Improving Automatic Confluence Analysis of Rewrite Systems by Redundant Rules*

Julian Nagele, Bertram Felgenhauer, and Aart Middeldorp

Institute of Computer Science, University of Innsbruck, Austria
`{julian.nagele|bertram.felgenhauer|aart.middeldorp}@uibk.ac.at`

Abstract

We describe how to utilize redundant rewrite rules, i.e., rules that can be simulated by other rules, when (dis)proving confluence of term rewrite systems. We demonstrate how automatic confluence provers benefit from the addition as well as the removal of redundant rules. Due to their simplicity, our transformations were easy to formalize in a proof assistant and are thus amenable to certification. Experimental results show the surprising gain in power.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases term rewriting, confluence, automation, transformation

Digital Object Identifier 10.4230/LIPIcs.RTA.2015.257

1 Introduction

Confluence of first-order term rewrite systems (TRSs) is an important property which is intimately connected to uniqueness of normal forms, and hence to determinism of programs. In recent years there has been tremendous progress in establishing confluence or non-confluence of TRSs automatically, with a number of tools being developed, like ACP [3], Saigawa [11,14], CoLL[1] and our own tool, CSI [29]. There is an annual confluence competition[2] where these tools compete. To increase the trust in the proofs produced by these tools, a certifier like CeTA [27] can be used to verify the proofs. (CeTA is a certifier for termination and confluence proofs for TRSs. Other certifiers already exist for termination proofs, notably Rainbow [5] and CiME3 [6].) The approach taken by CeTA is to formalize various termination and confluence criteria in an interactive theorem prover, together with executable functions that can be used to verify that the criteria are applied correctly. From this formalization, the certifier is automatically extracted, which produces highly trustworthy code. An alternative approach is to convert certificates produced by automated termination (or confluence) tools into proofs that can be replayed in a theorem prover, thereby formally proving the property for the original TRS.

In this paper we present a remarkably simple technique based on the removal and addition of redundant rules, which can significantly enhance the power of automatic confluence provers. The technique is also straightforward to formalize, making it amenable to certification.

▶ **Example 1.** Consider the TRS $\mathcal{R}$ consisting of the two rewrite rules

$$\mathsf{f}(\mathsf{f}(x)) \to x \qquad\qquad \mathsf{f}(x) \to \mathsf{f}(\mathsf{f}(x))$$

* This work is supported by FWF (Austrian Science Fund) project P27528.

[1] `http://www.jaist.ac.jp/~s1310032/coll/`

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

The two non-trivial critical pairs

$\mathsf{f}(\mathsf{f}(\mathsf{f}(x))) \leftarrow\!\rtimes\!\!\!\ltimes\!\rightarrow x \qquad\qquad x \leftarrow\!\rtimes\!\!\!\ltimes\!\rightarrow \mathsf{f}(\mathsf{f}(\mathsf{f}(x)))$

are obviously joinable

$\mathsf{f}(\mathsf{f}(\mathsf{f}(x))) \rightarrow_{\mathcal{R}} \mathsf{f}(x) \rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{f}(x)) \rightarrow_{\mathcal{R}} x$

but not by a multistep (cf. Definition 3). Consequently, the result of van Oostrom [19] on development-closed critical pairs does not apply. After adding the rewrite rule $\mathsf{f}(x) \rightarrow x$ to $\mathcal{R}$, we obtain four new critical pairs

$\mathsf{f}(x) \leftarrow\!\rtimes\!\!\!\ltimes\!\rightarrow x \qquad x \leftarrow\!\rtimes\!\!\!\ltimes\!\rightarrow \mathsf{f}(x) \qquad \mathsf{f}(\mathsf{f}(x)) \leftarrow\!\rtimes\!\!\!\ltimes\!\rightarrow x \qquad x \leftarrow\!\rtimes\!\!\!\ltimes\!\rightarrow \mathsf{f}(\mathsf{f}(x))$

The new rule ensures that $\mathsf{f}^n(x) \nrightarrow\!\!\!\!\!\!\circ\;\; x$ for all $n \geqslant 0$ and thus confluence of the extension follows from the main result of [19], cf. Example 4 in Section 2. Since the new rule can be simulated by the original rules (i.e., $\mathsf{f}(x) \rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{f}(x)) \rightarrow_{\mathcal{R}} x$), also $\mathcal{R}$ is confluent.

None of the aforementioned tools can prove confluence of the TRS $\mathcal{R}$ of Example 1, but every tool can prove confluence of the extended TRS. Below we explain how such extensions can be found automatically.

The next example shows that also proving non-confluence may become easier after adding rules.

▶ **Example 2.** Consider the TRS $\mathcal{R}$ consisting of the eight rewrite rules

$\mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{g}(y)) \rightarrow \mathsf{b} \qquad \mathsf{f}(x, y) \rightarrow \mathsf{f}(x, \mathsf{g}(y)) \qquad \mathsf{g}(x) \rightarrow x \qquad \mathsf{a} \rightarrow \mathsf{g}(\mathsf{a})$

$\mathsf{f}(\mathsf{h}(x), \mathsf{h}(\mathsf{a})) \rightarrow \mathsf{c} \qquad \mathsf{f}(x, y) \rightarrow \mathsf{f}(\mathsf{h}(x), y) \qquad \mathsf{h}(x) \rightarrow x \qquad \mathsf{a} \rightarrow \mathsf{h}(\mathsf{a})$

All critical pairs are *deeply*[3] joinable but $\mathcal{R}$ is not confluent [7]. Two of the critical pairs are

$\mathsf{b} \leftarrow\!\rtimes\!\!\!\ltimes\!\rightarrow \mathsf{f}(\mathsf{h}(\mathsf{g}(\mathsf{a})), \mathsf{g}(x)) \qquad\qquad \mathsf{c} \leftarrow\!\rtimes\!\!\!\ltimes\!\rightarrow \mathsf{f}(\mathsf{h}(x), \mathsf{g}(\mathsf{h}(\mathsf{a})))$

