Decision Problems in Ordered Rewriting

H. Comon — LSV, CNRS, ENS Cachan, 61 Ave. Pr. Wilson, F-94235 Cachan cedex, France, comon@lsv.ens-cachan.fr

P. Narendran[*] — Dep. of Comp. Science, SUNY at Albany, Albany, NY 12222, USA, dran@cs.albany.edu

R. Nieuwenhuis — Dep. LSI, Tech. Univ. of Catalonia, Jordi Girona 1, 08034 Barcelona, Spain, roberto@lsi.upc.es

M. Rusinowitch — LORIA & INRIA-Lorraine, 615 rue du jardin botanique, BP 101, 54602 Villers les Nancy, France, rusi@loria.fr

Abstract

A term rewrite system (TRS) terminates iff its rules are contained in a reduction ordering $>$. In order to deal with any *set of equations, including inherently non-terminating ones (like commutativity), TRS have been generalised to* ordered *TRS $(E, >)$, where equations of E are applied in whatever direction agrees with $>$. The confluence of terminating TRS is well-known to be decidable, but for ordered TRS the decidability of confluence has been open.*

Here we show that the confluence of ordered TRS is decidable if ordering constraints for $>$ can be solved in an adequate way, which holds in particular for the class of LPO orderings. For sets E of constrained equations*, confluence is shown to be undecidable. Finally,* ground reducibility *is proved undecidable for ordered TRS.*

1 Introduction

Term rewrite systems (TRS) have been applied to many problems in symbolic computation, automated theorem proving, program synthesis and verification, and logic programming among others. Two fundamental properties of TRS are *termination* and *confluence,* which together ensure the existence and uniqueness of normal forms and hence a decidable word problem. Let us take a closer look at both properties.

Termination of TRS is undecidable even for left-linear one-rule systems [7]. However, a TRS R terminates if there is some reduction ordering $>$ such that $l > r$ for each $l \to r \in R$, and there are practical general-purpose methods for defining such $>$. But standard TRS cannot deal with inherently non-orientable axioms (like the commutativity axiom). Therefore they have been generalized to *ordered* TRS $(E, >)$, where rewriting is done by applying equations of E in whatever direction agrees with $>$ [12]. Hence ordered TRS can handle *any* set of equations E, being terminating by definition. For solving word problems by ordered rewriting, there must be one unique minimal element (w.r.t. $>$) in each E-congruence class. Therefore a standard choice in practice for $>$ is some lexicographic path ordering (LPO), because it can be made total (on ground terms, which suffices as we will see).

Confluence is undecidable in general, but for terminating TRS a decision procedure is given in Knuth and Bendix' landmark paper [17]: a TRS is confluent iff all its critical pairs are joinable. For ordered TRS, deciding confluence is more difficult. It has been a long standing open question, listed for instance as open problem #64 in the RTA'93 list [10]. The main problem is that, due to the ordering restrictions, different instances of a critical pair may require different joinability proofs. Here we prove decidability if $>$ is an LPO: then it is possible to finitely analyze the joinability of *all* ground instances —possibly with new symbols— of each critical pair. Dealing with new symbols is essential for deciding arbitrary word problems, and it is also a fundamental characteristic of the notion of confluence, since the weaker property of *ground* confluence, i.e., confluence when rewriting only ground terms over the given signature, is undecidable even for terminating standard TRS [15].

Example: Let E consist of the (clearly non-orientable) single equation $(x + y) + z = z + (x + y)$. Then there is a critical pair cp which is $(z + (x + y)) + u = u + ((x + y) + z)$. To show confluence, we have to prove that every instance $cp\sigma$ is joinable by ordered rewriting with E. In this example, this can be done

[*]Partially supported by the NSF grants CCR-9712396 and INT-9401087.

by considering a number of well-chosen ordering relations between terms that cover all possible cases. On the one hand, one considers (i) $z\sigma > x\sigma + y\sigma$, (ii) $z\sigma = x\sigma + y\sigma$, and (iii) $z\sigma < x\sigma + y\sigma$, and, on the other, the further possible relations with $u\sigma$. In all cases, joinability follows. Note that it is not sufficient here to consider only relations between variables, as observed in [18].

In general the picture is not so simple as in the previous example. Joinability proofs may have more than one step, and the cases of ordering relations considered must be compatible with the ones of previous steps, and also the new equality relations may introduce additional subterms that can be rewritten. Furthermore, one has to show that the search process of attempted joinability proofs by such a case analysis is finite, while covering all possible cases. In the following we do this by building for each critical pair *cp* an arbitrary *confluence tree*, whose nodes are ordering-constrained equations, and whose root is *cp*. Children of a node are obtained by three possible steps: *constrained rewriting*, *decomposition*, and *instantiation*. These steps precisely generate the right ordering relations that have to be analyzed. The whole process of building the trees importantly relies on existing results for LPO constraint solving [5, 20]. This is also the case for our proof of finiteness of the trees. Our main result is finally that $(E, >)$ is confluent iff all leaves of the trees are tautologies.

We emphasize that our decision procedure is not only of theoretical interest. Since Knuth and Bendix' paper in the late 60's, a lot of work has been devoted to the *completion* of standard TRS and to *ordered* (or *unfailing*) completion of ordered TRS (e.g., [2, 12, 23, 18, 3]). Ordered completion provably yields a confluent system after a (possibly infinite) number of iterations. It roughly amounts to a systematic closure of the TRS under inclusion of (simplifications of) its non-joinable critical pairs. In order to enhance the efficiency of completion and to find finite *complete* TRS whenever they exist, powerful methods for proving the joinability (and hence redundancy) of a critical pair are crucial. This is precisely what our method is able to do (automatically), making it now also possible to decide whether the completion has finished. In fact, in [21] we reported on the use of similar trees in the *Saturate* system [11] as a successful method for proving the redundancy of critical pairs. Furthermore, in case of non-joinability, instead of adding the critical pair to E, one may choose to add the non-tautology leaves of the tree.

Comparing our result with the undecidability of confluence for arbitrary finite term rewriting systems, it seems quite surprising (in favour of ordered rewriting), since extending the notion of reduction we go from undecidability to decidability. There is an obvious clue for this result: ordered rewriting is always terminating. However, the picture is not so simple. For instance, we also show that for finite sets of *constrained equations*, confluence is undecidable (while ordered rewriting is still terminating). This relativizes the scope of our decidability result and also shows its significance.

Unfortunately, in completion with constraint *inheritance* [16, 22], constrained equations appear even if we start with only unconstrained ones. In such an equation $e \mid c$, the constraint c records the conditions under which e is derivable. This restricts the number of critical pairs that have to be considered, but arbitrary simplification by rewriting is not allowed any more. In this context, our problem is to decide joinability of the critical pairs by rewriting with a set of constrained equations. This is not easy, because simplification with an arbitrary constrained rule is undecidable [6]: given a term s and a rule $l \to r \mid c$, it is undecidable whether or not all instances of s can be reduced.

Going further, we investigate the use of ordered rewriting in a classical application of rewrite systems: the proof-by-consistency approach to proving inductive theorems [14, 13, 1]. Here again, the result is the opposite of what happens in the case of finite term rewriting systems: we show that ground reducibility is undecidable for ordered rewriting.

