Improving Rewriting Induction Approach for Proving Ground Confluence*

Takahito Aoto[1], Yoshihito Toyama[2], and Yuta Kimura[3]

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

Abstract

In (Aoto&Toyama, FSCD 2016), a method to prove ground confluence of many-sorted term rewriting systems based on rewriting induction is given. In this paper, we give several methods that add wider flexibility to the rewriting induction approach for proving ground confluence. Firstly, we give a method to deal with the case in which suitable rules are not presented in the input system. Our idea is to construct additional rewrite rules that supplement or replace existing rules in order to obtain a set of rules that is adequate for applying rewriting induction. Secondly, we give a method to deal with non-orientable constructor rules. This is accomplished by extending the inference system of rewriting induction and giving a sufficient criterion for the correctness of the system. Thirdly, we give a method to deal with disproving ground confluence. The presented methods are implemented in our ground confluence prover AGCP and experiments are reported. Our experiments reveal the presented methods are effective to deal with problems for which state-of-the-art ground confluence provers can not handle.

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.2017.7

1 Introduction

Confluence is an important property of computational models having non-deterministic computations. In term rewriting, confluence has caught attention many years. Recent interest has been shifted toward the automated methods for proving confluence. Thus, many confluence tools have been developed in past years, and efforts to have yearly confluence competitions continues[1]. Confluence property considered on the set of ground terms is called *ground confluence*. Confluence implies ground confluence, but not vice versa. In equational logic, ground terms are concerned with the validity on the initial models (inductive validity), and thus, it arises when studying in topics such as inductive theorem proving and correct program transformations, which is closely related to inductive validity.

* This work is partially supported by JSPS KAKENHI No. 15K00003.
[1] http://coco.nue.riec.tohoku.ac.jp

Ground confluence of many-sorted term rewriting systems (TRSs, for short) has been studied in e.g. [4, 5, 8, 9]. Recently, two automated tools for ground confluence appeared [2, 12] – one tool is based on rewriting induction [7, 13], which is originally a method for proving inductive theorems, and the other is based on tree automata technique. However, both approaches have some limitations – in rewriting induction approach, the input TRS needs to have a quasi-reducible terminating set of rewrite rules, since the method is based on noetherian induction using reduction order of terminating rewriting systems; in tree automata approach, the input TRS needs to be left-linear right-ground.

In this paper, we give several methods that add wider flexibility to the rewriting induction approach for proving ground confluence. Firstly, we give a method to deal with the case in which suitable rewrite rules are not presented in the input system. Our idea is to construct additional rewrite rules that supplement or replace existing rules. This approach also provides a way to deal with some kind of non-orientable rewrite rules. But it turns out the approach is not capable of some rules, e.g. those for adding flexibility of the data structures. To deal also with such rules, we next extend the inference system of rewriting induction and give a sufficient criterion for the correctness of the system. The last ingredient of the paper is about disproving ground confluence – we design an inference rule of rewriting induction for refuting ground confluence in the course of rewriting induction derivation. All the presented methods are implemented in our ground confluence prover AGCP and experiments are reported.

The rest of the paper is organized as follows. Section 2 fixes notations used in this paper, and presents basic backgrounds on ground confluence proving based on rewriting induction. In Section 3, we present a method to find and construct a suitable defining rules and a method to replace some non-orientable constructor rules, for the rewriting induction to work. In Section 4, we give an extension of rewriting induction system for bounded ground convertibility that can also deals with non-orientable rewrite rules; and then, this rewriting induction system is applied to obtain a new ground confluence criterion. Section 5 deals with proving ground non-confluence. Implementation and experiments of these methods are reported in Section 6. Section 7 concludes.

2 Preliminaries

We assume basic familiarity with term rewriting (e.g. [15]).

For a quasi-order $\succsim$, its strict part and equivalent part are denoted by $\succ$ and $\approx$, respectively. The reflexive (reflexive transitive, symmetric, equivalent) closure of a relation $\to$ is denoted by $\to^=$ (resp. $\to^*$, $\leftrightarrow$, $\leftrightarrow^*$) and n-fold composition by $\to^n$ ($n \geq 0$). We write $\overset{\succ}{\to}$ ($\overset{\approx}{\to}$, $\overset{\succ}{\to}$) to denote $\succsim \cap \to$ (resp. $\approx \cap \to$, $\succ \cap \to$). Let $\to$ be a relation and $\succsim$ a quasi-order on a set. Elements x and y are *bounded convertible* if $x = x_0 \leftrightarrow x_1 \leftrightarrow \cdots \leftrightarrow x_n = y$ for some $x_0, \ldots, x_n$ such that $x \succsim x_i$ or $y \succsim x_i$ $(0 \leq i \leq n)$; we write $x \overset{*}{\leftrightarrow}_{\succsim} y$ if x and y are bounded convertible. We use $\circ$ for the function composition. A relation $\to$ is *confluent* if ${}^*\!\leftarrow \circ \to^* \subseteq \to^* \circ {}^*\!\leftarrow$.

A *many-sorted signature* is given by a non-empty set $\mathcal{S}$ of sorts and a set $\mathcal{F}$ of many-sorted *function symbols*. We use $\mathcal{V}$ to denote the set of many-sorted variables. We will not explicitly state the sort information if it is understood from the context. The sets of function symbols and variables in a term t are denoted by $\mathcal{F}(t)$ and $\mathcal{V}(t)$, respectively. A term t is *ground* if $\mathcal{V}(t) = \emptyset$; it is *linear* if any variable occurs at most once in t. We use $\uplus$ for the disjoint union. We consider a partition of function symbols $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$; function symbols in $\mathcal{D}$ are called *defined symbols* and those in $\mathcal{C}$ are *constructors*. For any $\mathcal{G} \subseteq \mathcal{F}$, the set of (ground) terms over function symbols $\mathcal{G}$ is denoted by $\mathrm{T}(\mathcal{G}, \mathcal{V})$ (resp. $\mathrm{T}(\mathcal{G})$). Terms in $\mathrm{T}(\mathcal{C}, \mathcal{V})$ are called *constructor terms*. A term t is *basic* if it has the form $f(c_1, \ldots, c_n)$ for some $f \in \mathcal{D}$ and $c_1, \ldots, c_n \in \mathrm{T}(\mathcal{C}, \mathcal{V})$. The set of basic subterms of t is denoted by $\mathcal{B}(t)$.

A *substitution* is a sort-preserving mapping $\theta : \mathcal{V} \to \mathrm{T}(\mathcal{F}, \mathcal{V})$ such that $\mathrm{dom}(\theta) = \{x \in \mathcal{V} \mid \theta(x) \neq x\}$ is finite. A substitution θ is *ground* (*constructor*, *ground constructor*) if $\theta(x) \in \mathrm{T}(\mathcal{F})$ (resp. $\mathrm{T}(\mathcal{C}, \mathcal{V})$, $\mathrm{T}(\mathcal{C})$) for all $x \in \mathrm{dom}(\theta)$; we assume $\mathrm{dom}(\theta) \subseteq \mathcal{V}(t)$ when we write $t\theta$ for a substitution θ. A set L of terms *covers* a term t if for any ground constructor substitution σ_{gc}, there exists $l \in L$ such that $\exists\theta.\, t\sigma_{gc} = l\theta$.

For a *position* p in a term t, $t(p)$ denotes the function symbol or variable at p and $t|_p$ denotes the subterm at position p. We use ϵ for the *root* position. A *context* $C[\,]$ is a term with a sorted hole, and $C[t]$ denotes the term obtained by filling the hole with a term t of the same sort. A relation is a *rewrite relation* if it closed under contexts and substitutions. A *rewrite quasi-order* is a quasi-order whose strict part and equivalent part are rewrite relations. A rewrite quasi-order is a *reduction quasi-order* if its strict part is well-founded.

For any equation $s \doteq t$, we assume s and t have the same sort. (Indirected) equations $s \doteq t$ and $t \doteq s$ are identified. A set $\{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ of equations is (ambiguously) abbreviated as $\{s_i \doteq t_i\}_i$ when no confusion arises. A *directed* equation is denoted by $s \to t$. For a set E of directed equations, we put $E^{\leftrightarrow} = \{l \doteq r \mid l \to r \in E\}$. For a set E of equations (directed equations) the smallest rewrite relation containing E is denoted by $\leftrightarrow_E$ (resp. $\to_E$). A directed equation $l \to r$ subjected to $l \notin \mathcal{V}$ and $\mathcal{V}(l) \supseteq \mathcal{V}(r)$ is a *rewrite rule*. A (many-sorted) *term rewriting system* (*TRS* for short) is a finite set of rewrite rules. A TRS $\mathcal{R}$ is *left-linear* if l is a linear term for all $l \to r \in \mathcal{R}$; it is *variable preserving* if $\mathcal{V}(l) = \mathcal{V}(r)$ for all $l \to r \in \mathcal{R}$. A TRS $\mathcal{R}$ is *(ground) confluent* if $\to_{\mathcal{R}}$ is confluent on $\mathrm{T}(\mathcal{F}, \mathcal{V})$ ($\mathrm{T}(\mathcal{F})$, respectively); $\mathcal{R}$ is *terminating* if $\to_{\mathcal{R}}$ is well-founded; $\mathcal{R}$ is *sufficiently complete* if for any $t_g \in \mathrm{T}(\mathcal{F})$ there exists $u_g \in \mathrm{T}(\mathcal{C})$ such that $t_g \to_{\mathcal{R}}^* u_g$. The set of rules $l \to r \in \mathcal{R}$ satisfying $l(\epsilon) \notin \mathcal{D}$ is denoted by $\mathcal{R}_c$ and called the set of *constructor rules*. For a rewrite quasi-order $\succsim$, the strict part of $\mathcal{R}$ is given by $\mathcal{R}^{\succ} = \{l \to r \in \mathcal{R} \mid l \succ r\}$ and the equivalent part of $\mathcal{R}$ by $\mathcal{R}^{\approx} = \{l \to r \in \mathcal{R} \mid l \approx r\}$. Note, as easily seen, $\mathcal{R}^{\approx}$ is variable preserving.

A *most general unifier* of terms s and t is denoted by $\mathrm{mgu}(s, t)$. Let $l_1 \to r_1, l_2 \to r_2$ be rewrite rules whose variables are w.l.o.g. renamed so that $\mathcal{V}(l_1) \cap \mathcal{V}(l_2) = \emptyset$. We say $l_1 \to r_1$ *overlaps on* $l_2 \to r_2$ when l_1 and non-variable subterm $l_2|_p$ of l_2 unify. The overlap arises a *critical pair* $l_2[r_1]_p\theta \mathrel{\widehat{\leftarrow}} r_2\theta$, where $\theta = \mathrm{mgu}(l_1, l_2|_p)$. The set of critical pairs arises from overlaps of rules in $\mathcal{R}$ on rules in $\mathcal{S}$ is denoted by $\mathrm{CP}(\mathcal{R}, \mathcal{S})$. We put $\mathrm{CP}(\mathcal{R}) = \mathrm{CP}(\mathcal{R}, \mathcal{R})$ and regard it as the set of equations induced from the critical pairs.

