Disproving Confluence of Term Rewriting Systems by Interpretation and Ordering

Takahito Aoto

RIEC, Tohoku University,
2-1-1 Katahira, Aoba-ku, Sendai 980-8577, Japan
aoto@nue.riec.tohoku.ac.jp

Abstract. In order to disprove confluence of term rewriting systems, we develop new criteria for ensuring non-joinability of terms based on interpretation and ordering. We present some instances of the criteria which are amenable for automation, and report on an implementation of a confluence disproving procedure based on these instances. The experiments reveal that our method is successfully applied to automatically disprove confluence of some term rewriting systems, on which state-of-the-art automated confluence provers fail. A key idea to make our method effective is the introduction of usable rules—this allows one to decompose the constraint on rewrite rules into smaller components that depend on starting terms.

Keywords: Confluence, Non-Joinability, Interpretation, Ordering, Term Rewriting Systems

1 Introduction

Confluence is a property that often turns out to be useful in vast topics of term rewriting; hence, it is conceived as one of the central properties of term rewriting (see e.g. [5, 30]). There is conceivably a long history for the development of techniques for proving confluence of term rewriting systems (TRSs, for short); see e.g. [5, 30, 31]. Recently, the area of proving confluence of TRSs *automatically* also caught an increasing attention. Indeed, recent works on confluence proving often address also automation of the methods [1, 2, 19, 21, 34]. Furthermore, several implementations of confluence provers are emerging [3, 17, 33], and the first competition on confluence provers (CoCo 2012) has been held last year.

For automated confluence provers, it is also important to *disprove* confluence so that they can give up unsuccessful attempts for confluence proving. In contrast to many dedicated techniques for proving confluence, however, not many techniques for disproving confluence are known. A typical approach to disprove confluence of (non-terminating) TRSs is first to construct a candidate of two terms that can be reduced from a common term, and then to show that they are not joinable, i.e. two terms do not have a common reduct. In this scenario, as well as the selection of the candidates, proving *non-joinability* of terms is

essential. So far, the only serious approach to prove the non-joinability of terms is to use approximation by tree automata [9, 12][1], implemented in CSI [33].

In this paper, we give new methods for proving that given two terms s, t are not joinable. The first method consists in giving an *interpretation*, e.g. a mapping from terms to natural numbers, that is preserved by the application of usable rules and such that the interpretation of s is different from that of t. The second method consists in giving an *ordering* $>$ such that $s > t$, and usable rules from s are non-decreasing and the usable rules from t are non-increasing. These methods are implemented using polynomial interpretations and recursive path orderings—interpretations and orderings that are widely used in the literature for termination proving. The experiments reveal that our methods can be applied to automatically disprove confluence of some term rewriting systems, on which state-of-the-art automated confluence provers fail.

The rest of the paper is organized as follows. In Section 2, we give some basic definitions and fix some notations to be used in the paper. In Section 3, we give an abstract non-joinability criterion using interpretation; the criterion is extended in Section 4 by a notion of usable rules for reachability. In Section 5, we give non-joinability criteria using ordering in terms of interpretation and rewrite relation; the latter is extended in Section 6 incorporating the notion of argument filtering from the area of termination proving. Related works are discussed in Section 7. In Section 8, we report on our implementation and experiments. Section 9 concludes.

2 Preliminaries

The *product* of two sets A and B is denoted by $A \times B$. We write A^2 for $A \times A$, and more generally, A^n (for the n-fold product of A) or $\Pi_{i \in I} A_i$, where I is an arbitrary index set. A *tuple* of elements is denoted by $\langle a_1, \ldots, a_n \rangle$ or $\langle a_i \rangle_{i \in I}$. The *disjoint union* of two sets A and B is denoted by $A \uplus B$, and that of all A_i $(i \in I)$ by $\biguplus_{i \in I} A_i$. The *composition* of relations R and S is denoted by $R \circ S$. The *reflexive transitive closure* of R is denoted by R^*; for a relation $\to$, its *reverse* is denoted by $\leftarrow$, the *reflexive closure* by $\xrightarrow{=}$ and the *reflexive transitive closure* by $\xrightarrow{*}$. We write $a_0 \to^n a_n$ to denote $a_0 \to a_1 \to \cdots \to a_n$ and $a \to^{\leq n} b$ if $a \to^m b$ for some $m \leq n$. We say an element a is a *normal form* (w.r.t. $\to$) if there exists no b such that $a \to b$. The relation $\to$ is *well-founded* if there exists no infinite chain $a_0 \to a_1 \to \cdots$. A relation is a *partial order* if it is reflexive, transitive and antisymmetric, and a *quasi-order* if it is reflexive and transitive.

We consider arity-fixed function symbols. The *arity* of function symbol f is denoted by $\mathrm{arity}(f)$. Function symbol f is a *constant* if $\mathrm{arity}(f) = 0$. The set of terms over a set $\mathcal{F}$ of function symbols and the set $\mathcal{V}$ of variables is denoted by $\mathrm{T}(\mathcal{F}, \mathcal{V})$. Terms which are not variables are referred to as *non-variable* terms and terms containing no variables are called *ground* terms. The set of variables in a term t is denoted by $\mathcal{V}(t)$. Let $\Box$ (called a *hole*) be a special constant that

[1] The technique is investigated in different literature. Applications of the technique are found also in the literature for termination proving [23, 25].

is not involved in $\mathcal{F}$. A *context* is a term in $\mathrm{T}(\mathcal{F} \cup \{\Box\}, \mathcal{V})$ that contains exactly one hole in the term. The term in $\mathrm{T}(\mathcal{F}, \mathcal{V})$ obtained by replacing the hole in a context C with a term t is denoted by $C[t]$. A context C is *empty* if $C = \Box$ and *non-empty* otherwise. A term s is said to be a *subterm* of t if $t = C[s]$ for some context C. We write $s \trianglelefteq t$ to denote that s is a subterm of t. The set of *positions* in a term t is denoted by $\mathrm{Pos}(t)$. A term t can be identified with a mapping $\mathrm{Pos}(t) \to \mathcal{F}$. The *root* position is denoted by ϵ; thus the root symbol of a term t is denoted by $t(\epsilon)$. The subterm at the position $p \in \mathrm{Pos}(t)$ is denoted by $t|_p$. We write $C[s]_p$ if $C(p) = \Box$.

A *substitution* is a mapping $\mathcal{V} \to \mathrm{T}(\mathcal{F}, \mathcal{V})$, which is homomorphically extended to the mapping $\mathrm{T}(\mathcal{F}, \mathcal{V}) \to \mathrm{T}(\mathcal{F}, \mathcal{V})$. For any substitution θ and term t, $\theta(t)$ is written as $t\theta$ if no confusion arises. Terms s and t are said to be *unifiable* if $s\theta = t\theta$ for some substitution θ. We write $\mathrm{Unif}(s, t)$ to denote that the terms s and t are unifiable.

A *rewrite rule* $l \to r$ is a pair of terms[2]. A *term rewriting system* (*TRS*, for short) is a set of rewrite rules. For a TRS $\mathcal{R}$, a *rewrite step* $s \to_{\mathcal{R}} t$ is given if $s = C[l\theta]$ and $t = C[r\theta]$ for some rewrite rule $l \to r \in \mathcal{R}$, context C and substitution θ. A relation R on terms is said to be *closed under contexts* if $s\ R\ t$ implies $C[s]\ R\ C[t]$ for any context C, is *closed under substitutions* if $s\ R\ t$ implies $s\theta\ R\ t\theta$ for any substitution θ. A *rewrite relation* is a relation on terms that is closed under contexts and substitutions. Given a TRS $\mathcal{R}$, it is readily checked that the relation $\to_{\mathcal{R}} = \{\langle s, t\rangle \in \mathrm{T}(\mathcal{F}, \mathcal{V})^2 \mid s \to_{\mathcal{R}} t\}$ is a rewrite relation, and is said to be the rewrite relation of $\mathcal{R}$. If no confusion arises, the subscript of $\to_{\mathcal{R}}$ will be omitted. A partial order (quasi-order) is a *rewrite* partial order (*rewrite* quasi-order, respectively) if it is a rewrite relation.

Given a term s, the sets of terms $\{t \in \mathrm{T}(\mathcal{F}, \mathcal{V}) \mid s \xrightarrow{*} t\}$ and $\{t \in \mathrm{T}(\mathcal{F}, \mathcal{V}) \mid t \xrightarrow{*} s\}$ are denoted by $[s](\xrightarrow{*})$ and $(\xrightarrow{*})[s]$, respectively. Terms s and t are said to be *joinable* if $[s](\xrightarrow{*}) \cap [t](\xrightarrow{*}) \neq \emptyset$, and *non-joinable* otherwise. We write $\mathrm{NJ}(s, t)$ to denote that the terms s and t are non-joinable. A TRS $\mathcal{R}$ is *confluent* if for any terms s, t, $(\xrightarrow{*})[s] \cap (\xrightarrow{*})[t] \neq \emptyset$ implies $[s](\xrightarrow{*}) \cap [t](\xrightarrow{*}) \neq \emptyset$. A TRS $\mathcal{R}$ is *terminating* if $\to_{\mathcal{R}}$ is well-founded. It is known that confluence of a TRS $\mathcal{R}$ is decidable if $\mathcal{R}$ is terminating.