The paper is organized as follows. We mainly focus on our decidability result: the decidability of confluence. We first recall in Section 2 the basic notions of ordered rewriting, as well as some results on ordering constraints. In Section 3, we introduce our confluence trees and we show how to solve constraints over the set of normal forms and derive the decidability of confluence. In Section 4 we sketch the undecidability of confluence for a finite set of constrained equations. In Section 5, we show that ground reducibility is undecidable for ordered rewriting. Finally, in Section 6, we state some possible extensions and implications of our decidability result in other areas.

2 Ordered Rewriting and LPO Constraints

We adopt the terminology and the notations of [9]. $T(\mathcal{F}, \mathcal{X})$ is the set of first-order terms built on alphabets $\mathcal{F}$ of *function symbols* and $\mathcal{X}$ of *variable symbols.*

$T(\mathcal{F})$ is the set of terms which do not contain any variables. An equation is a multiset of two terms, written $s = t$, where s and t are in $T(\mathcal{F}, \mathcal{X})$. *Positions* in a term are strings of positive integers, corresponding to paths in the tree representation of the term. The set of positions of a term t is written $Pos(t)$. If p is a position of t, then $t|_p$ is the subterm of t at position p and $t[u]_p$ is the term obtained by replacing $t|_p$ with u at position p in t. The *topmost symbol* of a term $f(\ldots)$, denoted $top(f(\ldots))$ is f. The *size* of a term t, denoted $|t|$, is 1 if t is a variable and $1 + |t_1| + \ldots + |t_n|$ if t is a term $f(t_1, \ldots, t_n)$. Strict equality between two terms (identity) will be written $\equiv$ in order to distinguish it from the equality symbol in equations.

If $\rightarrow$ is a binary relation on a set S, then $\leftarrow$ is its inverse, $\leftrightarrow$ its symmetric closure, $\rightarrow^*$ its reflexive-transitive closure and $\rightarrow^+$ its transitive closure. We write $s \rightarrow^! t$ if $s \rightarrow^* t$ and there is no t' such that $t \rightarrow t'$. Then t is called a *normal form* of s. The relation $\rightarrow$ is *well-founded* or *terminating* if there exists no infinite sequence $s_1 \rightarrow s_2 \rightarrow \ldots$ and it is *confluent* if the relation $\leftarrow^* \circ \rightarrow^*$ is contained in the joinability relation $\rightarrow^* \circ \leftarrow^*$. It is *locally confluent* if the relation $\leftarrow \circ \rightarrow$ is contained in the joinability relation $\rightarrow^* \circ \leftarrow^*$. A relation $\rightarrow$ on terms is *monotonic* if $s \rightarrow t$ implies $u[s]_p \rightarrow u[t]_p$ for all terms u and positions p.

The *word problem* for a set of equations E is to check, given a (universally quantified) equation $s = t$, whether or not it is a logical consequence of E (denoted as usual $E \models s = t$). We denote by $\leftrightarrow_E$ the smallest monotonic relation on terms such that $s\sigma \leftrightarrow_E t\sigma$ for all substitutions σ and $s = t \in E$. By Birkhoff's theorem $E \models s = t$ iff $s \leftrightarrow_E^* t$.

2.1 Ordered rewriting

An *ordered* term rewrite system (TRS) is a pair $(E, \succ)$, where E is a set of equations, and $\succ$ is an ordering on terms. The *ordered rewriting relation* defined by $(E, \succ)$ is the smallest monotonic binary relation $\rightarrow_{E,\succ}$ on terms such that $s\sigma \rightarrow_{E,\succ} t\sigma$ whenever $s = t \in E$ and $s\sigma \succ t\sigma$.

Ordered rewriting terminates if $\succ$ is well-founded and monotonic. But our aim is to decide word problems by ordered rewriting, that is, we would like it to be that $E \models s = t$ iff the (unique) normal forms of s and t by $\rightarrow_{E,\succ}$ coincide. Then some further requirements on $\succ$ are needed. For example, if $\succ$ is the empty ordering, then $\rightarrow_{E,\succ}$ is empty as well (and hence confluent and terminating), but useless for our purpose. A necessary condition on $\succ$ is clearly that there should be a unique minimal element w.r.t. $\succ$ in each E-congruence class. But even this is insufficient: if E is $\{a = b, b = c\}$ and $\succ$ is the smallest relation containing $a \succ c$ and $b \succ c$, then $\rightarrow_{E,\succ}$ is confluent and terminating but a and c are different E-equivalent normal forms. Therefore, it is usually required that $\succ$ is total on ground terms, and $\rightarrow_{E,\succ}$ is defined only on ground terms. This is what we will assume in the sequel as well. Then clearly for all ground terms s and t, we have $s \leftrightarrow_E t$ iff $s \leftrightarrow_{E,\succ} t$, and, as in the standard case, confluence and termination imply that $s \leftrightarrow_E t$ iff $E \models s = t$ iff the (unique) normal forms of s and t coincide.

Fortunately, the restriction to ground terms is no limitation for solving word problems with (universally quantified) variables, since clearly $s \leftrightarrow_E t$ iff $s\sigma \leftrightarrow_E t\sigma$ where σ is a substitution replacing each variable by a distinct new constant. In fact, it even suffices to have one new unary symbol $\texttt{succ}$: then, if 0 is the smallest constant symbol (and hence the smallest term), the ground terms $\texttt{succ}(0), \texttt{succ}(\texttt{succ}(0)), \ldots$ can play the role of the new constants. This is how we will proceed in the following: we assume that the terms of the equations in E are in $T(\mathcal{F}, \mathcal{X})$ and we write $\mathcal{F}^e$ to denote $\mathcal{F} \cup \{\texttt{succ}\}$.

Lemma 2.1 *Let $(E, \succ)$ be an ordered rewrite system where $\succ$ is well-founded, monotonic and total on $T(\mathcal{F}^e)$ and where $\rightarrow_{E,\succ}$ is confluent on $T(\mathcal{F}^e)$. Then the word problem is decidable for E.*

This lemma shows the relevance of the following

Definition 2.2 *Given a set of equations E built on $T(\mathcal{F}, \mathcal{X})$ and a total, well-founded, monotonic ordering $\succ$ on $T(\mathcal{F}^e)$, the ordered rewrite system $(E, \succ)$ is* confluent *(resp.* ground confluent*) if $\rightarrow_{E,\succ}$ is confluent on $T(\mathcal{F}^e)$ (resp. on $T(\mathcal{F})$).*

As for classical term rewriting, there is a gap between confluence and ground confluence: confluence implies ground confluence, but the converse is false. An easy example if given by $\{f(x) \rightarrow x;\ f(x) \rightarrow 0\}$. This rewrite system is not confluent. However, if $\mathcal{F}$ only consists of $0, f$, then the system is ground confluent. This example also applies to ordered rewrite systems: introducing new symbols in $\mathcal{F}^e$ allows to form critical peaks that cannot be reduced. In the next section, these new symbols will also be used to build solutions of ordering constraints; they are crucial in this respect.

