relation by $\rhd$ and define $\succ_{\mathsf{st}} = (\succ \cup \rhd)^+$ for any order $\succ$. If $\succ$ is a reduction order, then an SDTRS $\mathcal{R}$ is *quasi-reductive with respect to* $\succ$ if for every substitution σ and every rule $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n$ in $\mathcal{R}$ we have that $s_j\sigma \succeq t_j\sigma$ for $1 \leqslant j < i$ implies $\ell\sigma \succ_{\mathsf{st}} s_i\sigma$ for all $1 \leqslant i \leqslant n$, and $s_j\sigma \succeq t_j\sigma$ for $1 \leqslant j \leqslant n$ implies $\ell\sigma \succ r\sigma$. An SDTRS $\mathcal{R}$ is *quasi-decreasing* if there exists a well-founded order $\succ$ such that $\succ = \succ_{\mathsf{st}}$, $\to_{\mathcal{R}} \subseteq \succ$, and for all rules $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n$ in $\mathcal{R}$, all substitutions σ, and $1 \leqslant i \leqslant n$, if $s_j\sigma \to_{\mathcal{R}}^* t_j\sigma$ for all $1 \leqslant j < i$ then $\ell\sigma \succ s_i\sigma$. Quasi-reductivity implies quasi-decreasingness—a fact that is available in IsaFoR.

3 Confluence of Quasi-Decreasing SDTRSs

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

Theorem 1 ([4, Theorem 4.1]). *Let the SDTRS $\mathcal{R}$ be quasi-reductive with respect to $\succ$. Then $\mathcal{R}$ is confluent iff all CCPs are joinable.*

That all CCPs of a CTRS $\mathcal{R}$ (no need for strong determinism or quasi-reductivity) are joinable if $\mathcal{R}$ is confluent is straightforward. Thus, we concentrate on the other direction. Our formalization is quite close to the original proof. The good news is: we could not find any errors (besides typos) in the original proof but as is often the case with formalizations there are places where the paper proof is vague or does not spell out the technical details in favor of readability. For example, we heavily rely on an earlier formalization of permutations [14] in order to formalize



(a) (b)

Fig. 1

variants of rules up to renaming. In contrast, the change from quasi-reductivity to quasi-decreasingness was rather smooth.

Below, we give our main theorem and walk through the formalized proof.

Theorem 2. *Let the SDTRS $\mathcal{R}$ be quasi-decreasing with respect to $\succ$. Then $\mathcal{R}$ is confluent if all CCPs are joinable.*

Proof. Assume that all critical pairs are joinable. We consider an arbitrary peak $t \,{}^*_{\mathcal{R}}\!\leftarrow s \to^*_{\mathcal{R}} u$ and prove $t \downarrow_{\mathcal{R}} u$ by well-founded induction with respect to $\succ_{\mathsf{st}}$.

By induction hypothesis (IH) we have that for all terms t_0, t_1, t_2 such that $s \succ_{\mathsf{st}} t_0$ and $t_1 \,{}^*_{\mathcal{R}}\!\leftarrow t_0 \to^*_{\mathcal{R}} t_2$ there exists a join $t_1 \to^*_{\mathcal{R}} \cdot \,{}^*_{\mathcal{R}}\!\leftarrow t_2$.

If $s = t$ or $s = u$ then t and u are trivially joinable and we are done. So we may assume that the peak contains at least one step in each direction: $t \,{}^*_{\mathcal{R}}\!\leftarrow t' \,{}_{\mathcal{R}}\!\leftarrow s \to_{\mathcal{R}} u' \to^*_{\mathcal{R}} u$.

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

1. a CCP which is joinable by assumption or we have

2. a root-overlap of variants of the same rule. Unlike in the unconditional case this could lead to non-joinability of the ensuing critical pair because of the extra-variables in the right-hand sides of conditional rules. We have $\rho_1\pi = \rho_2$ for some permutation π. Moreover, $s = \ell_1\sigma_1 = \ell_2\sigma_2$ and we have

$$\pi^{-}\sigma_1 = \sigma_2\ [\mathcal{V}(\ell_2)] \tag{7}$$

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

$$x\pi^{-}\sigma_1 \xrightarrow[\mathcal{R}]{*} x\sigma'_1 \text{ and } x\sigma_2 \xrightarrow[\mathcal{R}]{*} x\sigma'_2 \text{ for all variables } x. \tag{8}$$

We prove $x\sigma'_1 = x\sigma'_2$ for $x \in \mathcal{EV}(\rho_2)$ by an inner induction on the length of $c_2 = s_1 \approx t_1, \ldots, s_n \approx t_n$. If ρ_2 has no conditions this holds vacuously because there are no extra variables. In the step case the inner induction hypothesis (IH_i) is that $x\sigma'_1 = x\sigma'_2$ for $x \in \mathcal{V}(s_1, t_1, \ldots, s_i, t_i) - \mathcal{V}(\ell_2)$ and we have to show that $x\sigma'_1 = x\sigma'_2$ for $x \in \mathcal{V}(s_1, t_1, \ldots, s_{i+1}, t_{i+1}) - \mathcal{V}(\ell_2)$. If $x \in \mathcal{V}(s_1, t_1, \ldots, s_i, t_i, s_{i+1})$ we are done by the IH_i and strong determinism of $\mathcal{R}$. So assume $x \in \mathcal{V}(t_{i+1})$. From strong determinism of $\mathcal{R}$, (7), (8), and the IH_i we have that $y\sigma'_1 = y\sigma'_2$ for all $y \in \mathcal{V}(s_{i+1})$ and thus $s_{i+1}\sigma'_1 = s_{i+1}\sigma'_2$. With this we can find a join between $t_{i+1}\sigma'_1$ and $t_{i+1}\sigma'_2$ by applying the IH twice as shown in Figure 1b. Since t_{i+1} is strongly irreducible and σ'_1 and σ'_2 are normalized, this yields $t_{i+1}\sigma'_1 = t_{i+1}\sigma'_2$ and thus $x\sigma'_1 = x\sigma'_2$.
3. We are left with the case that there is a variable position q' in ℓ_1 such that $q = q'r'$ for some position r'. Let x be the variable $\ell_1|_{q'}$. Then $x\sigma_1|_{r'} = \ell_2\sigma_2$, which implies $x\sigma_1 \to^*_{\mathcal{R}} x\sigma_1[r_2\sigma_2]_{r'}$. Now let τ be the substitution such that $\tau(x) = x\sigma_1[r_2\sigma_2]_{r'}$ and $\tau(y) = \sigma_1(y)$ for all $y \neq x$, and τ' some normalization, that is, $y\tau \to^*_{\mathcal{R}} y\tau'$ for all y. Moreover, note that

