Ground Confluence Prover Based on Rewriting Induction

Takahito Aoto[1] **and Yoshihito Toyama**[2]

1 **Faculty of Engineering, Niigata University, Niigata, Japan**
aoto@ie.niigata-u.ac.jp
2 **RIEC, Tohoku University, Sendai, Japan**
toyama@riec.tohoku.ac.jp

Abstract

Ground confluence of term rewriting systems guarantees that all ground terms are confluent. Recently, interests in proving confluence of term rewriting systems automatically has grown, and confluence provers have been developed. But they mainly focus on confluence and not ground confluence. In fact, little interest has been paid to developing tools for proving ground confluence automatically. We report an implementation of a ground confluence prover based on rewriting induction, which is a method originally developed for proving inductive theorems.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems, I.2.3 Deduction and Theorem Proving

Keywords and phrases Ground Confluence, Rewriting Induction, Non-Orientable Equations, Term Rewriting Systems

Digital Object Identifier 10.4230/LIPIcs.FSCD.2016.33

1 Introduction

Ground confluence of term rewriting systems (TRSs for short) guarantees that all ground terms are confluent. Not (general) confluence but ground confluence often matters in applications where not equational validity but *inductive validity* is of concern, including refutational completeness of inductive theorem proving and correctness of program transformations (e.g. [10, 11, 21]).

Classical works on ground confluence include [7, 16]. These works stem from *inductionless induction* which has been studied in the context of proving inductive validity by variations of Knuth-Bendix completion. Further studies on ground confluence orient for dealing with expressive rewrite rules over complex data structures, such as order-sorted signature, conditional rewrite rules and regular tree language constraints [7, 8, 15]. In this context, Bouhoula [8] reported on a tool for proving ground confluence. However, not only his procedure assumes reductivity of the system (a stronger notion of termination for conditional TRSs), but it also depends on a procedure for proving joinable inductive theorems which have to be dealt with a specialized proof procedure [9].

Confluence implies ground confluence but not vice versa as witnessed by:

▶ **Example 1.** Let $\mathcal{F} = \{\mathsf{plus} : \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Nat}, \mathsf{s} : \mathsf{Nat} \to \mathsf{Nat}, 0 : \mathsf{Nat}\}$ and

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{plus}(0,0) & \to & 0 & (a) \\ \mathsf{plus}(x,\mathsf{s}(y)) & \to & \mathsf{s}(\mathsf{plus}(y,x)) & (c) \end{array} \quad \begin{array}{lcll} \mathsf{plus}(\mathsf{s}(x),y) & \to & \mathsf{s}(\mathsf{plus}(x,y)) & (b) \end{array} \right\}$$

$\mathcal{R}$ is not confluent, as $\mathsf{s}(\mathsf{s}(\mathsf{plus}(y,x))) \leftarrow \mathsf{s}(\mathsf{plus}(x,\mathsf{s}(y))) \leftarrow \mathsf{plus}(\mathsf{s}(x),\mathsf{s}(y))) \to \mathsf{s}(\mathsf{plus}(y,\mathsf{s}(x))) \to \mathsf{s}(\mathsf{s}(\mathsf{plus}(x,y)))$. However, $\mathcal{R}$ is ground confluent.

Recently, several confluence provers have been developed (e.g. [6, 23, 17, 27]) and results in automatable techniques for confluence proving. In contrast, it seems that little interest has been paid to the development of tools for proving ground confluence automatically.

In this paper, we report an implementation of a ground confluence prover. Key features of our prover are as follows:

- It is based on a simple framework: our framework is many-sorted (first-order) TRSs; considering not uni-sorted but many-sorted signature is a minimal requirement for natural setting of the problem as it involves inductive arguments.
- It requires a minimal input: the input of our tool is only a description of a many-sorted TRS. In particular, we do not assume the ordering for making systems reductive and a partition of function symbols into constructors and defined symbols—this is in contrast to the setting often found in the literature of ground confluence [7, 8, 15].
- It employs a simple method: our tool is based on rewriting induction, which is nowadays a well-understood method for inductive theorem proving (e.g. [1]). We anticipate it should be easy to develop similar (or even more sophisticated) tools based on our method.

Furthermore, we have prepared a collection of examples which can be used to estimate the status of the power of ground confluence proving tools.

2 Preliminaries

We assume basic familiarity with (many-sorted) term rewriting (e.g. [24]).

We use $\uplus$ for the disjoint union and $\setminus$ for the subtraction. The transitive reflexive (reflexive, symmetric, reflexive symmetric, equivalence) closure of a relation $\to$ is denoted by $\xrightarrow{*}$ (resp. $\xrightarrow{=}$, $\leftrightarrow$, $\xleftrightarrow{=}$, $\xleftrightarrow{*}$). For any quasi-order $\succsim$, we put $\succ = \succsim \setminus \precsim$ and $\approx = \succsim \cap \precsim$. A quasi-order $\succsim$ is *well-founded* if so is its strict part $\succ$. We abuse a set notation $\{a_1, \ldots, a_n\}$ with multisets. The *multiset extension* of a partial order $\succ$ is denoted by $\succ_m$.

Let $\mathcal{S}$ be a set of *sorts*. Each *many-sorted function* f is equipped with its *sort declaration* $f : \alpha_1 \times \cdots \times \alpha_n \to \alpha_0$, where $\alpha_0, \ldots, \alpha_n \in \mathcal{S}$ $(n \geq 0)$; the *arity* n is denoted by $ar(f)$. The set of *terms* over the set of many-sorted function symbols $\mathcal{F}$ and the set of variables $\mathcal{V}$ is denoted by $\mathrm{T}(\mathcal{F}, \mathcal{V})$. The set of function symbols (variables) contained in a term t is denoted by $\mathcal{F}(t)$ (resp. $\mathcal{V}(t)$). The set of *ground terms* over $\mathcal{G} \subseteq \mathcal{F}$ is denoted by $\mathrm{T}(\mathcal{G})$. The set of *positions* of a term t is denoted by $\mathrm{Pos}(t)$. The *empty* position is denoted by ϵ. The symbol in t at position p is denoted by $t(p)$.

A *context* is a term containing a special constant $\Box$, called a *hole*. Let C be a context containing precisely one hole. Then the term obtained from C by replacing the hole with t is denoted by $C[t]$. A *substitution* is a mapping from $\mathcal{V}$ to $\mathrm{T}(\mathcal{F}, \mathcal{V})$. A *ground substitution* is a mapping from $\mathcal{V}$ to $\mathrm{T}(\mathcal{F})$. For substitutions σ, usually it is required that the domain $\{x \in \mathcal{V} \mid \sigma(x) \neq x\}$ of σ is finite, but we omit that condition to ease the notation so that $t\sigma_g$ is a ground term for any term t and ground substitution σ_g. A *most general unifier* of terms s and t is denoted by $\mathrm{mgu}(s, t)$. A *rewrite relation (quasi-order)* is a relation (resp. quasi-order) on terms *closed under contexts and substitutions*. A rewrite relation (quasi-order) is a *reduction* relation (resp. quasi-order) if it is well-founded.