In order to disprove that a (non-terminating) TRS $\mathcal{R}$ is confluent, we construct two terms s and t such that $(\xrightarrow{*})[s] \cap (\xrightarrow{*})[t] \neq \emptyset$ in some way, and then prove $\mathrm{NJ}(s, t)$. In order to check non-joinability of terms, it suffices to check ground instances of them using fresh constants [33]. From here on, we concentrate on the problem of proving non-joinability of ground terms.

3 Proving Non-Joinability by Interpretation

In this section, we give an abstract criterion to prove non-joinability of terms based on their interpretations in $\mathcal{F}$-algebras. Then we point out why it is not useful for our purpose—we will fix the problem in the next section.

[2] Here, we drop the usual restriction that $l \notin \mathcal{V}$ and $\mathcal{V}(r) \subseteq \mathcal{V}(l)$; we will deal with rewrite rules that do not satisfy these restrictions in Section 6.

We first recall some basic terminology on semantics of the equational logic and fix our notations (see e.g. [5]). An *$\mathcal{F}$-algebra* $\mathcal{A} = \langle A, \langle f^{\mathcal{A}} \rangle_{f \in \mathcal{F}} \rangle$ is a pair of a set A and a tuple of functions $f^{\mathcal{A}} : A^n \to A$ for each n-ary function symbol $f \in \mathcal{F}$; the set A is called the *carrier set* of the $\mathcal{F}$-algebra $\mathcal{A}$. A *valuation* on $\mathcal{A}$ is a mapping $\mathcal{V} \to A$. The *interpretation* $[\![t]\!]_{\mathcal{A},\sigma}$ (which is abbreviated as $[\![t]\!]_\sigma$ for brevity) of a term $t \in \mathrm{T}(\mathcal{F}, \mathcal{V})$ w.r.t. a valuation σ on $\mathcal{A}$ is recursively defined by $[\![x]\!]_\sigma = \sigma(x)$ and $[\![f(t_1, \ldots, t_n)]\!]_\sigma = f^{\mathcal{A}}([\![t_1]\!]_\sigma, \ldots, [\![t_n]\!]_\sigma)$. For any substitution θ and valuation σ, a valuation $[\![\theta]\!]_\sigma$ is given by $[\![\theta]\!]_\sigma(x) = [\![\theta(x)]\!]_\sigma$. We note that the interpretation of ground terms is independent of valuation, i.e., for any ground term t, $[\![t]\!]_\sigma = [\![t]\!]_\rho$ holds for any valuations σ, ρ. Hence, w.l.o.g. we drop valuations to denote the interpretation of ground terms.

The next property is well-known (e.g. [5]).

Lemma 1. *Fix an $\mathcal{F}$-algebra $\mathcal{A}$. For any term t, substitution θ and valuation σ, $[\![t\theta]\!]_\sigma = [\![t]\!]_{(\sigma \circ [\![\theta]\!]_\sigma)}$.* □

Using the notion of interpretation of terms in $\mathcal{F}$-algebras, the following criterion for non-joinability of (ground) terms naturally arises.

Theorem 2. *Let s, t be ground terms and $\mathcal{A} = \langle A, \langle f^{\mathcal{A}} \rangle_{f \in \mathcal{F}} \rangle$ an $\mathcal{F}$-algebra such that $A = \biguplus_{i \in I} A_i$. Suppose (i) for any valuation σ and $l \to r \in \mathcal{R}$, if $[\![l]\!]_\sigma \in A_i$ then $[\![r]\!]_\sigma \in A_i$, (ii) for any $f \in \mathcal{F}$, $a \in A$ and $i, j \in I$, if $a \in A_i$ implies $f^{\mathcal{A}}(\ldots, a, \ldots) \in A_j$, then $f^{\mathcal{A}}(\ldots, b, \ldots) \in A_j$ for any $b \in A_i$ and (iii) $[\![s]\!] \in A_i$ and $[\![t]\!] \in A_j$ with $i \neq j$. Then $\mathrm{NJ}(s, t)$.*

Proof. It is straightforward to show by induction on C that for any valuation σ and $l \to r \in \mathcal{R}$, $[\![C[l\theta]]\!]_\sigma \in A_i$ implies $[\![C[r\theta]]\!]_\sigma \in A_i$. Thus for any valuation σ, $u \to_{\mathcal{R}} v$ and $[\![u]\!]_\sigma \in A_i$ imply $[\![v]\!]_\sigma \in A_i$. Suppose that $\mathrm{NJ}(s, t)$ does not hold, i.e. $u \in [s](\xrightarrow{*}) \cap [t](\xrightarrow{*})$ for some u. Then we obtain $[\![u]\!]_\sigma \in A_i$ from $[\![s]\!]_\sigma \in A_i$ and $[\![u]\!]_\sigma \in A_j$ from $[\![t]\!]_\sigma \in A_j$. This contradicts our assumption that $A_i \cap A_j = \emptyset$. □

Although Theorem 2 may be applied to prove non-joinability of two terms in general, it is not effective in our setting—in the context of disproving confluence, one wants to show $\mathrm{NJ}(s, t)$ for s, t satisfying $(\xrightarrow{*})[s] \cap (\xrightarrow{*}])[t] \neq \emptyset$. For such s, t, if the conditions (i), (ii) of the theorem are satisfied, then $s \in A_i \Leftrightarrow t \in A_i$ holds, and hence the condition (iii) never holds.

The trick to apply the idea in our setting is to relax the condition (i) so that the constraint is applied not to all rules but only to rules usable in the reductions starting from s or t, which will be explored in the next section.

4 Usable Rules for Reachability

In the literature for proving termination of TRSs, a notion of usable rules is known to be very useful in the dependency pairs technique [4, 14, 18, 32]. There, the name 'usable' originally comes from the fact that usable rules are rules

possibly used to connect dependency pairs. Naturally, it is not very suitable for our purpose, because, in our setting, usable rules are to be the collection of rules that are possibly used in reductions from an initial term.

To suit our setting, we introduce a notion of *usable rules for reachability*. For this, the notion of TCAP [13] (introduced for defining usable rules for dependency pairs) is helpful. For terms t, $\mathrm{TCAP}(t)$ is defined recursively by: $\mathrm{TCAP}(x) = x'$, $\mathrm{TCAP}(f(t_1,\dots,t_n)) = x'$ if $\mathrm{Unif}(f(u_1,\dots,u_n), l)$ for some $l \to r \in \mathcal{R}$, and $\mathrm{TCAP}(f(t_1,\dots,t_n)) = f(u_1,\dots,u_n)$ otherwise, where $u_i = \mathrm{TCAP}(t_i)$ $(1 \le i \le \mathrm{arity}(f))$. Here, a new fresh variable is taken for x' every time it is used.

Definition 3 (usable rules for reachability). *For any TRS $\mathcal{R}$ and term s, let $\mathcal{U}_0(\mathcal{R}, s)$ be the smallest set $\mathcal{U}_0(\mathcal{R}, s) \subseteq \mathcal{R}$ satisfying the following conditions: (i) if $l \to r \in \mathcal{R}$ with $l \in \mathcal{V}$, then $l \to r \in \mathcal{U}_0(\mathcal{R}, s)$; (ii) for any $l \to r \in \mathcal{R}$ and non-variable subterm $f(u_1,\dots,u_n) \trianglelefteq s$, if $\mathrm{Unif}(f(\mathrm{TCAP}(u_1),\dots,\mathrm{TCAP}(u_n)), l)$ then $l \to r \in \mathcal{U}_0(\mathcal{R}, s)$; (iii) if $l' \to r' \in \mathcal{U}_0(\mathcal{R}, s)$ and $l \to r \in \mathcal{U}_0(\mathcal{R}, r')$, then $l \to r \in \mathcal{U}_0(\mathcal{R}, s)$. The set $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$ of* usable rules for reachability *w.r.t. a TRS $\mathcal{R}$ and a term s is defined by $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) = \mathcal{R}$ if there exists $l \to r \in \mathcal{U}_0(\mathcal{R}, s)$ such that $\mathcal{V}(r) \not\subseteq \mathcal{V}(l)$ and $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) = \mathcal{U}_0(\mathcal{R}, s)$ otherwise.*

Under the usual variable restrictions on rewrite rules, our notion of usable rules corresponds to the one obtained from the usable rules for innermost termination [13] by replacing ICAP with TCAP. (Standard) usable rules $\mathcal{U}(\mathcal{R}, s)$ in dependency pairs [4] is different from $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$—for example, for $\mathcal{R} = \{\mathsf{f}(\mathsf{a}) \to \mathsf{b}\}$, we have $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, \mathsf{f}(\mathsf{b})) = \emptyset \neq \mathcal{R} = \mathcal{U}(\mathcal{R}, \mathsf{f}(\mathsf{b}))$. Under the usual restriction of rewrite rules on variables, it is easy to check $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) \subseteq \mathcal{U}(\mathcal{R}, s)$.