$$y\sigma_1 \xrightarrow[\mathcal{R}]{*} y\tau \text{ for all } y. \tag{9}$$

We have $u' = \ell_1\sigma_1[r_2\sigma_2]_q = \ell_1\sigma_1[x\tau]_{q'} \to^*_{\mathcal{R}} \ell_1\tau$, and thus $u' \to^*_{\mathcal{R}} \ell_1\tau'$. From (9) we have $r_1\sigma_1 \to^*_{\mathcal{R}} r_1\tau$ and thus $t' = r_1\sigma_1 \to^*_{\mathcal{R}} r_1\tau'$. Finally, we will show that $\ell_1\tau' \to_{\mathcal{R}} r_1\tau'$, concluding the proof of $t' \downarrow_{\mathcal{R}} u'$. To this end, let $s_i \approx t_i \in c_1$. By (9) and the definition of τ' we obtain $s_i\sigma_1 \to^*_{\mathcal{R}} t_i\sigma_1 \to^*_{\mathcal{R}} t_i\tau'$ and $s_i\sigma_1 \to^*_{\mathcal{R}} s_i\tau'$. Then $s_i\tau' \downarrow_{\mathcal{R}} t_i\tau'$ by IH and also $s_i\tau' \to^*_{\mathcal{R}} t_i\tau'$, since t_i is strongly irreducible. □

4 Certification

There are some complications for employing Theorem 2 in practice. Quasi-decreasingness, strong irreducibility, and joinability of CCPs are all undecidable in general. For quasi-decreasingness we fall back to the sufficient criterion that a deterministic 3-CTRS is quasi-decreasing if its unraveling (a transformation to

an unconditional term rewrite system) is terminating. This result was formalized by Winkler and Thiemann [32] and is already available in IsaFoR. A sufficient condition for strong irreducibility is *absolute irreducibility*:

Definition 1. *A term t is* absolutely $\mathcal{R}$-irreducible *if none of its non-variable subterms unify with any variable-disjoint variant of left-hand sides of rules in the CTRS $\mathcal{R}$. A DTRS is called* absolutely deterministic *(or ADTRS for short) if for each rule all right-hand sides of conditions are absolutely $\mathcal{R}$-irreducible.*

The proof of the following lemma [4, Lemma 4.1(a,b)] is immediate.

Lemma 1. *For a term t and a CTRS $\mathcal{R}$:*
- *If t is absolutely $\mathcal{R}$-irreducible, then t is also strongly $\mathcal{R}$-irreducible.*
- *If $\mathcal{R}$ is absolutely deterministic, then $\mathcal{R}$ is also strongly deterministic.* □

We replace joinability of CCPs by infeasibility [26] (already part of IsaFoR) together with two further criteria which rely on *contextual rewriting*.

Definition 2. *Consider a set C of equations between terms which we will call a* context*. First we define a function $\bar{\cdot}$ on terms such that $\bar{t}$ is the term t where each variable $x \in \mathcal{V}(C)$ is replaced by a fresh constant $\overline{x}$. Moreover, let $\overline{C}$ denote the set C where all variables have been replaced by fresh constants $\overline{x}$. For a CTRS $\mathcal{R}$ we can make a* contextual rewrite step*, denoted by $s \to_{\mathcal{R},C} t$, if we can make a conditional rewrite step with respect to the CTRS $\mathcal{R} \cup \overline{C}$ from $\bar{s}$ to $\bar{t}$.*

We formalized soundness of contextual rewriting [4, Lemma 4.2] as follows:

Lemma 2. *If $s \to^*_{\mathcal{R},C} t$ then $s\sigma \to^*_{\mathcal{R}} t\sigma$ for all substitutions σ satisfying C.*

This lemma is stated as obvious without proof by Avenhaus and Loría-Sáenz. However, we deem the strengthened statement ($\star$) below intricate enough to warrant a full proof (since without this strengthening, as far as we can tell, the outermost induction fails).

Proof. Consider the auxiliary function $[t]_\sigma$, which substitutes each Skolem constant $\overline{x}$ in t by $\sigma(x)$, that is, it works like applying a substitution to a term, but to Skolem constants instead of variables. Note that $[\,\bar{t}\,]_\sigma = t\sigma$ whenever $\mathcal{V}(t) \subseteq \mathcal{V}(C)$. Now we show by induction on n that

$$s \to_{\mathcal{R}\cup\overline{C},n} t \text{ implies } [s]_\sigma \to^*_{\mathcal{R},n} [t]_\sigma \qquad (\star)$$

for any σ satisfying C. The base case is trivial. In the inductive step we have a rule $\ell \to r \Leftarrow c \in \mathcal{R} \cup \overline{C}$, a position p, and a substitution τ such that $s|_p = \ell\tau$, $t = s[r\tau]_p$, and $u\tau \to^*_{\mathcal{R}\cup\overline{C},n} v\tau$ for all $u \approx v \in c$. If $\ell \to r \Leftarrow c \in \mathcal{R}$, then we obtain $[u\tau]_\sigma \to^*_{\mathcal{R}\cup\overline{C},n} [v\tau]_\sigma$ for all $u \approx v \in c$ by IH. Then $s \to^*_{\mathcal{R}\cup\overline{C},n+1} t$ can be shown by induction on the context $s[\cdot]_p$. Otherwise, $\ell \to r \Leftarrow c \in \overline{C}$ and thus c is empty, $\ell\tau = \ell$, and $r\tau = r$, since $\overline{C}$ is an unconditional ground TRS. Moreover, there is a rule $\ell' \to r' \in C$ (thus also $\mathcal{V}(\ell', r') \subseteq \mathcal{V}(C)$) such that $\overline{\ell'} = \ell$ and $\overline{r'} = r$. Again, the final result follows by induction on $s[\cdot]_p$.

Assume $s \to_{\mathcal{R},C} t$. Then $\overline{s} \to_{\mathcal{R}\cup\overline{C},n} \overline{t}$ for some level n. Let $\widetilde{t}$ denote the extension of $\overline{t}$ where all variables x in t (that is, not just those in $\mathcal{V}(C)$) are replaced by fresh constants $\overline{x}$. Note that $\widetilde{t} = (\overline{t})(\lambda x.\,\overline{x})$ for every term t. But then also $\widetilde{s} \to_{\mathcal{R}\cup\overline{C},n} \widetilde{t}$ since conditional rewriting is closed under substitutions. Further note that $[\widetilde{t}]_\sigma = t\sigma$ for all t. Thus taking $\widetilde{s}$ and $\widetilde{t}$ for s and t in $(\star)$ we obtain $s\sigma \to^*_{\mathcal{R},n} t\sigma$. Since we just established the desired property for single contextual rewrite steps it is straightforward to extend it to rewrite sequences. □

The above lemma is the key to overcome the undecidability issues of conditional rewriting. For example, for joinability of CCPs the problem is that a single joining sequence (as is usual in certificates for TRSs) does not prove joinability for all satisfying substitutions. However, contextual rewriting has this property.