(Indirected) *equations* $l \doteq r$ and $r \doteq l$ are identified. A *directed equation* is denoted by $l \to r$. For a set E of equations (directed equations) the smallest rewrite relation containing E is denoted by $\leftrightarrow_E$ (resp. $\to_E$). For a set of directed equations E, let $\mathrm{LHS}(E) = \{l \mid l \to r \in E\}$ and $\mathrm{LHS}(f, E) = \{l \mid l \to r \in E, l(\epsilon) = f\}$ for each $f \in \mathcal{F}$. A directed equation $l \to r$ is a *rewrite rule* if $l \notin \mathcal{V}$ and $\mathcal{V}(l) \supseteq \mathcal{V}(r)$ hold. A (many-sorted) *term rewriting system* (*TRS* for short) is a finite set of rewrite rules. The set of $\mathcal{R}$-normal forms is denoted by $\mathrm{NF}(\mathcal{R})$. The set of *critical pairs* of a TRS $\mathcal{R}$ is denoted by $\mathrm{CP}(\mathcal{R})$.

A TRS $\mathcal{R}$ is *terminating* if $\to_{\mathcal{R}}$ is well-founded. Terms s and t are *joinable* w.r.t. the rewrite relation $\to_{\mathcal{R}}$ (denoted by $s \downarrow_{\mathcal{R}} t$) if $s \xrightarrow{*}_{\mathcal{R}} u$ and $t \xrightarrow{*}_{\mathcal{R}} u$ for some u. A TRS $\mathcal{R}$ is *(ground) confluent* if $s \downarrow_{\mathcal{R}} t$ holds for any (ground) terms s, t such that $u \xrightarrow{*}_{\mathcal{R}} s$ and $u \xrightarrow{*}_{\mathcal{R}} t$ for some (resp. ground) term u. Terms s and t are *ground convertible* if $s\sigma_g \overset{*}{\leftrightarrow}_{\mathcal{R}} t\sigma_g$ holds for any ground substitution σ_g. An equation $s \doteq t$ is an *inductive theorem* of a TRS $\mathcal{R}$, or *inductively valid* in $\mathcal{R}$, if s and t are ground convertible. We write $\mathcal{R} \models_{ind} E$ for a set E of equations if every $s \doteq t \in E$ is an inductive theorem. Let $\succsim$ be a rewrite quasi-order. We write $s \overset{*}{\leftrightarrow}_{\succsim\mathcal{R}} t$ if there exists $s = u_0 \leftrightarrow u_1 \leftrightarrow_{\mathcal{R}} \cdots \leftrightarrow_{\mathcal{R}} u_n = t$ such that $s \succsim u_i$ or $t \succsim u_i$ for every u_i $(1 \le i \le n)$. Terms s and t are *bounded ground convertible* w.r.t. $\succsim$ if $s\sigma_g \overset{*}{\leftrightarrow}_{\succsim\mathcal{R}} t\sigma_g$ holds for any ground substitution σ_g. A set E of equations is bounded ground convertible if s and t are bounded ground convertible for any $s \doteq t \in E$.

The next lemma is a direct consequence of a generalized Newman's Lemma [26]; see Exercise 1.3.12 in [24].

▶ **Lemma 2.** *Let $\succsim$ be a reduction quasi-order and $\mathcal{R}$ be a TRS such that $\mathcal{R} \subseteq \succ$. If $\mathrm{CP}(\mathcal{R})$ is bounded ground convertible w.r.t. $\succsim$, then $\mathcal{R}$ is ground confluent.*

We consider a partition of function symbols into the set $\mathcal{D}$ of *defined symbols*, and the set $\mathcal{C}$ of *constructors* i.e. $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$. Terms in $\mathrm{T}(\mathcal{C}, V)$ are *constructor terms*. Then a mapping from $\mathcal{V}$ to $\mathrm{T}(\mathcal{C})$ is called a *ground constructor substitution*. A term of the form $f(c_1, \ldots, c_n)$ for some $f \in \mathcal{D}$ and $c_1, \ldots, c_n \in \mathrm{T}(\mathcal{C}, \mathcal{V})$ is said to be *basic*. The set of basic subterms of s is written as $\mathcal{B}(s)$. A TRS $\mathcal{R}$ is said to be *quasi-reducible* if no ground basic terms are normal. Clearly, if $\mathcal{R}$ is a quasi-reducible terminating TRS then for any ground term s there exists t such that $s \xrightarrow{*} t \in \mathrm{T}(\mathcal{C})$.

3 Rewriting Induction for Ground Confluence

We now provide a background theory of our tool. In this section, we put $\mathcal{D} = \{l(\epsilon) \mid l \to r \in \mathcal{R}\}$ and $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$. The fundamental ingredient of our ground confluence prover is the following inference system of rewriting induction.

▶ **Definition 3** (rewriting induction for ground confluence)**.** The input of a rewriting induction procedure is a TRS $\mathcal{R}$, a set E of equations and a reduction quasi-order $\succsim$ such that $\mathcal{R} \subseteq \succ$. In Figure 1, we list the inference rules of the rewriting induction that act on pairs of a set of equations and a set of directed equations. We write $\langle E, H\rangle \rightsquigarrow \langle E', H'\rangle$ when an inference rule is applied (from upper to lower). The procedure (non-deterministically) generates a derivation starting from $\langle E, \emptyset\rangle$. If the derivation ends with some $\langle \emptyset, H\rangle$ (i.e. $\langle E, \emptyset\rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H\rangle$ for some H), then the procedure succeeds. The procedure fails if there are no inference rules to apply. The derivation may also diverge.

In the figure, we use $\circ$ for the composition of relations and Expd is defined as:

$$\mathrm{Expd}_u(s, t) = \{C[r]\sigma \to t\sigma \mid s = C[u], \sigma = \mathrm{mgu}(u, l), l \to r \in \mathcal{R}\}.$$

For a set H of directed equations and a reduction quasi-order $\succsim$, we let

$$\mathrm{inv}(H) = \{r \to l \mid l \to r \in H\} \quad \text{and} \quad H^{\diamond} = \{l\sigma \to r\sigma \mid l \to r \in H, l\sigma \diamond r\sigma\}$$

where $\diamond \in \{\succ, \succsim\}$. Note that, for each $\diamond \in \{\succ, \succsim\}$, $s \to_{H^{\diamond}} t$ iff $s \to_H t$ and $s \diamond t$.

▶ Remark. In contrast to usual rewriting induction system for proving inductive theorems (see e.g. [1]), $s \succ t$ is not assumed in *Expand* rule. The point is essential to deal with

Expand

$$\frac{\langle E \uplus \{s \doteq t\},\ H\rangle}{\langle E \cup \{s'_i \doteq t_i\}_i,\ H \cup \{s \to t\}\rangle} \quad \begin{array}{l} u \in \mathcal{B}(s), \{s_i \to t_i\}_i = \mathrm{Expd}_u(s,t), \\ s_i \xrightarrow[H\cup \mathrm{inv}(H)^{\succsim}]{*} s'_i \end{array}$$

Simplify

$$\frac{\langle E \uplus \{s \doteq t\},\ H\rangle}{\langle E \cup \{s' \doteq t\},\ H\rangle}\ s \to_{\mathcal{R}\cup H^{\succ}} \circ \xrightarrow[H\cup \mathrm{inv}(H)^{\succsim}]{*} s'$$

Delete

$$\frac{\langle E \uplus \{s \doteq t\},\ H\rangle}{\langle E,\ H\rangle}\ s \overset{=}{\leftrightarrow}_H t$$

Figure 1 Inference rules of rewriting induction.

non-orientable equations. In the system of [2], $s \to_{H\cup \mathrm{inv}(H)^{\succ}} s'$ is allowed in *Simplify* rule, but here only $s \to_{H^{\succ}} s'$ is allowed. Compared to the system in [13], elements of H are directed equations to keep record of information on which side is expanded in *Expand* rule. These modifications are required to guarantee the bounded ground convertibility.