The following is a key lemma to prove our theorem below.

Lemma 4. *Let $\mathcal{R}$ be a TRS, $l \to r \in \mathcal{R}$ and s, t terms. If $s \xrightarrow{*}_{\mathcal{R}} \circ \to_{\{l \to r\}} t$ then $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$.*

Proof. The proof consists of proving the following three claims step by step.

1. If $s \to_{\{l \to r\}} t$ then $l \to r \in \mathcal{U}_0(\mathcal{R}, s)$ (and hence, $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$).
2. If $s \to_{\{l \to r\}} t$ then $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, t) \subseteq \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$.
3. If $s \xrightarrow{*}_{\mathcal{R}} \circ \to_{\{l \to r\}} t$ then $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$.

1. Suppose $s \to_{\{l \to r\}} t$. If $l \in \mathcal{V}$ then $l \to r \in \mathcal{U}_0(\mathcal{R}, s)$ by definition. Otherwise, there exists non-variable subterm $u \trianglelefteq s$ such that $u = f(u_1,\dots,u_n) = l\theta$ for some substitution θ. Then, as $u_i = \mathrm{TCAP}(u_i)\theta_i$ for some θ_i, we have $f(\mathrm{TCAP}(u_1)\theta_1,\dots,\mathrm{TCAP}(u_n)\theta_n) = l\theta$. Because one can assume w.l.o.g. that $\mathcal{V}(\mathrm{TCAP}(u_i)) \cap \mathcal{V}(\mathrm{TCAP}(u_j)) = \emptyset$ for $i \neq j$ and $\mathcal{V}(\mathrm{TCAP}(u_i)) \cap \mathcal{V}(l) = \emptyset$, it follows that $f(\mathrm{TCAP}(u_1),\dots,\mathrm{TCAP}(u_n))$ and l are unifiable, and hence $l \to r \in \mathcal{U}_0(\mathcal{R}, s)$.
2. Since $l \to r \in \mathcal{U}_0(\mathcal{R}, s)$ by the claim 1, if $\mathcal{V}(r) \not\subseteq \mathcal{V}(l)$ then $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) = \mathcal{R} \supseteq \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$. So, suppose $\mathcal{V}(r) \subseteq \mathcal{V}(l)$. It suffices to show $\mathcal{U}_0(\mathcal{R}, t) \subseteq \mathcal{U}_0(\mathcal{R}, s)$. We show the claim by induction on the definition of $\mathcal{U}_0(\mathcal{R}, t)$. Suppose

$l' \to r' \in \mathcal{U}_0(\mathcal{R}, t)$. (i) Suppose $l' \in V$. Then by definition $l' \to r' \in \mathcal{U}_0(\mathcal{R}, s)$. (ii) Suppose there exists a non-variable subterm $u = f(u_1, \ldots, u_n) \trianglelefteq t$ such that $\mathrm{Unif}(f(\mathrm{TCAP}(u_1), \ldots, \mathrm{TCAP}(u_n)), l')$. Let $t = C[r\theta]$. Then either (a) $u \trianglelefteq \theta(x)$ for some $x \in \mathcal{V}(r)$, (b) $u \trianglelefteq C$, (c) $u = v\theta$ for some non-variable subterm $v = f(v_1, \ldots, v_n) \trianglelefteq r$ or (d) $u = C'[r\theta]$ for some non-empty context $C' \trianglelefteq C$. In the cases of (a) and (b), because $\mathcal{V}(r) \subseteq \mathcal{V}(l)$, we have $u \trianglelefteq s = C[l\theta]$ and hence $l' \to r' \in \mathcal{U}_0(\mathcal{R}, s)$ by definition. In the case of (c), by $\mathrm{Unif}(f(\mathrm{TCAP}(u_1), \ldots, \mathrm{TCAP}(u_n)), l')$ and $f(u_1, \ldots, u_n) = f(v_1, \ldots, v_n)\theta$, we have $\mathrm{Unif}(f(\mathrm{TCAP}(v_1), \ldots, \mathrm{TCAP}(v_n)), l')$. Hence because $v = f(v_1, \ldots, v_n) \trianglelefteq r$, it follows $l' \to r' \in \mathcal{U}_0(\mathcal{R}, r)$. Thus, since $l \to r \in \mathcal{U}_0(\mathcal{R}, s)$ by our claim 1, we have $l' \to r' \in \mathcal{U}_0(\mathcal{R}, s)$ by definition. In the case of (d), let $C' = f(u_1, \ldots, \tilde{C}, \ldots, u_n)$. Then because of $\mathrm{Unif}(f(\mathrm{TCAP}(u_1), \ldots, \mathrm{TCAP}(\tilde{C}[r\theta]), \ldots, \mathrm{TCAP}(u_n)), l')$, it follows that $\mathrm{Unif}(f(\mathrm{TCAP}(u_1), \ldots, \mathrm{TCAP}(\tilde{C}[x']), \ldots, \mathrm{TCAP}(u_n)), l')$, and thus we have $\mathrm{Unif}(f(\mathrm{TCAP}(u_1), \ldots, \mathrm{TCAP}(\tilde{C}[l\theta]), \ldots, \mathrm{TCAP}(u_n)), l')$. Thus $l' \to r' \in \mathcal{U}_0(\mathcal{R}, s)$ by definition. (iii) Suppose there exists $l'' \to r'' \in \mathcal{U}_0(\mathcal{R}, t)$ and $l' \to r' \in \mathcal{U}_0(\mathcal{R}, r'')$. Then, by induction hypothesis, $l'' \to r'' \in \mathcal{U}_0(\mathcal{R}, s)$ and hence $l' \to r' \in \mathcal{U}_0(\mathcal{R}, s)$ by definition.

3. We show the claim by induction on the length k of $s \xrightarrow{*}_{\mathcal{R}} t$. (B.S.) $k = 1$. Then $s \to_{\{l \to r\}} t$ and thus $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$ by the claim 1. (I.S.) $k > 1$. Suppose $s \to_{\mathcal{R}} s' \xrightarrow{*}_{\mathcal{R}} \circ \to_{\{l \to r\}} t$. Then by induction hypothesis $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s')$. Since $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, s') \subseteq \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$ by the claim 2, we obtain $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$. □

Now, Theorem 2 in the previous section is refined by replacing "$l \to r \in \mathcal{R}$" with "$l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) \cup \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$."

Theorem 5. *Let s, t be ground terms and $\mathcal{A} = \langle A, \langle f^{\mathcal{A}} \rangle_{f \in \mathcal{F}} \rangle$ an $\mathcal{F}$-algebra such that $A = \biguplus_{i \in I} A_i$. Suppose (i) for any valuation σ and $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) \cup \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$, if $[\![l]\!]_\sigma \in A_i$ then $[\![r]\!]_\sigma \in A_i$, (ii) for any $f \in \mathcal{F}$, $a \in A$ and $i, j \in I$, if $a \in A_i$ implies $f^{\mathcal{A}}(\ldots, a, \ldots) \in A_j$, then $f^{\mathcal{A}}(\ldots, b, \ldots) \in A_j$ for any $b \in A_i$ and (iii) $[\![s]\!] \in A_i$ and $[\![t]\!] \in A_j$ with $i \neq j$. Then* $\mathrm{NJ}(s, t)$.

Proof. Similar to the proof of Theorem 2, using Lemma 4. □

The criterion of Theorem 5, in general, is not amenable for automation, and one has to use more concrete instances of the theorem such as given below.