After adding them as rules

$\mathsf{f}(\mathsf{h}(\mathsf{g}(\mathsf{a})), \mathsf{g}(x)) \rightarrow \mathsf{b} \qquad\qquad \mathsf{f}(\mathsf{h}(x), \mathsf{g}(\mathsf{h}(\mathsf{a}))) \rightarrow \mathsf{c}$

new critical pairs are obtained, one of which is

$\mathsf{b} \leftarrow\!\rtimes\!\!\!\ltimes\!\rightarrow \mathsf{c}$

Since b and c are different normal forms, the extension is obviously non-confluent. Since the new rules can be simulated by the original rules, also $\mathcal{R}$ is non-confluent. Of the tools mentioned, ACP shows confluence by first deriving the rule $\mathsf{g}(\mathsf{a}) \rightarrow \mathsf{a}$ (which can be simulated by existing rules), which then gives rise to a critical pair that extends to a non-joinable peak:

$\mathsf{b} \leftarrow \mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a})) \rightarrow \mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{a}) \rightarrow^* \mathsf{c}$

Saigawa also shows non-confluence but exceeded the 60s time limit in our experiments; it considers critical pairs of the (extended) TRS $\mathcal{R} \cup \mathcal{R}^{-1}$, which includes the rule $\mathsf{g}(\mathsf{a}) \rightarrow \mathsf{a}$. Hence Saigawa finds the same non-joinable peak as ACP. However, CoLL and CSI (without the techniques from this paper) fail.

The remainder of the paper is structured as follows. In Section 3, we describe the theory underlying the addition and removal of rules, and Section 4 is devoted to its integration into CeTA. In Section 5 we briefly sketch our implementation in CSI and present experimental results. Related work is presented in Section 6 before we conclude in Section 7.

[3] A critical pair $s \leftarrow\!\rtimes\!\!\!\ltimes\!\rightarrow t$ is deeply joinable if $u \downarrow v$ for any two reducts u of s and v of t. The example defeats any non-confluence check based on proving non-joinability of peaks starting from critical peaks.

2 Preliminaries

Throughout the paper we assume familiarity with term rewriting; for an introduction to this topic see [4, 25]. We use s and t to denote terms. Given a position p in a term t, $t|_p$ is the subterm at position p of t, and $t[s]_p$ is the result of replacing the subterm $t|_p$ by the term s in t. The letter σ represents a substitution (mapping variables to terms) and $t\sigma$ is the result of applying σ to the term t. For a binary relation R on terms we write $\sigma\ R\ \sigma'$ if $\sigma(x)\ R\ \sigma'(x)$ for all variables x. Term rewrite systems $\mathcal{R}$, $\mathcal{S}$ consist of rewrite rules $\ell \to r$, and induce rewrite relations (e.g., $\to_{\mathcal{R}}$). The relations $\leftarrow$, $\leftrightarrow$, $\to^*$ denote the inverse, the symmetric closure, and the reflexive, transitive closure of $\to$, respectively. Joinability $\downarrow$ and meetability $\uparrow$ are defined by $\downarrow = \to^* \cdot {}^*\!\leftarrow$ and $\uparrow = {}^*\!\leftarrow \cdot \to^*$. Consider two rules $\ell \to r$ and $\ell' \to r' \in \mathcal{R}$ that have been renamed such that ℓ and ℓ' have no variables in common, and a non-variable position p of ℓ. If p is the root position, then we demand that $\ell \to r$ and $\ell' \to r'$ are not variants of each other. If ℓ' and $\ell|_p$ unify with most general unifier σ, then we obtain a critical peak $\ell[r']_p\sigma \leftarrow \ell\sigma \to r\sigma$. We write $\ell[r']_p\sigma \leftarrow\!\rtimes\!\rightarrow r\sigma$ for the corresponding critical pair. We also write $s \leftarrow\!\bowtie\!\rightarrow t$ to denote overlays, i.e., critical pairs that stem from overlaps at the root position, and $s \leftarrow\!\rtimes\!\rightarrow t$ for the other critical pairs.

▶ **Definition 3.** For a TRS $\mathcal{R}$, *multisteps* $\mathrel{\circ\!\!\!\!\rightarrow}_{\mathcal{R}}$ are defined inductively by

- $x \mathrel{\circ\!\!\!\!\rightarrow}_{\mathcal{R}} x$ if x is a variable,
- $\ell\sigma \mathrel{\circ\!\!\!\!\rightarrow}_{\mathcal{R}} r\sigma'$ if $\ell \to r \in \mathcal{R}$ and σ, σ' are substitutions with $\sigma \mathrel{\circ\!\!\!\!\rightarrow}_{\mathcal{R}} \sigma'$, and
- $f(s_1, \ldots, s_n) \mathrel{\circ\!\!\!\!\rightarrow}_{\mathcal{R}} f(t_1, \ldots, t_n)$ if f is a function symbol of arity n and $s_i \mathrel{\circ\!\!\!\!\rightarrow}_{\mathcal{R}} t_i$ for $1 \leqslant i \leqslant n$.

The TRS $\mathcal{R}$ is *development-closed* if every critical pair $s \leftarrow\!\rtimes\!\rightarrow t$ satisfies $s \mathrel{\circ\!\!\!\!\rightarrow} t$. It is *almost development-closed* if $s \mathrel{\circ\!\!\!\!\rightarrow} \cdot {}^*\!\leftarrow t$ for all overlays $s \leftarrow\!\bowtie\!\rightarrow t$ and $s \mathrel{\circ\!\!\!\!\rightarrow} t$ for all other critical pairs $s \leftarrow\!\rtimes\!\rightarrow t$.