▶ **Example 4.** Let us consider $\mathcal{R}$ of Example 1. Then, we have the following rewriting induction derivation of from $\langle E_0, \emptyset\rangle$ where $E_0 = \mathrm{CP}(\mathcal{R})$.

$$\begin{array}{ll}
 & \langle \{\mathsf{s}(\mathsf{plus}(x,\mathsf{s}(y))) \doteq \mathsf{s}(\mathsf{plus}(y,\mathsf{s}(x)))\},\ \emptyset\rangle \\
\overset{*}{\rightsquigarrow}_{Simplify} & \langle \{\mathsf{s}(\mathsf{s}(\mathsf{plus}(y,x))) \doteq \mathsf{s}(\mathsf{s}(\mathsf{plus}(x,y)))\},\ \emptyset\rangle \\
\rightsquigarrow_{Expand} & \langle \{\mathsf{s}(\mathsf{s}(0)) \doteq \mathsf{s}(\mathsf{s}(0)),\ \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{plus}(y',x)))) \doteq \mathsf{s}(\mathsf{s}(\mathsf{plus}(x,\mathsf{s}(y')))), \\
 & \qquad\qquad \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{plus}(x',y)))) \doteq \mathsf{s}(\mathsf{s}(\mathsf{plus}(\mathsf{s}(x'),y)))\}, \\
 & \{\mathsf{s}(\mathsf{s}(\mathsf{plus}(y,x))) \to \mathsf{s}(\mathsf{s}(\mathsf{plus}(x,y)))\}\rangle \\
\overset{*}{\rightsquigarrow}_{Simplify} & \langle \{\mathsf{s}(\mathsf{s}(0)) \doteq \mathsf{s}(\mathsf{s}(0)),\ \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{plus}(y',x)))) \doteq \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{plus}(y',x)))), \\
 & \qquad\qquad \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{plus}(x',y)))) \doteq \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{plus}(x',y))))\}, \\
 & \{\mathsf{s}(\mathsf{s}(\mathsf{plus}(y,x))) \to \mathsf{s}(\mathsf{s}(\mathsf{plus}(x,y)))\}\rangle \\
\overset{*}{\rightsquigarrow}_{Delete} & \langle \emptyset,\ \{\mathsf{s}(\mathsf{s}(\mathsf{plus}(y,x))) \to \mathsf{s}(\mathsf{s}(\mathsf{plus}(x,y)))\}\rangle
\end{array}$$

Then, the rewriting induction procedure returns success.

Henceforth, we assume a reduction quasi-order $\succsim$ such that $\mathcal{R} \subseteq \succ$—using standard techniques in automated termination proof of TRSs, such a reduction quasi-order can be searched efficiently using SAT-solvers (e.g. [12]).

A rewriting induction derivation $\langle E_0, H_0\rangle \rightsquigarrow \langle E_1, H_1\rangle \rightsquigarrow \cdots$ is said to be *fair* if $\bigcup_{j\geq 0}\bigcap_{i\geq j} E_i = \emptyset$. In the following lemmas, let us fix a fair derivation $\langle E_0, H_0\rangle \rightsquigarrow \langle E_1, H_1\rangle \rightsquigarrow \cdots$ with $H_0 = \emptyset$. Let $E_\infty = \bigcup_i E_i$. As $H_0 = \emptyset$, it easily follows $H_i \subseteq E_\infty$ from the inference rules of rewriting induction. The next relations are used to characterize an ordering constraint induced by fair derivations.

▶ **Definition 5.** Let $\succsim$ be a well-founded quasi-order, $\underset{1}{\to}, \underset{2}{\to}$ binary relations, and $\underset{1,2}{\leftrightarrow} = \underset{1}{\leftrightarrow} \cup \underset{2}{\leftrightarrow}$.

- $x \overset{*}{\underset{\succ 2}{\leftrightarrow}} y$ iff there exists $x = x_0 \underset{2}{\leftrightarrow} x_1 \underset{2}{\leftrightarrow} \cdots \underset{2}{\leftrightarrow} x_n = y$ such that $\{x,y\} \succ_m \{x_i, x_{i+1}\}$ for every $x_i \underset{2}{\leftrightarrow} x_{i+1}$.
- $x \overset{*}{\underset{\succsim 1,\succ 2}{\leftrightarrow}} y$ iff there exists $x = x_0 \underset{1,2}{\leftrightarrow} x_1 \underset{1,2}{\leftrightarrow} \cdots \underset{1,2}{\leftrightarrow} x_n = y$ such that $x \succsim x_i$ or $y \succsim x_i$ for every x_i and $\{x,y\} \succ_m \{x_i, x_{i+1}\}$ for every $x_i \underset{2}{\leftrightarrow} x_{i+1}$.

Note that $\{x,y\} \succ_m \{x_i, x_{i+1}\}$ implies $x \succsim x_i$ or $y \succsim x_i$ (and $x \succsim x_{i+1}$ or $y \succsim x_{i+1}$).

Before showing the characterization of fair rewriting induction derivations, let us show a consequence of the ordering constraint.

▶ **Lemma 6** (conversion lemma). *Suppose $x \underset{2}{\leftrightarrow} y$ implies $x \overset{*}{\leftrightarrow}_{\gtrsim 1,\succ 2} y$ for any x, y. Then, $x \underset{2}{\leftrightarrow} y$ implies $x \overset{*}{\leftrightarrow}_{\gtrsim 1} y$ for any x, y.*

Proof. By induction on the multiset $\{x, y\}$ w.r.t. $\succ_m$. Suppose $x \underset{2}{\leftrightarrow} y$. Then by our assumption, $x = x_0 \underset{1,2}{\leftrightarrow} x_1 \underset{1,2}{\leftrightarrow} \cdots \underset{1,2}{\leftrightarrow} x_n = y$ for some $x_0, \ldots, x_n$ $(n \leq 0)$. If $x_i \underset{1}{\leftrightarrow} x_{i+1}$ for all $0 \leq i \leq n-1$, then the claim follows. Suppose $x_i \underset{2}{\leftrightarrow} x_{i+1}$ for some i. Then, by our assumption, $\{x, y\} \succ_m \{x_i, x_{i+1}\}$ holds. Thus, by induction hypothesis, $x_i \overset{*}{\leftrightarrow}_{\gtrsim 1} x_{i+1}$ holds. Let $x_i = z_0 \underset{1}{\leftrightarrow} z_1 \underset{1}{\leftrightarrow} \cdots \underset{1}{\leftrightarrow} z_p = x_{i+1}$. Then, every z_j satisfies $x_i \gtrsim z_j$ or $x_{i+1} \gtrsim z_j$. Hence, we have $x \gtrsim z_j$ or $y \gtrsim z_j$ as $\{x, y\} \succ_m \{x_i, x_{i+1}\}$. Thus, by replacing each $x_i \underset{2}{\leftrightarrow} x_{i+1}$ by $x_i \overset{*}{\leftrightarrow}_{\gtrsim 1} x_{i+1}$, we obtain $x \overset{*}{\leftrightarrow}_{\gtrsim 1} y$. ◀

Thus the characterization is directly connected to bounded (ground) convertibility, and consequently, to ground confluence via Lemma 2. To show the characterization, we need the following property of the operation Expd [1].