Corollary 6. *Let $\mathcal{A}$ be an $\mathcal{F}$-algebra and s, t be ground terms. Suppose (i) $[\![l]\!]_\sigma = [\![r]\!]_\sigma$ for any valuation σ and $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) \cup \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$ and (ii) $[\![s]\!] \neq [\![t]\!]$. Then* $\mathrm{NJ}(s, t)$. □

Proof. Take the carrier set A itself as the index set and the singleton set $\{a\}$ as A_a for each $a \in A$. □

Corollary 7. *Let s, t be ground terms and $\mathcal{A}$ an $\mathcal{F}$-algebra whose carrier set is a set of integers. Suppose there exists an integer $k \geq 2$ such that (i) for any valuation σ and $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) \cup \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$, $[\![l]\!]_\sigma \equiv [\![r]\!]_\sigma \pmod{k}$ and (ii) $[\![s]\!] \not\equiv [\![t]\!] \pmod{k}$. Then* $\mathrm{NJ}(s, t)$. □

Proof. Take $I = \{0, 1, \ldots, k-1\}$ and $A_i = \{n \in A \mid n \bmod k = i\}$ for each $i \in I$ and use Theorem 2. □

One way to automate non-joinability check using (instances of) Corollaries 6 and 7 is to use linear polynomial interpretations [5]: Take the set of integers as the carrier set, and for each n-ary function symbol $f \in \mathcal{F}$, let $f^{\mathcal{A}}(x_1, \ldots, x_n) = a_{f,0} + a_{f,1}x_1 + \cdots + a_{f,n}x_n$ where $a_{f,0}, \ldots, a_{f,n}$ are selected from a finite range of integers. Then the criteria of Corollaries 6 and 7 can be encoded as constraint solving problems assigning suitable values for each $a_{f,i}$ ($f \in \mathcal{F}, 0 \le i \le \mathrm{arity}(f)$). Indeed, this kind of constraint solving for polynomial interpretations is commonly used in termination tools, and its automation techniques are widely known (e.g. [7, 14]).

In following examples, non-confluence is shown using these corollaries.

Example 8. Let

$$\mathcal{R} = \left\{ \begin{array}{llll} (1) & \mathsf{a} \to \mathsf{h}(\mathsf{c}), & (2) & \mathsf{a} \to \mathsf{h}(\mathsf{f}(\mathsf{c})) \\ (3) & \mathsf{h}(x) \to \mathsf{h}(\mathsf{h}(x)), & (4) & \mathsf{f}(x) \to \mathsf{f}(\mathsf{g}(x)) \end{array} \right\}.$$

Let $s = \mathsf{h}(\mathsf{c})$ and $t = \mathsf{h}(\mathsf{f}(\mathsf{c}))$. As $\mathsf{a} \in (\xrightarrow{*})[s] \cap (\xrightarrow{*})[t]$, it suffices to show $\mathrm{NJ}(s, t)$ to disprove the confluence of $\mathcal{R}$. We have $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) \cup \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t) = \{(3), (4)\}$. Take an $\mathcal{F}$-algebra $\mathcal{A} = \langle \{0, 1\}, \langle f^{\mathcal{A}} \rangle_{f \in \mathcal{F}} \rangle$ as $\mathsf{a}^{\mathcal{A}} = \mathsf{c}^{\mathcal{A}} = 0$, $\mathsf{f}^{\mathcal{A}}(n) = 1 - n$, $\mathsf{h}^{\mathcal{A}}(n) = \mathsf{g}^{\mathcal{A}}(n) = n$. Then for any valuation σ, we have $[\![\mathsf{h}(x)]\!]_\sigma = \sigma(x) = [\![\mathsf{h}(\mathsf{h}(x))]\!]_\sigma$ and $[\![\mathsf{f}(x)]\!]_\sigma = 1 - \sigma(x) = [\![\mathsf{f}(\mathsf{g}(x))]\!]_\sigma$; thus, $[\![l]\!]_\sigma = [\![r]\!]_\sigma$ for each $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) \cup \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$. Thus $[\![s]\!] = [\![\mathsf{h}(\mathsf{c})]\!] = 0 \neq 1 = [\![t]\!] = [\![\mathsf{h}(\mathsf{f}(\mathsf{c}))]\!]$. Therefore, $\mathrm{NJ}(s, t)$ by Corollary 6.

Example 9. Let

$$\mathcal{R} = \left\{ \begin{array}{llll} (1) & \mathsf{a} \to \mathsf{f}(\mathsf{c}), & (2) & \mathsf{a} \to \mathsf{h}(\mathsf{c}) \\ (3) & \mathsf{f}(x) \to \mathsf{h}(\mathsf{g}(x)), & (4) & \mathsf{h}(x) \to \mathsf{f}(\mathsf{g}(x)) \end{array} \right\}.$$

Let $s = \mathsf{f}(\mathsf{c})$ and $t = \mathsf{h}(\mathsf{c})$. We have $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) \cup \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t) = \{(3), (4)\}$. Take an $\mathcal{F}$-algebra $\mathcal{A} = \langle \mathbb{N}, \langle f^{\mathcal{A}} \rangle_{f \in \mathcal{F}} \rangle$ as $\mathsf{a}^{\mathcal{A}} = \mathsf{c}^{\mathcal{A}} = 0$, $\mathsf{g}^{\mathcal{A}}(n) = n + 1$, $\mathsf{f}^{\mathcal{A}}(n) = n$, $\mathsf{h}^{\mathcal{A}}(n) = n + 1$. Then $[\![\mathsf{f}(x)]\!]_\sigma - [\![\mathsf{h}(\mathsf{g}(x))]\!]_\sigma = \sigma(x) - (\sigma(x) + 2) = -2$ and $[\![\mathsf{h}(x)]\!]_\sigma - [\![\mathsf{f}(\mathsf{g}(x))]\!]_\sigma = (\sigma(x) + 1) - (\sigma(x) + 1) = 0$. Take $k = 2$. Then $[\![\mathsf{f}(x)]\!]_\sigma \equiv [\![\mathsf{h}(\mathsf{g}(x))]\!]_\sigma \pmod k$ and $[\![\mathsf{h}(x)]\!]_\sigma \equiv [\![\mathsf{f}(\mathsf{g}(x))]\!]_\sigma \pmod k$ for any valuation σ. Furthermore, since we have $[\![s]\!] = [\![\mathsf{f}(\mathsf{c})]\!] = 0$ and $[\![t]\!] = [\![\mathsf{h}(\mathsf{c})]\!] = 1$, $[\![s]\!] \not\equiv [\![t]\!] \pmod k$. Hence, $\mathrm{NJ}(s, t)$ by Corollary 7.

5 Proving Non-Joinability by Ordering

In Corollary 7, we considered the case that the carrier set is a set of integers. In such a case, another obvious choice to obtain a partition of the carrier set is to divide it as $A = \{n \in A \mid n < k\} \uplus \{n \in A \mid k \le n\}$ for some k. We first formulate this idea in a more abstract setting, using the notion of ordered $\mathcal{F}$-algebra [35].

An *ordered $\mathcal{F}$-algebra* $\mathcal{A} = \langle A, \le, \langle f^{\mathcal{A}} \rangle_{f \in \mathcal{F}} \rangle$ is a triple of a set A, a partial order $\le$ on it and a tuple of functions $f^{\mathcal{A}} : A^n \to A$ for each n-ary function

symbol $f \in \mathcal{F}$. We use $<$ to denote strict part of $\leq$, i.e. $< = \leq \setminus \geq$. An ordered $\mathcal{F}$-algebra $\mathcal{A} = \langle A, \leq, \langle f^{\mathcal{A}} \rangle_{f \in \mathcal{F}} \rangle$ is said to be *weakly monotone* if $a \leq b$ implies $f^{\mathcal{A}}(\ldots, a, \ldots) \leq f^{\mathcal{A}}(\ldots, b, \ldots)$ for any $a, b \in A$ and $f \in \mathcal{F}$. Interpretations of terms on ordered $\mathcal{F}$-algebras are defined in the same way as on $\mathcal{F}$-algebras.

Theorem 10. *Let $\mathcal{A}$ be a weakly monotone ordered $\mathcal{F}$-algebra and s, t be ground terms. Suppose (i) $[\![l]\!]_\sigma \leq [\![r]\!]_\sigma$ for any valuation σ and any $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$, (ii) $[\![l]\!]_\sigma \geq [\![r]\!]_\sigma$ for any valuation σ and any $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$ and (iii) $[\![s]\!] > [\![t]\!]$. Then* $\mathrm{NJ}(s, t)$.

Proof. By weak monotonicity, for any valuation σ, $u \to_{\mathcal{U}_{\mathsf{r}}(\mathcal{R},s)} v$ implies $[\![u]\!]_\sigma \leq [\![v]\!]_\sigma$ and $u \to_{\mathcal{U}_{\mathsf{r}}(\mathcal{R},t)} v$ implies $[\![u]\!]_\sigma \geq [\![v]\!]_\sigma$. Hence the claim follows. □

Remark that well-foundedness of the ordering is not necessary, in contrast to orderings used in termination proving.

We now consider the case that term algebras are taken as $\mathcal{F}$-algebras, and formulate the theorem in a more general way using the notion of rewrite relation. For this, the following notion is useful.