We write $\mathcal{R} \otimes f = \{f(\ldots, x_{i-1}, l, x_{i+1}, \ldots) \to f(\ldots, x_{i-1}, r, x_{i+1}, \ldots) \mid 1 \leq i \leq n, l \to r \in \mathcal{R}\}$, where $x_1, \ldots, x_n$ are pairwise distinct variables not in l. Let $\mathrm{LHS}(\mathcal{R}) = \{l \mid l \to r \in \mathcal{R}\}$ and $\mathrm{LHS}(f, \mathcal{R}) = \{l \mid l(\epsilon) = f, l \to r \in \mathcal{R}\}$. A TRS $\mathcal{R}$ is said to be a *strongly quasi-reducible* (w.r.t. $\mathcal{D}$) if $\mathrm{LHS}(\mathcal{R}_c \otimes f) \cup \mathrm{LHS}(f, \mathcal{R})$ covers $f(x_1, \ldots, x_n)$ for each $f \in \mathcal{D}$. It easily follows from the definition that any strongly quasi-reducible TRS is quasi-reducible TRS [2]. Thus, any terminating and strongly quasi-reducible TRS is sufficiently complete [10].

In [2], the first two authors give a system of rewriting induction for proving ground confluence of TRSs – its inference rules are given in Figure 1. In *Expand* rule, we put $\mathrm{Expd}_u(s, t) = \{C[r]\mu \to t\mu \mid \mu = \mathrm{mgu}(l, u), l \to r \in (\mathcal{R} \setminus \mathcal{R}_c) \cup (\mathcal{R}_c \otimes f)\}$, where $s = C[u]$ and $u(\epsilon) = f$. We write $\langle E, H\rangle \rightsquigarrow_{\mathcal{R},\mathcal{D},\succsim} \langle E', H'\rangle$ (or $\langle E, H\rangle \rightsquigarrow \langle E', H'\rangle$ if no confusion arises) when $\langle E', H'\rangle$ is obtained from $\langle E, H\rangle$ by an inference rule.

Let $\succsim$ be a quasi-order on terms. An equation $s \doteq t$ is *bounded ground convertible* (by $\mathcal{R}$ w.r.t. $\succsim$) if all ground instantiation $s\theta_g$ and $t\theta_g$ are bounded convertible for the relation $\to_{\mathcal{R}}$ and quasi-order $\succsim$. Then we have the following proposition.

▶ **Proposition 1** ([2]). *Let $\succsim$ be a reduction quasi-order and $\mathcal{R}$ a strongly quasi-reducible TRS w.r.t. $\mathcal{D}$ such that $\mathcal{R} \subseteq \succ$. (1) If $\langle E, \emptyset\rangle \rightsquigarrow^*_{\mathcal{R},\mathcal{D},\succsim} \langle \emptyset, H\rangle$ for some H, then E is bounded ground convertible by $\mathcal{R}$ w.r.t. $\succsim$. (2) If $\langle \mathrm{CP}(\mathcal{R}), \emptyset\rangle \rightsquigarrow^*_{\mathcal{R},\mathcal{D},\succsim} \langle \emptyset, H\rangle$ for some H, then $\mathcal{R}$ is ground confluent.*

Expand

$$\frac{\langle E \uplus \{s \doteq t\},\ H\rangle}{\langle E \cup \{s'_i \doteq t_i\}_i,\ H \cup \{s \to t\}\rangle} \quad \begin{array}{l} u \in \mathcal{B}(s), \{s_i \to t_i\}_i = \mathrm{Expd}_u(s,t), \\ s_i \overset{\succ}{\rightsquigarrow}^*_{H\leftrightarrow} s'_i \end{array}$$

Simplify

$$\frac{\langle E \uplus \{s \doteq t\},\ H\rangle}{\langle E \cup \{s' \doteq t\},\ H\rangle} \quad s \overset{\succ}{\to}_{\mathcal{R}\cup H} \circ \overset{\succ}{\rightsquigarrow}^*_{H\leftrightarrow} s'$$

Delete

$$\frac{\langle E \uplus \{s \doteq t\},\ H\rangle}{\langle E,\ H\rangle} \quad s \to^{\equiv}_{H} t$$

Figure 1 Inference rules of rewriting induction for ground confluence proof.

An equation $s \doteq t$ is an *inductive theorem of* $\mathcal{R}$ if $s\theta_g \leftrightarrow^*_{\mathcal{R}} t\theta_g$ for any ground substitution θ_g. We write $\mathcal{R} \models_{ind} E$ for a set E of equations if any equation $s \doteq t \in E$ is an inductive theorem of $\mathcal{R}$. Clearly, if an equation is bounded ground convertible by $\mathcal{R}$, then it is an inductive theorem of $\mathcal{R}$. In practice, it is often useful to incorporate the following easily obtained property into ground confluence proofs by rewriting induction:

▶ **Proposition 2** ([2]). *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.*

All in all, the procedure for ground confluence proof is described as follows:

Basic GCR Procedure

Input: many-sorted TRS $\mathcal{R}$
Output: `YES` or `MAYBE`

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. Find a reduction quasi-order $\succsim$ such that $\mathcal{R}_0 \subseteq \succ$.
4. Run rewriting induction for deriving $\langle \mathrm{CP}(\mathcal{R}_0) \cup (\mathcal{R} \setminus \mathcal{R}_0), \emptyset\rangle \overset{*}{\rightsquigarrow}_{\mathcal{R}_0,\mathcal{D},\succsim} \langle \emptyset, H\rangle$ for some H.
5. Return `YES` if it succeeds in step 4, otherwise `MAYBE`.

---

3 Rule Complementation and Instantiation

For our basic GCR procedure to work, it is required that there exists strongly quasi-reducible and terminating subset $\mathcal{R}_0 \subseteq \mathcal{R}$. Experiments in [2], however, reveal that there are cases that there does not exist such an $\mathcal{R}_0$.

▶ **Example 3.** Let $\mathcal{F} = \{\mathsf{eq} : \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Bool},\ \mathsf{neq} : \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Bool},\ \mathsf{not} : \mathsf{Bool} \to \mathsf{Bool},\ \mathsf{s} : \mathsf{Nat} \to \mathsf{Nat},\ \mathsf{0} : \mathsf{Nat},\ \mathsf{true} : \mathsf{Bool},\ \mathsf{false} : \mathsf{Bool}\}$ and

$$\mathcal{R} = \left\{ \begin{array}{llll} \mathsf{eq}(\mathsf{0},\mathsf{0}) & \to \mathsf{true} & (a) & \mathsf{eq}(\mathsf{0},\mathsf{s}(y)) \to \mathsf{false} \quad (b) \\ \mathsf{eq}(\mathsf{s}(x),\mathsf{s}(y)) & \to \mathsf{eq}(x,y) & (c) & \mathsf{eq}(x,y) \to \mathsf{eq}(y,x) \quad (d) \\ \mathsf{not}(\mathsf{true}) & \to \mathsf{false} & (e) & \mathsf{not}(\mathsf{false}) \to \mathsf{true} \quad (f) \\ \mathsf{neq}(x,y) & \to \mathsf{not}(\mathsf{eq}(x,y)) & (g) & \mathsf{neq}(x,\mathsf{s}(x)) \to \mathsf{true} \quad (h) \end{array} \right\}$$

Then, only possible reduction $\mathsf{eq}(\mathsf{s}(\mathsf{0}),\mathsf{0}) \to^*_{\mathcal{R}} \mathsf{false}$ has the following form: $\mathsf{eq}(\mathsf{s}(\mathsf{0}),\mathsf{0}) \to_{\{(d)\}} \mathsf{eq}(\mathsf{0},\mathsf{s}(\mathsf{0})) \to_{\{(b)\}} \mathsf{false}$. Hence, in order to make $\mathcal{R}_0$ strong quasi-reducible, one needs

$(d) \in \mathcal{R}_0$. But having $(d) \in \mathcal{R}_0$ makes $\mathcal{R}_0$ non-terminating. Thus, there exists no quasi-reducible and terminating subset $\mathcal{R}_0$ of $\mathcal{R}$. A natural candidate of $\mathcal{R}_0$ here would be

$$\mathcal{R}_0' = \left\{ \begin{array}{llllllll} \mathsf{eq}(0,0) & \to & \mathsf{true} & (a) & \mathsf{eq}(0,\mathsf{s}(y)) & \to & \mathsf{false} & (b) \\ \mathsf{eq}(\mathsf{s}(x),0) & \to & \mathsf{false} & (b') & \mathsf{eq}(\mathsf{s}(x),\mathsf{s}(y)) & \to & \mathsf{eq}(x,y) & (c) \end{array} \right\} \cup \{(e),(f),(g)\}$$

Indeed, the rewrite rule (b') is equationally valid as: $\mathsf{eq}(\mathsf{s}(x),0) \to_{\{(d)\}} \mathsf{eq}(0,\mathsf{s}(x)) \to_{\{(b)\}} \mathsf{false}$. However, the rewrite rule (b') is not contained in $\mathcal{R}$. Let $\mathcal{R}' = \mathcal{R} \cup \{(b')\}$. Then, we have $\to_{\mathcal{R}} \subseteq \to_{\mathcal{R}'} \subseteq \to_{\mathcal{R}}^*$, and thus, ground confluence of $\mathcal{R}'$ implies that of $\mathcal{R}$. Indeed, $\mathrm{CP}(\mathcal{R}_0') = \emptyset$ and $\mathcal{R}_0' \models_{ind} \{(b),(h)\}$ is obtained in our basic GCR procedure, and thus ground confluence of $\mathcal{R}'$ can be shown. The key part of this procedure is to find the lacking pattern $\mathsf{eq}(\mathsf{s}(x),0)$ and obtain a suitable rewrite rule (b').

In the previous example, we supplement rewrite rules. There are cases that we need to replace the rewrite rules, instead of supplementing new ones.

▶ **Example 4.** Let $\mathcal{F} = \{\mathsf{zero} : \mathsf{Nat} \to \mathsf{Bool},\ \mathsf{if} : \mathsf{Bool} \times \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Nat},\ + : \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Nat},\ - : \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Nat},\ * : \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Nat},\ \mathsf{fact} : \mathsf{Nat} \to \mathsf{Nat},\ \mathsf{s} : \mathsf{Nat} \to \mathsf{Nat},\ 0 : \mathsf{Nat},\ \mathsf{true} : \mathsf{Bool},\ \mathsf{false} : \mathsf{Bool}\}$ and

$$\mathcal{R} = \left\{ \begin{array}{llllll} \mathsf{zero}(0) & \to & \mathsf{true} & \mathsf{if}(\mathsf{true},y,z) & \to & y \\ \mathsf{zero}(\mathsf{s}(x)) & \to & \mathsf{false} & \mathsf{if}(\mathsf{false},y,z) & \to & z \\ +(0,y) & \to & y & -(0,y) & \to & 0 \\ +(\mathsf{s}(x),y) & \to & \mathsf{s}(+(x,y)) & -(x,0) & \to & x \\ +(x,\mathsf{s}(y)) & \to & \mathsf{s}(+(y,x)) & -(\mathsf{s}(x),\mathsf{s}(y)) & \to & -(x,y) \\ *(0,y) & \to & 0 & *(\mathsf{s}(x),y) & \to & +(*(x,y),y) \\ \mathsf{fact}(x) & \to & \mathsf{if}(\mathsf{zero}(x),\mathsf{s}(0),*(x,\mathsf{fact}(-(x,\mathsf{s}(0))))) & & & (a) \end{array} \right\}$$

The rewrite rule (a) is the only rule that defines the function fact. Thus, we need to have $(a) \in \mathcal{R}_0$, but the lhs of (a) embeds in the rhs, and thus there is no reduction quasi-order $\succsim$ such that $(a) \in \succ$. To make the rewriting induction work, we replace the rule (a) with the following two rules that inductively defines the function fact:

$$\left\{ \begin{array}{llll} \mathsf{fact}(0) & \to & \mathsf{s}(0) & (a') \\ \mathsf{fact}(\mathsf{s}(x)) & \to & *(\mathsf{s}(x),\mathsf{fact}(x)) & (a'') \end{array} \right\}$$

These new rules are obtained by:

$$\mathsf{fact}(0) \to_{\mathcal{R}} \mathsf{if}(\mathsf{zero}(0),\mathsf{s}(0),\ldots) \to_{\mathcal{R}} \mathsf{if}(\mathsf{true},\mathsf{s}(0),\ldots) \to_{\mathcal{R}} \mathsf{s}(0)$$
$$\mathsf{fact}(\mathsf{s}(x)) \to_{\mathcal{R}} \mathsf{if}(\mathsf{zero}(\mathsf{s}(x)),\ldots,*(\mathsf{s}(x),\mathsf{fact}(-(\mathsf{s}(x),\mathsf{s}(0))))) \to_{\mathcal{R}}^* *(\mathsf{s}(x),\mathsf{fact}(x))$$

In this example, we need to construct a pattern $\{\mathsf{fact}(0),\mathsf{fact}(\mathsf{s}(x))\}$ that covers $\mathsf{fact}(x)$.