▶ **Proposition 7** (property of Expd). *Suppose $\mathcal{R}$ is a quasi-reducible TRS and $u \in \mathcal{B}(s)$. Then, for any ground constructor substitution σ_{gc}, we have $s\sigma_{gc} \to_{\mathcal{R}} \circ \to_{\mathrm{Expd}_u(s,t)} t\sigma_{gc}$.*

We now prove the announced characterization.

▶ **Lemma 8.** *For any $s \doteq t \in E_\infty$ and ground substitution σ_g, $s\sigma_g \overset{*}{\leftrightarrow}_{\gtrsim \mathcal{R}, \succ E_\infty} t\sigma_g$ holds.*

Proof. If σ_g is not a ground constructor substitution on $\mathcal{V}(s) \cup \mathcal{V}(t)$, then $s\sigma_g \overset{+}{\to}_{\mathcal{R}} s\rho_g$ (or $t\sigma_g \overset{+}{\to}_{\mathcal{R}} t\rho_g$) for some ground constructor substitution ρ_g. Then, we have $s\sigma_g \overset{+}{\to}_{\mathcal{R}} s\rho_g \leftrightarrow_{E_\infty} t\rho_g \overset{*}{\leftarrow}_{\mathcal{R}} t\sigma_g$. Thus, from $\mathcal{R} \subseteq \succ$, one easily obtains $s\sigma_g \overset{*}{\leftrightarrow}_{\gtrsim \mathcal{R}, \succ E_\infty} t\sigma_g$. It remains to consider the case that σ_g is a ground constructor substitution. By the fairness assumption, some inference rule is applied to $s \doteq t$ in some step. We distinguish three cases by the inference rule applied. Note that $H \subseteq E_\infty$ in the following cases.

(Expand) Then we have $\langle E \uplus \{s \doteq t\}, H\rangle \rightsquigarrow \langle E \cup \{s'_i \doteq t_i\}_i, \{s \to t\} \cup H\rangle$, where $\mathrm{Expd}_u(s,t) = \{s_i \to t_i\}_i$, $u \in \mathcal{B}(s)$, and $s_i \overset{*}{\to}_{H \cup \mathrm{inv}(H)^\gtrsim} s'_i$ for each i. Since σ_g is a ground constructor substitution, by Proposition 7, we have $s\sigma_g \to_{\mathcal{R}} s_i\theta_g \to_{\mathrm{Expd}_u(s,t)} t\sigma_g$ for some θ_g and i. Thus, $s\sigma_g \to_{\mathcal{R}} s_i\theta_g \overset{*}{\to}_{H \cup \mathrm{inv}(H)^\gtrsim} s'_i\theta_g \leftrightarrow_{E_\infty} t\sigma_g$ for each i. By $\mathcal{R} \subseteq \succ$, we have $s\sigma_g \succ s_i\theta_g$. Then, for any step $u_g \leftrightarrow_H v_g$ in $s_i\theta_g \overset{*}{\leftrightarrow}_{H \cup \mathrm{inv}(H)^\gtrsim} s'_i\theta_g$, we have $s\sigma_g \succ u_g, v_g$, and hence $\{s\sigma_g, t\sigma_g\} \succ_m \{u_g, v_g\}$. Thus, $s\sigma_g \overset{*}{\leftrightarrow}_{\gtrsim \mathcal{R}, \succ E_\infty} t\sigma_g$.

(Simplify) Then we have $\langle E \uplus \{s \doteq t\}, H\rangle \rightsquigarrow \langle E \cup \{s' \doteq t\}, H\rangle$ for some E, H, where $s \to_{\mathcal{R} \cup H^\succ} \hat{s} \overset{*}{\to}_{H \cup \mathrm{inv}(H)^\gtrsim} s'$. Then, $s \to_{\mathcal{R} \cup H^\succ} \hat{s} = s_1 \leftrightarrow_H s_2 \leftrightarrow_H \cdots \leftrightarrow_H s_k = s' \leftrightarrow_{E_\infty} t$ with $s_i \gtrsim s_{i+1}$ for $i = 1, \ldots, k-1$. We distinguish two cases.

1. Case $s \to_{\mathcal{R}} \hat{s}$. Then by $\mathcal{R} \subseteq \succ$, $s\sigma_g \succ \hat{s}\sigma_g$ and thus $s\sigma_g \succ s_i\sigma_g$ for $i = 1, \ldots, k$. Hence, we have $\{s\sigma_g, t\sigma_g\} \succ_m \{s_i\sigma_g, s_{i+1}\sigma_g\}$ for $i = 1, \ldots, k-1$ and $\{s\sigma_g, t\sigma_g\} \succ_m \{s'\sigma_g, t\sigma_g\}$. Thus, $s\sigma_g \overset{*}{\leftrightarrow}_{\gtrsim \mathcal{R}, \succ E_\infty} t\sigma_g$.
2. Case $s \to_{H^\succ} \hat{s}$. Then $s \leftrightarrow_{E_\infty} \hat{s}$ with $s \succ \hat{s}$ and $s\sigma_g \succ s_i\sigma_g$ for all $i = 1, \ldots, k$. Hence, we have $\{s\sigma_g, t\sigma_g\} \succ_m \{s_i\sigma_g, s_{i+1}\sigma_g\}$ for $i = 1, \ldots, k-1$ and $\{s\sigma_g, t\sigma_g\} \succ_m \{s'\sigma_g, t\sigma_g\}$. It remains to show there exists $s \overset{*}{\leftrightarrow}_{\mathcal{R} \cup E_\infty} \hat{s}$ such that $s\sigma_g \succ u_g$ or $t \succ u_g$ for any midpoint u_g in $s \overset{*}{\leftrightarrow}_{\mathcal{R} \cup E_\infty} \hat{s}$ and $\{s\sigma_g, t\sigma_g\} \succ_m \{u_g, v_g\}$ for any $u_g \leftrightarrow_{E_\infty} v_g$ in $s \overset{*}{\leftrightarrow}_{\mathcal{R} \cup E_\infty} \hat{s}$. By $s \to_{H^\succ} \hat{s}$, there exists $w \to \hat{w} \in H$ such that $s = C[w\theta]$ and $\hat{s} = C[\hat{w}\theta]$ for some context

C and substitution θ. If $\theta_g = \sigma_g \circ \theta$ is not a constructor ground substitution on $\mathcal{V}(w) \cup \mathcal{V}(\hat{w})$, then the claim follows as in the case σ_g is not a constructor ground substitution. Thus, suppose otherwise. Then, as *Expand* rule is applied to $w \doteq \hat{w}$ with $u \in \mathcal{B}(w)$, it follows using Proposition 7 that $w\theta_g \to_{\mathcal{R}} \circ \xrightarrow{*}_{H \cup \mathrm{inv}(H)^{\succsim}} \circ \leftrightarrow_{E_\infty} \hat{w}\theta_g$. Hence, $s\sigma_g = C[w\theta]\sigma_g = C\sigma_g[w\theta_g] \to_{\mathcal{R}} w_g \xrightarrow{*}_{H \cup \mathrm{inv}(H)^{\succsim}} \circ \leftrightarrow_{E_\infty} C\sigma_g[\hat{w}\theta_g] = C[\hat{w}\theta]\sigma_g = \hat{s}\sigma_g$ Then $s\sigma_g \succ w_g$. Furthermore, for any step $u_g \leftrightarrow_H v_g$ in $w_g \xrightarrow{*}_{H \cup \mathrm{inv}(H)^{\succsim}} \circ \leftrightarrow_{E_\infty} \hat{s}\sigma_g$, we have $\{s\sigma_g, t\sigma_g\} \succ_m \{u_g, v_g\}$. Thus, one obtains $s\sigma_g \overset{*}{\leftrightarrow}_{\succsim_{\mathcal{R}}, \succ E_\infty} t\sigma_g$.