Definition 11 (discrimination pair). *A pair $\langle \gtrsim, \succ \rangle$ of two relations $\gtrsim$ and $\succ$ is said to be a* discrimination pair *if (i) $\gtrsim$ is a rewrite relation, (ii) $\succ$ is a irreflexive relation and (iii) $\gtrsim \circ \succ \subseteq \succ$ and $\succ \circ \gtrsim \subseteq \succ$.*

Remark that neither transitivity, well-foundedness nor closure under substitutions and contexts is needed for the relation $\succ$, unlike a similar notion used in the termination proving, called reduction pair [24]. Note also that in the condition (iii), both of $\gtrsim \circ \succ \subseteq \succ$ and $\succ \circ \gtrsim \subseteq \succ$ are requested—this is again contrasted with the reduction pair where either $\gtrsim \circ \succ \subseteq \succ$ or $\succ \circ \gtrsim \subseteq \succ$ suffices; both conditions will be required in the proof of the theorem given below.

Clearly, for any rewrite quasi-order $\gtrsim$, the pair $\langle \gtrsim, \gtrsim \setminus \lesssim \rangle$ forms a discrimination pair.

Theorem 12. *Let $\mathcal{R}$ be a TRS and s, t ground terms. Suppose there exists a discrimination pair $\langle \gtrsim, \succ \rangle$ such that $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) \subseteq \lesssim$, $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, t) \subseteq \gtrsim$ and $s \succ t$. Then* $\mathrm{NJ}(s, t)$.

Proof. Since $\gtrsim$ is a rewrite relation, it follows that $u \to_{\{l \to r\}} v$ implies $u \lesssim v$ for any $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$, and $u \to_{\{l \to r\}} v$ implies $u \gtrsim v$ for any $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$. Suppose $u \in [s](\overset{*}{\to}) \cap [t](\overset{*}{\to})$. Let $s = s_0 \to s_1 \to \cdots \to s_n = u$. Then, by Lemma 4, $s = s_0 \to_{l_{i_1} \to r_{i_1}} s_1 \to_{l_{i_2} \to r_{i_2}} \cdots \to_{l_{i_n} \to r_{i_n}} s_n = u$ with $l_{i_j} \to r_{i_j} \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$ for all $j = 1, \ldots, n$. Thus $s \lesssim \cdots \lesssim u$. Since $t \prec s \lesssim \cdots \lesssim u$, we obtain $t \prec u$ by the property $\gtrsim \circ \succ \subseteq \succ$ of the discrimination pair. Similarly, from $t \to \cdots \to u$, we obtain $t \gtrsim \cdots \gtrsim u$. By $u \succ t \gtrsim \cdots \gtrsim u$, we obtain $u \succ u$ by the property $\succ \circ \gtrsim \subseteq \succ$ of the discrimination pair. This contradicts our assumption that $\succ$ is irreflexive. □

In terms of interpretations, Theorem 12 amounts to taking term algebras as $\mathcal{F}$-algebras, while Theorem 10 allows to take any $\mathcal{F}$-algebra. On the other hand, in terms of discrimination pairs, Theorem 10 amounts to taking a discrimination pair of the form $\langle \gtrsim, \gtrsim \setminus \lesssim \rangle$. Hence Theorem 10 is not subsumed by Theorem 12 and vice versa.

6 Argument Filtering for Non-Joinability

The criterion of Theorem 12 has a typical style used in criteria for termination proving. Therefore, similarly to the termination proving case, a discrimination pair can be obtained using various path orders combined with argument filtering. For dependency pairs, usable rules can be considered after performing argument filtering [14] and this sometimes decreases usable rules that need to be considered. In this section, we show that such an extension is possible also for non-joinability proving.

An *argument filtering* [4] is a mapping $\pi : \mathcal{F} \to (\mathrm{List}(\mathbb{N}^+) \cup \mathbb{N}^+)$ such that $\pi(f) \in \{[i_1, \ldots, i_k] \mid 1 \leq i_1 < \cdots < i_k \leq \mathrm{arity}(f)\} \cup \{i \mid 1 \leq i \leq \mathrm{arity}(f)\}$. Here, $\mathbb{N}^+$ denotes the set of positive integers and $\mathrm{List}(\mathbb{N}^+)$ the set of lists of positive integers. The application of the argument filtering π to terms is recursively defined as $x^\pi = x$ for $x \in \mathcal{V}$, $f(t_1, \ldots, t_n)^\pi = f(t_{i_1}^\pi, \ldots, t_{i_k}^\pi)$ if $\pi(f) = [i_1, \ldots, i_k]$, $f(t_1, \ldots, t_n)^\pi = t_i^\pi$ if $\pi(f) = i$. Hence $t^\pi \in \mathrm{T}(\mathcal{F}^\pi, \mathcal{V})$ where $\mathcal{F}^\pi = \{f \in \mathcal{F} \mid \pi(f) \in \mathrm{List}(\mathbb{N}^+)\}$ with $\mathrm{arity}(f) = |\pi(f)|$. For a TRS $\mathcal{R}$, we put $\mathcal{R}^\pi = \{l^\pi \to r^\pi \mid l \to r \in \mathcal{R}\}$. For substitution θ, we put $\theta^\pi(x) = \theta(x)^\pi$.

The following properties of the argument filtering are well-known (e.g. [4]).

Lemma 13. *(1)* $(s\theta)^\pi = s^\pi\theta^\pi$. *(2)* $s \to_{\{l\to r\}} t$ *implies* $s^\pi \mathrel{\overline{=}}\!\!\to_{\{l^\pi \to r^\pi\}} t^\pi$.

Theorem 14. *Let* $\mathcal{R}$ *be a TRS and* s, t *ground terms. Suppose there exist a discrimination pair* $\langle \gtrsim, \succ \rangle$ *and an argument filtering* π *such that* $\mathcal{U}_{\mathsf{r}}(\mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)^\pi, s^\pi) \subseteq \lesssim$, $\mathcal{U}_{\mathsf{r}}(\mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)^\pi, t^\pi) \subseteq \gtrsim$ *and* $s^\pi \succ t^\pi$. *Then* $\mathrm{NJ}(s, t)$.

Proof. Suppose $u \in [s](\overset{*}{\to}) \cap [t](\overset{*}{\to})$. Let $s = s_0 \to_{\{l_1\to r_1\}} \cdots \to_{\{l_n\to r_n\}} s_n = u$. Then by Lemma 4, $l_i \to r_i \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$ for all $i = 1, \ldots, n$. Then $s^\pi = s_0^\pi \mathrel{\overline{=}}\!\!\to_{\{l_1^\pi \to r_1^\pi\}} s_1^\pi \mathrel{\overline{=}}\!\!\to_{\{l_2^\pi \to r_2^\pi\}} \cdots \mathrel{\overline{=}}\!\!\to_{\{l_n^\pi \to r_n^\pi\}} s_n^\pi = u^\pi$ by Lemma 13 where $l_i^\pi \to r_i^\pi \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)^\pi$ for all $i = 1, \ldots, n$. Then by Lemma 4, $l_{i_j}^\pi \to r_{i_j}^\pi \in \mathcal{U}_{\mathsf{r}}(\mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)^\pi, s^\pi)$. Hence $t^\pi \prec s^\pi \lesssim s_1^\pi \lesssim \cdots \lesssim u^\pi$. Thus, by the definition of the discrimination pair, $t^\pi \prec u^\pi$. Similarly, we have $t^\pi \gtrsim \cdots \gtrsim u^\pi$. Hence $u^\pi \succ t^\pi \gtrsim \cdots \gtrsim u^\pi$, and $u^\pi \succ u^\pi$. This contradicts $\succ$ is irreflexive. □

If one takes an argument filtering π such that $\pi(f) = [1, 2, \ldots, \mathrm{arity}(f)]$ for all $f \in \mathcal{F}$, then we have $\pi(t) = t$ for any term t. Thus, Theorem 14 subsumes Theorem 12.

Some readers may wonder whether Theorem 14 can be obtained from Theorem 12 by taking the discrimination pair $\langle \gtrsim', \succ' \rangle$ defined by $s \gtrsim' t$ iff $s^\pi \gtrsim t^\pi$ and $s \succ' t$ iff $s^\pi \succ t^\pi$ for some discrimination pair $\langle \gtrsim, \succ \rangle$. Indeed, it can be shown that the discrimination pair $\langle \gtrsim', \succ' \rangle$ given like this is again a discrimination pair. However, the direct application of Theorem 12 only yields $\mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$ in the place of $\mathcal{U}_{\mathsf{r}}(\mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)^\pi, t^\pi)$ in Theorem 14. Since the inclusion $\mathcal{U}_{\mathsf{r}}(\mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)^\pi, t^\pi) \subseteq \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$ may be proper, Theorem 14 is not subsumed by Theorem 12.