Van Oostrom [19] has shown that (almost) development closed TRSs are confluent, extending results by Huet [13] and Toyama [28].

▶ **Example 4.** We revisit Example 1 and show confluence of $\mathcal{R} \cup \{\mathsf{f}(x) \to x\}$. First we establish that $\mathsf{f}^n(x) \mathrel{\circ\!\!\!\!\rightarrow} x$ by induction on n. The claim is trivially true for $n = 0$. Given $\mathsf{f}^{n-1}(x) \mathrel{\circ\!\!\!\!\rightarrow} x$, we can take substitutions σ and σ' that map x to $\mathsf{f}^{n-1}(x)$ and x, respectively, and obtain $\mathsf{f}(x)\sigma \mathrel{\circ\!\!\!\!\rightarrow} x\sigma'$, i.e., $\mathsf{f}^n(x) \mathrel{\circ\!\!\!\!\rightarrow} x$. For each of the critical pairs $s \leftarrow\!\rtimes\!\rightarrow t$, we have either $s \mathrel{\circ\!\!\!\!\rightarrow} t$ or $s \leftarrow\!\bowtie\!\rightarrow t$ and $s \mathrel{\leftarrow\!\!\!\!\circ} t$ (which implies $s \mathrel{\circ\!\!\!\!\rightarrow} \cdot {}^*\!\leftarrow t$). Therefore the TRS is almost development closed and thus confluent.

3 Theory

In this section we present the easy theory behind the use of redundant rules for proving confluence. For adding such rules we use the following folklore result.

▶ **Lemma 5.** *If $\ell \to^*_{\mathcal{R}} r$ for every rule $\ell \to r$ from $\mathcal{S}$ then $\to^*_{\mathcal{R}} = \to^*_{\mathcal{R}\cup\mathcal{S}}$.*

Proof. The inclusion $\to^*_{\mathcal{R}} \subseteq \to^*_{\mathcal{R}\cup\mathcal{S}}$ is obvious. For the reverse direction it suffices to show that $s \to^*_{\mathcal{R}} t$ whenever $s \to_{\mathcal{S}} t$. The latter ensures the existence of a position p in s, a rewrite rule $\ell \to r$ in $\mathcal{S}$, and a substitution σ such that $s|_p = \ell\sigma$ and $t = s[r\sigma]_p$. We obtain $\ell \to^*_{\mathcal{R}} r$ from the assumption of the lemma. Closure (of $\to^*_{\mathcal{R}}$) under contexts and substitutions yields the desired $s \to^*_{\mathcal{R}} t$. ◀

▶ **Corollary 6.** *If $\ell \to^*_{\mathcal{R}} r$ for every rule $\ell \to r$ from $\mathcal{S}$ then $\mathcal{R}$ is confluent if and only if $\mathcal{R} \cup \mathcal{S}$ is confluent.*

Proof. We obtain $\to^*_{\mathcal{R}} = \to^*_{\mathcal{R}\cup\mathcal{S}}$ from the preceding lemma. Hence also $\downarrow_{\mathcal{R}} = \downarrow_{\mathcal{R}\cup\mathcal{S}}$ and $\uparrow_{\mathcal{R}} = \uparrow_{\mathcal{R}\cup\mathcal{S}}$. Therefore

$$\uparrow_{\mathcal{R}} \subseteq \downarrow_{\mathcal{R}} \qquad \iff \qquad \uparrow_{\mathcal{R}\cup\mathcal{S}} \subseteq \downarrow_{\mathcal{R}\cup\mathcal{S}}$$ ◀

▶ **Definition 7.** A rule $\ell \to r \in \mathcal{R}$ is *redundant* if $\ell \to^*_{\mathcal{R}\setminus\{\ell\to r\}} r$.

By Corollary 6, if $\ell \to r \in \mathcal{R}$ is redundant, then $\mathcal{R}$ is confluent if and only if $\mathcal{R} \setminus \{\ell \to r\}$ is confluent. In other words, removing a redundant rule does not affect confluence of a TRS. For removing rules while reflecting[4] confluence (or adding rules while reflecting non-confluence) it suffices that the left- and right-hand side are convertible with respect to the remaining rules.

▶ **Lemma 8.** *If $\ell \leftrightarrow^*_{\mathcal{R}} r$ for every rule $\ell \to r$ from $\mathcal{S}$ then $\leftrightarrow^*_{\mathcal{R}\cup\mathcal{S}} = \leftrightarrow^*_{\mathcal{R}}$.*

Proof. The inclusion $\leftrightarrow^*_{\mathcal{R}} \subseteq \leftrightarrow^*_{\mathcal{R}\cup\mathcal{S}}$ is obvious. For the reverse direction it suffices to show that $s \leftrightarrow^*_{\mathcal{R}} t$ whenever $s \to_{\mathcal{S}} t$. The latter ensures the existence of a position p in s, a rewrite rule $\ell \to r$ in $\mathcal{S}$, and a substitution σ such that $s|_p = \ell\sigma$ and $t = s[r\sigma]_p$. We obtain $\ell \leftrightarrow^*_{\mathcal{R}} r$ from the assumption of the lemma. Closure (of $\leftrightarrow^*_{\mathcal{R}}$) under contexts and substitutions yields the desired $s \leftrightarrow^*_{\mathcal{R}} t$. ◀

▶ **Corollary 9.** *If $\mathcal{R}$ is confluent and $\ell \leftrightarrow^*_{\mathcal{R}} r$ for every rule $\ell \to r$ from $\mathcal{S}$ then $\mathcal{R} \cup \mathcal{S}$ is confluent.*

Proof. From the preceding lemma and the confluence of $\mathcal{R}$ we obtain

$$\leftrightarrow^*_{\mathcal{R}\cup\mathcal{S}} \;=\; \leftrightarrow^*_{\mathcal{R}} \;\subseteq\; \downarrow_{\mathcal{R}} \;\subseteq\; \downarrow_{\mathcal{R}\cup\mathcal{S}}$$