(Delete) Then we have $\langle E \uplus \{s \doteq t\}, H\rangle \rightsquigarrow \langle E, H\rangle$, where $s = t$ or $s \leftrightarrow_H t$. The case $s = t$ is obvious. If $s \leftrightarrow_H t$ then there exists $s' \doteq t' \in E_\infty$ such that $s = C[s'\theta]$, $t = C[r'\theta]$ for some θ, and *Expand* is applied to $s' \doteq t'$. Thus $s\sigma_g \overset{*}{\leftrightarrow}_{\succsim_{\mathcal{R}}, \succ E_\infty} t\sigma_g$ is shown in the same way as the case (*Simplify*)-b. ◀

▶ **Lemma 9.** *If $\langle E_0, \emptyset\rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H\rangle$ then E_0 is bounded ground convertible.*

Proof. We have $\mathcal{R} \subseteq \succ$. Clearly, the derivation $\langle E_0, \emptyset\rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H\rangle$ is fair. Thus, by Lemma 8, for any $s \doteq t \in E_\infty$, ground substitution σ_g and ground context C, $C[s\sigma_g] \leftrightarrow_{E_\infty} C[t\sigma_g]$ implies $C[s\sigma_g] \overset{*}{\leftrightarrow}_{\succsim_{\mathcal{R}}, \succ E_\infty} C[t\sigma_g]$. Hence, by Lemma 6, $C[s\sigma_g] \leftrightarrow_{E_\infty} C[t\sigma_g]$ implies $C[s\sigma_g] \overset{*}{\leftrightarrow}_{\succsim_{\mathcal{R}}} C[t\sigma_g]$. The claim follows as $E_0 \subseteq E_\infty$. ◀

▶ Remark. If E_0 is bounded ground convertible then $\mathcal{R} \models_{ind} E_0$. Thus, the lemma above implies the correctness of our rewriting induction system as an inductive theorem proving procedure. In particular, the soundness of the basic rewriting induction system (e.g. Figure 1 of [1]) follows. For this system several correctness proofs are known: e.g. the one using minimal counterexample [13] and the one based on retrogressive property [1].

By Lemmas 2 and 9, our method for ground confluence proving is obtained.

▶ **Theorem 10** (ground confluence check by rewriting induction)**.** *Let $\mathcal{D} = \{l(\epsilon) \mid l \to r \in \mathcal{R}\}$, $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$, $\mathcal{R}$ a quasi-reducible TRS, and $\succsim$ a reduction quasi-order such that $\mathcal{R} \subseteq \succ$. If $\langle \mathrm{CP}(\mathcal{R}), \emptyset\rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H\rangle$ for some H, then $\mathcal{R}$ is ground confluent.*

4 Relaxing the Free Constructor Restriction

In the previous section, we have fixed a partition $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$ as $\mathcal{D} = \{l(\epsilon) \mid l \to r \in \mathcal{R}\}$ and $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$. This setting and the quasi-reducibility assumption on $\mathcal{R}$ induce a rather strong constraint on $\mathcal{R}$. To see this, consider the following example.

▶ **Example 11.** Let

$$\mathcal{R} = \left\{ \ \mathsf{plus}(0, y) \ \to \ y, \quad \mathsf{plus}(\mathsf{s}(0), y) \ \to \ \mathsf{s}(y), \quad \mathsf{s}(\mathsf{s}(x)) \ \to \ x \ \right\}.$$

Then $\{l(\epsilon) \mid l \to r \in \mathcal{R}\} = \{\mathsf{plus}, \mathsf{s}\}$. However, if we put $\mathcal{D} = \{\mathsf{plus}, \mathsf{s}\}$ and $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$, then $\mathsf{s}(0)$ is a basic ground term which is a normal form. Then $\mathcal{R}$ is not quasi-reducible. On the other hand, we could have put $\mathcal{D} = \{\mathsf{plus}\}$ and $\mathcal{C} = \{\mathsf{s}, 0\}$. In that case $\mathcal{R}$ is quasi-reducible.

In other words, the partition $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$ in the previous section can deal with only TRSs $\mathcal{R}$ having *free constructors*, i.e. the case $\mathrm{T}(\mathcal{C}) \subseteq \mathrm{NF}(\mathcal{R})$. In this section, we relax this restriction. From now on, we assume some partition $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$ has been fixed, and the set of rules $l \to r \in \mathcal{R}$ satisfying $l(\epsilon) \notin \mathcal{D}$ is denoted by $\mathcal{R}_c$. To extend the notion of quasi-reducibility to deal with TRSs with non-free constructors, we first prepare several notions.

▶ **Definition 12** (cover)**.** A set L of terms *covers* a term t if for any ground constructor substitution σ_{gc}, there exists $l \in L$ such that $\exists\theta.\, t\sigma_{gc} = l\theta$.

The set L (and its variants) appears under various names in the literature [10, 19, 20].

▶ **Definition 13** (pattern). For any $f \in \mathcal{D}$, we put

$$\mathrm{Pat}(f, \mathcal{R}_c) = \{f(x_1, \ldots, x_{i-1}, w, x_{i+1}, \ldots, x_n) \mid 1 \leq i \leq n, w \in \mathrm{LHS}(\mathcal{R}_c)\}.$$

where $n = ar(f)$ and $x_1, \ldots, , x_n$ are pairwise distinct variables not in w.

The notion of quasi-reducibility is replaced as follows for arbitrary partition of $\mathcal{D} \uplus \mathcal{C}$.

▶ **Definition 14** (strongly quasi-reducible). A TRS $\mathcal{R}$ is said to be a *strongly quasi-reducible* if for each $f \in \mathcal{D}$, $\mathrm{LHS}(f, \mathcal{R}) \cup \mathrm{Pat}(f, \mathcal{R}_c)$ covers $f(x_1, \ldots, x_n)$.