Example 15. Let

$$\mathcal{R} = \left\{ \begin{array}{llll} (1) & \mathsf{c} \to \mathsf{f}(\mathsf{c}, \mathsf{d}), & (2) & \mathsf{c} \to \mathsf{h}(\mathsf{c}, \mathsf{d}) \\ (3) & \mathsf{f}(x, y) \to \mathsf{h}(\mathsf{g}(y), x), & (4) & \mathsf{h}(x, y) \to \mathsf{f}(\mathsf{g}(y), x) \end{array} \right\}.$$

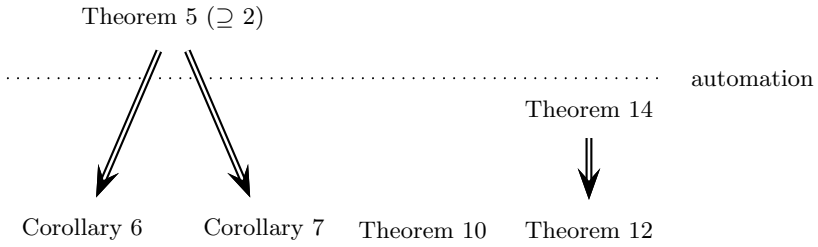

Fig. 1. Relations of theorems and corollaries

Let $s = \mathsf{h}(\mathsf{f}(\mathsf{c},\mathsf{d}),\mathsf{d})$ and $t = \mathsf{f}(\mathsf{c},\mathsf{d})$. First consider to apply Theorem 12. Then we need to solve the following constraint:

$$\left\{ \begin{array}{llll} \mathsf{h}(\mathsf{f}(\mathsf{c},\mathsf{d}),\mathsf{d}) \succ \mathsf{f}(\mathsf{c},\mathsf{d}), & \mathsf{c} & \simeq \mathsf{f}(\mathsf{c},\mathsf{d}), & \mathsf{c} \simeq \mathsf{h}(\mathsf{c},\mathsf{d}) \\ \mathsf{f}(x,y) & \simeq \mathsf{h}(\mathsf{g}(y),x), & \mathsf{h}(x,y) \simeq \mathsf{f}(\mathsf{g}(y),x) & \end{array} \right\}.$$

This constraint can not be satisfied using a discrimination pair $\langle \gtrsim_{rpo}, \gtrsim_{rpo} \setminus \lesssim_{rpo} \rangle$ based on recursive path orders. Next, we consider applying Theorem 14. For this, take an argument filtering π as $\pi(\mathsf{g}) = 1$, $\pi(\mathsf{f}) = [2]$ and $\pi(\mathsf{h}) = [1]$. Then we have $\mathcal{U}_{\mathsf{r}}(\mathcal{U}_{\mathsf{r}}(\mathcal{R},s)^{\pi},s^{\pi}) = \{(3)^{\pi},(4)^{\pi}\}$ and $\mathcal{U}_{\mathsf{r}}(\mathcal{U}_{\mathsf{r}}(\mathcal{R},t)^{\pi},t^{\pi}) = \{(3)^{\pi},(4)^{\pi}\}$. Then we need to solve the following constraint:

$$\left\{ \mathsf{h}(\mathsf{f}(\mathsf{d})) \succ \mathsf{f}(\mathsf{d}), \quad \mathsf{f}(y) \simeq \mathsf{h}(y), \quad \mathsf{h}(x) \simeq \mathsf{f}(x) \right\}.$$

Then the constraint is satisfied by a discrimination pair $\langle \gtrsim_{rpo}, \gtrsim_{rpo} \setminus \lesssim_{rpo} \rangle$, where $\gtrsim_{rpo}$ is the recursive path order based on the precedence $\mathsf{f} \simeq \mathsf{h}$. Thus $\mathrm{NJ}(s,t)$ by Theorem 14.

In Figure 1, we summarize relations of theorems and corollaries presented in the paper. The dotted line at the middle of the figure indicates that criteria below this line are suitable for automation.

7 Related Works

Non-Joinability Check in Confluence Provers We now review methods for proving non-joinability employed in the state-of-the-art confluence provers ACP [3], CSI [33] and Saigawa [17] that participated in the 1st Confluence Competition (CoCo 2012). Obviously, if the termination proof of the input TRS succeeds, the well-known criterion that confluence coincides with joinability of all critical pairs (Knuth-Bendix criterion [22]) can be used to prove non-confluence. Below we describe other approaches employed by these provers for proving non-confluence and non-joinability.

ACP basically uses the following three conditions to show non-joinability.

ACP(1) Fix some $n > 0$. Check $[s](\xrightarrow{*}) = [s](\rightarrow^{\leq n})$ and $[t](\xrightarrow{*}) = [t](\rightarrow^{\leq n})$. If it is the case and $[s](\rightarrow^{\leq n}) \cap [t](\rightarrow^{\leq n}) = \emptyset$ then conclude $\mathrm{NJ}(s,t)$.

ACP(2) Check $s(\epsilon) \neq t(\epsilon)$. If it is the case, check (by an approximation) that $\forall s' \in [s](\xrightarrow{*})$. $s(\epsilon) = s'(\epsilon)$ and $\forall t' \in [t](\xrightarrow{*})$. $t(\epsilon) = t'(\epsilon)$ hold. If they hold, conclude $\mathrm{NJ}(s,t)$.

ACP(3) If $s(\epsilon) = t(\epsilon) \notin \{l(\epsilon) \mid l \rightarrow r \in \mathcal{R}\}$, then check $\mathrm{NJ}(s|_i, t|_i)$ holds for some i. If it is the case, conclude $\mathrm{NJ}(s,t)$. This is used in conjunction with ACP(1) and ACP(2).

CSI uses the following two conditions to show non-joinability [33].

CSI(1) If $\mathrm{TCAP}(s)$ and $\mathrm{TCAP}(t)$ are not unifiable, then conclude $\mathrm{NJ}(s,t)$.

CSI(2) Use the approximation technique based on tree automata: Try to construct tree automata $\mathcal{A}_s$ and $\mathcal{A}_t$ such that $[s](\xrightarrow{*}) \subseteq \mathcal{L}(\mathcal{A}_s)$ and $[t](\xrightarrow{*}) \subseteq \mathcal{L}(\mathcal{A}_t)$ (using the method in [23]). If they succeed, then check $\mathcal{L}(\mathcal{A}_s) \cap \mathcal{L}(\mathcal{A}_t) = \emptyset$. If it is the case, then conclude $\mathrm{NJ}(s,t)$.

Saigawa uses CSI(1) above and the following extension of the Knuth-Bendix criterion [21] to disprove confluence.

Saigawa(1) Suppose $\mathcal{S}$ is confluent, $\mathcal{R}$ is terminating relative to $\mathcal{S}$, and $\mathcal{R}$ is not strongly overlapping on $\mathcal{R}$ and vice versa. Then $\mathcal{R} \cup \mathcal{S}$ is confluent iff all $\mathcal{S}$-critical pairs of $\mathcal{R}$ is joinable by $\mathcal{R} \cup \mathcal{S}$-rewrite steps.

Apparently the approach presented in the paper is very different from those employed already in these confluence provers, and the criteria given in the paper are not subsumed by any of the techniques employed in these confluence provers[3].

Decidable Classes Besides Knuth-Bendix criterion saying that confluence is decidable for terminating TRSs [22], decision procedures for deciding confluence of the TRSs in some classes of TRSs have been investigated.

One of the most basic such classes is the class of ground TRSs [8, 26]. For ground TRSs a polynomial time algorithm for deciding confluence is known [6, 10]. Such a decision procedure is implemented in CSI [33].

Other well-known such classes include the class of shallow right-linear TRSs [15], the class of right-ground TRSs [16, 20] and the class of monadic right-linear TRSs [28]. (The former two classes include the class of ground TRSs.) To the best of our knowledge, however, no implementations of the decision procedures other than the one mentioned above have been reported.

The criteria presented in this paper are free of syntactic restrictions on rewrite rules characterizing such classes. In particular, Examples 8, 9 and 15 are neither shallow, right-ground nor monadic; hence, they do not belong to any classes mentioned above for which confluence is decidable.

[3] Meanwhile, new versions of tools have been released, and CoCo 2013 have been held. The methods described in the present paper have been incorporated to the latest version of ACP.

8 Implementations and Experiments

Implementations The following instances of presented criteria have been implemented. The implementation is built on ACP. The language used in the implementation is the functional programming language SML/NJ [29].