Now we are able to define the two promised criteria for CCPs that employ contextual rewriting: *context-joinability* and *unfeasibility*.

Definition 3. *Let $s \approx t \Leftarrow c$ be a CCP induced by an overlap between variable-disjoint variants $\ell_1 \to r_1 \Leftarrow c_1$ and $\ell_2 \to r_2 \Leftarrow c_2$ of rules in $\mathcal{R}$ with mgu μ. We say that the CCP is* unfeasible *if we can find terms u, v, and w such that (1) for all σ that satisfy c we have $\ell_1\mu\sigma \succ u\sigma$, (2) $u \to^*_{\mathcal{R},c} v$, (3) $u \to^*_{\mathcal{R},c} w$, and (4) v and w are both strongly irreducible and $v \not\sim w$. Moreover, we call the CCP* context-joinable *if there exists some term u such that $s \to^*_{\mathcal{R},c} u$ and $t \to^*_{\mathcal{R},c} u$.*

Example 3. Consider the CTRS $\mathcal{R}_{\mathsf{last}}$ consisting of the two rules

$$\mathsf{last}(\mathsf{cons}(x,y)) \to x \Leftarrow y \approx \mathsf{nil} \quad \mathsf{last}(\mathsf{cons}(x,y)) \to \mathsf{last}(y) \Leftarrow y \approx \mathsf{cons}(z,v)$$

having the CCP $x \approx \mathsf{last}(y) \Leftarrow c$ with $c = \{y \approx \mathsf{nil},\ y \approx \mathsf{cons}(z,v)\}$. This CCP is unfeasible because for all satisfying substitutions σ we have $\mathsf{last}(\mathsf{cons}(x,y))\sigma \succ y\sigma$, $y \to^*_{\mathcal{R}_{\mathsf{last}},c} \mathsf{cons}(z,v)$, $y \to^*_{\mathcal{R}_{\mathsf{last}},c} \mathsf{nil}$, and both $\mathsf{cons}(z,v)$ and nil are strongly irreducible and not unifiable. Now, look at the arbitrary CCP $x \approx \mathsf{min}(\mathsf{nil}) \Leftarrow c$ with $c = \{\mathsf{min}(\mathsf{nil}) \approx x\}$. Since $x \to^*_{\mathcal{R},c} x$ and $\mathsf{min}(\mathsf{nil}) \to^*_{\mathcal{R},c} x$ it is context-joinable (regardless of the actual CTRS $\mathcal{R}$).

Due to Lemma 2 above, context-joinability implies joinability of a CCP for arbitrary satisfying substitutions. The rationale for the definition of unfeasibility is a little bit more technical, since it only makes sense inside the proof (by induction) of the theorem below. Basically, unfeasibility is defined in such a way that unfeasible CCPs contradict the confluence of all $\succ$-smaller terms, which we obtain as induction hypothesis.

In the original paper the definition of quasi-reductivity requires its order to be closed under substitutions. This property is used in the proof of [4, Theorem 4.2]. By a small change to the definition of unfeasibility we avoid this requirement for our extension to quasi-decreasingness.

We are finally ready to state a concrete version of Theorem 2:

Theorem 3. *Let the ADTRS $\mathcal{R}$ be quasi-decreasing with respect to $\succ$. Then $\mathcal{R}$ is confluent if all CCPs are context-joinable, unfeasible, or infeasible.*

Proof. Unfortunately, we cannot directly reuse Theorem 2 and its proof, since we need our sufficient criteria in the induction hypothesis. However, the new proof is quite similar. It only differs in case (1), where we consider a CCP:

1. If the CCP is context-joinable, we obtain a join with respect to contextual rewriting which we can easily transform into a join with respect to $\mathcal{R}$ by an application of Lemma 2 because we have a substitution satisfying the conditions of the CCP.
2. If the CCP is unfeasible, we obtain two diverging contextual rewrite sequences. Again since there is a substitution satisfying the conditions of the CCP we may employ Lemma 2 to get two diverging conditional $\mathcal{R}$-rewrite sequences. Because $\ell_1\sigma \succ_{\mathsf{st}} t_0$ we can use the induction hypothesis to get a join between the two end terms. But from the definition of unfeasibility we also know that the end points are not unifiable (and hence are not the same) and cannot be rewritten (because of strong irreducibility), leading to a contradiction.
3. Finally, if the CCP is infeasible, then there is no substitution that satisfies its conditions, contradicting the fact that we already have such a substitution. □

Example 4. The CTRS $\mathcal{R}_{\mathsf{min}}$ from Example 1 is actually an ADTRS and also quasi-decreasing. To conclude confluence of the system it remains to check its CCPs which are listed in Example 2. The first one, (1,2), is trivially context-joinable because the left- and right-hand sides coincide. Unfortunately, the methods used in ConCon are not able to handle either of the CCPs (1,3) and (2,3). So we are not able to conclude confluence of $\mathcal{R}_{\mathsf{min}}$ at this point.

We give a transformation on CTRSs which is often helpful in practice:

Definition 4 (Inlining of Conditions). *Given a conditional rewrite rule* $\rho : \ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n$ *and an index* $1 \leqslant i \leqslant n$ *such that* $t_i = x$ *for some variable* x*, let* $\mathsf{inl}_i(\rho)$ *denote the rule resulting from* inlining *the* i*th condition of* ρ*, that is,* $\ell \to r\sigma \Leftarrow s_1\sigma \approx t_1, \ldots, s_{i-1}\sigma \approx t_{i-1}, s_{i+1}\sigma \approx t_{i+1}, \ldots, s_n\sigma \approx t_n$ *with* $\sigma = \{x \mapsto s_i\}$*.*

Lemma 3. *Let* $\rho \in \mathcal{R}$ *and* $s \approx x$ *be the* i*th condition of* ρ*. Whenever we have* $x \notin \mathcal{V}(\ell, s, t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n)$*, then the relations* $\to^*_{\mathcal{R}}$ *and* $\to^*_{\mathcal{R}'}$*, where* $\mathcal{R}' = (\mathcal{R} \setminus \{\rho\}) \cup \{\mathsf{inl}_i(\rho)\}$*, coincide.*