Such lacking patterns can be characterized by the notion of complement of patterns: A finite set P of basic terms is called a *pattern*. Intuitively, the set P can be regarded as expressing a set of ground terms given as $\mathrm{Inst}(P) = \{p\sigma_{gc} \mid p \in P, \sigma_{gc} : \mathcal{V} \to \mathrm{T}(\mathcal{C})\}$. $\mathrm{T_B}(\mathcal{D},\mathcal{C})$ denotes the set of ground basic terms over $\mathcal{D}$ and $\mathcal{C}$. A finite set Q of terms is said to be a *complement* (w.r.t. $\mathrm{T_B}(\mathcal{D},\mathcal{C})$) of P if $\mathrm{Inst}(P) \uplus \mathrm{Inst}(Q) = \mathrm{T_B}(\mathcal{D},\mathcal{C})$. We (ambiguously) denote Q as $\mathrm{T_B}(\mathcal{D},\mathcal{C}) \ominus P$.

▶ **Example 5.** Let $\mathcal{D} = \{\mathsf{eq}\}$, $\mathcal{C} = \{\mathsf{s},0,\mathsf{true},\mathsf{false}\}$ and $P = \{\mathsf{eq}(0,0),\mathsf{eq}(0,\mathsf{s}(y)),\mathsf{eq}(\mathsf{s}(x),\mathsf{s}(y))\}$. Then $\mathrm{T_B}(\mathcal{D},\mathcal{C}) \ominus P = \{\mathsf{eq}(\mathsf{s}(x),0)\}$.

A pattern P is *linear* if so are all its elements. Theorem 1 of [11] gives an algorithm to compute Q from P (complementation algorithm) for any linear pattern P, and we use it in the following procedure.

▶ **Definition 6** (Rule Complementation Procedure)**.**

1. For each $f \in \mathcal{D}$,
   - **a.** take $\mathcal{S}_f \subseteq \{l \to r \in \mathcal{R} \mid l(\epsilon) = f\}$ such that $\mathrm{LHS}(\mathcal{S}_f)$ is a linear pattern, and
   - **b.** take $l_1, \ldots, l_n$ such that $\{l_1, \ldots, l_n\} = \mathrm{T_B}(\{f\}, \mathcal{C}) \ominus \mathrm{LHS}(\mathcal{S}_f)$ and $r_1, \ldots, r_n$ such that $l_i \to_{\mathcal{R}}^* r_i$ for each i, and let $\mathcal{S}'_f = \mathcal{S}_f \cup \{l_i \to r_i \mid 1 \le i \le n\}$.
2. Finally, put $\mathcal{R}_1 = \mathcal{R}_c \cup \bigcup_{f \in \mathcal{D}} \mathcal{S}'_f$.

▶ **Theorem 7.** *Suppose $\mathcal{R}_1$ is obtained from $\mathcal{R}$ by the rule complementation procedure. If $\mathcal{R}_1$ is ground confluent and $\mathcal{R}_1 \models_{ind} \mathcal{R} \setminus \mathcal{R}_1$, then $\mathcal{R}$ is ground confluent.*

Proof. Suppose $\mathcal{R}_1$ is ground confluent and $\mathcal{R}_1 \models_{ind} \mathcal{R} \setminus \mathcal{R}_1$. Then $\mathcal{R}_1 \cup \mathcal{R}$ is ground confluent by Proposition 2. From the procedure, we have $\to_{\mathcal{R}} \subseteq \to_{\mathcal{R}_1 \cup \mathcal{R}} \subseteq \to_{\mathcal{R}}^*$. Thus, ground confluence of $\mathcal{R}$ follows. ◀

▶ **Example 8.** Let $\mathcal{R}$ be a TRS given in Example 4. Take $\mathcal{S}_{\mathsf{fact}} = \emptyset$, and one can put $P = \mathrm{T_B}(\{\mathsf{fact}\}, \mathcal{C}) \ominus \emptyset = \{\mathsf{fact}(\mathsf{0}), \mathsf{fact}(\mathsf{s}(x))\}$. By $\mathsf{fact}(\mathsf{0}) \to_{\mathcal{R}}^* \mathsf{0}$ and $\mathsf{fact}(\mathsf{s}(x)) \to_{\mathcal{R}}^* *(\mathsf{s}(x), \mathsf{fact}(x))$, one gets $\mathcal{R}_1 = \mathcal{R} \setminus \{(a)\} \cup \{(a'), (a'')\}$. Then one successfully proves that $\mathcal{R}_1$ is ground confluent and $\mathcal{R}_1 \models_{ind} \{(a)\}$. Thus, it follows that $\mathcal{R}$ is ground confluent.

The following example shows the condition $\mathcal{R}_1 \models_{ind} \mathcal{R} \setminus \mathcal{R}_1$ in Theorem 7 can not be dropped.

▶ **Example 9.** Let $\mathcal{F} = \{\mathsf{zero} : \mathsf{Nat} \to \mathsf{Bool},\ \mathsf{if} : \mathsf{Bool} \times \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Nat},\ \mathsf{f} : \mathsf{Nat} \to \mathsf{Nat},\ \mathsf{s} : \mathsf{Nat} \to \mathsf{Nat},\ \mathsf{0} : \mathsf{Nat},\ \mathsf{true} : \mathsf{Bool},\ \mathsf{false} : \mathsf{Bool}\}$ and

$$\mathcal{R} = \left\{ \begin{array}{llllll} \mathsf{zero}(\mathsf{0}) & \to & \mathsf{true} & \mathsf{if}(\mathsf{true}, y, z) & \to & y \\ \mathsf{zero}(\mathsf{s}(x)) & \to & \mathsf{false} & \mathsf{if}(\mathsf{false}, y, z) & \to & z \\ \mathsf{f}(\mathsf{0}) & \to & \mathsf{0} & & & (a) \\ \mathsf{f}(x) & \to & \mathsf{if}(\mathsf{zero}(x), \mathsf{s}(\mathsf{0}), \mathsf{0}) & & & (b) \end{array} \right\}$$

Take $\mathcal{S}_{\mathsf{f}} = \{(a)\}$ and $\mathrm{T_B}(\{\mathsf{f}\}, \mathcal{C}) \ominus \mathrm{LHS}(\mathcal{S}_{\mathsf{f}}) = \{\mathsf{f}(\mathsf{s}(x))\}$. By $\mathsf{f}(\mathsf{s}(x)) \to \mathsf{if}(\mathsf{zero}(\mathsf{s}(x)), \mathsf{s}(\mathsf{0}), \mathsf{0}) \to \mathsf{if}(\mathsf{false}, \mathsf{s}(\mathsf{0}), \mathsf{0}) \to \mathsf{0}$, one gets $\mathcal{S}'_{\mathsf{f}} = \{(a)\} \cup \{\mathsf{f}(\mathsf{s}(x)) \to \mathsf{0}\}$. Now as $\mathcal{R}_1$ is orthogonal, $\mathcal{R}_1$ is ground confluent. On the other hand, we have $\mathsf{0} \leftarrow_{\mathcal{R}} \mathsf{f}(\mathsf{0}) \to_{\mathcal{R}} \mathsf{if}(\mathsf{zero}(\mathsf{0}), \mathsf{s}(\mathsf{0}), \mathsf{0}) \to_{\mathcal{R}} \mathsf{if}(\mathsf{true}, \mathsf{s}(\mathsf{0}), \mathsf{0}) \to_{\mathcal{R}} \mathsf{s}(\mathsf{0})$, and thus $\mathcal{R}_1$ is not ground confluent. Note here $\mathcal{R}_1 \not\models_{ind} \{(b)\}$.

Instantiation technique is also useful for dealing with non-orientable constructor rules. The following example illustrates this.

▶ **Example 10** (COPS #74)**.** Let $\mathcal{R}$ be a (uni-sorted) TRS as follows:

$$\left\{ \begin{array}{llllllllll} \mathsf{a} & \to & \mathsf{c} & (a) & \mathsf{b} & \to & \mathsf{c} & (b) & \mathsf{f}(\mathsf{a}, \mathsf{b}) & \to & \mathsf{d} & (c) \\ \mathsf{f}(x, \mathsf{c}) & \to & \mathsf{f}(\mathsf{c}, \mathsf{c}) & (d) & \mathsf{f}(\mathsf{c}, x) & \to & \mathsf{f}(\mathsf{c}, \mathsf{c}) & (e) & & & & \\ \mathsf{d} & \to & \mathsf{f}(\mathsf{a}, \mathsf{c}) & (f) & \mathsf{d} & \to & \mathsf{f}(\mathsf{c}, \mathsf{b}) & (g) & & & & \end{array} \right.$$

Let $\mathcal{C} = \{\mathsf{f}, \mathsf{c}\}$. Take $\mathcal{R}_c = \{(d), (e)\}$, $\mathcal{R}_0 = \{(a), (b), (g)\} \cup \mathcal{R}_c$ and $\mathcal{R} \setminus \mathcal{R}_0 = \{(c), (f)\}$. Then $\mathcal{R}_0$ is strongly quasi-reducible. However, $\mathcal{R}_0 \subseteq \succ$ does no hold because $\mathsf{f}(x, \mathsf{c}) \not\succ \mathsf{f}(\mathsf{c}, \mathsf{c})$ as well as $\mathsf{f}(\mathsf{c}, x) \not\succ \mathsf{f}(\mathsf{c}, \mathsf{c})$.

Instantiation technique can be used as follows. First, add some instantiation $\mathcal{R}'_c$ of rewrite rules (d) and (e) and replace partitions as follows:

$$\mathcal{R}'_c = \left\{ \begin{array}{lcll} \mathsf{f}(\mathsf{c},\mathsf{c}) & \to & \mathsf{f}(\mathsf{c},\mathsf{c}) & (h) \\ \mathsf{f}(\mathsf{f}(x,y),\mathsf{c}) & \to & \mathsf{f}(\mathsf{c},\mathsf{c}) & (h') \\ \mathsf{f}(\mathsf{c},\mathsf{f}(x,y)) & \to & \mathsf{f}(\mathsf{c},\mathsf{c}) & (h'') \end{array} \right\}, \qquad \begin{array}{l} \mathcal{R}'_0 = \{(a),(b),(g)\} \cup \mathcal{R}'_c \\ \mathcal{R} \setminus \mathcal{R}'_0 = \{(c),(d),(e),(f)\} \end{array}$$

Because $\mathcal{C} = \{\mathsf{f},\mathsf{c}\}$, $\mathsf{f}(\mathsf{c},x) \doteq \mathsf{f}(\mathsf{c},\mathsf{c})$ are inductive theorems of $\mathcal{R}'_0$. Thus, $\xrightarrow{*}_{\mathcal{R}}$ and $\xrightarrow{*}_{\mathcal{R}\cup\mathcal{R}'_c}$ coincide on ground terms. Evidently, one can further remove trivial rule (h), and thus one can use $\mathcal{R}''_0 = \mathcal{R}'_0 \setminus \{(h)\} \subseteq \succ$. Then the ground confluence of $\mathcal{R}''_0$ can be shown, and from $\mathcal{R}''_0 \models_{ind} \{(c),(d),(e),(f)\}$, one can conclude ground confluence of $\mathcal{R}$.