2.2 LPO and LPO constraints

Assuming that $>_{\mathcal{F}}$ is a well-founded ordering on $\mathcal{F}$ (called the *precedence*), the *lexicographic path ordering* (LPO) on $T(\mathcal{F}, \mathcal{X})$ is defined by: $s >_{\mathrm{lpo}} t$ iff at least one of the following three conditions hold:

- $s \equiv f(s_1,\ldots,s_n)$ and for some $i \in [1..n]$ either $s_i >_{\text{lpo}} t$ or $s_i \equiv t$
- $t \equiv f(t_1,\ldots,t_n)$, s is not a variable and $top(s) >_{\mathcal{F}} f$, and $s >_{\text{lpo}} t_i$ for all $i \in [1..n]$
- $s \equiv f(s_1,\ldots,s_n)$ and $t \equiv f(t_1,\ldots,t_n)$ and for some $i \in [1..n]$ it holds that $s_1 \equiv t_1 \ \wedge \ \ldots \ \wedge \ s_{i-1} \equiv t_{i-1} \ \wedge \ s_i >_{\text{lpo}} t_i \ \wedge \ s >_{\text{lpo}} t_{i+1} \ \wedge \ \ldots \ \wedge \ s >_{\text{lpo}} t_n$

LPO is a strict ordering (i.e., an irreflexive transitive relation) on terms that is well-founded, monotonic, stable under substitutions ($s >_{\text{lpo}} t$ implies $s\sigma >_{\text{lpo}} t\sigma$ for all σ) and it has the subterm property ($s >_{\text{lpo}} t$ if t is a proper subterm of s). Hence it also contains the (strict) *tree embedding* relation, the smallest transitive monotonic relation $>_{emb}$ such that $s >_{emb} t$ if t is a proper subterm of s. Moreover, it is total on $T(\mathcal{F})$ if $>_{\mathcal{F}}$ is total on $\mathcal{F}$ (see e.g. [8, 9] for more details).

LPO is a standard choice for ordered TRS, since it fulfills the necessary requirements and it is easily and efficiently implementable (note that other general-purpose orderings like RPO do not fulfill the totality requirement). In the following we will apply LPO as an ordering on $T(\mathcal{F}^e)$ with a linear precedence $>_{\mathcal{F}^e}$ where 0 is the smallest constant symbol in $\mathcal{F}$ (if there is no constant in $\mathcal{F}$ we can add one wlog.) and the unary symbol succ is the smallest function symbol.

Example 1 Let E be the set $\{x+y = y+x, \ x+(y+z) = (x+y)+z\}$. If $s \equiv f(0, \mathtt{succ}(f(\mathtt{succ}(0), 0)))$ and $t \equiv \mathtt{succ}(f(0, \mathtt{succ}(\mathtt{succ}(0))))$, then $s >_{\text{lpo}} t$ and hence $s + t \rightarrow_{E,>_{\text{lpo}}} t + s$. We also have $(\mathtt{succ}(0)+0)+0 \rightarrow_{E,>_{\text{lpo}}} (0+\mathtt{succ}(0))+0 \rightarrow_{E,>_{\text{lpo}}} 0+(\mathtt{succ}(0)+0) \rightarrow_{E,>_{\text{lpo}}} 0+(0+\mathtt{succ}(0))$, the latter term being irreducible.

Example 2 Let E be the set $\{f(x,y) = g(x,z)\}$ and let $f >_{\mathcal{F}^e} g >_{\mathcal{F}^e} a >_{\mathcal{F}^e} b$. Then $f(a,a) \rightarrow_{E,>_{\text{lpo}}} g(a,a)$, but $f(a,a)$ also rewrites into $g(a,b)$ or into $g(a,0)$ or into $g(a,\mathtt{succ}(0))$, etc., since there is a choice on how to instantiate the so-called *extra* variable z, as long as the step is reductive w.r.t. $>_{\text{lpo}}$.
Of course, if $\rightarrow_{E,>_{\text{lpo}}}$ is confluent all choices lead to the same normal form. For the equivalent set $E' \equiv E \cup \{f(x,y) = f(x,z), \ g(x,y) = g(x,z)\}$, the relation $\rightarrow_{E',>_{\text{lpo}}}$ is confluent and all terms of the form $f(s,t)$ or $g(s,t)$ have $g(s,0)$ as unique normal form. Clearly 0 is the most "efficient" choice for instantiating extra variables like z; in fact, to reach a normal form here, one is eventually forced to chose 0.

See [5, 20, 22] for the general definitions and the main results on *ordering constraints*. In order to keep this paper simple and self-contained, here we restrict ourselves to LPO-constraints with *extended signature* semantics. In the following, an *LPO constraint* is a Boolean combination (using the connectives $\wedge, \vee, \neg$) of atoms $s = t$ or $s > t$ where $s, t \in T(\mathcal{F}, \mathcal{X})$. An ordering constraint is interpreted on $T(\mathcal{F}^e)$; the Boolean connectives have their usual meaning and an assignment σ of the variables of $s > t$ (resp. $s = t$) *satisfies* $s > t$ (resp. $s = t$) iff $s\sigma >_{\text{lpo}} t\sigma$ (resp. $s\sigma \equiv t\sigma$). We write $\sigma \models c$ when the assignment σ satisfies the constraint c; then σ is called a *solution* for c. We sometimes write chains $s > t > u > \ldots$ as a shorthand for $s > t \wedge s > u \wedge \ldots \wedge t > u \wedge \ldots$, and also $s \geq t$ as a shorthand for $s > t \vee s = t$.

Example 3 Let σ be the assignment $\{x \mapsto \mathtt{succ}(0); \ y \mapsto \mathtt{succ}(\mathtt{succ}(0))\}$. Then $\sigma \models f(a,x) > y > x$. The constraint $f(x,y) > f(y,x) > y > x$ is unsatisfiable.

An interesting property of LPO is that for non-variable terms s and t, the relation $s >_{\text{lpo}} t$ follows from at least one conjunction of relations of the form $s' >_{\text{lpo}} t'$ or $s' \equiv t'$ where s' and t' are subterms of s and t respectively and $|s| + |t| > |s'| + |t'|$. This gives rise to the so-called *LPO decomposition* of a relation $s > t$: if $s \equiv f(s_1,\ldots,s_n)$ and $t \equiv g(t_1,\ldots,t_m)$, then

- for all $i \in 1\ldots n$, the constraints $s_i > t$ and $s_i = t$ are LPO decompositions of $s > t$
- if $f >_{\mathcal{F}^e} g$ then $s > t_1 \wedge \ldots \wedge s > t_m$ is an LPO decomposition of $s > t$
- if $g \equiv f$ (and hence $n = m$) then, for all $i \in 1\ldots n$, the constraints of the form: $s_1 = t_1 \ \wedge \ \ldots \ \wedge \ s_{i-1} = t_{i-1} \ \wedge \ s_i > t_i \ \wedge \ s > t_{i+1} \ \wedge \ \ldots \ \wedge \ s > t_n$ are LPO decompositions of $s > t$.

An LPO constraint c without disjunctions is called *LPO-closed* iff $0 > s$ is not in c and for each relation $s > t$ in c where s and t are non-variable terms, c contains an LPO-decomposition of $s > t$ (a relation $s = t$ is contained as well if $s \equiv t$).