Hence $\mathcal{R} \cup \mathcal{S}$ is confluent. ◀

▶ **Example 10.** Consider the TRS from [10, Example 2] consisting of the five rewrite rules

$$\begin{array}{lll}
\mathsf{hd}(x : y) \to x & \mathsf{inc}(x : y) \to \mathsf{s}(x) : \mathsf{inc}(y) & \mathsf{nats} \to 0 : \mathsf{inc}(\mathsf{nats}) \\
\mathsf{tl}(x : y) \to y & \mathsf{inc}(\mathsf{tl}(\mathsf{nats})) \to \mathsf{tl}(\mathsf{inc}(\mathsf{nats})) &
\end{array}$$

While this system can be shown to be confluent using decreasing diagrams, simply removing the last rule would make confluence obvious, since the remaining four rules constitute an orthogonal TRS. And indeed, because of the following joining sequences, the last rule is superfluous and can be dropped:

$$\begin{array}{l}
\mathsf{inc}(\mathsf{tl}(\mathsf{nats})) \to \mathsf{inc}(\mathsf{tl}(0 : \mathsf{inc}(\mathsf{nats}))) \to \mathsf{inc}(\mathsf{inc}(\mathsf{nats})) \\
\mathsf{tl}(\mathsf{inc}(\mathsf{nats})) \to \mathsf{tl}(\mathsf{inc}(0 : \mathsf{inc}(\mathsf{nats}))) \to \mathsf{tl}(\mathsf{s}(0) : \mathsf{inc}(\mathsf{inc}(\mathsf{nats}))) \to \mathsf{inc}(\mathsf{inc}(\mathsf{nats}))
\end{array}$$

▶ Remark. Some other examples from [10] can be dealt with in a similar fashion: In [10, Example 1] the first rule is joinable using the other rules, and the remaining system is orthogonal. The same argument (with a different joining conversion) applies to [10, Example 5].

Corollary 9 can also be beneficial when dealing with non-left-linear systems, as demonstrated by the following example.

[4] We are interested in transformations that reflect (rather than preserve) confluence, because our goal is automation, and it is natural to work from the conclusion for finding (non-)confluence proofs.

▶ **Example 11.** Consider the TRS from [24] consisting of the four rewrite rules

$$\mathsf{f}(x,x) \to \mathsf{f}(\mathsf{g}(x),\mathsf{g}(x)) \qquad \mathsf{f}(x,y) \to \mathsf{f}(\mathsf{h}(x),\mathsf{h}(y))$$
$$\mathsf{g}(x) \to \mathsf{p}(x) \qquad \mathsf{h}(x) \to \mathsf{p}(x)$$

Because of the conversion

$$\mathsf{f}(x,x) \to \mathsf{f}(\mathsf{h}(x),\mathsf{h}(x)) \to \mathsf{f}(\mathsf{p}(x),\mathsf{g}(x)) \to \mathsf{f}(\mathsf{p}(x),\mathsf{p}(x)) \leftarrow \mathsf{f}(\mathsf{g}(x),\mathsf{p}(x)) \leftarrow \mathsf{f}(\mathsf{g}(x),\mathsf{g}(x))$$

we can remove the first rule. Since the resulting TRS is orthogonal and the removed rule is convertible using the other rules, also the original TRS is confluent.

It can also be beneficial to both add and remove rules. In particular adding a redundant rule can help with removing other, problematic rules, as shown in the following example.

▶ **Example 12.** Consider the TRS consisting of the three rewrite rules[5]

$$\mathsf{f}(x,y) \to \mathsf{f}(\mathsf{g}(x),\mathsf{g}(x)) \qquad \mathsf{f}(x,x) \to \mathsf{a} \qquad \mathsf{g}(x) \to x$$

After adding the rule $\mathsf{f}(x,y) \to \mathsf{a}$, which is justified since $\mathsf{f}(x,y) \to \mathsf{f}(\mathsf{g}(x),\mathsf{g}(x)) \to \mathsf{a}$, we can remove the first two original rules, due to the following conversions:

$$\mathsf{f}(x,y) \to \mathsf{a} \leftarrow \mathsf{f}(\mathsf{g}(x),\mathsf{g}(x)) \qquad \mathsf{f}(x,x) \to \mathsf{a}$$

The resulting TRS is orthogonal and hence confluent. Since the added rule can be simulated by the original rules, and the removed rules are convertible using the new rule, also the original TRS is confluent.

While adding (or removing) rules using Corollary 6 is always safe in the sense that we cannot lose confluence, it is easy to see that the reverse direction of Corollary 9 does not hold in general. That is, removing convertible rules can make a confluent TRS non-confluent as for example witnessed by the two TRSs $\mathcal{R} = \{\mathsf{a} \to \mathsf{b},\ \mathsf{a} \to \mathsf{c}\}$ and $\mathcal{S} = \{\mathsf{b} \to \mathsf{a}\}$. Clearly $\mathcal{R}$ is not confluent, $\mathcal{S} \cup \mathcal{R}$ is confluent, and $\mathsf{b} \leftrightarrow^*_{\mathcal{R}} \mathsf{a}$.

We give one more example, showing that using removal of redundant rules can considerably speed up finding a confluence proof.