Cor. 7 ($k = 2, 3$) Corollary 7 applied for the polynomial interpretation with linear polynomials, i.e. $f^{\mathcal{A}}$ has the form $a_{0,f} + a_{f,1}x_1 + \cdots + a_{n,f}x_n$ for each function symbol f of arity n. In case $k = 2$, we check whether $[\![l]\!]_\sigma - [\![r]\!]_\sigma$ is even for all rewrite rules $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s) \cup \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$ and whether $[\![s]\!] - [\![t]\!]$ is odd. We encode these constraints in boolean formulas and check the constraints by an external SAT solver (or SMT solver). We deal with integer variables of the range between 0 and 15 that are encoded by four bits. Thus the constraint that an integer variable $x = (b_3\ b_2\ b_1\ b_0)_{10}$ is even is encoded by $\bar{b}_0$ (i.e. b_0 equals false). The condition that a monomial ax ($a \in \mathbb{Z}$) is even is encoded by true if a is even and by "x is even" otherwise. The condition that a polynomial $a_0 + a_1x_1 + \cdots + a_nx_n$ is even is encoded recursively by the disjunct of both of the monomial a_0 and the polynomial $a_1x_1 + \cdots + a_nx_n$ are even and both are odd, recursively. Finally, the condition that meta polynomial $\Phi = \varphi_o + \varphi_1X_1 + \cdots + \varphi_nX_n$ where φ_i are polynomials, is even is encoded by all polynomials $\varphi_o, \ldots, \varphi_n$ are even and Φ is odd is encoded by the constant part φ_0 is odd and polynomials $\varphi_1, \ldots, \varphi_n$ are even. The case $k = 3$ is more complicated but encoding is again straightforward. For example, $(b_3\ b_2\ b_1\ b_0)_{10} \equiv (c_3\ c_2\ c_1\ c_0)_{10} \pmod 3$ can be encoded by $(b_3 \wedge b_2 \wedge b_1 \wedge b_0) \vee ((b_3 \otimes b_1) \wedge (b_2 \otimes b_0)) \vee (\bar{b_3} \wedge \bar{b_2} \wedge \bar{b_1} \wedge \bar{b_0})$.

Th. 10 (poly) Theorem 10 applied for polynomial interpretation with linear polynomials. Similar to the case Cor. 7 ($k = 2, 3$), we encode the constraints in boolean formulas and check the constraints by an external SAT solver. We also deal with integer variables of the range between 0 and 15 which are encoded by four bits. To ensure the weak monotonicity, we restrict all coefficients of $f^{\mathcal{A}} = a_{f,0} + a_{f,1}x_1 + \cdots + a_{f,n}x_n$ to be non-negative. Our implementation tries two possible applications of the Theorem to show $\mathrm{NJ}(s, t)$, namely that (1) $[\![s]\!] > [\![t]\!]$, $[\![l]\!]_\sigma \geq [\![r]\!]_\sigma$ for $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$ and $[\![l]\!]_\sigma \leq [\![r]\!]_\sigma$ for $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$, and (2) $[\![t]\!] > [\![s]\!]$, $[\![l]\!]_\sigma \geq [\![r]\!]_\sigma$ for $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$ and $[\![l]\!]_\sigma \leq [\![r]\!]_\sigma$ for $l \to r \in \mathcal{U}_{\mathsf{r}}(\mathcal{R}, t)$. Because of our bounds on integer variables, it may be the case that only one of (1) or (2) works and the other doesn't.

Th. 14 (rpo) Theorem 14 applied for recursive path order with argument filtering. Similar to the cases Cor. 7 ($k = 2, 3$) and Th. 10 (poly), we encode the constraints in boolean formulas and check the constraints by an external SAT solver. Let $\mathcal{S} = \mathcal{U}_{\mathsf{r}}(\mathcal{R}, s)$. We approximate the set of usable rules $\mathcal{U}_{\mathsf{r}}(\mathcal{S}^\pi, s^\pi)$ by $\hat{\mathcal{U}}_{\mathsf{r}}(\mathcal{S}^\pi, s^\pi)$ where $\hat{\mathcal{U}}_{\mathsf{r}}$ is given by $\hat{\mathcal{U}}_{\mathsf{r}}(\mathcal{R}', s') = \mathcal{U}(\mathcal{R}', s') \cup \{l \to r \in \mathcal{R}' \mid l \in \mathcal{V}\}$ and $\mathcal{U}$ returns the set of usable rules for dependency pairs [4]. The soundness of the approximation follows from $\mathcal{U}_{\mathsf{r}}(\mathcal{R}', s') \cap \succsim \subseteq \hat{\mathcal{U}}_{\mathsf{r}}(\mathcal{R}', s') \cap \succsim$ for well-founded rewrite quasi-order $\succsim$. Here, $\mathcal{S}$ is computed before the encoding and the constraint $\hat{\mathcal{U}}_{\mathsf{r}}(\mathcal{S}^\pi, s^\pi) \subseteq \succsim$ is encoded in a similar way as the encoding of termination criteria using dependency pairs [14]. Currently, we don't know

whether a direct encoding of $\mathcal{U}_{\mathsf{r}}(\mathcal{S}^\pi, s^\pi)$ is possible. Finally, as in the case for Th. 10 (poly), our implementation tries two possible applications of the Theorem (the $s^\pi \succ t^\pi$ version and the $t^\pi \succ s^\pi$ version).

Selecting candidates for the non-joinability test For all these implementations, candidates for the non-joinability test are generated from the input TRS $\mathcal{R}$ like this: (1) first compute the one-step unfolding $\mathcal{R}'$ of $\mathcal{R}$ [27] and then (2) compute critical pairs of $\mathcal{R} \cup \mathcal{R}'$, and finally, (3) all critical pairs are sorted w.r.t. term size and at most 100 critical pairs are considered as candidates.

Apparently, various ways to compute candidates for the non-joinability test are possible. Note that considering just reducts of critical pairs of $\mathcal{R}$ for candidates is not enough for proving non-confluence [11].

Here we explain very roughly which candidates for the non-joinability test are considered in the state-of-the-art confluence provers. According to [33], CSI considers candidates from the set $\mathcal{C} = \bigcup_{\langle s_1,t_1\rangle}\{\langle s,t\rangle \mid s_1 \to^{\leq m} s, t_1 \to^{\leq n} t\}$ where $\langle s_1, t_1\rangle$ ranges over those satisfying $s_1 = C[r_1\sigma]_p \leftarrow C[l_1\sigma]_p = C[l_2\sigma]_q \to C[r_2\sigma]_q = t_1$ with $l_1 \to r_1, l_2 \to r_2 \in \mathcal{R}$, $p \leqslant q$ and $p \in \mathrm{Pos}(C[l_2]_q)$. ACP uses similar candidates, with (probably very) different heuristics for choosing C, p, q, m, n and choices of the candidates from $\mathcal{C}$. Saigawa considers, for testing CSI(1), candidates from the set $\bigcup_{\langle -,v,-\rangle}\{\langle s,t\rangle \mid v \to^{\leq n} s, v \to^{\leq n} t, s \neq t\}$ where $\langle -, v, -\rangle$ ranges over critical peaks of $\mathcal{R} \cup \mathcal{R}^{-1}$.

Experiments Experiments have been performed on our implementation and the state-of-the-art confluence provers ACP (ver. 0.31), CSI (ver. 0.2) and Saigawa (ver. 1.4). Each test is performed on a PC with one 2.50GHz CPU and 4G memory; the timeout is set to 60 seconds. We have tested a collection of 23 new examples which includes Examples 8, 9, 15 developed in the course of experiments, and a collection of 35 examples from the 1st Confluence Competition (CoCo 2012) that were not proved to be confluent by any of participating provers.

A summary of the experiments is shown in Table 1. Each column shows success(✓) or failure(×) of confluence disproving on Examples 8, 9 and 15, the numbers of examples from the collections that are successfully proved to be non-confluent and of those that timeout (except CSI, for which one can not distinguish timeout and failure), and the total time in seconds. The column below all shows the result for the combination of the four instances. Note that ACP, CSI and Saigawa consume considerable time for proving confluence while our implementation concentrates on disproving confluence.

All provers ACP, CSI and Saigawa fail on Examples 8, 9 and 15. Both of Cor. 7 ($k = 2$) and Cor. 7 ($k = 3$) succeed on Examples 8 and 9. Th. 10 (poly) succeeds on Example 8. Th. 14 (rpo) succeed on Examples 8 and 15. Hence, incomparability of Cor. 7 and Th. 14 (rpo) is observed.

In the experiments on the collection of 23 new examples, the following are observed: Th. 14 (rpo) succeeds most. Cor. 7 ($k = 2$) and Cor. 7 ($k = 3$) succeed on the same examples. The examples handled by Th. 10 (poly) are also handled by Th. 14 (rpo) and also by Cor. 7. Examples handled by any of the provers ACP, CSI and Saigawa are also handled by all.

Table 1. Summary of experiments

	ACP	CSI	Saigawa	Cor. 7 ($k=2$)	Cor. 7 ($k=3$)	Th. 10 (poly)	Th. 14 (rpo)	all
Example 8	×	×	×	✓	✓	✓	✓	✓
Example 9	×	×	×	✓	✓	×	×	✓
Example 15	×	×	×	×	×	×	✓	✓
23 examples (success)	9	12	3	16	16	14	19	21
23 examples (timeout)	0	–	1	0	3	0	0	1
23 examples (time in sec.)	2	2107	228	25	293	206	26	84
35 examples (success)	18	21	17	17	16	17	17	16
35 examples (timeout)	1	–	6	5	8	3	1	9
35 examples (time in sec.)	71	485	482	318	562	446	106	761

In the experiments on the collection of 35 examples from CoCo 2012, the following are observed. All instances succeed on the same examples, except for Cor. 7 ($k=3$), in which one timeouts. The numbers of examples on which ACP, CSI and Saigawa succeed but all fails are 4, 5, 3, respectively. There is one example (Cops Problem 15) which is proved by our implementation but by none of the provers. Unfortunately, the success on this example is not due to our new technique—this difference arises by the way one computes the candidates for non-joinability test.