Now we formalize this idea. A substitution σ is linear if $\sigma(x)$ is linear for all $x \in \mathrm{dom}(\sigma)$ and $\mathcal{V}(\sigma(x)) \cap \mathcal{V}(\sigma(y)) = \emptyset$ for any distinct $x, y \in \mathrm{dom}(\sigma)$. A complement of a linear constructor substitution σ is a set of linear constructor substitutions $\mathrm{Comp}(\sigma)$ such that for any term t and ground constructor substitution θ_{gc}, there exists $\rho \in \mathrm{Comp}(\sigma) \cup \{\sigma\}$ such that $t\theta_g$ is an instance of $t\rho$. Definition 11 of [11] gives an algorithm to compute a complement $\mathrm{Comp}(\sigma)$ of a linear constructor substitution σ, and we use it in the following procedure.

▶ **Definition 11** (Rule Instantiation Procedure)**.** Let $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$ be an arbitrary partition. Suppose $l \to r \in \mathcal{R}_c$ and there exists $\sigma = \mathrm{mgu}(l,r)$ such that σ is a linear substitution. Put $\mathcal{R}_2 = (\mathcal{R} \setminus \{l \to r\}) \cup \{l\rho \to r\rho \mid \rho \in \mathrm{Comp}(\sigma)\}$.

▶ **Theorem 12.** *Suppose that $\mathcal{R}^{\succ}$ is strongly quasi-reducible and $\mathcal{R}_2$ is obtained from $\mathcal{R}$ by the rule instantiation procedure. If $\mathcal{R}_2$ is ground confluent, then so is $\mathcal{R}$.*

Proof. Suppose $\mathcal{R}^{\succ}$ is strongly quasi-reducible, $l \to r \in \mathcal{R}$ and $\sigma = \mathrm{mgu}(l,r)$ is a linear substitution. Let $\mathcal{S} = \{l\delta \to r\delta \mid \delta \in \mathrm{Comp}(\sigma)\}$ and $\mathcal{S}' = \mathcal{S} \cup \{l\sigma \to r\sigma\}$. Let $\mathcal{R}' = \mathcal{R} \cup \mathcal{S}$. Then, since $l \to r \in \mathcal{R}$, we have $\to_{\mathcal{R}'} = \to_{\mathcal{R}}$, and thus $\mathcal{R}$ is ground confluent iff so is $\mathcal{R}'$. We now show if $\mathcal{R}_2$ is ground confluent then $\mathcal{R}'$ is ground confluent. To show this, from $\mathcal{R}' = \mathcal{R}_2 \cup \{l \to r\}$ and Proposition 2, it suffices to show $\mathcal{R}_2 \models_{ind} l \doteq r$. Since l and r are unifiable and $\succ$ is well-founded, we have $l \to r \notin \mathcal{R}^{\succ}$. Thus $\mathcal{R}^{\succ} \subseteq \mathcal{R}_2$. Hence, as $\mathcal{R}^{\succ}$ is strongly quasi-reducible and terminating, for any ground substitution θ_g there exists a ground constructor substitution θ_{gc} such that $\theta_g(x) \to^*_{\mathcal{R}_2} \theta_{gc}(x)$. Thus for any ground substitution θ_g, we have $l\theta_g \to^*_{\mathcal{R}_2} l\theta_{gc}$ and $r\theta_g \to^*_{\mathcal{R}_2} r\theta_{gc}$. By the property of complement, mentioned above, for any $l\theta_{gc}$, there exists $\delta \in \mathrm{Comp}(\sigma) \cup \{\sigma\}$ such that $l\theta_{gc}$ is an instance of $l\delta$. Thus, $l\theta_{gc} \to_{\mathcal{S}'} r\theta_{gc}$. Since $l\sigma = r\sigma$, we have $l\theta_{gc} \to^{=}_{\mathcal{S}} r\theta_{gc}$. As $\mathcal{S} \subseteq \mathcal{R}_2$, we have $l\theta_g \leftrightarrow^*_{\mathcal{R}_2} r\theta_g$. Hence $\mathcal{R}_2 \models_{ind} l \doteq r$. ◀

4 Relaxing Ordering-Constraints in Rewriting Induction

The technique in the previous section enables us to deal with non-orientable rule $l \to r \in \mathcal{R}$ if it can be moved to conjecture part and replaced with strictly decreasing rule. However, if the rule is a constructor rule, this may be not possible in nature.

▶ **Example 13.** Let $\mathcal{F} = \{\mathsf{sum} : \mathsf{Btree} \to \mathsf{Nat},\ + : \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Nat},\ \mathsf{node} : \mathsf{Nat} \times \mathsf{Btree} \times \mathsf{Btree} \to \mathsf{Btree},\ \mathsf{leaf} : \mathsf{Nat} \to \mathsf{Btree},\ \mathsf{s} : \mathsf{Nat} \to \mathsf{Nat},\ \mathsf{0} : \mathsf{Nat}\}$ and $\mathcal{R}$ be the following TRS:

$$\left\{ \begin{array}{llllll} \mathsf{sum}(\mathsf{leaf}(x)) & \to & x & \mathsf{sum}(\mathsf{node}(x,yt,zt)) & \to & +(x,+(\mathsf{sum}(yt),\mathsf{sum}(zt))) \\ +(x,\mathsf{0}) & \to & x & +(x,\mathsf{s}(y)) & \to & \mathsf{s}(+(x,y)) \\ \mathsf{node}(x,yt,zt) & \to & \mathsf{node}(x,zt,yt) & & & \end{array} \right\}$$

Modify

$$\frac{\langle E \uplus \{s \doteq t\},\ H\rangle}{\langle E \cup \{s' \doteq t\},\ H\rangle} \quad s \overset{\approx}{\to}_{\mathcal{R}} s'$$

Figure 2 *Modify* inference rule.

Now the non-orientable $\mathsf{node}(x, yt, zt) \to \mathsf{node}(x, zt, yt)$ can not be replaced with strictly decreasing rules nor moved to the conjecture part. Thus, the previous techniques cannot deal with this case.

To deal with such a case, we first extend our rewriting induction system. The idea is to allow a constructor rewrite rule $l \to r$ such that $l \approx r$. We assume that a TRS $\mathcal{R}$ and a reduction quasi-order $\succsim$ satisfy $\mathcal{R} \subseteq \succsim$ and $\mathcal{R}^{\succ}$ is strongly quasi-reducible.

▶ **Definition 14.** Inference rules of rewriting induction for proving bounded ground convertibility of many-sorted TRSs is obtained from those given in Figure 1 by (i) adding *Modify* rule in Figure 2, and (ii) replacing Expd operation in *Expand* rule with $\mathrm{Expd}^{\succ}$ defined as follows:

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

where $s = C[u]$ and $u(\epsilon) = f$.

We now provide the key property of the system (see Appendix B for the proof).

▶ **Theorem 15.** *If $\langle E, \emptyset\rangle \rightsquigarrow^* \langle \emptyset, H\rangle$ for some H, then E is bounded ground convertible by $\mathcal{R}$ w.r.t. $\succsim$.*

Allowing (constructor) rewrite rules $l \to r$ such that $l \approx r$ makes the bounded ground convertibility alone insufficient for guaranteeing ground confluence. Thus, we need an extension of Newman's Lemma [15] to suit our situation (see Appendix A for the proof).

▶ **Lemma 16.** *Let $\succsim$ be a well-founded quasi-order. Suppose $\mathcal{R}_d = \mathcal{R} \setminus \mathcal{R}_c$, $\mathcal{R}_d \subseteq \succ$, $\mathcal{R}_c \subseteq \succsim$ and $\mathcal{R}_c$ is ground confluent. Then, $\mathcal{R}$ is ground confluent if the following two conditions are satisfied for any ground terms u_g, v_g:*

(i) *$u_g \overset{\prec}{\leftarrow}_{\mathcal{R}} \circ \to_{\mathcal{R}_d} v_g$ implies $u_g \leftrightarrow^*_{\mathcal{R}^{\prec}_{\succsim}} v_g$, and*

(ii) *$u_g \overset{\approx}{\leftarrow}_{\mathcal{R}} \circ \to_{\mathcal{R}_d} v_g$ implies $u_g \to_{\mathcal{R}_d} \circ \leftrightarrow^*_{\mathcal{R}^{\prec}_{\succsim}} v_g$.*

Henceforth, we put $\mathcal{R}_d = \mathcal{R} \setminus \mathcal{R}_c$ and assume $\mathcal{R}_d \subseteq \succ$, $\mathcal{R}_c \subseteq \succsim$ (thus, $\mathcal{R}^{\approx} \subseteq \mathcal{R}_c$) and $\mathcal{R}^{\succ}$ is strongly quasi-reducible. The idea to obtain a ground confluence proof of $\mathcal{R}$ by our new rewriting induction system for bounded ground convertibility is to apply Lemma 16 with the help of the assumption that $\mathcal{R}_c$ is ground confluent and Theorem 15. The condition (i) of the lemma is guaranteed by requesting $\langle \mathrm{CP}(\mathcal{R}^{\succ}) \setminus \mathrm{CP}(\mathcal{R}_c^{\succ}), \emptyset\rangle \rightsquigarrow^* \langle \emptyset, H\rangle$, thanks to Theorem 15.

To guarantee the condition (ii) of the lemma, we assume $\mathcal{R}$ is supplemented by $\{l \to r \mid l \succ r, l \to r \in \mathcal{R}_c \otimes f, f \in \mathcal{D}\}$ so that $\mathcal{R}_d$ is quasi-reducible. This is possible since we assume that $\mathcal{R}^{\succ}$ is strongly quasi-reducible, and the addition does not affect whether $\mathcal{R}$ is ground confluent or not. Furthermore, we need to modify inference rules of rewriting induction and the starting set of equations from $\mathrm{CP}(\mathcal{R}_c^{\approx}, \mathcal{R}_d)$ or $\mathrm{CP}(\mathcal{R}_d, \mathcal{R}_c^{\approx})$ by marking the left-hand side

Expand

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

Simplify

$$\frac{\langle E \uplus \{s^\circ \doteq t\},\ H\rangle}{\langle E \cup \{s' \doteq t\},\ H\rangle} \quad s \overset{\succ}{\to}_{\mathcal{R}^\circ \cup H} \circ \overset{\succsim}{\to}{}^*_{H\leftrightarrow} s'$$

Modify

$$\frac{\langle E \uplus \{s \doteq t\},\ H\rangle}{\langle E \cup \{s' \doteq t\},\ H\rangle} \quad s \overset{\approx}{\to}_{\mathcal{R}} s'$$

Delete

$$\frac{\langle E \uplus \{s^\circ \doteq t\},\ H\rangle}{\langle E,\ H\rangle} \quad s \to^{=}_{H} t$$

Figure 3 Inference rules of rewriting induction for ground confluence proof.

or the right-hand side which is required to apply $\mathcal{R}_d$. We denote a marked term s as $s^\bullet$ and a marked equation as $s^\bullet \doteq t$ or $s \doteq t^\bullet$. Then we have the following set of equations.

$$\begin{aligned} \mathrm{CP}_{\gtrsim}(\mathcal{R}) \quad = \quad & \mathrm{CP}(\mathcal{R}^{\succ}) \setminus \mathrm{CP}(\mathcal{R}^{\succ}_c) \cup \{s^\bullet \doteq t \mid s \mathrel{\overset{\frown}{\cup}} t \in \mathrm{CP}(\mathcal{R}^{\approx}_c, \mathcal{R}_d)\} \\ & \cup \{s \doteq t^\bullet \mid s \mathrel{\overset{\frown}{\cup}} t \in \mathrm{CP}(\mathcal{R}_d, \mathcal{R}^{\approx}_c)\} \end{aligned}$$

By extending *Expand*, *Simplify* and *Delete* rules of rewriting induction for treating marked equations, we obtain inference rules of Figure 3 for proving ground confluence of $\mathcal{R}$.