▶ **Example 13.** Consider the TRS consisting of the following two rules:

$$\mathsf{f}(x) \to \mathsf{g}(x,\mathsf{f}(x)) \qquad \mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{f}(x)))) \to \mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{g}(x,\mathsf{f}(x)))))$$

This TRS is confluent by the simultaneous critical pair criterion of Okui [18][6] which is implemented by ACP. Alas, there are 58 simultaneous critical pairs and indeed ACP, which implements Okui's criterion, does not terminate in five minutes. While 58 looks small, the simultaneous critical pairs become quite big. For example, with $t = \mathsf{g}(\mathsf{f}^3(\mathsf{g}(x,\mathsf{f}(x))),\mathsf{f}^4(\mathsf{g}(x,\mathsf{f}(x))))$, one of the simultaneous critical pairs is

$$\mathsf{f}^3(\mathsf{g}(\mathsf{f}(\mathsf{g}(\mathsf{f}(t),\mathsf{f}(\mathsf{f}(t)))),\mathsf{f}(\mathsf{f}(\mathsf{g}(\mathsf{f}(t),\mathsf{f}(\mathsf{f}(t)))))))) \mathrel{\leftarrow\!\!\!+\!\!\!\rtimes\!\!\rightarrow} \mathsf{f}^5(\mathsf{g}(\mathsf{f}^3(x),\mathsf{f}^4(x)))$$

and testing joinability using development steps is very expensive. In general, if one takes the rules $\mathsf{f}(x) \to \mathsf{g}(x,\mathsf{f}(x))$ and $\mathsf{f}^n(\mathsf{f}(x)) \to \mathsf{f}^n(\mathsf{g}(x,\mathsf{f}(x)))$, then the number and size of the simultaneous critical pair will grow exponentially in n. However, Corollary 9 is applicable—the second rule can be simulated by the first rule in one step—and showing confluence of the first rule is trivial.

[5] `http://www.nue.riec.tohoku.ac.jp/tools/acp/experiments/rtatlca14/examples/u1.trs`

[6] Note that the given TRS is feebly orthogonal [21]. The key observation here is that any simultaneous critical pair arises from a peak of a development step and a plain rewrite step. By the orthogonalization procedure from [21], we can obtain an equivalent peak of two orthogonal development steps, which is joinable by two development steps, thus satisfying Okui's criterion.

4 Formalization and Certification

Due to the ever increasing interest in automatic analysis of term rewrite systems in the recent years, it is of great importance whether a proof, automatically generated by some tool, is indeed correct. The complexity of the generated proofs makes checking correctness, i.e., certification, impractical for humans. Thus there is a strong interest in automated certification of proofs generated by e.g. confluence or termination tools. This led to the common approach of using proof assistants for certification.

Our technique is particularly well suited for certification for the following reasons. First, since the theory we use is elementary, formalizing it in a proof assistant is entirely straightforward. Moreover the generated proofs, while simple in nature, can become very large, which makes checking them infeasible by hand, but easy for a machine. Finally, as demonstrated in Section 5, the existing certifiable confluence techniques heavily benefit from our transformations.

As certifier we use CeTA [27], originally developed as a tool for certifying termination proofs which have to be provided as certificates in CPF (certification problem format) [23]. Given a certificate CeTA will either answer `CERTIFIED`, or return a detailed error message why the proof was `REJECTED`. Its correctness is formally proven as part of IsaFoR, the **Isa**belle **F**ormalization **o**f **R**ewriting. IsaFoR contains executable "check"-functions for each formalized proof technique together with formal proofs that whenever such a check succeeds, the technique was indeed applied correctly. Isabelle's code-generation facility is used to obtain a trusted Haskell program from these check functions: the certifier CeTA.[7]

Since 2012 CeTA supports checking (non-)confluence certificates [16,26]. Checkable criteria that ensure confluence are:

- Knuth and Bendix' criterion [15],
- (weak) orthogonality [22],
- Huet's result on strongly closed critical pairs [13], and
- the rule labeling heuristic for decreasing diagrams [17,20].

For non-confluence CeTA can check that, given derivations $s \to^* t_1$ and $s \to^* t_2$, t_1 and t_2 cannot be joined. Here the supported justifications are:

- testing that t_1 and t_2 are distinct normal forms,
- testing that $\mathsf{tcap}(t_1\sigma)$ and $\mathsf{tcap}(t_2\sigma)$ are not unifiable [29],
- usable rules, discrimination pairs, argument filters, and interpretations [1], and
- reachability analysis using tree automata [8].

To add support for our transformations to CeTA we formalized the results from Section 3 in Isabelle and integrated them into IsaFoR. The theory `Redundant_Rules.thy` contains the theoretical results, whose formalization, directly following the paper proof, requires a mere 100 lines of Isabelle, stressing the simplicity of the transformations.

We extended CPF for representing proofs using addition and removal of redundant rules and implemented dedicated check functions in the theory `Redundant_Rules_Impl.thy`, enabling CeTA to inspect, i.e., certify such (non-)confluence proofs. A certificate for (non-)confluence of a TRS $\mathcal{R}$ by an application of the redundant rules transformation consists of three parts:

- the modified TRS $\mathcal{R}'$,
- a certificate for the (non-)confluence of $\mathcal{R}'$, and

[7] IsaFoR/CeTA and CPF are available at `http://cl-informatik.uibk.ac.at/software/ceta/`.

- a justification for redundancy of the added and removed rules. Here for the rules that were added, i.e., all $\ell \to r$ in $\mathcal{S} = \mathcal{R}' \setminus \mathcal{R}$, we simply require a bound on the length of the derivations showing $\ell \to_{\mathcal{R}}^{*} r$.[8] For the deleted rules in a non-confluence certificate, i.e., all $\ell \to r$ in $\mathcal{S} = \mathcal{R} \setminus \mathcal{R}'$, the same bound is used for $\ell \to_{\mathcal{R}'}^{*} r$. For a confluence proof one can either give explicit conversions $\ell \leftrightarrow_{\mathcal{R}'}^{*} r$ or rely on the bound again, which then has to ensure $\ell \downarrow_{\mathcal{R}'} r$.

Implementing check functions for such a certificate is then straightforward. We simply compute $\mathcal{S} \setminus \mathcal{R}$ and $\mathcal{R} \setminus \mathcal{S}$ and use the given bound and conversions to ensure redundancy.