Example 16 (Cops Problem 15). Let

$$\mathcal{R} = \left\{ \begin{array}{ll} (1) & \mathsf{f}(x,\mathsf{f}(y,z)) \to \mathsf{f}(\mathsf{f}(x,y),\mathsf{f}(x,z)) \\ (2) & \mathsf{f}(\mathsf{f}(x,y),z) \to \mathsf{f}(\mathsf{f}(x,z),\mathsf{f}(y,z)) \\ (3) & \mathsf{f}(\mathsf{f}(x,y),\mathsf{f}(y,z)) \to y \end{array} \right\}.$$

Let $s = \mathsf{f}(\mathsf{a},\mathsf{a})$ and $t = \mathsf{a}$. Note $s,t \in [\mathsf{f}(\mathsf{f}(\mathsf{a},\mathsf{a}),\mathsf{f}(\mathsf{a},\mathsf{a}))](\overset{*}{\to})$. Then s,t are normal forms and hence it is easy to see that $\mathrm{NJ}(s,t)$.

Finally, the running time is observed like this: Th. 14 (rpo) $<$ Cor. 7 ($k=2$) $\ll$ Th. 10 (poly) $\ll$ Cor. 7 ($k=3$).

All details of the experiments are available on the webpage: `http://www.nue.riec.tohoku.ac.jp/tools/acp/experiments/frocos13/all.html`.

9 Conclusion

We have presented sufficient criteria of non-joinability of terms that can be used to disprove confluence of TRSs. Our criteria are based on interpretation and ordering, and are using new notions of usable rules and discrimination pairs. The combination of arguments filtering and our notion of usable rules have been also considered. We have given some concrete instances of our criteria which are amenable for automation—implementations of these instances have been

described and experiments have been reported. Experiments have shown that the presented methods can automatically disprove confluence of TRSs, on which state-of-the-art automated confluence provers fail.

Since our criteria are parametrized by $\mathcal{F}$-algebras or orderings, other concrete instances of our criteria can be possibly used. We note that all of our instances are highly non-optimal, i.e. they do not use the full strength of discrimination pairs; for example, they are all well-founded although this is not required for discrimination pairs. Future work would involve exploring other possibilities to obtain effective interpretations and orderings.

Acknowledgements

Thanks are due to anonymous referees for many valuable comments. This work was partially supported by a grant from JSPS No. 23500002.

References

1. Aoto, T.: Automated confluence proof by decreasing diagrams based on rule-labelling. In: Proc. of 21st RTA. LIPIcs, vol. 6, pp. 7–16. Schloss Dagstuhl (2010)
2. Aoto, T., Toyama, Y.: A reduction-preserving completion for proving confluence of non-terminating term rewriting systems. Logical Methods in Computer Science 1(31), 1–29 (2012)
3. Aoto, T., Yoshida, Y., Toyama, Y.: Proving confluence of term rewriting systems automatically. In: Proc. of 20th RTA. LNCS, vol. 5595, pp. 93–102. Springer-Verlag (2009)
4. Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theoretical Computer Science 236(1–2), 133–178 (2000)
5. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
6. Comon, H., Godoy, G., Nieuwenhuis, R., Tiwari, A.: The confluence of ground term rewrite systems is decidable in polynomial time. In: Proc. of 42nd LICS. pp. 263–297. IEEE Computer Society Press (2001)
7. Contejean, E., Marché, C., Tomás, A.P., Urbain, X.: Mechanically proving termination using polynomial interpretation. Journal of Automated Reasoning 34, 325–363 (2005)
8. Dauchet, M., Heuillard, T., Lescanne, P., Tison, S.: Decidability of the confluence of finite ground term rewrite systems and of other related term rewrite systems. Information and Computation 88, 187–201 (1990)
9. Durand, I., Middeldorp, A.: Decidable call by need computations in term rewriting. In: Proc. of 14th CADE. LNAI, vol. 1249, pp. 4–18. Springer-Verlag (1997)
10. Felgenhauer, B.: Deciding confluence of ground term rewrite systems in cubic time. In: Proc. of 23rd RTA. LIPIcs, vol. 15, pp. 165–175. Schloss Dagstuhl (2012)
11. Felgenhauer, B.: A proof order for decreasing diagrams. In: Proc. of 1st IWC. pp. 9–15 (2012)
12. Genet, T.: Decidable approximations of sets of descendants and sets of normal forms. In: Proc. of 9th RTA. LNCS, vol. 1379, pp. 151–165. Springer-Verlag (1998)

13. Giesl, J., Thiemann, R., Schneider-Kamp, P.: Proving and disproving termination of higher-order functions. In: Proc. of 5th FroCoS. vol. 3717, pp. 216–231. Springer-Verlag (2005)
14. Giesl, J., Thiemann, R., Schneider-Kamp, P.: Mechanizing and improving dependency pairs. Journal of Automated Reasoning 37(3), 155–203 (2006)
15. Godoy, G., Tiwari, A.: Confluence of shallow right-linear rewrite systems. In: Proc. of 14th CSL. LNCS, vol. 3634, pp. 541–556. Springer-Verlag (2005)
16. Godoy, G., Tiwari, A., Verma, R.: Characterizing confluence by rewrite closure and right ground term rewriting systems. Applicable Algebra in Engineering, Communication and Computing 15, 13–36 (2004)
17. Hirokawa, N., Klein, D.: Saigawa: A confluence tool. In: Proc. of 1st IWC. p. 49 (2012)
18. Hirokawa, N., Middeldorp, A.: Tyrolean termination tool: Techniques and features. Information and Computation 205(4), 474–511 (2007)
19. Hirokawa, N., Middeldorp, A.: Decreasing diagrams and relative termination. Journal of Automated Reasoning 47(4), 481–501 (2011)
20. Kaiser, L.: Confluence of right ground term rewriting system is decidable. In: Proc. of 8th FoSSaCS. LNCS, vol. 3441, pp. 470–489. Springer-Verlag (2005)
21. Klein, D., Hirokawa, N.: Confluence of non-left-linear TRSs via relative termination. In: Proc. of 18th LPAR. LNCS, vol. 7180, pp. 258–2012. Springer-Verlag (2012)
22. Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. In: Leech, J. (ed.) Computational Problems in Abstract Algebra, pp. 263–297. Pergamon Press (1970)
23. Korp, M., Middeldorp, A.: Match-bounds revisited. Information and Computation 207(11), 1259–1283 (2009)
24. Kusakari, K., Nakamura, M., Toyama, Y.: Argument filtering transformation. In: Proc. of 1st PPDP. vol. 1702, pp. 47–61. Springer-Verlag (1999)
25. Middeldorp, A.: Approximating dependency graphs using tree automata techniques. In: Proc. of 1st IJCAR. LNAI, vol. 2083, pp. 593–610. Springer-Verlag (2001)
26. Oyamaguchi, M.: The Church-Rosser property for ground term rewriting systems is decidable. Theoretical Computer Science 49, 43–79 (1987)
27. Payet, É.: Loop detection in term rewriting using eliminating unfoldings. Theoretical Computer Science 403, 307–327 (2008)
28. Salomaa, K.: Decidability of confluence and termination of monadic term rewriting systems. In: Proc. of 4th RTA. LNCS, vol. 488, pp. 275–286. Springer-Verlag (1991)
29. Standard ML of New Jersey, `http://www.sml.org/`
30. Terese: Term Rewriting Systems. Cambridge University Press (2003)
31. Toyama, Y.: Confluent term rewriting systems (invited talk). In: Proc. of 16th RTA. LNCS, vol. 3467, p. 1. Springer-Verlag (2005), slides are available from `http://www.nue.riec.tohoku.ac.jp/user/toyama/slides/toyama-RTA05.pdf`
32. Urbain, X.: Modular & incremental automated termination proofs. Journal of Automated Reasoning 32, 315–355 (2004)
33. Zankl, H., Felgenhauer, B., Middeldorp, A.: CSI – A confluence tool. In: Proc. of 23rd CADE. LNAI, vol. 6803, pp. 499–505. Springer-Verlag (2011)
34. Zankl, H., Felgenhauer, B., Middeldorp, A.: Labelings for decreasing diagrams. In: Proc. of 22nd RTA. LIPIcs, vol. 10, pp. 377–392. Schloss Dagstuhl (2011)
35. Zantema, H.: Termination of term rewriting by semantic labelling. Fundamenta Informaticae 24, 89–105 (1995)