Proof. We show $\to_{\mathcal{R},n} \subseteq \to^*_{\mathcal{R}',n}$ and $\to_{\mathcal{R}',n} \subseteq \to_{\mathcal{R},n}$ by induction on the level n. For $n = 0$ the result is immediate. Consider a step $s = C[\ell\sigma] \to_{\mathcal{R},n+1} C[r\sigma] = t$ employing rule ρ (for the other rules of $\mathcal{R}$ the result is trivial). Thus, $u\sigma \to^*_{\mathcal{R},n} v\sigma$ for all $u \approx v \in c$. In particular $s\sigma \to^*_{\mathcal{R},n} x\sigma$. Thus, using the IH, for each condition $u \approx v$ of $\mathsf{inl}_i(\rho)$ we have $1 \leqslant j \leqslant n$ such that $u\sigma = s_j\{x \mapsto s\}\sigma \to^*_{\mathcal{R}',n} s_j\sigma \to^*_{\mathcal{R}',n} t_j\sigma = v\sigma$. Hence, $\ell\sigma \to_{\mathcal{R}',n+1} r\{x \mapsto s\}\sigma \to^*_{\mathcal{R}',n+1} r\sigma$ and thus $s \to^*_{\mathcal{R}',n+1} t$.

Now, consider a step $s = C[\ell\sigma] \to_{\mathcal{R}',n+1} C[r\{x \mapsto s\}\sigma]$ employing rule $\mathsf{inl}_i(\rho)$. Together with the IH this implies that $u\sigma \to^*_{\mathcal{R},n} v\sigma$ for all conditions $u \approx v$ in $\mathsf{inl}_i(\rho)$. Let τ be a substitution such that $\tau(x) = s\sigma$ and $\tau(y) = \sigma(y)$ for all $y \neq x$. We have $s_i\tau = s\tau = x\tau = t_i\tau$ and $s_j\tau = s_j\{x \mapsto s\}\sigma \to^*_{\mathcal{R},n} t_j\sigma = t_j\tau$ for all $1 \leqslant j \leqslant n$ with $i \neq j$, since x neither occurs in s nor the right-hand sides of

conditions in $\mathsf{inl}_i(\rho)$. Therefore, $u \to^*_{\mathcal{R},n} v$ for all $u \approx v \in c$. In total, we have $s = C[\ell\sigma] = C[\ell\tau] \to_{\mathcal{R},n+1} C[r\tau] = C[r\{x \mapsto s\}\sigma]$, concluding the proof. □

We are not aware of any mention of this simple method in the literature, but found that in practice, exhaustive application of inlining increases the applicability of other methods like infeasibility via tcap and non-confluence via plain rewriting: for the former inlining yields more term structure, which may prevent tcap from replacing a subterm by a fresh variable and thus makes non-unifiability more likely; while for the latter inlining may yield CCPs without conditions and thereby make them amenable to non-joinability techniques for plain term rewriting [34].

Example 5. Rules (2) and (3) of $\mathcal{R}_{\mathsf{min}}$ from Example 1 are both susceptible to inlining of conditions. For each of them, we may remove the first condition and replace y by $\mathsf{min}(xs)$ resulting in

$$\mathsf{min}(\mathsf{cons}(x, xs)) \to x \Leftarrow x < \mathsf{min}(xs) \approx \mathsf{true} \tag{2′}$$
$$\mathsf{min}(\mathsf{cons}(x, xs)) \to \mathsf{min}(xs) \Leftarrow x < \mathsf{min}(xs) \approx \mathsf{false} \tag{3′}$$

Now, instead of the CCPs from Example 2 we have the following CCPs (modulo symmetry as before):

$$x \approx x \Leftarrow x < \mathsf{min}(\mathsf{nil}) \approx \mathsf{true} \tag{1,2′}$$
$$x \approx \mathsf{min}(\mathsf{nil}) \Leftarrow x < \mathsf{min}(\mathsf{nil}) \approx \mathsf{false} \tag{1,3′}$$
$$x \approx \mathsf{min}(xs) \Leftarrow x < \mathsf{min}(xs) \approx \mathsf{true},\ x < \mathsf{min}(xs) \approx \mathsf{false} \tag{2′,3′}$$

Again, the first CCP (1,2′) is trivially context-joinable, (1,3′) is infeasible since $\mathsf{tcap}(x < \mathsf{min}(\mathsf{nil})) = x < \mathsf{min}(\mathsf{nil})$ and false are not unifiable, and (2′,3′) is unfeasible because with contextual rewriting we can reach the two non-unifiable normal forms true and false starting from $x < \mathsf{min}(xs)$. Hence, we conclude confluence of the quasi-decreasing ADTRS $\mathcal{R}_{\mathsf{min}}$ by Theorem 3.

Inlining of conditions is implemented in ConCon 1.4.0 as a first preprocessing step and is certifiable by CeTA.

5 Certification Challenges

One of the main challenges towards actual certification is typically disregarded on paper: the definition of critical pairs may yield an infinite set of CCPs even for finite CTRSs. This is because we have to consider arbitrary variable-disjoint variants of rules. However, a hypothetical certificate would only contain those CCPs that were obtained from some specific variable-disjoint variants of rules. Now the argument typically goes as follows: *modulo variable renaming there are only finitely many CCPs. Done.*

However, this reasoning is valid only for properties that are either closed under substitution or at least invariant under renaming of variables. For joinability of plain critical pairs—arguably the most investigated case—this is indeed easy. But

when it comes to contextual rewriting we spent a considerable amount of work on some results about permutations that were not available in IsaFoR.

To illustrate the issue, consider the abstract specification of the check function *check-CCPs*, such that *isOK* (*check-CCPs* $\mathcal{R}$) implies that each of the CCPs of $\mathcal{R}$ is either unfeasible, context-joinable, or infeasible. To this end we work modulo the assumption that we already have sound check functions for the latter three properties, which is nicely supported by Isabelle's locale mechanism:[2]

locale *al94-spec* =
  fixes v_x **and** v_y
    and *check-context-joinable*
    and *check-infeasible*
    and *check-unfeasible*
  assumes v_x and v_y are injective
    and $\mathrm{ran}(v_x) \cap \mathrm{ran}(v_y) = \varnothing$
    and *isOK* (*check-context-joinable* $\mathcal{R}$ s t C) $\Longrightarrow \exists u.\ s \to^*_{\mathcal{R},C} u \wedge t \to^*_{\mathcal{R},C} u$)
      ...