Whereas for certification we only need to check that the modified rules really are redundant, the question of how to automatically find suitable rules for addition and deletion is more intricate. In the next section we discuss and evaluate our implementation of three possible approaches in the confluence prover CSI.

5 Implementation and Experiments

CSI features a powerful strategy language [29], which allows to combine confluence techniques in a modular and flexible manner, making it easy to test different strategies that exploit redundant rules.

We tested our implementation on the Cops database[9] using the following three strategies to add and remove rules.

(js) Our first strategy is to add (minimal) joining sequences of critical pairs as rules, i.e., in Corollary 6 choose $\mathcal{S} \subseteq \{s \to u, t \to u \mid s \leftarrow\!\rtimes\!\to t \text{ with } s \to_{\mathcal{R}}^{*} u \text{ and } t \to_{\mathcal{R}}^{*} u\}$. The underlying idea here is that critical peaks become joinable in a single step, which is advantageous for other confluence criteria, for example rule labeling [20].

(rhs) The second strategy for obtaining redundant rules to add, is to rewrite right-hand sides of rules, i.e., in Corollary 6 set $\mathcal{S} = \{\ell \to t \mid \ell \to r \in \mathcal{R} \text{ and } r \to_{\mathcal{R}} t\}$. (This idea has already been used for termination by Zantema [30].) Again the motivation is to produce shorter joining sequences for critical pairs, facilitating the use of other confluence criteria.

(del) For removing rules we search for rules whose left- and right-hand sides are joinable, i.e., in Corollary 9 set $\mathcal{S} = \{\ell \to r \mid \ell \downarrow_{\mathcal{R}} r\}$. This decision is motivated by simplicity of implementation and the fact that for confluent TRSs, joinability and convertibility coincide. Removing rules can benefit confluence proofs by eliminating critical pairs. Since our strategy here is a simple greedy one that removes as many rules as possible, we also lose confluence in some cases.

In the case of adding rules we also discard rules that can be simulated by other rules in a single step. Without this refinement, the gain in power would become smaller, and even disappear for CSI's full strategy. We also implemented and tested three other strategies, which did not yield any additional proofs.

- Inspired by Example 1 we tried to add rules specifically for making rewrite systems development closed. That is, we used $\mathcal{S} = \{s \to t \mid s \leftarrow\!\rtimes\!\to t \text{ with } s \to_{\mathcal{R}}^{*} t \text{ and } s \not\twoheadrightarrow_{\mathcal{R}} t\}$ in Corollary 6. All examples gained by this strategy can also be handled by (js) or (rhs).
- To help with systems containing AC-like rules we tried to add inverted reversible rules, by setting $\mathcal{S} = \{r \to \ell \mid \ell \to r \in \mathcal{R} \text{ with } r \to_{\mathcal{R}}^{*} \ell\}$ in Corollary 6. Again we gained no additional proofs compared to (js) and (rhs).

[8] This bound is necessary, because in Isabelle all functions have to be total and an unbounded search might not terminate.
[9] All TRS problems (276 at the time of writing) from `http://coco.nue.riec.tohoku.ac.jp/cops/`.

Table 1 Experimental Results (✓ = certified).

	CSI	$\mathsf{CSI}_{\mathsf{js}}$	$\mathsf{CSI}_{\mathsf{rhs}}$	$\mathsf{CSI}_{\mathsf{del}}$	$\mathsf{CSI}_{\mathsf{all}}$
yes	155	156	159	163	166
no	47	48	47	47	48
maybe	74	72	70	66	62
	✓CSI	✓$\mathsf{CSI}_{\mathsf{js}}$	✓$\mathsf{CSI}_{\mathsf{rhs}}$	✓$\mathsf{CSI}_{\mathsf{del}}$	✓$\mathsf{CSI}_{\mathsf{all}}$
yes	71	86	73	78	104
no	47	48	47	47	48
maybe	158	142	156	151	124

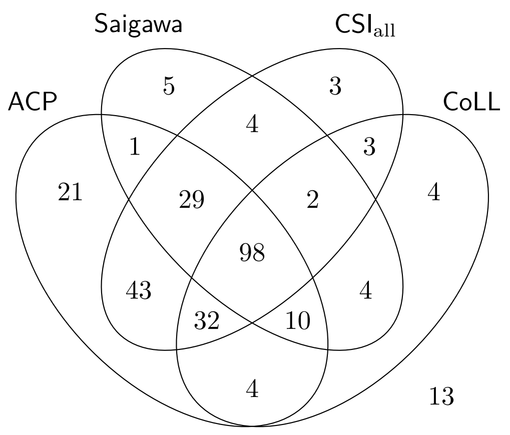

Figure 1 Overlap Between Solved Examples.

- When removing rules we also tried to search for conversions that are not valleys, by using rules in the reverse direction when searching for a join. More precisely, we tried $\mathcal{S} = \{\ell \to r \mid \ell \downarrow_{\mathcal{R}\cup\mathcal{R}^{-1}} r\}$ in Corollary 9. However, this variation only lost examples compared to (del).

The results are shown in Table 1.[10] The experiments were performed on a 48 core 2.2 GHz Opteron 6174 server with 256 GB RAM. We performed two sets of benchmarks, based on CSI's full and certifiable strategies, respectively. For the full strategy, adding joining sequences of critical pairs (js) or rewriting right-hand sides (rhs) show very limited effect, gaining 2 and 4 proofs, respectively. Removing rules (del) is the most effective technique and gains 10 systems while losing 2 other ones. With all techniques combined, 12 new systems can be shown (non-)confluent. Interestingly, the picture for the certifiable strategy is a bit different. Here, (rhs) gains 2 proofs, (js) gains 16 systems and (del) gains 17 proofs while losing 10. Remarkably, in all of those 17 proofs the TRS becomes orthogonal after removing redundant rules, which emphasizes that our transformations can considerably simplify confluence proofs. In total, 34 new systems are shown (non-)confluent.

