Deciding the Confluence of Ordered Term Rewrite Systems

HUBERT COMON
LSV, CNRS
and
PALIATH NARENDRAN
SUNY at Albany
and
ROBERT NIEUWENHUIS
Tech. Univ. Catalonia
and
MICHAEL RUSINOWITCH
LORIA & INRIA-Lorraine,

A term rewrite system (TRS) terminates if, and only if, 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 generalized 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 $>$ belongs to a large class of path orderings (including most practical orderings like LPO, MPO, RPO (with status), KNS and RDO), since then *ordering constraints* for $>$ can be solved in an adequate way. For ordered TRS $(E, >)$ where E consists of *constrained equations*, confluence is shown to be undecidable. Finally, also *ground reducibility* is proved undecidable for ordered TRS.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Mechanical theorem proving*; F.4.2 [**Mathematical Logic and Formal Languages**]: Grammars and Other Rewriting Systems—*Decision problems*

General Terms: Theory

Additional Key Words and Phrases: Rewrite Systems, Confluence, Path Orderings, Ordered Rewriting

Authors adresses: H. Comon, LSV, CNRS, ENS Cachan, 61 Ave. Pr. Wilson, F-94235 Cachan cedex, France (comon@lsv.ens-cachan.fr). P. Narendran, Dept. Comp. Science, SUNY at Albany, Albany, NY 12222, USA (dran@cs.albany.edu). R. Nieuwenhuis , Dept. LSI, Tech. Univ. Catalonia, Jordi Girona 1, 08034 Barcelona, Spain (roberto@lsi.upc.es). M. Rusinowitch, INRIA-Lorraine, 615 rue du Jardin Botanique, BP 101, 54602 Villers les Nancy, France (rusi@loria.fr)
P. Narendran partially supported by the NSF grants CCR-9712396 and INT-9401087. R. Nieuwenhuis partially supported by the Spanish CICYT project HEMOSS ref. TIC98-0949-C02-01.
An earlier version of this work, presented at the LICS'98 conference, included similar results limited to the LPO ordering.

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 [Dau89]. However, a TRS R terminates if, and only 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 $>$ [HR87]. Hence ordered TRS can handle *any* set of equations E, being terminating by definition. A standard choice in practice for $>$ is some of the existing general-purpose *path orderings*, since they are easily defined and implemented. In Section 2 we characterize the requirements on $>$ that are needed for solving word problems by ordered rewriting, and we show that they can be obtained in a simple way for path orderings.

Confluence is undecidable in general, but for terminating TRS a decision procedure is given in Knuth and Bendix' landmark paper [KB70]: a TRS is confluent if, and only if, 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 [DJK93]. The main problem is that, due to the ordering restrictions, different instances of a critical pair may require different joinability proofs.

Here we prove the decidability of confluence if $>$ belongs to a class of path orderings: 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 characteristics 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 [KNO90].

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 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 [MN90].

In general the picture is not as 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* a finite *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. Our main result is finally that $(E, >)$ is confluent if, and only if, all leaves of the trees are tautologies.

The whole process of building the trees importantly relies on existing results for ordering constraint solving. This is also the case for our proof of finiteness of the trees. Here we adapt the constraint solving algorithms of [Nie93] with extended signature semantics (i.e., new symbols may appear in solutions) for the recursive path ordering with status (RPO). In RPO, every symbol is assigned a lexicographic or a multiset status. The lexicographic path ordering (LPO) or the multiset path ordering (MPO) are obtained as particular cases of RPO by making all symbols lexicographic or multiset, respectively (see, e.g., [Der82]). More refined path orderings like KNS [KNS85] or RDO [Les90] are included as well, since all these orderings coincide on ground terms [Der87], and hence the same constraint solving algorithms apply.