Example 4 Suppose $f >_{\mathcal{F}^e} g$. The LPO decomposition of $f(x,y) > g(x,z)$ is $f(x,y) > x \ \wedge \ f(x,y) > z$. The four LPO decompositions of $g(x,y) > f(y,z)$ are $x > f(y,z)$, $x = f(y,z)$, $y > f(y,z)$, and $y = f(y,z)$. There are six LPO decompositions of $f(x,y) > f(y,z)$, namely $x > f(y,z)$, $y > f(y,z)$, $x = f(y,z)$, $y = f(y,z)$, $x > y \ \wedge \ f(x,y) > z$ and $x = y \ \wedge \ y > z$.

A *simple system* S is a particular constraint of the form $s_n \ \#_n \ s_{n-1} \ \#_{n-1} \ \ldots \ \#_1 \ s_0$, where $\#_i$ is either $=$ or $>$, every strict subterm of an s_i is some s_j and $i \neq j$ implies $s_i \not\equiv s_j$. Note that we write here e.g. $s > t > u$ as a shorthand for $s > t \wedge t > u$. Then $=_S$ is the least equivalence relation on $\{s_0, \ldots, s_n\}$ which contains all pairs (s_i, s_{i-1}) such that $\#_i$ is $=$. $>_S$ is the least transitive relation on $\{s_0, \ldots, s_n\}$ containing all pairs (s_i, s_{i-1}) such that $\#_i$ is $>$ and such that $s =_S t$ and $t >_S u$ and $u =_S v$ implies $s >_S v$. In short, $=_S$ and $>_S$ are respectively the equalities and ordering constraints that can be deduced from S.

The *equational part* of a simple system S, which we write *eqpart*(S), is $\{s = t \mid s =_S t\}$. The *inequational part* of S, denoted *ineqpart*(S), is $\{s > t \mid s\sigma >_S t\sigma\}$ if σ is the most general simultaneous unifier of *eqpart*(S) and *ineqpart*(S) $= \bot$ if *eqpart*(S) is not unifiable. S is called *purely inequational* if $S \equiv$ *ineqpart*(S).

The satisfiability of LPO constraints was shown decidable in [5] (and NP-complete in [20]). Key steps for these results are as follows:

1. Any constraint c is (effectively) equivalent to a finite disjunction of simple systems $S_1 \vee \ldots \vee S_n$, and hence c is satisfiable iff S_i is satisfiable for some $i \in 1 \ldots n$.
2. A simple system S is unsatisfiable if *eqpart*(S) is not unifiable.
3. A simple system S is equivalent to *ineqpart*(S) if *eqpart*(S) is unifiable.
4. A purely inequational simple system is satisfiable iff it is LPO-closed.

Proof: We only sketch the proof for the last statement, since it is less obvious and the construction will be used throughout the paper. First, note that if $s >_S t$ and S contains no LPO decomposition for $s > t$, then it must be unsatisfiable since S is closed under subterms and hence S must be in contradiction with *all* LPO decompositions of $s > t$. This happens for example with the constraint $f(x,y) > f(y,x) > y > x$.

For the reverse implication, if S is of the form $s_n > \ldots > s_0$, we can build a solution σ from right to left, i.e., by induction on the index i: if s_i is a variable, then let $s_i\sigma$ be $\mathtt{succ}(0)$ if $i = 0$ and $\mathtt{succ}(s_{i-1}\sigma)$ otherwise. To see that this σ is indeed a solution, we use the fact that S is LPO-closed: using an induction argument on $|s| + |t|$ for each relation $s >_S t$, it can be shown that the substitution σ satisfies S iff it satisfies all relations $s >_S t$ where s or t is a variable, which in turn holds since `succ` is a new smallest symbol (see [20] for details). □

In the following, the previous kind of solution for S will be called the *minimal* solution for S. Note that it is *alien*: for every variable x, the solution $x\sigma$ is headed by the new symbol `succ`.

3 Decidability of confluence of ordered rewriting

In the following, $(E, >_{\text{lpo}})$ will be an ordered TRS. According to the previous section, we assume that terms of E belong to $T(\mathcal{F}, \mathcal{X})$ and that substitutions, and the interpretation of constraints are in $T(\mathcal{F}^e)$. A *constrained equation* is a pair $(s = t, c)$, denoted $s = t \mid c$, where $s = t$ is an equation and c is a constraint. It denotes all its *instances*: the equations $s\sigma = t\sigma$ such that $\sigma \models c$. Hence it is a *tautology* if $s\sigma \equiv t\sigma$ for all such σ. A *critical pair* between two equations $u = v$ and $s = t$ of E is a constrained equation $u[t]_p = v \mid s > t \wedge u > v \wedge u|_p = s$, for some position p such that $u|_p$ is not a variable. Newmann's lemma states that, for a terminating relation, confluence is equivalent to local confluence, which for term rewriting (resp. ordered rewriting) reduces to the joinability of critical pairs. The following result is essentially due to J. Hsiang and M. Rusinowitch [12] (see also [2]):

Lemma 3.1 *$\rightarrow_{E,>_{\text{lpo}}}$ is confluent iff for every critical pair $s = t \mid c$ between equations in E all its instances $s\sigma = t\sigma$ are* joinable, *i.e., there is a term u such that $s\sigma \rightarrow^*_{E,>_{\text{lpo}}} u \leftarrow^*_{E,>_{\text{lpo}}} t\sigma$.*

A *confluence tree* for $(E, >_{\text{lpo}})$ and a critical pair $s = t \mid c$ is a tree T where the nodes of T are constrained equations, the root of T is $s = t \mid c$, and the children of a node $e \mid c$ in T are the constrained equations obtained by one of the following three kinds of steps:

1. By *constrained rewriting*, $e \mid c$ can be rewritten with $l = r \in E$ into $e[r\sigma]_p \mid c \wedge l\sigma > r\sigma$ and into the *complementary* equation $e \mid c \wedge r\sigma \geq l\sigma$ iff
   - $e|_p \equiv l\sigma$
   - $c \wedge l\sigma > r\sigma$ is satisfiable
   - $x\sigma \equiv 0$ for every variable x in r not occurring in l
2. By *decomposition*, $e \mid c$ can be rewritten into $\{e \mid S_1, \ldots, e \mid S_n\}$ if c is satisfiable and not a simple system and $\{S_1, \ldots, S_n\}$ is an equivalent set of simple systems for c.

3. By *instantiation*, $e \mid c$ can be rewritten into $e\sigma \mid \mathit{ineqpart}(c)$, if c is a satisfiable and not purely inequational simple system, and σ is the most general unifier of $\mathit{eqpart}(c)$.

Example 5 Consider again the set $E \equiv \{x + y = x + z,\ \ (x + y) + z = x + (y + z)\}$. Critical pairs are $(x + y) + z = y + (x + z) \mid y > x$ and $x + (y + z) = z + (x + y) \mid x + y > z$. Other pairs yield unsatisfiable constraints or renamings of the above pairs. Then a confluence tree rooted with the first one, as it was automatically generated by the *Saturate* system [21], is depicted in Figure 1. Note that in a constrained rewrite step with the associativity axiom (like the one applied to the root) the complementary equation always has an unsatisfiable constraint (and is hence not shown). The three framed nodes are leaves. Only the leftmost one is a tautology and hence E is not confluent. The last step for the two rightmost framed nodes is by decomposition and instantiation. The three other nodes without descendants become leaves after one step of decomposition and instantiation followed in some cases by one rewrite step with associativity.