▶ **Definition 17.** Inference rules of rewriting induction for proving ground confluence of many-sorted TRSs is given in Figure 3. Here, E is the set of marked or unmarked equations, and s° denotes a marked term $s^\bullet$ or an unmarked term s. In *Simplify* rule, $\mathcal{R}^\circ$ denotes $\mathcal{R}_d$ if s° is a marked term or $\mathcal{R}$ if s° is an unmarked term.

Note that only *Expand*, *Simplify* or *Delete* rule is applied to a marked equation, and then, the mark is removed; in other words, marked equations can be removed only by applying one of them. For unmarked equations, any of four inference rules is applied.

Using our new rewriting induction system for ground confluence proof, we obtain the following new ground confluence criterion (see Appendix B for the proof).

▶ **Theorem 18.** *Let $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$ be an arbitrary partition. Suppose that $\gtrsim$ is a reduction quasi-order such that $\mathcal{R} \subseteq \gtrsim$. Suppose that $\mathcal{R}^{\succ}$ is left-linear and strongly quasi-reducible, $\mathcal{R}^{\approx} \subseteq \mathcal{R}_c$ and $\mathcal{R}_c$ is ground confluent. If $\langle \mathrm{CP}_{\gtrsim}(\mathcal{R}), \emptyset\rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H\rangle$ for some H, then $\mathcal{R}$ is ground confluent.*

▶ **Example 19.** Let $\mathcal{R}$ be the TRS given in Example 13. Take $\mathcal{C} = \{0, \mathsf{s}, \mathsf{leaf}, \mathsf{node}\}$, $\mathcal{R}_c = \mathcal{R}^{\approx} = \{\mathsf{node}(x, yt, zt) \to \mathsf{node}(x, zt, yt)\}$ and $\mathcal{R}_d = \mathcal{R}^{\succ}$. It is obvious that $\mathcal{R}^{\succ}$ is left-linear and strongly quasi-reducible and $\mathcal{R}_c$ is ground confluent. Then we have $\mathrm{CP}_{\gtrsim}(\mathcal{R}) = \{\mathsf{sum}(\mathsf{node}(x, zt, yt))^\bullet \doteq +(x, +(\mathsf{sum}(yt), \mathsf{sum}(zt)))\}$. Applying inference rules of Figure 3, we obtain $\langle \mathrm{CP}_{\gtrsim}(\mathcal{R}), \emptyset\rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H\rangle$, where

$$H = \left\{ \begin{array}{lcl} +(x, +(\mathsf{s}(y), z)) & \to & \mathsf{s}(+(x, +(z, y))), \\ +(x, +(0, y)) & \to & +(x, y), \\ +(x, +(y, z)) & \to & +(x, +(z, y)), \\ +(x, +(y, \mathsf{sum}(zt))) & \to & +(x, +(\mathsf{sum}(zt), y)), \\ +(x, +(\mathsf{sum}(yt), \mathsf{sum}(zt))) & \to & +(x, +(\mathsf{sum}(zt), \mathsf{sum}(yt))) \end{array} \right\}.$$

Thus, from Theorem 18 it follows that $\mathcal{R}$ is ground confluent.

5 Disproving Ground Confluence

In this section, we deal with methods to disprove ground confluence. We present two mechanisms to disprove ground confluence – the first one is given as an additional inference rule of rewriting induction and the second one is a general method that is obtained by modifying a method for disproving confluence [1]. From here on, for some set A of terms, rules, etc. we denote those of sort τ by A^τ if the sort information is necessary.

Non-confluence is usually shown by selecting some candidates of non-joinable term-pair $\langle t, u\rangle$ such that $t \,{}^*\!\leftarrow \circ \to^* u$, and proving t and u are not joinable. This idea is naturally incorporated into the setting of ground confluence disproving. However, one needs to be careful about the rewrite sequence for counterexample can be instantiated by ground terms; that is, it may happen that $t_g \,{}^*\!\leftarrow s \to^* u_g$ for ground terms u_g, t_g and term s, but there is no ground instantiation of s, and in such a case, even if t_g and u_g are not joinable, this does not imply ground non-confluence of the system. To avoid such pitfalls, it is better to exclude rewrite rules that can not be instantiated by ground terms. This motivates us to introduce the following definitions.

▶ **Definition 20** (redundant rewrite rules)**.** A sort τ is said to be *redundant* if $\mathcal{T}(\mathcal{F})^\tau = \emptyset$. A rule $l \to r \in \mathcal{R}$ is redundant if there exists a redundant sort τ such that (1) $l \to r \in \mathcal{R}^\tau$ or (2) $\mathcal{V}(l) \cap \mathcal{V}^\tau \neq \emptyset$. A TRS $\mathcal{R}$ is non-redundant if $\mathcal{R}$ contains no redundant rules.

It is easy to see redundancy is decidable. Since removing redundant rules preserves ground (non-)confluence, one can work with non-redundant TRSs to prove or disprove ground confluence:

▶ **Theorem 21.** *Let $\mathcal{R}'$ be obtained by removing redundant rules in $\mathcal{R}$. Then, $\mathcal{R}'$ is ground confluent if and only if so is $\mathcal{R}$.*

Proof. It suffices to have $\to_{\mathcal{R}} = \to_{\mathcal{R}'}$ on $\mathrm{T}(\mathcal{F})$. ($\supseteq$) is clear as $\mathcal{R}' \subseteq \mathcal{R}$. Suppose $s_g \to_{\mathcal{R}} t_g$. Then $s_g \to_{l\to r} t_g$ for some non-redundant rule $l \to r \in \mathcal{R}$ since no redundant rule is applied in a ground rewrite step. Thus, $s_g \to_{\mathcal{R}'} t_g$. ◀

Any rewrite sequence $s \to^* t$ by non-redundant rules has a ground instantiation. Thus, for non-redundant TRS $\mathcal{R}$, if $t \,{}^+_{\mathcal{R}}\!\leftarrow \circ \to^+_{\mathcal{R}} u$ and there exists ground instantiations $t\theta_g$ and $u\theta_g$ which are not joinable, then $\mathcal{R}$ is not ground confluent. *In the remainder of this section, we assume $\mathcal{R}$ is non-redundant.*

Several inference rules of rewriting induction for refuting inductive conjectures are known (e.g. [6, 14]) – for example, when $\mathcal{R}$ has free constructors, if there exists $s \doteq t \in E$ with $s(\epsilon), t(\epsilon) \in \mathcal{C}$ and $s(\epsilon) \neq t(\epsilon)$, one can infer $\bot$ from $\langle E, H\rangle$, meaning that the initial equation of the derivation is not an inductive theorem of $\mathcal{R}$. Here, the correctness of such rules is guaranteed by the assumption that $\mathcal{R}$ *is confluent* [14]. On the other hand, since we are dealing with proving ground confluence of $\mathcal{R}$, it is not appropriate to assume *confluence* of $\mathcal{R}$. However, as we will see below, it turns out that not the same but a similar idea can be used for refuting ground confluence.

First, we need a couple of preparations before presenting our inference rule for refuting ground confluence.

▶ **Definition 22.** A function symbol f is said to be *stable* (in a TRS $\mathcal{R}$) if for any rewrite rule $l \to r \in \mathcal{R}$, $f = l(\epsilon)$ implies $f = r(\epsilon)$. A term s is *root-stable* if $s(\epsilon) \in \mathcal{F}$ and $s(\epsilon)$ is stable.

Disprove

$$\frac{\langle E \cup \{s \doteq t\},\ H\rangle}{\bot} \quad s \not\approx_{\mathcal{R}} t$$

Figure 4 Inference rule for disproving.

A term s is a *minimal form* (of a TRS $\mathcal{R}$) if $s \to_{\mathcal{R}} t$ implies $s = t$. A term is a ground minimal form if it is ground and a minimal form. We denote the set of ground minimal forms of $\mathcal{R}$ of sort τ by $\mathrm{GMF}(\mathcal{R})^\tau$.

We are now ready to present the key definition of our inference rule.

▶ **Definition 23.** For terms s and t of the same sort, we write $s \not\approx_{\mathcal{R}} t$ if either

(i) s, t are root-stable (in $\mathcal{R}$) with $s(\epsilon) \neq t(\epsilon)$,
(ii) $s(\epsilon) = t(\epsilon) \notin \{l(\epsilon) \mid l \to r \in \mathcal{R}\}$ and $s|_i \not\approx t|_i$ for some $1 \leq i \leq \mathrm{arity}(s(\epsilon))$,
(iii) $s \in \mathcal{V}^\tau \setminus \mathcal{V}(t)$ and there exist $u_g, v_g \in \mathrm{GMF}(\mathcal{R})^\tau$ such that $u_g \neq v_g$, or
(iv) $s \in \mathcal{V}^\tau$, t is root-stable and there exists a root-stable term u of sort τ such that $u(\epsilon) \neq t(\epsilon)$.

The intended property of the relation $\not\approx_{\mathcal{R}}$ is as follows (see Appendix C for the proof).

▶ **Lemma 24.** *Suppose $\mathcal{V}(s) \cup \mathcal{V}(t)$ contains no variable of redundant sort. If $s \not\approx_{\mathcal{R}} t$, then there exists a ground substitution θ_g such that $s\theta_g$ and $t\theta_g$ are not joinable by $\mathcal{R}$.*

▶ **Definition 25.** Rewriting induction with disproof is obtained by adding the inference rule in Figure 4, where $\mathcal{R}$ is the input TRS of the problem.

▶ **Lemma 26.** *Let $\langle E_0, \emptyset\rangle \overset{*}{\rightsquigarrow} \langle E_i, H_i\rangle$. (1) $l \leftrightarrow^*_{E_0 \cup \mathcal{R}} r$ for any $l \doteq r \in E_i$ and $l \to r \in H_i$. (2) $\mathcal{R} \models_{ind} E_0$ then $\mathcal{R} \models_{ind} E_i$.*

Proof. (1) Straightforward, using induction on the length of $\langle E_0, \emptyset\rangle \overset{*}{\rightsquigarrow} \langle E_i, H_i\rangle$. (2) Immediately follows from (1). ◀

▶ **Theorem 27.** *Suppose $\mathcal{R}$ is non-redundant and $\mathcal{R} \models_{ind} \mathcal{R}_0$. If $\langle \mathrm{CP}(\mathcal{R}_0) \cup (\mathcal{R} \backslash \mathcal{R}_0), \emptyset\rangle \overset{*}{\rightsquigarrow} \bot$ then $\mathcal{R}$ is not ground confluent.*

Proof. Suppose $\langle \mathrm{CP}(\mathcal{R}_0) \cup (\mathcal{R} \backslash \mathcal{R}_0), \emptyset\rangle \overset{*}{\rightsquigarrow} \langle E, H\rangle \rightsquigarrow \bot$. Then, by the definition of *Disprove* inference rule, there exists $s \doteq t \in E$ such that $s \not\approx_{\mathcal{R}} t$. Hence, by Lemma 24, there exists ground instances $s\theta_g, t\theta_g$ that are not joinable by $\mathcal{R}$. On the other hand, since $\mathcal{R} \models_{ind} \mathrm{CP}(\mathcal{R}_0) \cup (\mathcal{R} \backslash \mathcal{R}_0)$, we have $\mathcal{R} \models_{ind} s \doteq t$ by Lemma 26. Hence, by non-redundancy of $\mathcal{R}$, we have a ground rewrite sequence $s\theta_g \leftrightarrow^*_{\mathcal{R}} t\theta_g$. Thus, $\mathcal{R}$ is not ground confluent. ◀

Note here the Disprove inference rule should be used with the side condition $s \not\approx_{\mathcal{R}} t$ and not with $s \not\approx_{\mathcal{R}_0} t$. This is a sharp contrast with other inference rules, which are concerned with $\mathcal{R}_0$ (and not with $\mathcal{R}$). The following example illustrates such replacement is incorrect.