▶ **Example 15** (checking strong quasi-reduciblity). Consider $\mathcal{R}$ in Example 11. Take $\mathcal{C} = \{\mathsf{s}, \mathsf{0}\}$ and $\mathcal{D} = \{\mathsf{plus}\}$. Then $\mathcal{R}_c = \{\mathsf{s}(\mathsf{s}(x)) \to x\}$. We have $\mathrm{LHS}(\mathsf{plus}, \mathcal{R}) = \{\mathsf{plus}(\mathsf{0}, y), \mathsf{plus}(\mathsf{s}(\mathsf{0}), y)\}$ and $\mathrm{Pat}(f, \mathcal{R}_c) = \{\mathsf{plus}(\mathsf{s}(\mathsf{s}(x)), y), \mathsf{plus}(x, \mathsf{s}(\mathsf{s}(y)))\}$. Now one can check $\{\mathsf{plus}(\mathsf{0}, y), \mathsf{plus}(\mathsf{s}(\mathsf{0}), y), \mathsf{plus}(\mathsf{s}(\mathsf{s}(x)), y), \mathsf{plus}(x, \mathsf{s}(\mathsf{s}(y)))\}$ covers $\mathsf{plus}(x, y)$, and thus $\mathcal{R}$ is strongly quasi-reducible.

The following lemma easily follows from these definitions.

▶ **Lemma 16.** *Any strongly quasi-reducible TRS is quasi-reducible.*

We also have to replace the operation Expd in our rewriting induction system.

▶ **Definition 17** ($\otimes$, Expd_u). Let, for any $f \in \mathcal{D}_f$,

$$\mathcal{R}_c \otimes f = \{f(x_1, \ldots, l', \ldots, x_n) \to f(x_1, \ldots, r', \ldots, x_n) \mid l' \to r' \in \mathcal{R}_c\},$$

where $x_1, \ldots, x_n$ are supposed to be distinct variables not in l', r'. Let $s = C[u]$, $u \in \mathcal{B}(s)$ and $u(\epsilon) = f$. We put

$$\mathrm{Expd}_u(s, t) \quad = \quad \{C[r]\mu \to t\mu \mid \mu = \mathrm{mgu}(l, u), l \to r \in (\mathcal{R} \setminus \mathcal{R}_c) \cup (\mathcal{R}_c \otimes f)\}.$$

Note $l \to_{\mathcal{R}_c} r$ for any $l \to r \in \mathcal{R}_c \otimes f$ by definition.

▶ **Example 18.** Let $\mathcal{R}$ be in Example 11, and $s = \mathsf{plus}(x, \mathsf{s}(\mathsf{0}))$ and $t = \mathsf{plus}(\mathsf{s}(x), \mathsf{0})$. Then $\mathrm{Expd}_s(s, t) = \{\mathsf{s}(\mathsf{0}) \doteq \mathsf{plus}(\mathsf{s}(\mathsf{0}), \mathsf{0}),\ \mathsf{s}(\mathsf{s}(\mathsf{0})) \doteq \mathsf{plus}(\mathsf{s}(\mathsf{s}(\mathsf{0})), \mathsf{0}),\ \mathsf{plus}(y, \mathsf{s}(\mathsf{0})) \doteq \mathsf{plus}(\mathsf{s}(\mathsf{s}(\mathsf{s}(y))), \mathsf{0})\}$.

▶ **Lemma 19** (property of Expd). *Suppose $\mathcal{R}$ is a strongly quasi-reducible TRS and $u \in \mathcal{B}(s)$. Then, for any ground constructor substitution σ_{gc}, we have $s\sigma_{gc} \to_{\mathcal{R}} \circ \to_{\mathrm{Expd}_u(s,t)} t\sigma_{gc}$.*

Proof. Since $u \in \mathcal{B}(s)$, $u = f(u_1, \ldots, u_n)$ for some $f \in \mathcal{D}$ and constructor terms $u_1, \ldots, u_n$. Then, since $\mathcal{R}$ is strongly quasi-reducible, for any ground constructor substitution σ_{gc}, there exists $l \in \mathrm{LHS}(f, \mathcal{R}) \cup \mathrm{Pat}(f, \mathcal{R}_c)$ such that $u\sigma_{gc}$ is an instance of l. Then, since one can assume $\mathcal{V}(l) \cap \mathcal{V}(u) = \emptyset$, w.l.o.g. one can let $u\sigma_{gc} = l\sigma_{gc}$. Then, there exist substitutions $\mu = \mathrm{mgu}(u, l)$ and ground substitution θ_g such that $\sigma_{gc} = \theta_g \circ \mu$, and we have $s\sigma_{gc} = C[u]\sigma_{gc} = C\sigma_{gc}[u\sigma_{gc}] = C\sigma_{gc}[l\sigma_{gc}] \to_{\mathcal{R}} C\sigma_{gc}[r\sigma_{gc}] = C[r]\mu\theta_g \to_{\mathrm{Expd}_u(s,t)} t\mu\theta_g = t\sigma_{gc}$. ◀

Now by replacing Proposition 7 with Lemma 19, the next theorem follows in the same way as Theorem 10 for the rewriting induction using the Expd given in Definition 17.

▶ **Theorem 20** (ground confluence for strongly quasi-reducible TRSs). *Let $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$ be an arbitrary partition. Let $\mathcal{R}$ be a strongly quasi-reducible TRS, and $\succsim$ a reduction quasi-order such that $\mathcal{R} \subseteq \succ$. If $\langle \mathrm{CP}(\mathcal{R}), \emptyset\rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H\rangle$ for some H, then $\mathcal{R}$ is ground confluent.*

It is easy to see that Theorem 20 generalizes theorem 10, because of the following observation:

▶ **Lemma 21.** *Suppose $\mathcal{D} = \{l(\epsilon) \mid l \to r \in \mathcal{R}\}$ and $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$. Then (1) $\mathcal{R}_c = \emptyset$ and (2) any quasi-reducible TRS is strongly quasi-reducible.*

Input: many-sorted TRS $\mathcal{R}$

Output: Success/Failure

1. Compute (possibly multiple) candidates for the partition $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$ of function symbols.
2. Compute (possibly multiple) candidates for strongly quasi-reducible $\mathcal{R}_0 \subseteq \mathcal{R}$.
3. Choose one $\mathcal{R}_0$ from the candidates, and remove $\mathcal{R}_0$ from the candidates list. If no candidate remains, then choose another candidate of partition to go Step 2 if exists, or else return Failure.
4. Find a reduction quasi-order $\succsim$ such that $\mathcal{R}_0 \subseteq \succ$. If it fails, go to Step 3 to examine another candidate.
5. Run rewriting induction for proving bounded ground convertibility of $\mathrm{CP}(\mathcal{R}_0)$ with $\succsim$. If it fails, go to Step 3 to examine another candidate.
6. Run rewriting induction for proving $\mathcal{R}_0 \models_{ind} (\mathcal{R} \setminus \mathcal{R}_0)$. If it fails, go to Step 3 to examine another candidate. If it succeeds, to return Success.

Figure 2 A ground confluence proving procedure based on rewriting induction.

5 Ground Confluence Proving Procedure

Before presenting our procedure, we introduce the last ingredient of our ground confluence proof, which one can prove easily:

▶ **Theorem 22.** *Let $\mathcal{R}$ be a TRS. Suppose $\mathcal{R}_0 \subseteq \mathcal{R}$ is ground confluent. If $\mathcal{R}_0 \models_{ind} \mathcal{R} \setminus \mathcal{R}_0$ then $\mathcal{R}$ is ground confluent.*

Hence, we divide the problem of ground confluence of $\mathcal{R}$ into the problem of ground confluence of $\mathcal{R}_0$ and the problem of inductive validity $\mathcal{R}_0 \models_{ind} \mathcal{R} \setminus \mathcal{R}_0$. In Figure 2, we present our procedure for proving ground confluence of many-sorted TRSs.