Proof plan: Our decision procedure will be based on the construction of one (arbitrary) confluence tree for each critical pair. The main result will be that $\rightarrow_{E,>_{\mathrm{lpo}}}$ is confluent iff all these trees have only tautology leaves. For this purpose, we first show that the trees are finite (Lemma 3.2), and that it is easy to decide whether a leaf $s = t \mid c$ is a tautology (Lemma 3.3). Then, in Lemma 3.4 we show that every instance of the critical pair at the root can be rewritten into some leaf, and hence $\rightarrow_{E,>_{\mathrm{lpo}}}$ is confluent if all leaves are tautologies, since then all instances of the critical pairs are joinable. Finally, for the (harder) reverse implication (Lemma 3.5), from every non-tautology leaf $s = t \mid c$ we can reconstruct a substitution σ (not necessarily a solution of c) such that $s\sigma$ and $t\sigma$ are distinct and in normal form, which contradicts confluence, since $E \models s = t$ for every leaf $s = t \mid c$.

Lemma 3.2 *Every confluence tree is finite.*

Proof: The tree is finitely branching. Hence by König's lemma it suffices to show that every path is finite. Assume the contrary. Only a finite number of instantiations can be applied on a branch since they reduce the number of variables of the descendant nodes, and decomposition cannot be applied more than once to a node. Hence there must be an infinite branch with only constrained rewriting steps (each one followed by zero or one decompositions). This branch has no infinite subsequence of only complementary steps, since the number of possible applications of equations to a finite e is finite, and no complementary steps can be applied twice at the same position (then the non-complementary constraint becomes unsatisfiable). Hence there must be an infinite number of non-complementary steps. By Kruskal's theorem, since all terms in the tree are built over a finite set of symbols, on such an infinite branch there must be a node $e_i \mid c_i$ and a descendant $e_j \mid c_j$ (obtained, among other steps, by one or more non-complementary steps), such that $e_i \equiv e_j$ or $e_j >_{emb} e_i$. But by construction of the tree, for all σ such that $\sigma \models c_j$, we have $e_i\sigma >_{\mathrm{lpo}} e_j\sigma$, which contradicts $e_i \equiv e_j$ or $e_j >_{emb} e_i$. □

Lemma 3.3 *Let $s = t \mid c$ be a constrained equation where c is a purely inequational simple system. Then $s = t \mid c$ is a tautology iff either c is unsatisfiable or $s \equiv t$.*

Proof: Clearly if c is unsatisfiable or $s \equiv t$ then $s = t \mid c$ is a tautology. For the reverse direction, suppose c is satisfiable and $s \not\equiv t$. Then c has an alien solution σ (for every variable x, the solution $x\sigma$ is headed by the new symbol `succ`), and it is easy to see that such alien σ cannot unify two different terms in $T(\mathcal{F}, \mathcal{X})$. Hence $s\sigma \not\equiv t\sigma$ and $s = t \mid c$ is not a tautology. □

Lemma 3.4 *Let T be a confluence tree rooted by $s = t \mid c$. If all leaves of T are tautologies, then all instances of $s = t \mid c$ are joinable.*

Proof: Let $s\sigma = t\sigma$ be an instance of $s = t \mid c$ with $s\sigma$ and $t\sigma$ in $T(\mathcal{F}^e)$. We show by induction on the depth of the tree that $s\sigma = t\sigma$ is joinable. If $s = t \mid c$ is already a leaf, then it must be a tautology, and all instances of tautologies are trivially joinable. Otherwise the children of $s = t \mid c$ are obtained by one of the three different steps.

By constrained rewriting with some $l = r \in E$, the children are $s[r\theta]_p = t \mid c \wedge l\theta > r\theta$, and $s = t \mid c \wedge r\theta \geq l\theta$. Clearly $s\sigma = t\sigma$ is either an instance of one of the children or it rewrites into one of them. In each case joinability follows by induction hypothesis (each subtree has a smaller depth).

By decomposition, the children are $\{s = t \mid S_1, \ldots, s = t \mid S_n\}$. Since c is equivalent to a disjunction of the S_i, $s\sigma = t\sigma$ is an instance of one of the children and joinability follows by the induction hypothesis.

By instantiation, the only descendant is $s\theta = t\theta \mid \mathit{ineqpart}(c)$. Since $\sigma \models c$ again $s\sigma = t\sigma$ is still

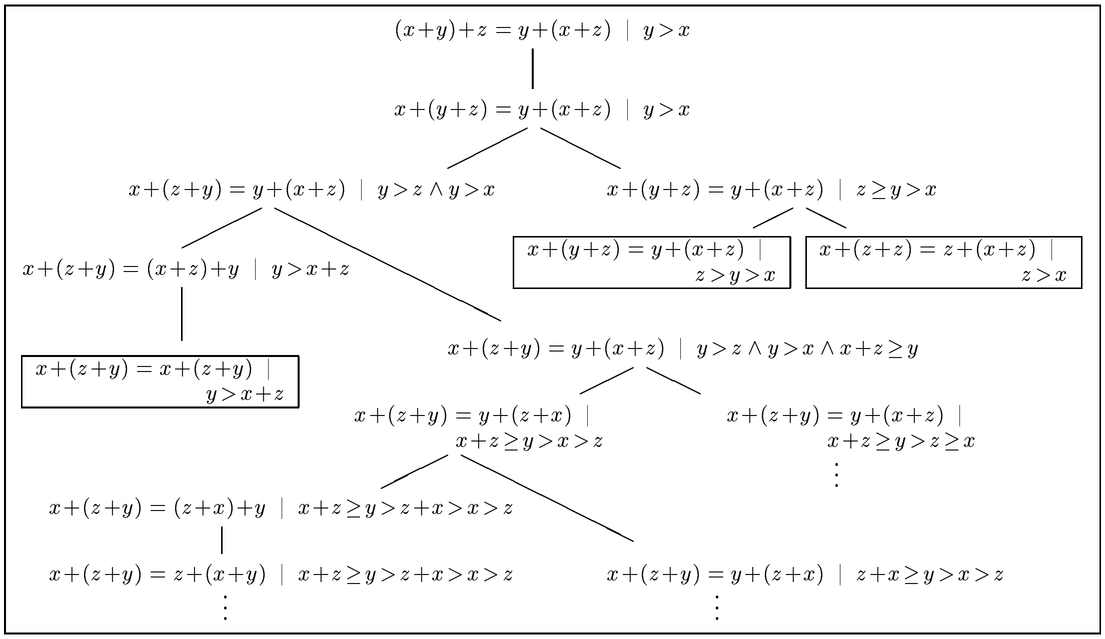
$(x+y)+z = y+(x+z) \mid y > x$
$x+(y+z) = y+(x+z) \mid y > x$
$x+(z+y) = y+(x+z) \mid y > z \wedge y > x$
$x+(y+z) = y+(x+z) \mid z \geq y > x$
$x+(z+y) = (x+z)+y \mid y > x+z$
$x+(y+z) = y+(x+z) \mid z > y > x$
$x+(z+z) = z+(x+z) \mid z > x$
$x+(z+y) = x+(z+y) \mid y > x+z$
$x+(z+y) = y+(x+z) \mid y > z \wedge y > x \wedge x+z \geq y$
$x+(z+y) = y+(z+x) \mid x+z \geq y > x > z$
$x+(z+y) = y+(x+z) \mid x+z \geq y > z \geq x$
$x+(z+y) = (z+x)+y \mid x+z \geq y > z+x > x > z$
$x+(z+y) = z+(x+y) \mid x+z \geq y > z+x > x > z$
$x+(z+y) = y+(z+x) \mid z+x \geq y > x > z$