We just list the required properties of the renaming functions v_x and v_y and the soundness assumption for *check-context-joinable*.

Now what would a certificate contain and how would we have to check it? Amongst other things, the certificate would contain a finite set of CCPs $\mathcal{C}'$ that were computed by some automated tool. Internally, our certifier computes its own finite set of CCPs $\mathcal{C}$ where variable-disjoint variants of rules are created by fixed injective variable renaming functions v_x and v_y, whose ranges are guaranteed to be disjoint. The former prefixes the character "x" and the latter the character "y" to all variable names, hence the names. At this point we have to check that for each CCP in $\mathcal{C}$ there is one in $\mathcal{C}'$ that is its variant, which is not too difficult. More importantly, we have to prove that whenever some desired property P, say context-joinability, holds for any CCP, then P also holds for all of its variants (including the one that is part of $\mathcal{C}$).

To this end, assume that we have a CCP resulting from a critical overlap of the two rules $\ell_1 \to r_1 \Leftarrow c_1$ and $\ell_2 \to r_2 \Leftarrow c_2$ at position p with mgu μ. This means that there exist permutations π_1 and π_2 such that $(\ell_1 \to r_1 \Leftarrow c_1)\pi_1$ and $(\ell_2 \to r_2 \Leftarrow c_2)\pi_2$ are both in $\mathcal{R}$. In our certifier, mgus are computed by the function $\mathrm{mgu}(s,t)$ which either results in *None*, if $s \not\sim t$, or in *Some* μ such that μ is an mgu of s and t, otherwise. Moreover, variable-disjointness of rules is ensured by v_x and v_y, so that we actually call $\mathrm{mgu}(\ell_1|_p\pi_1 v_x, \ell_2\pi_2 v_x)$ for computing a concrete CCP corresponding to the one we assumed above. Thus, we need to show that $\mathrm{mgu}(\ell_1|_p, \ell_2) = \mathit{Some}\ \mu$ also implies that $\mathrm{mgu}(\ell_1|_p\pi_1 v_x, \ell_2\pi_2 y_v) = \mathit{Some}\ \mu'$ for some mgu μ'. Moreover, we are interested in the relationship between μ and μ' with respect to the variables in both rules. Previously—for an earlier formalization of infeasibility [25]—IsaFoR only contained a result that related both unifiers modulo some arbitrary substitution (that is, not necessarily a renaming).

[2] For technical reasons, our formalization uses two locales (*al94-ops*, *al94-spec*) here.

Unfortunately, contextual rewriting is not closed under arbitrary substitutions. Nevertheless, contextual rewriting is closed under permutations, provided the permutation is also applied to C.

Lemma 4. *For every permutation π we have that $s\pi \to^*_{R,C\pi} t\pi$ iff $s \to^*_{R,C} t$.* □

It remains to show that μ and μ' differ basically only by a renaming (at least on the variables of our two rules), which is covered by the following lemma.

Lemma 5. *Let $\mathrm{mgu}(s,t) = \mathit{Some}\ \mu$ and $\mathcal{V}(s,t) \subseteq S \cup T$ for two finite sets of variables S and T with $S \cap T = \varnothing$. Then, there exist a substitution μ' and a permutation π such that for arbitrary permutations π_1 and π_2: $\mathrm{mgu}(s\pi_1 v_x, t\pi_2 v_y) = \mathit{Some}\ \mu'$, $\mu = \pi_1\mu' v_x\pi\ [S]$, and $\mu = \pi_2\mu' v_y\pi\ [T]$.*

Proof. Let $h(x) = xv_x\pi_1$ if $x \in S$ and $h(x) = xv_y\pi_2$, otherwise. Then, since h is bijective between $S \cup T$ and $h(S \cup T)$ we can obtain a permutation π for which $\pi = h\ [S \cup T]$. We define $\mu' = -\pi\mu$ and abbreviate $s\pi_1 v_x$ and $t\pi_2 v_y$ to s' and t', respectively. Note that $s' = s\pi$ and $t' = t\pi$. Since μ is an mgu of s and t we have $s\mu = t\mu$, which further implies $s'\mu' = t'\mu'$. But then μ' is a unifier of s' and t' and thus there exists some μ'' for which $\mathrm{mgu}(s',t') = \mathit{Some}\ \mu''$ and $s'\mu'' = t'\mu''$.

We now show that μ' is also most general. Assume $s'\tau = t'\tau$ for some τ. Then $s\pi\tau = t\pi\tau$ and thus there exists some δ such that $\pi\tau = \mu\delta$ (since μ is most general). But then $\pi^-\pi\tau = \pi^-\mu\delta$ and thus $\tau = \mu'\delta$. Hence, μ' is most general.

Since μ'' is most general too, it only differs by a renaming, say π', from μ', that is, $\mu'' = \pi'\mu'$. This yields $\mu = \pi_1\mu'' v_x\pi'^-\ [S]$ and $\mu = \pi_2\mu'' v_y\pi'^-[T]$, and thus concludes the proof. □

6 Available Check Functions

Before we can actually certify the output of CTRS confluence tools with CeTA, we have to provide an executable check function for each property that is required to apply Theorem 3 and prove its soundness. It is worth mentioning that the return type of these check functions is only "morally" bool. In order to have nice error messages we actually employ a monad. So whenever we need to handle the result of a check function as bool we encapsulate it in a call to *isOK* which results in *False* if there was an error and *True*, otherwise.

As mentioned earlier, the check functions for quasi-decreasingness and infeasibility are already in place. It remains to provide new check functions for absolute irreducibility, absolute determinism, contextual rewrite sequences, context-joinability, and unfeasibility together with their corresponding soundness proofs. For absolute irreducibility we provide the check function *check-airr*, employing existing machinery from IsaFoR for renaming and unification, and prove:

Lemma 6. *isOK (check-airr $\mathcal{R}$ t) iff the term t is absolutely $\mathcal{R}$-irreducible.* □

This, in turn, is used to define the check function *check-adtrs* and the accompanying lemma for ADTRSs.