▶ **Example 23** (strongly quasi-reducible subsets)**.** Consider a TRS

$$\mathcal{R} = \left\{ \begin{array}{llllllll} \mathsf{plus}(0,y) & \to & y & (a) & \mathsf{plus}(\mathsf{s}(x),y) & \to & \mathsf{s}(\mathsf{plus}(x,y)) & (b) \\ \mathsf{plus}(x,0) & \to & \mathsf{plus}(0,x) & (c) & \mathsf{plus}(x,\mathsf{s}(y)) & \to & \mathsf{plus}(\mathsf{s}(y),x) & (d) \end{array} \right\}$$

Then $\{(a),(b)\}$, $\{(c),(d)\}$ and $\mathcal{R}$ are all strongly quasi-reducible. However, the only choice of $\mathcal{R}_0 = \{(a),(b)\}$ is successful for finding ground confluence proof in our method.

▶ **Example 24** (strongly quasi-reducible subsets)**.** Consider a TRS

$$\mathcal{R} = \left\{ \begin{array}{llllllll} \mathsf{plus}(0,y) & \to & y & (a) & \mathsf{plus}(\mathsf{s}(0),y) & \to & \mathsf{s}(y) & (b) \\ \mathsf{plus}(\mathsf{plus}(x,y),z) & \to & \mathsf{plus}(x,\mathsf{plus}(y,z)) & (c) & \mathsf{s}(\mathsf{s}(x)) & \to & \mathsf{s}(x) & (e) \\ \mathsf{plus}(x,y) & \to & \mathsf{plus}(y,x) & (e) & \mathsf{s}(x) & \to & \mathsf{s}(\mathsf{s}(x)) & (f) \end{array} \right\}$$

Then $\{(a),(b),(e)\}$, $\{(a),(f)\}$ and $\{(d),(f)\}$ are all strongly quasi-reducible, where the first one takes $\mathcal{C} = \{0,\mathsf{s}\}$ while the latter two take $\mathcal{C} = \{0\}$. However, the only choice of $\mathcal{R}_0 = \{(a),(b),(e)\}$ is successful for finding ground confluence proof in our method.

6 Implementation and Experiments

Our tool AGCP is written in SML/NJ. Some program code are incorporated from confluence prover ACP [6] and an inductive theorem prover [2]. The tool is obtained from

```
(FUN
    plus : Nat,Nat -> Nat
    s : Nat -> Nat
    0 : Nat
)
(VAR
    x : Nat
    y : Nat
)
(RULES
    plus(0,0) -> 0
    plus(s(x),y) -> s(plus(x,y))
    plus(x,s(y)) -> s(plus(y,x))
)
```

Figure 3 An example of input: specification of a many-sorted TRS.

`http://www.nue.ie.niigata-u.ac.jp/tools/agcp/`. The format of input TRSs basically follows old TPDB format of first-order TRSs; the only difference is the addition of sort declarations. In Figure 3, we present an example of input file.

Some heuristics implicit in the procedure are described below.

- We impose a timeout for each of Steps 1, 5 and 6.
- In Step 2, as the candidates for $\mathcal{R}_0 \subseteq \mathcal{R}$, we only take those with minimal set of rewrite rules i.e. those $\mathcal{R}_0$ such that any $\mathcal{R}_0' \subsetneq \mathcal{R}_0$ is not a quasi-reducible.
- In Step 4, we restrict ourselves to multiset path orderings based on total precedence. We encoded the condition $\mathcal{R}_0 \subseteq \succsim$ as a constraint on precedence and find a precedence which meets the condition by a SMT solver.
- In *Expand* inferences, among equations, one with the smallest size are expanded. If $l \succ r$ or $r \succ l$ holds, then the larger side is expanded. Among basic subterms, the older one is expanded. Here, a subterm is younger if it contains a newer created variable.
- In *Expand* and *Simplify* inferences, $\xrightarrow{*}_{H \cup \mathrm{inv}(H)\succsim}$-part is tried only if successive applications of *Simplify* occur: just after *Expand*, the expanded side is simplified by $s\ (\xrightarrow{*}_{H \cup \mathrm{inv}(H)\succsim} \circ \to_{\mathcal{R}_0 \cup H^\succ})^*\ s'$, and multiple *Simplify* steps are applied by $\to_{\mathcal{R}_0 \cup H^\succ} \circ (\xrightarrow{*}_{H \cup \mathrm{inv}(H)\succsim} \circ \to_{\mathcal{R}_0 \cup H^\succ})^*$.

We have tested our ground confluence prover with 121 (many-sorted) TRSs—23 are constructed in the course of our study and 98 are incorporated from Cops[1]. Since Cops is a database of confluence problems, their signatures are unsorted. Also, there is no declaration of the signature. Hence, we have inspected the problems and identified a set of function symbols and attached them sorts naturally guessed. More than half Cops problems whose appropriate sorts are hardly imagined are dropped. Among newly constructed 23 problems, 4 problems are incorporated from the literature [7, 14, 18]. Others are constructed by starting with basic TRSs such as addition of natural numbers, and then add variations and extensions of them, trying to make them possibly ground confluent, where this collection includes Examples 1, 15, 23 and 24.

[1] Confluence Problem Database `http://cops.uibk.ac.at/`.

Table 1 Summary of experiments.

sources	number	success	timeout	time (msec)
Example 1	–	✓	–	8
Example 15	–	✓	–	8
Example 23	–	✓	–	11
Example 24	–	✓	–	10
Cops `confluent`	88	62	4	–
Cops `non_confluent`	3	1	0	–
Cops `!confluent !non_confluent`	7	3	0	–
Crafted	23	20	2	–

Tests are performed on a PC with one 2.50GHz CPU and 4G memory. We impose 60 (5, 1) seconds time limit total (resp. rewriting induction proof, computation of constructors).

Table 1 shows a summary of experiments. The columns below 'sources' and 'number' denote the number of TRSs and their sources. 'Cops `confluent`' ('Cops `non_confluent`', 'Cops `!confluent !non_confluent`') denotes problems incorporated from Cops problems that have been proved or non-proved (non-)confluent by state-of-the-art confluence provers. Note such confluence provers prove (non-)confluent of unsorted versions of the problems, but that of the many-sorted ones follow by persistency [5]. Similarly, among 'Crafted' problems, 11 problems are proved to be confluent and the others non-confluent by ACP [6]. The columns below 'Success' show results for each example in the present paper (✓ for success), and the numbers of problems from the collections that succeed. The columns below 'timeout' ('time (msec)') show the number of occurrences of timeout (run time shown in milliseconds, respectively).