Figure 1: An example of a confluence tree

an instance of the child and joinability follows by the induction hypothesis. □

Lemma 3.5 *Let $(E, >_{\rm lpo})$ be an ordered TRS and let T be a confluence tree for some critical pair between two equations in E. If T has some non-tautology leaf then $\rightarrow_{E,>_{\rm lpo}}$ is not confluent.*

Proof: From a non-tautology leaf $s = s' \mid c$ we can reconstruct a substitution σ (not necessarily a solution of c) such that $s\sigma$ and $s'\sigma$ are distinct and in normal form w.r.t. $\rightarrow_{E,>_{\rm lpo}}$, which contradicts confluence, since $E \models s = s'$ for every leaf $s = s' \mid c$.

We first build a satisfiable LPO-closed constraint G (for graph) expressing that s and s' are irreducible. Let α be an alien solution of c. Then α can be used to totally order the set of all subterms of s and s', so there is a simple system $s_n > \ldots > s_1$ (the *spine* of G) containing all subterms of s and s' such that α is a solution of the spine. This spine is the first part of G. Now we add to G all relations $r\theta > s_i$ (the *ribs* of G) such that $l\theta \equiv s_i$ for some i and some $l = r \in E$ with $l\theta \not\equiv r\theta$ and where $x\theta = 0$ for all $x \in vars(r) \setminus vars(l)$. Hence $r\theta$ does not contain any variables that are not in the spine, and all variables of G are in the spine of G.

Since α is a solution of c we have $r\theta\alpha \geq_{\rm lpo} l\theta\alpha$ (otherwise $s = s'|c$ would be further reducible and would not be a leaf). But $r\theta\alpha \equiv l\theta\alpha$ is impossible because α is alien and $l\theta \not\equiv r\theta$, so we have $r\theta\alpha >_{\rm lpo} l\theta\alpha$. Hence α satisfies G (both the spine and the ribs). Now we can close G under LPO while keeping α as a solution of G. Since the spine is already LPO-closed, this only has to be done for the ribs $r\theta > s_i$. It only produces new ribs of the form $t > s_j$ since equalities are again impossible because α is alien. Note that here t is not a variable since all variables are already on the spine.

We now complete the proof by showing that
a) the minimal solution σ of the spine satisfies the whole G and
b) all $s_i\sigma$ are distinct and irreducible wrt. $\rightarrow_{E,>_{\rm lpo}}$.

a) it suffices to show $t\sigma >_{\rm lpo} s\sigma$ for all pairs $t > s_i$ in G where s_i is in the spine and t is not (and hence t is not a variable). We proceed by induction on the ordering on pairs (s_i, t) whose first component is i (the situation of s_i in the spine) and whose second component is the subterm ordering on t. If s_i is a variable and $i = 1$ then $s_i\sigma$ is $\texttt{succ}(0)$, and then $t\sigma >_{\rm lpo} s_i\sigma$ since t contains at least one symbol g of $\mathcal{F}$ with $g >_{\mathcal{F}^e} 0$. If s_i is a variable and $i > 1$ then $s_i\sigma$ is $\texttt{succ}(s_{i-1})$ and by the induction hypothesis $t\sigma >_{\rm lpo} s_{i-1}\sigma$ and hence $t\sigma >_{\rm lpo} s_i\sigma$ since t is headed by a symbol of $\mathcal{F}$. If s_i is not a variable, then $t > s_i$ follows by the induction hypothesis from relations that are smaller under the measure since G

is LPO-closed.

b) Since σ satisfies the spine $s_n > \ldots > s_1$ clearly all $s_i\sigma$ are distinct. Now we prove that they are also irreducible wrt. $\rightarrow_{E,>_{\mathrm{lpo}}}$. We proceed by induction on the subindices i in $s_n > \ldots > s_1$. If s_1 is a variable, then $s_1\sigma$ is $\mathtt{succ}(0)$ which is clearly irreducible. If s_1 is ground then it is also irreducible since s_1 is a subterm of $s = s'$ in the leaf. For the induction step, if s_i is a variable, then $s_i\sigma$ is $\mathtt{succ}(s_{i-1}\sigma)$ which is irreducible since $s_{i-1}\sigma$ is irreducible by the induction hypothesis. If s_i is not a variable, it is of the form $f(t_1, \ldots, t_n)$ and all t_j are some s_k with $i > k$ and the $t_j\sigma$ are irreducible by the induction hypothesis and hence we only have to check reducibility at the topmost position.

Suppose $s_i\sigma \equiv l\theta$ for some $l = r \in E$. We show that for all such $l = r$ it is the case that $r\theta \geq_{\mathrm{lpo}} l\theta$ even if $x\theta = 0$ for all $x \in vars(r) \setminus vars(l)$. Since σ is alien, all variable positions of l must be positions in s_i (otherwise some non-variable position of l would be $\mathtt{succ}$). This means that $s_i \equiv l\rho$ for some ρ and that $\theta \equiv \rho\rho'$ for some ρ'. If $l\rho \equiv r\rho$ then $r\theta \equiv l\theta$ and we are done. If $l\rho \not\equiv r\rho$ then $r\rho > l\rho$ is a relation in G and hence $r\rho\sigma >_{\mathrm{lpo}} l\rho\sigma$ which implies $r\theta \geq_{\mathrm{lpo}} l\theta$. $\square$

Theorem 3.6 *The confluence of ordered TRS* $(E, >_{\mathrm{lpo}})$ *is decidable.*

Let us conclude this section by an example of application: we show how completion of associativity and commutativity axioms yields a confluent ordered rewrite system, making use of the above algorithm to check the confluence.

Example 6 We continue Example 5: we consider the axioms of associativity and commutativity of the binary symbol $+$. These axioms cannot be handled by standard completion as commutativity cannot be oriented without loosing termination.

The confluence tree of Example 5 shows that these two axioms alone are not confluent w.r.t. ordered rewriting since there are leaves of the confluence tree which are not tautologies. We may however add the equations which are leaves of the tree to the original set of axioms without modifying the original equational theory (this is a completion process) and check again for ordered confluence.

For instance, following Figure 1, we may add the equation $x + (y + z) = y + (x + z)$ to the original set of equations. Then we have a 3 equations presentation which turns out to be confluent w.r.t. ordered rewriting. There are more than 10 critical pairs to be considered and hence the associated confluence trees with tautology leaves cannot be depicted here (but they can be automatically reproduced by the *Saturate* system).