▶ **Example 28.** Let $\mathcal{R}$ be as follows:

$$\{\ \mathsf{b} \to \mathsf{c}\ \ (a) \qquad \mathsf{c} \to \mathsf{b}\ \ (b) \qquad \mathsf{b} \to \mathsf{a}\ \ (c)\ \}$$

Note that $\mathcal{R}$ is ground confluent. Take $\mathcal{C} = \{\mathsf{a}, \mathsf{c}\}$ and $\mathcal{R}_0 = \{(a)\}$. Then, $\mathrm{CP}(\mathcal{R}_0) \cup (\mathcal{R} \backslash \mathcal{R}_0) = \{(b), (c)\}$, and thus, a rewriting induction derivation $\langle \{(b), (c)\}, \emptyset\rangle \rightsquigarrow^* \langle \{\mathsf{c} \doteq \mathsf{a}\}, \emptyset\rangle$. Now, c, a are root-stable in $\mathcal{R}_0$ and $\mathsf{c} \neq \mathsf{a}$. Hence, $\mathsf{c} \not\approx_{\mathcal{R}_0} \mathsf{a}$. Thus, if we had replaced the condition $s \not\approx_{\mathcal{R}} t$ with $s \not\approx_{\mathcal{R}_0} t$, we could have derived $\bot$.

In [5] a procedure with SPIKE for disproving ground confluence is proposed, but it works on the assumption that $\mathcal{R}$ is terminating; thus, it cannot deal with non-orientable constructor rules. On the other hand, Theorem 27 can be applied to $\mathcal{R}$ having non-orientable constructor rules.

To end the section, we explain how methods for disproving confluence [1] can be incorporated to prove ground non-confluence. The basic idea of disproving confluence is to take some terms s, t such that $s \ {}^{+}\!\leftarrow \circ \rightarrow^{+} t$ and show s and t are not joinable. Since such s, t are obtained by taking forward closure or backward closure of rewrite rules and computing critical pairs.

In the case of disproving ground confluence we first instantiate s, t with some ground terms. Thus, we take some $\langle s\theta_g, t\theta_g \rangle$ as a candidate. The rest of technique to show non-joinability of $s\theta_g, t\theta_g$ remain the same. Note we first need to remove redundant rules beforehand, otherwise, $s \ {}^{+}\!\leftarrow \circ \rightarrow^{+} t$ may not have ground instantiation, and the non-joinability of $s\theta_g, t\theta_g$ does not necessarily implies ground non-confluence.

6 Implementation and Experiments

All the techniques presented in this paper have been implemented in our ground confluence prover AGCP [2]. Our improved procedure is described as follows.

Improved GCR Procedure

Input: many-sorted TRS $\mathcal{R}$
Output: `YES`, `NO` or `MAYBE`

1. Check redundancy of sorts and remove redundant rules (Section 5).
2. Compute (possibly multiple) candidates for the partition $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$ of function symbols.
3. Select constructor rules from $\mathcal{R}$ and construct a constructor subsystem $\mathcal{R}_c$ by the rule instantiation procedure (Section 3).
4. For each $f \in \mathcal{D}$, construct multiple candidates of defining rules for f using the rule complementation procedure (Section 3), and construct $\mathcal{R}_0$ from $\mathcal{R}_c$ by adding a candidate for each.
5. Find a reduction quasi-order $\succsim$ satisfying conditions of Theorem 18 for $\mathcal{R}_0$ (Section 4). If it fails try another candidate of defining rules; if the candidates are exhausted, try another partition $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$. Run rewriting induction to obtain $\langle \mathrm{CP}_{\succsim}(\mathcal{R}_0) \cup (\mathcal{R} \setminus \mathcal{R}_0), \emptyset \rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H \rangle$ for some H or $\langle \mathrm{CP}_{\succsim}(\mathcal{R}_0) \cup (\mathcal{R} \setminus \mathcal{R}_0), \emptyset \rangle \overset{*}{\rightsquigarrow} \bot$. (Sections 4 and 5). If it succeeds, return `YES` or `NO` accordingly. If the number of rewriting induction steps exceeds a limit, then try another candidate of defining rules.
6. Run the ground non-confluence check incorporated from the methods for disproving confluence [1] (Section 5). If it fails, return `MAYBE`.

Below we explain some details of the procedure and heuristics employed in our implementation.

- We take $l \rightarrow r \in \mathcal{R}$ satisfying $\mathcal{F}(l) \cup \mathcal{F}(r) \subseteq \mathcal{C}$ as a constructor rule; those $\mathcal{F}(l) \cup \mathcal{F}(r) \not\subseteq \mathcal{C}$ and $l(\epsilon) \notin \mathcal{D}$ are moved to the conjecture part. Succeedingly, $\mathcal{R}_c$ is modified by the rule instantiation procedure in Section 3. For checking ground confluence of $\mathcal{R}_c$, few basic techniques for checking confluence are employed.

Table 1 Test on 244 examples of CoCo 2016 GCR demonstration category.

	AGCP (A) (coco2016)	FORT (F) (coco2016)	ALL	$A \setminus ALL$	$F \setminus ALL$
YES	105	83	140	0	43
NO	0	32	34	0	2
YES+NO	105	115	174	0	45

- For each $f \in \mathcal{D}$, we construct multiple candidates of defining rules for f as follows. Firstly, from $\mathcal{R}$, we exclude the rules whose lhs is not a linear basic term and those having a subterm of r that unifies with l, and such rules are moved to the conjecture part; let the result be $\mathcal{S}$. Then, we try to select minimal subsets of f-rules in $\mathcal{S}$ that is strongly quasi-reducible. If there is no such subset, we perform three ways to supplement rewrite rules using the rule complementation procedure in Section 3.
  - If $\mathrm{LHS}(f, \mathcal{S})$ is non-empty, a complement pattern of $\mathrm{LHS}(f, \mathcal{S})$ is computed and rewrite rules for the such patterns are added.
  - If $\mathrm{LHS}(f, \mathcal{S})$ is empty, we take basic patterns such as only i-th argument is extended (i.e. $\{f(x_1, \ldots, x_{i-1}, u, x_{i+1}, \ldots, x_n) \mid u = c(\bar{y}), c \in \mathcal{C}\}$) and those and all arguments are extended, and rewrite rules for the such patterns are added.
  - If there is a rule such that recursive call on the rhs of the rule has an argument, say u, whose root symbol is a defined symbol, then we seek for rewrite rules $l \to r$ and σ such that $\sigma = \mathrm{mgu}(l, u)$ and take a rewrite rule for the pattern $l\sigma$.

  Each of these cases, for each pattern l, we seek for reducts of l within some steps, and take a term r of minimal size to generate an added rule $l \to r$.
- A reduction quasi-order used in the rewriting induction is selected from recursive path orders with multiset and lexicographic status. We obtain it by encoding ordering constraints expressing that there is at least one candidate of strictly decreasing strongly quasi-reducible defining rules for each defined function symbol and constructor rules are weakly decreasing, and solving them using an SMT-solver.
- For the marked term $t^\bullet$, we try to remove the mark at the very first steps of the rewriting induction.
- *Modify* rules is used before applying *Simplify* or *Delete* rules – this is a similar heuristics to the one how weakly decreasing rewrite steps by $H^{\leftrightarrow}$ are included [2].
- We impose a limitation of the length of rewriting induction derivation; when the length reaches the limit, we seek for a defined symbol, say f, whose defining rules may be a cause of the divergence by checking the root of innermost basic subterms of rewrite rules in H-part. We then switch the defining rules for f to the next candidate.

We have tested our prover on the collection of 244 examples that was used in the GCR demonstration category of Confluence Competition 2016. The collection is contributed by two participants AGCP [2] and FORT [12] of the category. The result is summarized in Table 1. Our new prover succeeds in proving ground confluence or non-confluence for 174 problems in total (shown in the column below ALL), where as AGCP and FORT prove 105 and 115, respectively (shown in the columns below AGCP and FORT). The row titled YES shows the number of success in ground confluence proving and the one titled NO shows the number of success in ground non-confluence proving. The total increase from AGCP is 65, and the increase in proving ground confluence is 35 and those in proving ground non-confluence is 34. $A \setminus ALL$ ($F \setminus ALL$) denotes the number of problems for which A (resp. F) succeeds but our prover fails. Table 2 shows the analysis on which technique contributes the success in 69

Table 2 Analyzing effective techniques in 69 examples from $ALL \setminus A$

	redun-ancy	RI disproof	general disproof	comple-mentation	instan-tiation	non-ori. con. rules	others	total
YES	2	–	–	13	13	4	6	35
NO	(34)	6	28	–	–	-	0	34

Table 3 Test on new 55 examples

	ACP	AGCP	ALL
YES	0	13	30
NO	43	0	3
MAYBE	12	42	22

examples in $ALL \setminus A$. It can be seen that each of presented techniques affects for increasing the power of the prover. "Other" refers to improvements using more sophisticated strategies and reduction orderings.

Many problems in the collection of 244 examples of CoCo 2016 GCR demonstration category stem from problems for confluence of first-order TRSs, and thus it contains many problems for which confluence of unsorted versions can be proved by the state-of-the-art confluence provers. This motivates us to construct problems which are non-confluent or for which confluence proof can not be handled by the state-of-the-art confluence provers. Thus, we constructed new 55 examples including Examples 3, 4, 9, 13, all of which are non-confluent (indicated by NO) or can not be handled (indicated by MAYBE) by the confluent prover ACP [3]. The result is summarized in Table 3. All the details of the experiments are available at `http://www.nue.ie.niigata-u.ac.jp/tools/agcp/experiments/fscd17/`.

7 Conclusion

In this paper, we have presented methods that strengthen the rewriting induction approach for proving ground confluence of many-sorted term rewriting systems. The first method is concerned with how to replace or supplement initial rewrite rules to obtain an appropriate set of rewrite rules for rewriting induction work. The second method is concerned with the non-orientable constructor rules for which the first method can not deal with. For this, we have extended rewriting induction inference system to deal with weakly decreasing rewrite rules. Then we obtain a correctness criteria for the ground confluence proving so that one can deal with non-orientable constructor rules. We believe that our extension provides a basis for dealing with term rewriting systems acting on more flexible data structures. As the last ingredient, we have presented methods to deal with proving ground non-confluence. All of these techniques have been implemented and experiments have shown that presented methods are effective to deal with problems for which state-of-the-art ground confluence provers can not handle.

Acknowledgements. Thanks are due to the anonymous reviewers.