Lemma 7. *isOK (check-adtrs $\mathcal{R}$) iff $\mathcal{R}$ is an ADTRS.* □

Concerning contextual rewriting, we provide the check function *check-csteps* for conditional rewrite sequences together with the following lemma:

Lemma 8. *Given a CTRS $\mathcal{R}$, a set of conditions C, two terms s and t, and a list of conditional rewrite proofs ps, we have that isOK (check-csteps ($\mathcal{R} \cup \overline{C}$) $\overline{s}$ $\overline{t}$ $\overline{ps}$) implies $s \to^*_{\mathcal{R},C} t$.* □

Although conditional rewriting is decidable in our setting (strong determinism and quasi-decreasingness), we require a *conditional rewrite proof* to provide all the necessary information for checking a single conditional rewrite step (the employed rule, position, and substitution; source and target terms; and recursively, a list of rewrite proofs for each condition of the applied rule). That way, we avoid having to formalize a rewriting engine for conditional rewriting in IsaFoR. With a check function for contextual rewrite sequences in place, we can easily give the check function *check-context-joinable* with the corresponding lemma:

Lemma 9. *Given a CTRS $\mathcal{R}$, three terms s, t, and u, a set of conditions C, and two lists of conditional rewrite proofs ps and qs, we have that isOK (check-context-joinable u ps qs $\mathcal{R}$ s t C) implies that there exists some term u' such that $s \to^*_{\mathcal{R},C} u' \,{}^*_{\mathcal{R},C}\!\leftarrow t$.* □

Here *check-context-joinable* is a concrete implementation of the homonymous function from the *al94-spec* locale. We further give the check function *check-unfeasible* and the accompanying soundness lemma:

Lemma 10. *Given a quasi-decreasing CTRS $\mathcal{R}$, two variable-disjoint variants of rules $\rho_1 \colon \ell_1 \to r_1 \Leftarrow c_1$ and $\rho_2 \colon \ell_2 \to r_2 \Leftarrow c_2$ in $\mathcal{R}$, an mgu μ of $\ell_1|_p$ and ℓ_2 for some position p, a set of conditions C such that $C = c_1\mu, c_2\mu$, three terms t, u, and v, and two lists of conditional rewrite proofs ps and qs, we have that isOK (check-unfeasible t u v ps qs ρ_1 ρ_2 $\mathcal{R}$ ℓ_1 μ C) implies that there exist three terms t', u', and v' such that for all σ we have $\ell_1\mu\sigma \succ t'\sigma$, whenever σ satisfies C, $u' \,{}^*_{\mathcal{R},C}\!\leftarrow t' \to^*_{\mathcal{R},C} v'$, u' and v' are both strongly irreducible, and $u' \not\sim v'$.* □

Again, *check-unfeasible* is a concrete implementation of the function of the same name from the *al94-spec* locale and it additionally performs various sanity checks.

At this point, interpreting the *al94-spec* locale using the three check functions *check-context-joinable*, *check-infeasible*, and *check-unfeasible* from above yields the concrete function *check-CCPs*, which is used in the final check *check-al94*.

Lemma 11. *Given a quasi-decreasing CTRS $\mathcal{R}$, a list of context-joinability certificates c, a list of infeasibility certificates i, and a list of unfeasibility certificates u. Then, isOK (check-al94 c i u $\mathcal{R}$) implies confluence of $\mathcal{R}$.* □

7 Experiments

The largest available collection of CTRSs we are aware of is the confluence problems database (Cops) [10]. At the time of writing it contains a total of

ConCon	A	A + i	N	N + i	T	T + i
1.3.2	42 (0) / 7 (0)	-	26 (0)	-	82 (38)	-
1.4.0	46 (43) / 8 (11)	47(44) / 8 (11)	27 (27)	29 (29)	84 (77)	86 (79)

Table 1: Comparison on 119 oriented 3-CTRSs from Cops.

152 CTRSs. Among these, there are 119 oriented 3-CTRSs from which exactly 100 are also ADTRSs. We compare ConCon 1.3.2, which participated in last years confluence competition (CoCo 2016) [3], to ConCon 1.4.0, the current version which implements the results of the paper at hand. Our experiments ran on the StarExec [29] platform with a timeout of 60 seconds per problem. The outcome is summarized in Table 1,[3] where columns labeled A, N, and T contain the results of applying Theorem 3, using non-confluence methods, and trying all methods implemented in ConCon concurrently, respectively. A suffix '+ i' indicates preprocessing by exhaustive inlining of conditions (Lemma 3). Results in parentheses are not just proved by ConCon but also certified by CeTA. For the two A-columns the numbers following the '/' indicate how many systems could only be solved by Theorem 3 but not by any other method.

In total, ConCon 1.3.2 can decide confluence of 82 systems. Of those, 56 are confluent and 26 are non-confluent. Using only Theorem 3, 42 systems can be shown confluent. For 7 of these, none of the other methods are successful. Neither Theorem 3 nor the non-confluence methods are certifiable in ConCon 1.3.2. However, in 38 cases (using other methods) the output of ConCon 1.3.2 is certifiable by CeTA. Also inlining of conditions is absent in ConCon 1.3.2.

The new version of ConCon can decide confluence of 86 systems. Of those, 57 are confluent and 29 are non-confluent. Seven of the generated confluence proofs cannot be certified by CeTA. This is due to an infeasibility method (using equational reasoning) that is not yet formalized. In contrast, all of the non-confluence proofs can be certified by CeTA. When we subtract the certifiably non-confluent systems we are left with 72 potentially confluent ADTRSs. From those 52 are certifiably quasi-decreasing. Theorem 3 succeeds on 46 of these quasi-decreasing ADTRSs (and can be certified for 43 of them). For three of these systems (288, 292, 326) testing for infeasibility is essential. When using inlining of conditions we gain another (certifiably) confluent system (493). Finally, independent of inlining of conditions, there are 8 systems where only Theorem 3 is successful. In the certifiable case this number increases to 11 systems (because for 3 systems the other methods are not certifiable). The most important message of Table 1 is that with the new versions of ConCon and CeTA the number of certifiably (non-)confluent systems has more than doubled from 38 to 79, which means that more than 90% of the (non-)confluence proofs for CTRSs are certifiable.

[3] Detailed results are available at http://cl-informatik.uibk.ac.at/experiments/2017/cade/.

8 Conclusion and Future Work

Even in the presence of a suitable notion of termination (like quasi-decreasingness), proving confluence of conditional term rewrite systems is still hard (unlike in the unconditional case, where confluence is decidable.)

We formalized a characterization of confluence of quasi-decreasing strongly deterministic CTRSs in Isabelle/HOL. It requires joinability of all conditional critical pairs, which is undecidable in general. Moreover, we formalized a more practical variant of the previous characterization for which each conditional critical pair must be either context-joinable, unfeasible, or infeasible. These properties, in turn, rely on strong irreducibility, which like strong determinism is undecidable in general. Thus, we further formalized decidable sufficient criteria.

In total, this paper constitutes the necessary work for the actual certification of confluence of quasi-decreasing SDTRSs, which complements our existing check functions for certifying confluence of CTRSs [26, 32]. We have extended our confluence tool ConCon and the certifier CeTA accordingly.