For this example, confluence can also be proved by means of the incomplete method given by Martin and Nipkow in [18]. Their method is based on the fact that every instance with some σ of a critical pair *cp* orders the variables of *cp* in some way, and sometimes one can prove confluence for each ordering. For example, one *cp* is $x_1 + (x_2 + (x_3 + x_4)) = x_3 + (x_1 + (x_2 + x_4))$, and if the ordering is $x_4\sigma >_{\mathrm{lpo}} x_3\sigma \equiv x_2\sigma >_{\mathrm{lpo}} x_1\sigma$, then no more information is needed to show that both sides of $cp\sigma$ rewrite into $x_1\sigma + (x_2\sigma + (x_2\sigma + x_4\sigma))$. (In fact, normalization in this rewrite system is simply sorting and associating to the right.)

Note that in many cases (e.g., the example given in the introduction) Martin and Nipkow's analysis is too coarse. In general, even a case analysis on all possible orderings between all subterms of the critical pair does not suffice.

4 Undecidability of confluence of constrained equations

If we use full constraints inheritance, then ordered completion generates constrained equations. Hence, it would be nice to be able to decide not only the confluence of an ordered rewrite system $(E, >_{\mathrm{lpo}})$, but also consider the case where E may contain constrained equations. Here, we show that this is impossible as going from unconstrained to constrained equations, confluence becomes undecidable.

Given a constrained equation $e : s = t \mid c$, a term $u \in T(\mathcal{F}^e)$ rewrites to v using e iff u rewrites to v using an instance $s\sigma = t\sigma$ such that $\sigma \models c \wedge s > t$. Confluence and ground confluence of sets of constrained equations are defined accordingly.

Theorem 4.1 *The problem of confluence of ordered rewriting for a finite set of constrained equations is undecidable.*

Proof sketch: We reduce the ground confluence problem for an lpo-terminating string rewriting system to the confluence of ordered rewriting of a finite set of constrained equations. The former is undecidable [15]. We consider the system $R_{u,v}$ of [15], Section 5, built on the alphabet $\mathcal{F}$ of unary and constant symbols. $R_{u,v}$ can be proved to terminate, using an lpo built on a precedence such that \$ (which is a constant) is the smallest symbol.

Then, the key idea of the encoding is the following: we can express the property that a term does not contain the symbol $\mathtt{succ}$ without any reference to $\mathtt{succ}$.

For this, we add a new symbol m, at the lower end of the precedence and the (constrained) equations E_0:

$$\begin{array}{rcll} m(x) & = & x & \\ y & = & f(x) & \mid \quad m(f(x)) > y > f(x) \\ & & & \quad\quad \text{for every } f \in \mathcal{F} \end{array}$$

Let $\mathcal{F}^e \cup \{m\}$ be the extended signature as before: the symbol $\mathtt{succ}$ is added at the lower end of the signature. The solutions for y of $m(f(x)) > y > f(x)$ are the terms $\mathtt{succ}^n(f(x))$ $(n > 1)$. This equation basically removes the $\mathtt{succ}$ symbols and that is why we can reduce confluence to ground confluence. More precisely, we can show that every term $t \in T(\mathcal{F}^e \cup \{m\})$ has a unique normal form $\hat{t} \in T(\mathcal{F})$ w.r.t. E_0. Then, we prove that $R_{u,v}$ is ground confluent iff $E = R_{u,v} \cup E_0$ is confluent:

First assume that $R_{u,v}$ is ground confluent (on $T(\mathcal{F})$). Since $\rightarrow_{E,>}$ is terminating, we only have to show the local confluence. Moreover, by induction on the number of rewriting steps it can be shown that on $T(\mathcal{F}^e)$ we have

$$\rightarrow_{R_{u,v}} \rightarrow_{E_0,>} ! \ \subseteq \rightarrow_{E_0,>} ! \ \rightarrow_{R_{u,v}}$$

i.e. one rewrite step with $R_{u,v}$ can be commuted with a normalization w.r.t. E_0. This implies more generally that $\rightarrow_{E,>} \rightarrow_{E_0,>} ! \ \subseteq \rightarrow_{E_0,>} ! \ \rightarrow_{R_{u,v}}$. Assume $s_1 \leftarrow_{E,>} s_0 \rightarrow_{E,>} s_2$ where $s_0, s_1, s_2 \in T(\mathcal{F}_1^e)$. Then, by the above commutation property, $\widehat{s_1} \leftarrow_{R_{u,v}} \widehat{s_0} \rightarrow_{R_{u,v}} \widehat{s_2}$ and by ground confluence of $R_{u,v}$, s_1 and s_2 are joinable by $R_{u,v}$: $s_1 \rightarrow^*_{E_0,>} \rightarrow^*_{R_{u,v},>} \leftarrow^*_{R_{u,v},>} \leftarrow^*_{E_0,>} s_2$, which shows the confluence.

Conversely, assume that $\rightarrow_{E,>}$ is confluent. Then $\rightarrow_{R_{u,v}}$ is confluent on $T(\mathcal{F})$ since a term in $T(\mathcal{F})$ can only be rewritten by rules in $R_{u,v}$. More details can be found in [4]. □

5 Undecidability of ground reducibility

Let us recall that a term t is *ground reducible* w.r.t. a rewrite system $\mathcal{R}$ iff all instances $t\sigma \in T(\mathcal{F})$ of t are reducible by $\mathcal{R}$. This definition extends to ordered rewriting, replacing $\mathcal{R}$ with $\rightarrow_{E,>}$ when E is a finite set of (unconstrained) equations.

Ground reducibility is decidable for arbitrary finite term rewriting systems [24]. We show here that it is undecidable for finite sets of equations:

Theorem 5.1 *The problem:*

Input: *A finite set of (unconstrained) equations E, a term t, a lexicographic path ordering.*

Question: *Is t ground reducible w.r.t. $\rightarrow_{E,>}$?*

is undecidable.

Proof Sketch: Our reduction is from the halting problem for deterministic two-counter machines. A *configuration* of the machine consists in two non-negative integers n_1, n_2 and a state $q \in Q$. The finite transition table of the machine is a mapping Δ from Q into $(\{1,2\} \times Q) \cup (\{1,2\} \times Q \times Q)$. A *move* of the machine from configuration (n_1, n_2, q) to (m_1, m_2, q') (written $(n_1, n_2, q) \vdash (m_1, m_2, q')$) is possible iff there is a transition $\Delta(q) = a$ and

1. Either $a = (i, q')$ and $m_i = n_i + 1$ and $m_j = n_j$ for $\{i, j\} = \{1, 2\}$
2. Or $a = (i, q', q'')$ and $m_i = n_i = 0$ and $m_j = n_j$ for $\{i, j\} = \{1, 2\}$
3. Or $a = (i, q'', q')$ and $m_i = n_i - 1 \geq 0$ and $m_j = n_j$ for $\{i, j\} = \{1, 2\}$

In order to encode two-counter machines, we consider an alphabet $\mathcal{F} = Q \cup \{a, b, 0\}$ where every symbol of Q (the set of states of the machine) is a ternary symbol, a, b are unary symbols and 0 is a constant. $\mathcal{F}$ is ordered according to $q > a > b > 0$ for every $q \in Q$ and the states are ordered in an arbitrary way.