References

1 T. Aoto. Disproving confluence of term rewriting systems by interpretation and ordering. In *Proc. of 9th FroCoS*, volume 8152 of *LNAI*, pages 311–326. Springer-Verlag, 2013.

2 T. Aoto and Y. Toyama. Ground confluence prover based on rewriting induction. In *Proc. of 1st FSCD*, volume 52 of *LIPIcs*, pages 33:1–12. Schloss Dagstuhl, 2016. doi:10.4230/LIPIcs.FSCD.2016.33.

3 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.

4 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.

5 A. Bouhoula. Simultaneous checking of completeness and ground conflunce for algebraic specifications. *ACM Transactions on Computational Logic*, 10(2):20:1–33, 2009.

6 A. Bouhoula, E. Kounalis, and M. Rusinowitch. Automated mathematical induction. *Journal of Logic and Computation*, 5(5):631–668, 1995.

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

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

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

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

11 A. Lazrek, P. Lescanne, and J. J. Thiel. Tools for proving inductive equalities, relative completeness, and ω-completeness. *Information and Computation*, 84:47–70, 1990.

12 F. Rapp and A. Middeldorp. Automating the first-order theory of rewriting for left-linear right-ground rewrite systems. In *Proc. of 1st FSCD*, volume 52 of *LIPIcs*, pages 36:1—-12. Schloss Dagstuhl, 2016.

13 U. S. Reddy. Term rewriting induction. In *Proc. of CADE-10*, volume 449 of *LNAI*, pages 162–177. Springer-Verlag, 1990.

14 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.

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

16 V. van Oostrom. Confluence by decreasing diagrams. *Theoretical Computer Science*, 126(2):259–280, 1994.

A Proof of Lemma 16

We first present a couple of preparations. Elements x and y are *convertible below* z if $x = x_0 \leftrightarrow x_1 \leftrightarrow \cdots \leftrightarrow x_n = y$ for some $x_0, \ldots, x_n$ such that $z \succ x_i$ $(0 \leq i \leq n)$; we write $x \leftrightarrow^*_{\prec z} y$ if x and y are convertible below z.

A labelled relation $\to = \bigcup_{i \in \mathcal{L}} \to_i$ is *locally decreasing* if there exists a well-founded partial order $\succ$ on $\mathcal{L}$ such that ${}_l\!\leftarrow \circ \to_m \subseteq \leftrightarrow^*_{\Upsilon l} \circ \to^{=}_m \circ \leftrightarrow^*_{\Upsilon l,m} \circ \overset{=}{{}_l\!\leftarrow} \circ \leftrightarrow^*_{\Upsilon m}$, where $\to_{\Upsilon l} = \bigcup_{i \prec l} \to_i$ and $\to_{\Upsilon l,m} = \to_{\Upsilon l} \cup \to_{\Upsilon m}$. Any locally decreasing relation is confluent [16].

▶ **Lemma 29.** *Let $\succsim$ be a well-founded quasi-order. Suppose $\to = \to_d \cup \to_e$, $\to_d \subseteq \succ$ and $\to_e \subseteq \succsim$. Then, $\to$ is confluent if the following three conditions are satisfied:*

(a) *$y \overset{\prec}{\leftarrow} x \to_d z$ implies $y \leftrightarrow^*_{\prec x} z$,*

(b) *$y \overset{\approx}{\leftarrow} x \to_d z$ implies $y \to_d \circ \leftrightarrow^*_{\prec x} z$, and*

(c) *$y \leftarrow_e \circ \to_e z$ implies $y \to^{=}_e \circ \overset{=}{\leftarrow}_e z$.*

Proof. Consider the source labeling, i.e. the labeling such that each $x \to y$ is labelled with x. Then, we easily obtain that $\to$ is locally decreasing; thus, it is confluent. ◀

The diamond property (c) in Lemma 29 can be relaxed to the confluence property (c') as follows.

▶ **Lemma 30.** *Let $\succsim$ be a well-founded quasi-order. Suppose $\to = \to_d \cup \to_c$, $\to_d \subseteq \succ$ and $\to_c \subseteq \succsim$. Then, $\to$ is confluent if the following three conditions are satisfied:*
(a') *$y \overset{\prec}{\leftarrow} x \to_d z$ implies $y \leftrightarrow^*_{\prec x} z$,*
(b') *$y \overset{\approx}{\leftarrow} x \to_d z$ implies $y \to_d \circ \leftrightarrow^*_{\prec x} z$, and*

(c') *$\to_c$ is confluent.*

Proof. Take $\to^*_c$ as $\to_e$ and apply Lemma 29. We have $\to^*_c \subseteq \succsim$, and the condition (c) in Lemma 29 is obviously satisfied. We next show (b) in Lemma 29. We show ${}^n\overset{\approx}{\leftarrow}_c x \to_d \subseteq \to_d \circ \leftrightarrow^*_{\prec x}$ ($n \geq 0$) by induction on n. Let $y \; {}^n\overset{\approx}{\leftarrow}_c x' \leftarrow_c x \to_d z$. From (b') we have $y \overset{n}{\leftarrow}_c x' \to_d u \leftrightarrow^*_{\prec x} z$ for some u. From induction hypothesis, it follows that $y \to_d \circ \leftrightarrow^*_{\prec x'} u \leftrightarrow^*_{\prec x} z$. As $x \approx x'$, we obtain $y \to_d \circ \leftrightarrow^*_{\prec x} z$. Thus, (b) has been proved. We now show (a) in Lemma 29. Suppose $y \overset{\prec}{\leftarrow} x \to_d z$. The case $y \; {}_d\!\leftarrow x \to_d z$ is clear. Otherwise, $y \; {}^*\!\leftarrow_c y'' \overset{\prec}{\leftarrow}_c y' \; {}^*\overset{\approx}{\leftarrow}_c x \to_d z$ for some y', y''. By (b), $y \; {}^*\!\leftarrow_c y'' \overset{\prec}{\leftarrow}_c y' \to_d \circ \leftrightarrow^*_{\prec x} z$. From $y' \approx x$, we obtain $y \leftrightarrow^*_{\prec x} z$. Thus, (a) has been proved. Now, applying Lemma 29, we obtain that $\to_d \cup \overset{*}{\to}_c$ is confluent. From $\to \subseteq \to_d \cup \to^*_c \subseteq \overset{*}{\to}$, it follows that $\to$ is confluent. ◀

The convertibility below x conditions (a') and (b') in Lemma 30 can be replaced with the bounded convertibility conditions (a") and (b") as follows.

▶ **Lemma 31.** *Let $\succsim$ be a well-founded quasi-order. Suppose $\to = \to_d \cup \to_c$, $\to_d \subseteq \succ$ and $\to_c \subseteq \succsim$. Then, $\to$ is confluent if the following three conditions are satisfied:*
(a") *$y \overset{\prec}{\leftarrow} \circ \to_d z$ implies $y \leftrightarrow^*_{\succsim} z$,*
(b") *$y \overset{\approx}{\leftarrow} \circ \to_d z$ implies $y \to_d \circ \leftrightarrow^*_{\succsim} z$, and*
(c") *$\to_c$ is confluent.*

Proof. From (a") and (b") it is obvious that (a') and (b') in Lemma 30 are satisfied. ◀

Therefore, Lemma 16 follows from Lemma 31 immediately by taking $\to = \to_{\mathcal{R}}$, $\to_c = \to_{\mathcal{R}_c}$ and $\to_d = \to_{\mathcal{R}_d}$.

B Proofs of Theorems 15 and 18

Proof of Theorem 15. Let $\langle E_0, H_0\rangle \rightsquigarrow \langle E_1, H_1\rangle \rightsquigarrow \cdots \rightsquigarrow \langle E_m, H_m\rangle$ with $H_0 = \emptyset$ and $E_m = \emptyset$. 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. We prove the theorem by contradiction. We first introduce a well-founded order on equations $s \doteq t \succ s' \doteq t'$ by $\{s,t\} \succ_m \{s',t'\}$, where $\succ_m$ denotes the multiset extension of $\succ$. Suppose that E_0 is not bounded ground convertible. Then there exists a minimal ground equation $s\sigma_g \doteq t\sigma_g$ that is an instance of $s \doteq t \in E_\infty$ and not bounded convertible. From the minimality of $s\sigma_g \doteq t\sigma_g$ and the strong quasi-reducibility of $\mathcal{R}^{\succ}$, σ_g must be a ground constructor substitution on $\mathcal{V}(s) \cup \mathcal{V}(t)$. We prove the following claim.

▶ **Claim.** *If $s \doteq t \in E_k$ then there exists $s' \doteq t \in E_{k+1}$ and $s'\sigma_g \doteq t\sigma_g$ is a minimal ground equation that is not bounded convertible.*

Proof of Claim. If no inference rule is applied to $s \doteq t$ in $\langle E_k, H_k\rangle \rightsquigarrow \langle E_{k+1}, H_{k+1}\rangle$, the claim is obvious. We distinguish four cases by the inference rule applied and show a contradiction except for (*Modify*). Note that $H \subseteq E_\infty$ in the following cases. Thus if $s \leftrightarrow_H t$ for ground terms s, t then there exist $u \doteq v \in E_\infty$ and θ_g such that $s = u\theta_g, t = v\theta_g$.