Here is a rough impression of the involved effort: our formalization comprises 28 definitions, 14 recursive functions, and 83 lemmas with proofs, on approximately 2500 lines of Isabelle code (in addition to everything that we could reuse from the IsaFoR library). The whole development took about 6 person-months.

Future Work. Concerning certification, our extension from quasi-reductive to quasi-decreasing CTRSs is at the moment only of theoretical relevance, since the only way of certifying quasi-decreasingness with CeTA is via quasi-reductivity.

In principle it may be useful to use methods for proving operational termination [20]—a notation equivalent to quasi-decreasingness [19]—in order to increase the applicability of Theorem 3. However, IsaFoR is currently lacking the proof that operational termination and quasi-decreasingness coincide. Also, none of the methods for proving operational termination have been formalized so far. Moreover, when running AProVE [12] and MU-TERM [1] on the 72 ADTRSs of Cops which have not already been shown to be non-confluent, the former can show operational termination of the same 52 systems for which ConCon could show quasi-reductivity, and the latter can show two additional systems (266, 278), while losing another one (362). Of course, this insignificant difference could be due to our example database.

Open Problem. After having finished our formalization, we realized that it is not known whether quasi-decreasingness differs from quasi-reductivity at all, that is, the question whether there exists a quasi-decreasing CTRS that is not quasi-reductive, is still open. Regardless, we agree with Ohlebusch [24] that quasi-decreasingness has two advantages: (1) it does not depend on signature extensions and (2) $\ell\sigma \succ_{\mathsf{st}} s_i\sigma$ is only required if $s_j\sigma \to_{\mathcal{R}}^* t_j\sigma$ instead of $s_j\sigma \succeq t_j\sigma$. Point (1) is illustrated by the quasi-decreasing CTRS $\mathcal{R}_{\mathsf{qd}} = \{\mathsf{f}(\mathsf{b}) \to \mathsf{f}(\mathsf{a}), \mathsf{b} \to \mathsf{c}, \mathsf{a} \to \mathsf{c} \Leftarrow \mathsf{b} \approx \mathsf{c}\}$. Assume that $\mathcal{R}_{\mathsf{qd}}$ is quasi-reductive with respect to $\succ$. Then, $\mathsf{f}(\mathsf{b}) \succ \mathsf{f}(\mathsf{a})$ and $\mathsf{a}\ (\succ \cup \rhd)^+\ \mathsf{b}$. If we are not allowed to introduce fresh function symbols, the latter implies $\mathsf{a} \succ \mathsf{b}$, for otherwise, we would have $\mathsf{a} \succ \mathsf{f}^k(\mathsf{b}) \trianglerighteq \mathsf{b}$ for some $k \geqslant 0$,

which together with closure under contexts and transitivity of $\succ$ contradicts the well-foundedness of $\succ$. But $\mathsf{a} \succ \mathsf{b}$ also contradicts the well-foundedness of $\succ$.

Proof Assistant. We found Sledgehammer [6, 7] to be an indispensable tool for our development. On the one hand, to quickly discharge subgoals that seemed intuitively obvious but turned out tedious to prove, and on the other, as fast "fact finder" for the huge IsaFoR library (especially for the second author, who has not been involved in IsaFoR from the start).

Acknowledgments. We thank Bertram Felgenhauer and Julian Nagele for fruitful discussions on the subject matter. Moreover, we would like to thank the anonymous reviewers for their constructive and helpful comments.

A Browsing Isabelle/HOL Theory Files

We provide the Isabelle/HOL theory files for the presented formalization (`AL94.thy`, `AL94_Impl.thy`, `Inline_Conditions.thy`, and `Inline_Conditions_Impl.thy` all in the subdirectory `thys/Conditional_Rewriting/`) as part of the formal IsaFoR library which depends on the Archive of Formal Proofs (AFP). First, get the AFP via

```
wget https://www.isa-afp.org/release/afp-current.tar.gz
```

and extract the archive. Then get IsaFoR via

```
hg clone \
  http://cl2-informatik.uibk.ac.at/rewriting/mercurial.cgi/IsaFoR
```

and from inside the `IsaFoR` directory update to tag `v2.29`:

```
hg update -r v2.29
```

For the remainder, you will need to have Isabelle2016-1 installed. Add the following lines to your `$HOME/.isabelle/Isabelle2016-1/etc/settings`

```
init_component "/path/to/afp/directory/"
init_component "/path/to/isafor/directory"
```

Finally—again from the `IsaFoR` directory—start Isabelle/jEdit in order to browse our formal development:

```
isabelle jedit -l TA thys/Conditional_Rewriting/AL94_Impl.thy
```

This will take some time, even on a (more than) decent machine, the first time around, but will be much faster thereafter.

References

1. Alarcón, B., Gutiérrez, R., Lucas, S., Navarro-Marset, R.: Proving termination properties with MU-TERM. In: Proceedings of the 13th International Conference on Algebraic Methodology And Software Technology (AMAST). Lecture Notes in Computer Science, vol. 6486, pp. 201–208. Springer (2011), doi:10.1007/978-3-642-17796-5_12
2. Antoy, S., Hanus, M.: Functional logic programming. Communications of the ACM 53(4), 74–85 (2010), doi:10.1145/1721654.1721675
3. Aoto, T., Hirokawa, N., Nagele, J., Nishida, N., Zankl, H.: Confluence competition 2015. In: Proceedings of the 25th International Conference on Automated Deduction (CADE). Lecture Notes in Computer Science, vol. 9195, pp. 101–104. Springer (2015), doi:10.1007/978-3-319-21401-6_5
4. Avenhaus, J., Loría-Sáenz, C.: 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 (LPAR). Lecture Notes in Computer Science, vol. 822, pp. 215–229. Springer (1994), doi:10.1007/3-540-58216-9_40
5. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
6. Blanchette, J., Paulson, L.: Hammering away – a user's guide to sledgehammer for Isabelle/HOL (2010), https://isabelle.in.tum.de/dist/doc/sledgehammer.pdf
7. Blanchette, J.C., Kaliszyk, C., Paulson, L.C., Urban, J.: Hammering towards QED. Journal of Formalized Reasoning 9(1), 101–148 (2016), doi:10.6092/issn.1972-5787/4593
8. Blanqui, F., Koprowski, A.: CoLoR: A Coq library on well-founded rewrite relations and its application to the automated verification of termination certificates. Mathematical Structures in Computer Science 21(4), 827–859 (2011), doi:10.1017/S0960129511000120
9. Contejean, É., Courtieu, P., Forest, J., Pons, O., Urbain, X.: Automated certified proofs with CiME 3. In: Proceedings of the 22nd International Conference on Rewriting Techniques and Applications (RTA). LIPIcs, vol. 10, pp. 21–30. Schloss Dagstuhl (2011), doi:10.4230/LIPIcs.RTA.2011.21
10. Cops: The confluence problems database, http://cops.uibk.ac.at/?q=ctrs
11. Giesl, J., Mesnard, F., Rubio, A., Thiemann, R., Waldmann, J.: Termination competition (termCOMP 2015). In: Proceedings of the 25th International Conference on Automated Deduction (CADE). Lecture Notes in Computer Science, vol. 9195, pp. 105–108. Springer (2015), doi:10.1007/978-3-319-21401-6_6
12. Giesl, J., Schneider-Kamp, P., Thiemann, R.: AProVE 1.2: Automatic termination proofs in the dependency pair framework. In: Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR). Lecture Notes in Computer Science, vol. 4130, pp. 281–286. Springer (2006), doi:10.1007/11814771_24
13. Giesl, J., Thiemann, R., Schneider-Kamp, P.: Proving and disproving termination of higher-order functions. In: Proceedings of the 5th International Symposium on Frontiers of Combining Systems (FroCoS). Lecture Notes in Computer Science, vol. 3717, pp. 216–231. Springer (2005)
14. Hirokawa, N., Middeldorp, A., Sternagel, C.: A new and formalized proof of abstract completion. In: Proceedings of the 5th International Conference on Interactive Theorem Proving (ITP). Lecture Notes in Computer Science, vol. 8558, pp. 292–307. Springer (2014), doi:10.1007/978-3-319-08970-6_19