We let t_n be the term $q_0(a^n(0), 0, x)$ and we are going to show that t_n is not ground reducible (w.r.t. a set of equations whose definition is sketched below) iff M halts on n. Intuitively, we are going to design E in such a way that irreducible ground instances of t_n encode (halting) sequences of successive configurations of the machine, as depicted on Figure 2. On this figure, configurations of the machine are (n_j, m_j, q_j).

The main tricks are the following: as in the previous section, we add a "dummy" symbol b, with the equation $b(x) = x$. Hence any term containing b is ground reducible. However, we can use it to enforce some equalities. For instance, the equations

$$\begin{array}{l} q(x, y, q'(a(x_1), y_1, z_1)) = q(x, y, q'(b(a(x)), y_1, z_1)) \\ q(x, y, q'(a(x_1), y_1, z_1)) = q(b(x_1), y, q'(a(x_1), y_1, z_1)) \end{array}$$

can be used from left to right (which is the only relevant direction) only if $a(x_1) > b(a(x))$ or $x > b(x_1)$. Assuming that ground irreducibility implies that $x = a^{n_1}(0)$ and $x_1 = a^{n_2}(0)$ for some n_1, n_2 (which is guaranteed by other equations), the rules do not apply only if $x = a^{n_1}(0) = x_1$. See [4] for more details.

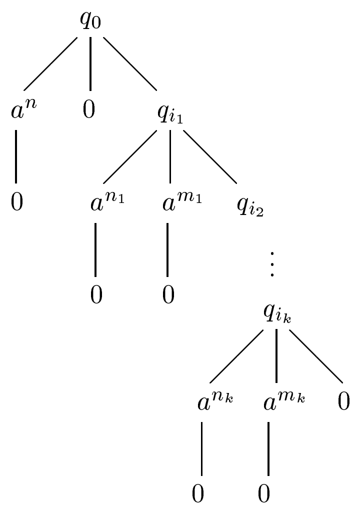

Figure 2: Representation of a sequence of configurations

6 Conclusion

We have shown that the behaviour of ordered rewriting is exactly the opposite of classical term rewriting for two important problems: confluence and ground reducibility. Confluence becomes decidable (whereas it is undecidable for term rewriting) and ground reducibility becomes undecidable (whereas it is decidable for term rewriting). These results provide interesting insights into the theory of ordered rewriting.

Regarding our proof of confluence of ordered TRS, in fact we show that confluence is equivalent to ground confluence over a signature with only the new symbol `succ`, which is what we finally show decidable. It is not difficult to show that for terminating TRS the same results hold. Regarding the applications to completion, apart from what has already been mentioned in the introduction, we also remark that our decision method works as well if, instead of building a tree for each critical pair $e \mid c \wedge u|_p = s$, we build it for $e\sigma \mid \top$ where $\sigma = mgu(u|_p, s)$, i.e., starting with an empty ordering constraint. However, the initial constraint c increases the efficiency of the procedure in practice by reducing the size of the tree.

Several questions remain open. For instance, our results heavily rely on a particular class of orderings: the lexicographic path orderings. Is there any other class of orderings that is useful in the context of ordered TRS for which confluence is also decidable?

References

[1] L. Bachmair. Proof by consistency in equational theories. In *Proc. 3rd IEEE Symp. Logic in Computer Science, Edinburgh*, July 1988.

[2] L. Bachmair, N. Dershowitz, and J. Hsiang. Orderings for equational proofs. In *Proc. 1st IEEE Symp. Logic in Computer Science, Cambridge, Mass.*, pages 346–357, June 1986.

[3] L. Bachmair, N. Dershowitz. Equational Inference, Canonical Proofs and Proof Orders. *Journal of the ACM* 41(2), pages 236–267, March 1994.

[4] H. Comon, P. Narendran and M. Rusinowitch. Decision Problems in Ordered Rewriting. Research Report 97-R-136, LORIA, Nancy, Nov. 1997. Available on the web at: `www.loria.fr/~rusi/publications.html.`

[5] H. Comon. Solving inequations in term algebras. In *Proc. 5th IEEE Symp. Logic in Computer Science, Philadelphia*, June 1990.

[6] H. Comon and R. Treinen. The first-order theory of lexicographic path orderings is undecidable. *Theoretical Computer Science*, 176, Apr. 1997.

[7] M. Dauchet. Simulation of Turing machines by a left-linear rewrite rule. N. Dershowitz, ed, *Rewriting Techniques and Applications, 3rd Int. Conf.*, LNCS 355,109–120, 1989.

[8] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1):69–115, Feb. 1987.

[9] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.

[10] N. Dershowitz, J-P. Jouannaud, and J-W. Klop. More problems in rewriting. In *5th Int. Conf. on Rewriting Techniques and Applications*, LNCS 690,468–487, Montreal, Canada, 1993.

[11] H. Ganzinger, R. Nieuwenhuis, and P. Nivela. The Saturate System, 1995. Software and documentation available: `www.mpi-sb.mpg.de`.

[12] J. Hsiang and M. Rusinowitch. On word problems in equational theories. In T. Ottmann, editor, *14th International Colloquium on Automata, Languages and Programming*, LNCS 267, 54–71, Karlsruhe, Germany, July 1987. Springer-Verlag.

[13] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in equational theories without constructors. In *Proc. 1st IEEE Symp. Logic in Computer Science, Cambridge, Mass.*, June 1986.

[14] D. Kapur and D. Musser. Proof by consistency. *Artificial Intelligence*, 31(2), Feb. 1987.

[15] D. Kapur, P. Narendran, and F. Otto. On ground confluence of term rewriting systems. *Inform. Comput.*, 86(1):14–31, 1989.

[16] C. Kirchner, H. Kirchner, and M. Rusinowitch. Deduction with symbolic constraints. *Revue Française d'Intelligence Artificielle*, 4(3):9–52, 1990. Special issue automatic deduction.

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

[18] U. Martin and T. Nipkow. Ordered rewriting and confluence. In Mark E. Stickel, editor, *10th International Conference on Automated Deduction*, LNAI 449, pages 366–380, Kaiserslautern, FRG, July 24–27, 1990. Springer-Verlag.

[19] M. Minsky. *Computation, Finite and Infinite Machines.* Prentice Hall, 1967.

[20] R. Nieuwenhuis. Simple LPO constraint solving methods. *Inf. Process. Lett.*, 47(2):65–69, Aug. 1993.

[21] P. Nivela and R. Nieuwenhuis. Practical results on the saturation of full first-order clauses: Experiments with the saturate system. (system description). In *5th Int. Conf. on Rewriting Techniques and Applications*, LNCS 690, Montreal, Canada, 1993. Springer-Verlag.

[22] R. Nieuwenhuis and A. Rubio. Theorem Proving with Ordering and Equality Constrained Clauses. *J. of Symbolic Computation*, 19(4):321–351, April 1995.

[23] G. E. Peterson. Complete sets of reductions with constraints. In M. E. Stickel, editor, *Proc. 10th Int. Conf. on Automated Deduction*, LNAI 449,381–395. Springer-Verlag, 1990.

[24] D. Plaisted. Semantic confluence tests and completion methods. *Information and Control*, 65:182–215, 1985.