1. (*Expand*) 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 $\mathrm{Expd}^{\succ}_u(s,t) = \{s_i \to t_i\}_i$, $u \in \mathcal{B}(s)$, and $s_i \overset{\succ}{\underset{H\leftrightarrow}{}}{}^{*} s'_i$ for each i. Since σ_g is a ground constructor substitution, we have $s\sigma_g \to_{\mathcal{R}} s_i\theta_g \to_{\mathrm{Expd}^{\succ}_u(s,t)} t\sigma_g$ and $s\sigma_g \succ s_i\theta_g$ for some θ_g and i. Thus, $s\sigma_g \to_{\mathcal{R}} s_i\theta_g \overset{\succ}{\underset{H\leftrightarrow}{}}{}^{*} s'_i\theta_g \leftrightarrow_{E_\infty} t\sigma_g$. Then, for any step $u_g \leftrightarrow_H v_g$ in $s_i\theta_g \overset{\succ}{\underset{H\leftrightarrow}{}}{}^{*} s'_i\theta_g$, we have $s\sigma_g \succ u_g, v_g$, and hence $s\sigma_g \doteq t\sigma_g \succ u_g \doteq v_g$. As $s\sigma_g \succ s'_i\theta_g$, $s\sigma_g \doteq t\sigma_g \succ s'_i\theta_g \doteq t\sigma_g$. From the minimality of $s\sigma_g \doteq t\sigma_g$, it follows that $u_g \doteq v_g$ and $s'_i\theta_g \doteq t\sigma_g$ are bounded convertible. Thus $s\sigma_g \doteq t\sigma_g$ is bounded convertible; this contradicts the choice of $s\sigma_g \doteq t\sigma_g$.
2. (*Simplify*) We have $\langle E \uplus \{s \doteq t\}, H\rangle \rightsquigarrow \langle E \cup \{s' \doteq t\}, H\rangle$ for some E, H, where $s \overset{\succ}{\underset{\mathcal{R}\cup H}{}}{} \hat{s} \overset{\succ}{\underset{H\leftrightarrow}{}}{}^{*} s'$. Then, $s \overset{\succ}{\underset{\mathcal{R}\cup H}{}}{} \hat{s} = s_1 \leftrightarrow_H s_2 \leftrightarrow_H \cdots \leftrightarrow_H s_k = s' \leftrightarrow_{E_\infty} t$ with $s_i \succsim s_{i+1}$ for $i = 1, \ldots, k-1$. We distinguish two cases.
   - **a.** Case $s \overset{\succ}{\underset{\mathcal{R}}{}}{} \hat{s}$. Then $s\sigma_g \succ \hat{s}\sigma_g$ and thus $s\sigma_g \succ s_i\sigma_g$ for $i = 1, \ldots, k$. Hence, we have $s\sigma_g \doteq t\sigma_g \succ s_i\sigma_g \doteq s_{i+1}\sigma_g$ for $i = 1, \ldots, k-1$ and $s\sigma_g \doteq t\sigma_g \succ s'\sigma_g \doteq t\sigma_g$. As $s_i\sigma_g \doteq s_{i+1}\sigma_g$ $(i = 1, \ldots, k-1)$ and $s'\sigma_g \doteq t\sigma_g$ are bounded convertible, $s\sigma_g \doteq t\sigma_g$ is bounded convertible, contradiction.
   - **b.** Case $s \overset{\succ}{\underset{H}{}}{} \hat{s}$. 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 $s\sigma_g \doteq t\sigma_g \succ s_i\sigma_g \doteq s_{i+1}\sigma_g$ for $i = 1, \ldots, k-1$ and $s\sigma_g \doteq t\sigma_g \succ s'\sigma_g \doteq t\sigma_g$. Thus, $s_i\sigma_g \doteq s_{i+1}\sigma_g$ $(i = 1, \ldots, k-1)$ and $s'\sigma_g \doteq t\sigma_g$ are bounded convertible. By $s \overset{\succ}{\underset{H}{}}{} \hat{s}$, 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 $s\sigma_g = C[w\theta]\sigma_g = C\sigma_g[w\theta_g] \overset{+}{\underset{\mathcal{R}}{}}{} s\rho_g$ (or $t\sigma_g \overset{+}{\underset{\mathcal{R}}{}}{} t\rho_g$) for some ground constructor substitution ρ_g and $s\sigma_g \doteq t\sigma_g \succ s\rho_g \doteq t\rho_g$. As $s\rho_g \doteq t\rho_g$ is bounded convertible, it follows that $s\sigma_g \doteq t\sigma_g$ is bounded convertible, contradiction. Thus, suppose that $\theta_g = \sigma_g \circ \theta$ is a constructor ground substitution. As $w \to \hat{w} \in H$, there exists some $p < k$ such that *Expand* rule is applied to $w \doteq \hat{w} \in E_p$ with $u \in \mathcal{B}(w)$ in $\langle E_p, H_p\rangle \rightsquigarrow \langle E_{p+1}, H_{p+1}\rangle$. Thus, from $H_p \subseteq H$ it follows that $w\theta_g \to_{\mathcal{R}} \circ \overset{\succ}{\underset{H\leftrightarrow}{}}{}^{*} \circ \leftrightarrow_{E_\infty} \hat{w}\theta_g$. Hence, $s\sigma_g = C[w\theta]\sigma_g = C\sigma_g[w\theta_g] \to_{\mathcal{R}} w_g \overset{\succ}{\underset{H\leftrightarrow}{}}{}^{*} w'_g \leftrightarrow_{E_\infty} C\sigma_g[\hat{w}\theta_g] = C[\hat{w}\theta]\sigma_g = \hat{s}\sigma_g$ and $s\sigma_g \succ w_g$. Furthermore, for any step $u_g \leftrightarrow_H v_g$ in $w_g \overset{\succ}{\underset{H\leftrightarrow}{}}{}^{*} w'_g$, we have $s\sigma_g \doteq t\sigma_g \succ u_g \doteq v_g$. As $s_i\sigma_g \doteq s_{i+1}\sigma_g$ $(i = 1, \ldots, k-1)$, $s'\sigma_g \doteq t\sigma_g$ and $u_g \doteq v_g$ are bounded convertible, it follows that $s\sigma_g \doteq t\sigma_g$ is bounded convertible, contradiction.
3. (*Modify*) We have $\langle E \uplus \{s \doteq t\}, H\rangle \rightsquigarrow \langle E \cup \{s' \doteq t\}, H\rangle$ for some E, H, where $s \overset{\approx}{\underset{\mathcal{R}}{}}{} s'$. If $s'\sigma_g \doteq t\sigma_g$ is bounded convertible, it follows that $s\sigma_g \doteq t\sigma_g$ is bounded convertible, contradiction. Hence $s'\sigma_g \doteq t\sigma_g$ is not bounded convertible. As $s \approx s'$, the minimality of $s'\sigma_g \doteq t\sigma_g$ is clear.
4. (*Delete*) We have $\langle E \uplus \{s \doteq t\}, H\rangle \rightsquigarrow \langle E, H\rangle$, where $s = t$ or $s \to_H t$. From the choice of $s\sigma_g \doteq t\sigma_g$, we have $s \neq t$. Let $s \to_H t$ and $w \to \hat{w} \in H$ such that $s = C[w\theta]$, $t = C[\hat{w}\theta]$ for some θ. Then, there exists some $p < k$ such that *Expand* rule is applied to $w \doteq \hat{w} \in E_p$ in $\langle E_p, H_p\rangle \rightsquigarrow \langle E_{p+1}, H_{p+1}\rangle$. Hence, in the same way as the case (*Simplify*)-b, it is shown that $s\sigma_g \doteq t\sigma_g$ is bounded convertible, contradiction.

Therefore, *Claim* holds. ◀

From *Claim*, we easily obtain $E_n \neq \emptyset$ $(k \leq n \leq m)$; this contradicts $E_m = \emptyset$. ◀

Proof of Theorem 18. We show the conditions of Lemma 16 are satisfied. To show the condition (i), suppose $u_g \leftarrow_{\mathcal{R}^\succ} \circ \rightarrow_{\mathcal{R}_d} v_g$. If the redexes contracted in the peak $u_g \leftarrow_{\mathcal{R}^\succ} \circ \rightarrow_{\mathcal{R}_d} v_g$ occur at independent positions or variable overlapping positions then it is trivial. If the peak is critical overlapping, then by $\mathrm{CP}(\mathcal{R}^\succ, \mathcal{R}_d) \subseteq \mathrm{CP}_{\succsim}(\mathcal{R})$ and Theorem 15, we obtain $u_g \overset{*}{\leftrightarrow}_{\mathcal{R}^\succsim} v_g$. It remains to show the condition (ii) of Lemma 16, that is, the claim that if $u_g \leftarrow_{\mathcal{R}^\approx} \circ \rightarrow_{\mathcal{R}_d} v_g$ then $u_g \rightarrow_{\mathcal{R}_d} \circ \overset{*}{\leftrightarrow}_{\mathcal{R}^\precsim} v_g$ for ground terms u_g, v_g. If the redexes contracted in the peak $u_g \leftarrow_{\mathcal{R}^\approx} \circ \rightarrow_{\mathcal{R}_d} v_g$ occur at independent positions, then it is trivial. If the peak is variable overlapping then from the left-linearity of $\mathcal{R}^\succ$ and the variable-preserving property of $\mathcal{R}^\approx$, it obviously follows. Suppose that the peak is critical overlapping and $u_g = C[s\theta_g]$, $v_g = C[t\theta_g]$, $s \mathbin{\widehat{\succ}} t \in \mathrm{CP}(\mathcal{R}^\approx_c, \mathcal{R}_d)$ or $t \mathbin{\widehat{\succ}} s \in \mathrm{CP}(\mathcal{R}_d, \mathcal{R}^\approx_c)$. Then, from $s^\bullet \doteq t \in \mathrm{CP}_{\succsim}(\mathcal{R})$ and $\langle \mathrm{CP}_{\succsim}(\mathcal{R}), \emptyset \rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H \rangle$ it follows that *Expand*, *Simplify* or *Delete* rule is eventually applied to $s^\bullet \doteq t$. Thus, in a similar way to the proof of Theorem 15 we obtain that $s\theta_g \rightarrow_{\mathcal{R}_d} \circ \overset{*}{\leftrightarrow}_{\mathcal{R}^\precsim} t\theta_g$ for each applied inference rule. Note here we have $s \neq t$ by $s \mathbin{\widehat{\succ}} t \in \mathrm{CP}(\mathcal{R}^\approx_c, \mathcal{R}_d)$ or $t \mathbin{\widehat{\succ}} s \in \mathrm{CP}(\mathcal{R}_d, \mathcal{R}^\approx_c)$ as $\mathcal{R}_d \subseteq \succ$, and thus $s^\bullet \rightarrow_H t$ if *Delete* is applied. Therefore, $u_g = C[s\theta_g] \rightarrow_{\mathcal{R}_d} \circ \overset{*}{\leftrightarrow}_{\mathcal{R}^\precsim} C[t\theta_g] = v_g$. ◀

C Proof of Lemma 24

Proof. By induction on $|s|$.

1. Suppose $s\theta_g \rightarrow^* u_g \ {}^*\!\leftarrow t\theta_g$. Since $s, t \notin \mathcal{V}$, $s\theta_g(\epsilon) = s(\epsilon)$ and $t\theta_g(\epsilon) = t(\epsilon)$. By root stability, $s\theta_g(\epsilon) = u_g(\epsilon)$ and $t\theta_g(\epsilon) = u_g(\epsilon)$, and thus $s(\epsilon) = t(\epsilon)$. A contradiction to $s(\epsilon) \neq t(\epsilon)$.
2. Suppose $s(\epsilon) = t(\epsilon) \notin \{l(\epsilon) \mid l \rightarrow r \in \mathcal{R}\}$. Then $s\theta_g \rightarrow^* s'$ implies $s|_i\theta_g \rightarrow^* s'|_i$, and $t\theta_g \rightarrow^* t'$ implies $t|_i\theta_g \rightarrow^* t'|_i$. Thus if $s\theta_g \rightarrow^* u_g \ {}^*\!\leftarrow t\theta_g$ then $s\theta_g|_i \rightarrow^* u_g|_i \ {}^*\!\leftarrow t\theta_g|_i$. Thus $s|_i\theta_g$ and $t|_i\theta_g$ are joinable. On the other hand, by induction hypothesis and our assumption $s|_i \not\approx t|_i$, there exists θ_g such that $s|_i\theta_g$ and $t|_i\theta_g$ are not joinable. Hence, $s\theta_g \rightarrow^* u_g \ {}^*\!\leftarrow t\theta_g$ does not hold.
3. Take an arbitrary ground instantiation $t\theta_g$ of t. If $t\theta_g \rightarrow^* u_g$, then take $\rho_g = \theta_g \uplus \{s \mapsto v_g\}$ and then $s\rho_g = v_g$ and u_g are not joinable; hence, so are $s\rho_g$ and $t\rho_g$ $(= t\theta_g)$. Next, suppose otherwise, i.e. $t\theta_g \not\rightarrow^* u_g$. Then, take $\rho_g = \theta_g \uplus \{s \mapsto u_g\}$ and then $s\rho_g = u_g$ and $t\theta_g$ $(= t\theta_g)$ are not joinable by the assumption.
4. Take any ground instance t_g of t. Since $t_g(\epsilon) = t(\epsilon)$ is stable, for any v_g such that $t_g \rightarrow^* v_g$, we have $v_g(\epsilon) = t(\epsilon)$. Let u_g be a ground instance of root-table term u of sort τ. Then u_g is root-stable and $u_g(\epsilon) = u(\epsilon) \neq t(\epsilon)$ by our assumption. Thus $u_g \rightarrow^* u'_g$ implies $u'_g(\epsilon) = u_g(\epsilon)$. Take $\rho_g = \theta_g \uplus \{s \mapsto u_g\}$. Then $s\rho_g = u_g \rightarrow^* \circ \ {}^*\!\leftarrow t\theta_g$ implies $u_g(\epsilon) = t(\epsilon)$, which is a contradiction. ◀