In Table 2, we compare CSI 0.5,to ACP 0.50, CoLL 1.1, and Saigawa 1.7. Figure 1 shows the examples solved by the four provers in relation to each other.

[10] Detailed results are available at `http://cl-informatik.uibk.ac.at/software/csi/rr-rta2015/`.

Table 2 Comparison of Confluence Provers.

	ACP	CoLL	CSI$_{\text{all}}$	Saigawa
yes	186	141	166	128
no	52	16	48	25
maybe	38	119	62	123

6 Related Work

Our work draws a lot of inspiration from existing literature. One starting point is [21], where van Oostrom introduces the notion of *feeble orthogonality.* A TRS is feebly orthogonal if the critical peaks arising from its non-redundant* rules are trivial or contain a trivial step (that rewrites a term to itself); a rule is redundant* if it can be simulated by another rule in a single step. Clearly our notion of redundancy generalizes redundancy*.

The most important prior work is [2]. In this paper, Aoto and Toyama describe an automated confluence criterion (which has been implemented in ACP) based on decomposing TRSs into a *reversible* part $\mathcal{P}$ and a terminating part $\mathcal{S}$. In order to help applicability of their criterion, they introduce a procedure based on the inference rules

$$\text{replace} \quad \frac{\langle \mathcal{S} \cup \{\ell \to r\}, \mathcal{P}\rangle}{\langle \mathcal{S} \cup \{\ell \to r'\}, \mathcal{P}\rangle} \; r \leftrightarrow_{\mathcal{P}}^* r' \qquad\qquad \text{add} \quad \frac{\langle \mathcal{S}, \mathcal{P}\rangle}{\langle \mathcal{S} \cup \{\ell \to r\}, \mathcal{P}\rangle} \; \ell \leftrightarrow_{\mathcal{P}}^* \cdot \to_{\mathcal{S}}^* r$$

The key is that because $\mathcal{P}$ is reversible, $\leftrightarrow_{\mathcal{P}}^*$ and $\to_{\mathcal{P}}^*$ coincide, and therefore confluence of $\mathcal{S} \cup \mathcal{P}$ is not affected by applying these inference rules. This very same idea underlies Lemma 5, which establishes *reduction equivalence*, and thus Corollary 6. Note that no rule removal is performed in [2].

There is a second connection between our work and [2] that seems noteworthy. Given a reversible $\mathcal{P}$, every rule from $\mathcal{P}^{-1}$ can be simulated by a sequence of $\mathcal{P}$-steps. Therefore, confluence of $\mathcal{S} \cup \mathcal{P}$ and $\mathcal{S} \cup \mathcal{P} \cup \mathcal{P}^{-1}$ coincide by Corollary 6. Using this observation, one could decompose the confluence criteria of [2] into two steps, one that replaces $\mathcal{P}$ by $\mathcal{P} \cup \mathcal{P}^{-1}$, and a respective underlying confluence criterion that does not make use of reversibility, but instead demands that $\mathcal{P}$ is symmetric, i.e., $\mathcal{P}^{-1} \subseteq \mathcal{P}$.

The idea of showing confluence by removing rules whose sides are convertible has already been used in the literature, e.g. [12, Example 11], which is a variation of Example 10.

Other works of interest are [9,30], where Gramlich and Zantema apply a similar idea to Corollary 6 to termination: If some additional requirements are met, then termination of $\mathcal{R} \cup \{\ell \to r\}$ is equivalent to termination of $\mathcal{R} \cup \{\ell \to r'\}$ where $r \to_{\mathcal{R}} r'$ by a non-erasing rule. This is true for non-overlapping TRSs [9, Theorem 4], or when the rule used in the $r \to_{\mathcal{R}} r'$ step is locally confluent by itself, left-linear, and furthermore it doesn't overlap with any rules from $\mathcal{R} \cup \{\ell \to r\}$ except itself [30, Theorem 4].

7 Conclusion

In this work we demonstrated how a very simple technique, namely adding and removing redundant rules, can boost the power of automated confluence provers. It is easy to implement and we believe that also confluence tools other than CSI could benefit from such transformations, not only increasing their power, but also simplifying the generated proofs. Moreover the technique is well-suited for certification, resulting in more trustworthy proofs. In particular we could significantly increase the number of certifiable confluence proofs in our experiments—by almost 50%. Interestingly we observed that all 17 of the systems gained by ✓(del) become orthogonal by removing redundant rules. This might be due to the fact that when designing example TRSs for new techniques, one often works by systematically making existing criteria non-applicable and removing rules can undo this effort.

As future work we plan to investigate more elaborate strategies for finding useful redundant rules, both for addition and removal (where the candidates are limited, but performing the transformation might lose confluence). Here one direction to explore might be the use of machine learning techniques to devise such strategies automatically.

Acknowledgments. The comments by the anonymous reviewers helped to improve the presentation.

References

1 T. Aoto. Disproving confluence of term rewriting systems by interpretation and ordering. In P. Fontaine, editor, *Proc. 9th International Workshop on Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Artificial Intelligence*, pages 311–326, 2013. doi: 10.1007/978-3-642-40885-4_22.

2 T. Aoto and Y. Toyama. A reduction-preserving completion for proving confluence of non-terminating term rewriting systems. *Logical Methods in Computer Science*, 8(1:31):1–29, 2012. doi: 10.2168/LMCS-8(1:31)2012.

3 T. Aoto, J. Yoshida, and Y. Toyama. Proving confluence of term rewriting systems automatically. In R. Treinen, editor, *Proc. 20th International Conference on Rewriting Techniques and Applications*, volume 5595 of *Lecture Notes in Computer Science*, pages 93–102, 2009. doi: 10.1007/978-3-642-02348-4_7.