Among 121 problems, our prover succeeded in proving ground confluence of 86 problems. Our procedures failed on some particular types of term rewriting systems. Firstly, those that have a defined symbol specified by non-terminating rules such as $\mathsf{nats} \to \mathsf{cons}(0, \mathsf{inc}(\mathsf{nats}))$. In such a case, the non-terminating rule is not an inductive theorem and hence it is included to the rule part. However, in the current approach, the rule part needs to be terminating, and thus our procedure failed to deal with such a case. Similarly, if AC-rules needs to be act as rewrite rules, then our method does not work—our method can deal with AC-rules only if they are inductive theorems:

▶ **Example 25** (Cops 183)**.**

$$\mathcal{R} = \left\{ \begin{array}{lcllcl} +(0,x) & \to & x & +(x,0) & \to & x \\ +(1,-(1)) & \to & 0 & +(-(1),1) & \to & 0 \\ -(0) & \to & 0 & -(-(x)) & \to & x \\ -(+(x,y)) & \to & +(-(x),-(y)) & & & \\ +(+(x,y),z) & \to & +(x,+(y,z)) & +(x,y) & \to & +(y,x) \end{array} \right\}$$

Here, for example, the computation of $+(+(1,1),+(-(1),-(1))) \xrightarrow{*} 0$ needs to use AC-rules. Thus AC-rules are included in the rule part, and hence its termination proof fails. Our approach can not handle problems of this type.

Some failures are due to incapability of non-ground-confluence checking. We expect some simple non-ground-confluence check should be useful, but currently it is not included in our tool. It also seems inclusion of stronger termination criteria would have stopped some failures in early stages of proofs, and inclusion of lemma generation methods in inductive

theorem proving would have solved at least one problem. Other reasons of failure include incapability of dealing with non-left-linear rules in strong quasi-reducibility checking.

Five timeouts are raised in rewriting induction proofs and one is in computation of constructor symbols. Two timeouts in a problem helped to switch the choice of $\mathcal{R}_0$ so as to succeed in that problem.

All details of the experiments are available on the webpage `http://www.nue.ie.niigata-u.ac.jp/tools/agcp/experiments/fscd16/`.

7 Conclusion

We have reported a ground confluence prover based on a variant of rewriting induction. We have also proved the correctness of our method. In contrast to many existing works on ground confluence, we focused on the pure many-sorted TRSs. Obviously, one can also use confluence provers such as [6, 27] to guarantee ground confluence, and to use more stronger inductive theorem proving methods or lemma generation methods such as [2, 3, 4, 22, 25] at inductive theorem proving part, in order to get a more powerful tool. Developing such a powerful tool stands as a long-term goal.

Acknowledgements. Thanks are due to the anonymous reviewers for helpful comments. This work is partially supported by JSPS KAKENHI Nos. 15K00003, 25280025.

References

1 T. Aoto. Dealing with non-orientable equations in rewriting induction. In *Proc. of 17th RTA*, volume 4098 of *LNCS*, pages 242–256. Springer-Verlag, 2006.
2 T. Aoto. Designing a rewriting induction prover with an increased capability of non-orientable equations. In *Proc. of 1st SCSS*, RISC Technical Report, pages 1–15, 2008.
3 T. Aoto. Sound lemma generation for proving inductive validity of equations. In *Proc. of 28th FSTTCS*, volume 2 of *LIPIcs*, pages 13–24. Schloss Dagstuhl, 2008.
4 T. Aoto and S. Stratulat. Decision procedures for proving inductive theorems without induction. In *Proc. of 16th PPDP*, pages 237–248. ACM Press, 2014.
5 T. Aoto and Y. Toyama. Persistency of confluence. *Journal of Universal Computer Science*, 3(11):1134–1147, 1997.
6 T. Aoto, Y. Yoshida, and Y. Toyama. Proving confluence of term rewriting systems automatically. In *Proc. of 20th RTA*, volume 5595 of *LNCS*, pages 93–102. Springer-Verlag, 2009.
7 K. Becker. Proving ground confluence and inductive validity in constructor based equational specifications. In *Proc. of 4th TAPSOFT*, volume 668 of *LNCS*, pages 46–60. Springer-Verlag, 1993.
8 A. Bouhoula. Simultaneous checking of completeness and ground conflunce for algebraic specifications. *ACM Transactions on Computational Logic*, 10(2):20:1–33, 2009.
9 A. Bouhoula and F. Jacquemard. Verifying regular trace properties of secuirty protocols with explicit destructors and implicit induction. In *Proc. of FCS-ARSPA*, pages 27–44, 2007.
10 A. Bouhoula, E. Kounalis, and M. Rusinowitch. Automated mathematical induction. *Journal of Logic and Computation*, 5(5):631–668, 1995.
11 Y. Chiba, T. Aoto, and Y. Toyama. Program transformation by templates based on term rewriting. In *Proc. of 7th PPDP*, pages 59–69. ACM Press, 2005.
12 M. Codish, V. Lagoon, and P. J. Stuckey. Solving partial order constraints for LPO termination. In *Proc. of 17th RTA*, volume 4098 of *LNCS*, pages 4–18. Springer-Verlag, 2006.

13 N. Dershowitz and U. S. Reddy. Deductive and inductive synthesis of equational programs. *Journal of Symbolic Computation*, 15:467–494, 1993.

14 L. Fribourg. A strong restriction of the inductive completion procedure. *Journal of Symbolic Computation*, 8:253–276, 1989.

15 H. Ganzinger. Ground term confluence in parametric conditional equational specifications. In *Proc. of 4th STACS*, volume 247 of *LNCS*, pages 286–298, 1987.

16 R. Göbel. Ground confluence. In *Proc. of 2nd RTA*, volume 256 of *LNCS*, pages 156–167, 1987.

17 N. Hirokawa and D. Klein. Saigawa: A confluence tool. In *Proc. of 1st IWC*, page 49, 2012.

18 D. Kapur, P. Narendran, and F. Otto. On ground-confluence of term rewriting systems. *Information and Computation*, 86:14–31, 1990.

19 D. Kapur, P. Narendran, and H. Zhang. On sufficient-completeness and related properties of term rewriting systems. *Acta Informatica*, 24(4):395–415, 1987.

20 D. Kapur, P. Narendran, and H. Zhang. Automating inductionless induction using test sets. *Journal of Symbolic Computation*, 11(1–2):81–111, 1991.

21 K. Sato, K. Kikuchi, T. Aoto, and Y. Toyama. Correctness of context-moving transformations for term rewriting systems. In *Proc. of 25th LOPSTR*, volume 9527 of *LNCS*, pages 331–345. Springer-Verlag, 2015.

22 S. Shimazu, T. Aoto, and Y. Toyama. Automated lemma generation for rewriting induction with disproof. *JSSST Computer Software*, 26(2):41–55, 2009. In Japanese.

23 T. Sternagel and A. Middeldorp. Conditional confluence (system description). In *Proc. of Joint 25th RTA and 12th TLCA*, pages 456–465. Springer-Verlag, 2014.

24 Terese. *Term Rewriting Systems*. Cambridge University Press, 2003.

25 T. Walsh. A divergence critic for inductive proof. *Journal of Artificial Intelligence Research*, 4:209–235, 1996.

26 F. Winkler and B. Buchberger. A criterion for eliminating unnecessary reductions in the Knuth-Bendix algorithm. In *Proc. of the Colloq. on Algebra, Combinatorics and Logic in Computer Science, Vol. II*, pages 849–869. Springer-Verlag, 1985.

27 H. Zankl, B. Felgenhauer, and A. Middeldorp. CSI – A confluence tool. In *Proc. of 23rd CADE*, volume 6803 of *LNAI*, pages 499–505. Springer-Verlag, 2011.