15. Huet, G.: Confluent reductions: Abstract properties and applications to term rewriting systems. Journal of the ACM 27(4), 797–821 (1980)
16. Knuth, D., Bendix, P.: Simple word problems in universal algebras. In: Leech, J. (ed.) Computational Problems in Abstract Algebra, pp. 263–297. Pergamon Press (1970)
17. Kop, C.: Higher-Order Termination: Automatable Techniques for Proving Termination of Higher-Order Term Rewriting Systems. Ph.D. thesis, VU University Amsterdam (2012), http://hdl.handle.net/1871/39346
18. Korp, M., Sternagel, C., Zankl, H., Middeldorp, A.: Tyrolean Termination Tool 2. In: Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA). Lecture Notes in Computer Science, vol. 5595, pp. 295–304. Springer (2009), doi:10.1007/978-3-642-02348-4_21
19. Lucas, S., Marché, C., Meseguer, J.: Operational termination of conditional term rewriting systems. Information Processing Letters 95(4), 446–453 (2005), doi:10.1016/j.ipl.2005.05.002
20. Lucas, S., Meseguer, J.: Dependency pairs for proving termination properties of conditional term rewriting systems. Journal of Logical and Algebraic Methods in Programming 86(1), 236–268 (2017), doi:10.1016/j.jlamp.2016.03.003
21. Marlow, S.: Haskell 2010 language report, https://www.haskell.org/definition/haskell2010.pdf
22. Newman, M.: On theories with a combinatorial definition of equivalence. Annals of Mathematics 43(2), 223–243 (1942)
23. Nipkow, T.: Equational reasoning in Isabelle. Science of Computer Programming 12(2), 123–149 (1989), doi:10.1016/0167-6423(89)90038-5
24. Ohlebusch, E.: Advanced Topics in Term Rewriting. Springer (2002)
25. Sternagel, C., Sternagel, T.: Level-confluence of 3-CTRSs in Isabelle/HOL. In: Proceedings of the 4th International Workshop on Confluence (IWC) (2015), arXiv:1602.07115
26. Sternagel, C., Sternagel, T.: Certifying confluence of almost orthogonal CTRSs via exact tree automata completion. In: Proceedings of the 1st International Conference on Formal Structures for Computation and Deduction (FSCD). LIPIcs, vol. 51, pp. 29:1–29:16 (2016), doi:10.4230/LIPIcs.FSCD.2016.29
27. Sternagel, T., Middeldorp, A.: Conditional confluence (system description). In: Proceedings of the Joint 25th International Conference on Rewriting Techniques and Applications (RTA) and 12th International Conference on Typed Lambda Calculi and Applications (TLCA). Lecture Notes in Computer Science, vol. 8560, pp. 456–465. Springer (2014), doi:10.1007/978-3-319-08918-8_31
28. Sternagel, T., Sternagel, C.: Formalized confluence of quasi-decreasing, strongly deterministic conditional TRSs. In: Proceedings of the 5th International Workshop on Confluence (IWC) (2016), arXiv:1609.03341
29. Stump, A., Sutcliffe, G., Tinelli, C.: StarExec: a cross-community infrastructure for logic solving. In: Proceedings of the 7th International Joint Conference on Automated Reasoning (IJCAR). Lecture Notes in Computer Science, vol. 8562, pp. 367–373. Springer (2014), doi:10.1007/978-3-319-08587-6_28
30. Thiemann, R., Sternagel, C.: Certification of termination proofs using CeTA. In: Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics (TPHOLs). Lecture Notes in Computer Science, vol. 5674, pp. 452–468. Springer (2009), doi:10.1007/978-3-642-03359-9_31
31. Waldmann, J.: Matchbox: A tool for match-bounded string rewriting. In: Proceedings of the 15th International Conference on Rewriting Techniques and Applications

(RTA). Lecture Notes in Computer Science, vol. 3091, pp. 85–94. Springer (2004), doi:10.1007/978-3-540-25979-4_6
32. Winkler, S., Thiemann, R.: Formalizing soundness and completeness of unravelings. In: Proceedings of the 10th International Symposium on Frontiers of Combining Systems (FroCoS). Lecture Notes in Computer Science, vol. 9322, pp. 239–255. Springer (2015), doi:10.1007/978-3-319-24246-0_15
33. Yamada, A., Kusakari, K., Sakabe, T.: Nagoya termination tool. In: Proceedings of the Joint 25th International Conference on Rewriting Techniques and Applications (RTA) and 12th International Conference on Typed Lambda Calculi and Applications (TLCA). Lecture Notes in Computer Science, vol. 8560, pp. 466–475. Springer (2014), doi:10.1007/978-3-319-08918-8_32
34. Zankl, H., Felgenhauer, B., Middeldorp, A.: CSI – A confluence tool. In: Proceedings of the 23rd International Conference on Automated Deduction (CADE). Lecture Notes in Computer Science, vol. 6803, pp. 499–505. Springer (2011), doi:10.1007/978-3-642-22438-6_38