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

5 F. Blanqui and A. Koprowski. 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.

6 E. Contejean, P. Courtieu, J. Forest, O. Pons, and Xavier Urbain. Automated certified proofs with CiME3. In M. Schmidt-Schauß, editor, *Proc. 22nd International Conference on Rewriting Techniques and Applications*, pages 21–30, 2011. doi: 10.4230/LIPIcs.RTA.2011.21.

7 B. Felgenhauer. A proof order for decreasing diagrams. In N. Hirokawa and A. Middeldorp, editors, *Proc. 1st International Workshop on Confluence*, pages 7–14, 2012. `http://cl-informatik.uibk.ac.at/events/iwc-2012/`.

8 B. Felgenhauer and R. Thiemann. Reachability analysis with state-compatible automata. In A.-H. Dediu, editor, *Proc. 8th International Conference on Language and Automata Theory and Applications*, volume 8370 of *Lecture Notes in Computer Science*, pages 347–359, 2013. doi: 10.1007/978-3-319-04921-2_28.

9 B. Gramlich. Simplifying termination proofs for rewrite systems by preprocessing. In F. Pfenning, editor, *Proc. 2nd ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, pages 139–150, 2000. doi: 10.1145/351268.351286.

10 B. Gramlich and S. Lucas. Generalizing Newman's lemma for left-linear rewrite systems. In F. Pfenning, editor, *Proc. 17th International Conference on Rewriting Techniques and Applications*, volume 4098 of *Lecture Notes in Computer Science*, pages 66–80, 2006. doi: 10.1007/11805618_6.

11 N. Hirokawa and D. Klein. Saigawa: A confluence tool. In N. Hirokawa and A. Middeldorp, editors, *Proc. 1st International Workshop on Confluence*, page 49, 2012. `http://cl-informatik.uibk.ac.at/events/iwc-2012/`.

12 N. Hirokawa and A. Middeldorp. Decreasing diagrams and relative termination. *Journal of Automated Reasoning*, 47(4):481–501, 2011. doi: 10.1007/s10817-011-9238-x.

13 G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980. doi: 10.1145/322217.322230.

14 D. Klein and N. Hirokawa. Confluence of non-left-linear TRSs via relative termination. In N. Bjørner and A. Voronkov, editors, *Proc. 18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 7180 of *Lecture Notes in Computer Science (Advanced Research in Computing and Software Science)*, pages 258–273, 2012.

15 D.E. Knuth and P.B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

16 J. Nagele and R. Thiemann. Certification of confluence proofs using CeTA. In T. Aoto and D. Kesner, editors, *Proc. 3rd International Workshop on Confluence*, pages 19–23, 2014. `http://www.nue.riec.tohoku.ac.jp/iwc2014/`.

17 J. Nagele and H. Zankl. Certified rule labeling. In M. Fernández, editor, *Proc. 26th International Conference on Rewriting Techniques and Applications*, volume 36 of *Leibniz International Proceedings in Informatics*, 2015. This volume.

18 S. Okui. Simultaneous critical pairs and Church-Rosser property. In T. Nipkow, editor, *Proc. 9th International Conference on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 2–16, 1998. doi: 10.1007/BFb0052357.

19 V. van Oostrom. Developing developments. *Theoretical Computer Science*, 175(1):159–181, 1997. doi: 10.1016/S0304-3975(96)00173-9.

20 V. van Oostrom. Confluence by decreasing diagrams – converted. In A. Voronkov, editor, *Proc. 19th International Conference on Rewriting Techniques and Applications*, volume 5117 of *Lecture Notes in Computer Science*, pages 306–320, 2008. doi: 10.1007/978-3-540-70590-1_21.

21 V. van Oostrom. Feebly not weakly. In *Proc. 7th International Workshop on Higher-Order Rewriting*, Vienna Summer of Logic flash drive, 2014.

22 B.K. Rosen. Tree-manipulating systems and Church-Rosser theorems. *Journal of the ACM*, 20(1):160–187, 1973. doi: 10.1145/321738.321750.

23 C. Sternagel and R. Thiemann. The certification problem format. In G. Klein and R. Gamboa, editors, *Proc. 11th International Workshop on User Interfaces for Theorem Provers*, volume 167 of *Electronic Proceedings in Theoretical Computer Science*, pages 61–72, 2014. doi: 10.4204/EPTCS.167.8.

24 T. Suzuki, T. Aoto, and Y. Toyama. Confluence proofs of term rewriting systems based on persistency. *Computer Software*, 30(3):148–162, 2013. In Japanese, doi: 10.11309/jssst.30.3_148.

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

26 R. Thiemann. Certification of confluence proofs using CeTA. In N. Hirokawa and A. Middeldorp, editors, *Proc. 1st International Workshop on Confluence*, page 45, 2012. `http://cl-informatik.uibk.ac.at/events/iwc-2012/`.

27 R. Thiemann and C. Sternagel. Certification of termination proofs using CeTA. In S. Berghofer, editor, *Proc. 22nd International Conference on Theorem Proving in Higher Order Logics*, volume 5674 of *Lecture Notes in Computer Science*, pages 452–468, 2009. doi: 10.1007/978-3-642-03359-9_31.

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

29 H. Zankl, B. Felgenhauer, and A. Middeldorp. CSI – A confluence tool. In N. Bjørner and V. Sofronie-Stokkermans, editors, *Proc. 23rd International Conference on Automated Deduction*, volume 6803 of *Lecture Notes in Artificial Intelligence*, pages 499–505, 2011. doi: 10.1007/978-3-642-22438-6_38.

30 H. Zantema. Reducing right-hand sides for termination. In V. van Oostrom A. Middeldorp and and F. van Raamsdonk, editors, *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion of his 60th Birthday*, volume 3838 of *Lecture Notes in Computer Science*, pages 173–197, 2005. doi: 10.1007/11601548_12.