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., [BDH86; HR87; BDP89; Pet90; MN90; BD94]). 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 [NN93] we reported on the use of similar trees in the *Saturate* system [GNN01] 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* [KKR90; NR95], 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 [CT97]: given a term s and a rule $l \rightarrow r \mid c$, 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 [KM87; JK86; Bac88]. Here again, the result is the opposite of what holds in the case of finite term rewriting systems: we show that *ground reducibility*, a crucial ingredient for the proof-by-consistency approach, 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 RPO CONSTRAINTS

We follow the terminology and the notations of [DP01; NR01]. $T(\mathcal{F},\mathcal{X})$ 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. A *multiset* over a set S is a function $M\colon S \rightarrow I\!N$. The union and intersection of multisets are defined as usual by $M_1 \cup M_2(x) = M_1(x) + M_2(x)$, and $M_1 \cap M_2(x) = min(M_1(x), M_2(x))$. We also use a set-like notation: $M = \{a, a, b\}$ denotes $M(a) = 2$, $M(b) = 1$, and $M(x) = 0$ for $x \not\equiv a$ and $x \not\equiv b$. A multiset M is *empty* if $M(x) = 0$ for all $x \in S$. 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 $f(t_1, \ldots, t_n)$. Syntactic 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.

A (strict, partial) ordering is an irreflexive transitive binary relation. An ordering $\succ$ on $T(\mathcal{F}, \mathcal{X})$ is a *reduction* ordering if it is well-founded, monotonic, and stable under substitutions ($s \succ t$ implies $s\sigma \succ t\sigma$ for all σ with range in $T(\mathcal{F}, \mathcal{X})$).

The *word problem* for a set of equations E is to check, given an 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 $E \models s = t$ if, and only if, $s \leftrightarrow_E^* t$.

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 a reduction ordering. 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 $s\sigma \succ t\sigma$.

Our aim is to apply ordered rewriting with $(E, \succ)$ for solving word problems $E \models s = t$ built over $T(\mathcal{F}, \mathcal{X})$. In general $s = t$ will be universally quantified, but it is well-known that it suffices to consider only *ground* $s = t$ over a signature with sufficiently many new constant symbols: if σ is a Skolem substitution mapping each variable to a different new constant symbol, then $s \leftrightarrow_E^* t$ if, and only if, $s\sigma \leftrightarrow_E^* t\sigma$.

But for our purposes, we need to go slightly beyond: we consider only one new symbol `succ`. In the following, let 0 be a constant symbol in $\mathcal{F}$ (if there is no constant in $\mathcal{F}$, then 0 is added as usual), let `succ` be a new unary function symbol not in $\mathcal{F}$, and let $\mathcal{F}^e$ denote $\mathcal{F} \cup \{\texttt{succ}\}$.

Proposition 2.1. *Let E be a set of equations and let $s = t$ be an equation, both built over $T(\mathcal{F}, \mathcal{X})$. Let $vars(s = t)$ be $\{x_1, \ldots, x_n\}$, and let σ be the substitution $\{\ x_1 \mapsto \texttt{succ}(0),\ \ldots,\ x_n \mapsto \texttt{succ}^{(n)}(0)\ \}$.*
Then $s \leftrightarrow_E^* t$ if, and only if, $s\sigma \leftrightarrow_E^* t\sigma$.

In standard rewriting, termination and confluence of $\rightarrow_R$ suffice for solving word problems in the theory of a TRS R, since then $s \leftrightarrow_R^* t$ if, and only if, $s \rightarrow_R^* \circ \leftarrow_R^* t$, i.e., if the (unique) normal forms of s and t by $\rightarrow_R$ coincide.

In ordered rewriting this is no longer the case if the relation $\rightarrow_{E,\succ}$ is restricted by a too weak ordering $\succ$. 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 with respect to $\succ$ 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.

Since it suffices to consider word problems where the equation to be proved is Skolemized (i.e., ground), in the literature this problem is overcome by requiring $\succ$ to be total on ground terms.

Below we show that this can be weakened: it is sufficient that $\succ$ orients all non-trivial instances of the equations, i.e., these instances have a one-step rewrite proof using $\rightarrow_{E,\succ}$. But in fact it suffices to consider rewrite proofs with an arbitrary number of steps. This leads to the following precise characterization of what is needed for deciding word problems:

LEMMA 2.2. *Let $(E, \succ)$ be an ordered term rewrite system such that $\rightarrow_{E,\succ}$ is confluent and terminating on $T(\mathcal{F}^e)$. Then the following two statements are equivalent:*

(1) *$u\sigma \rightarrow^*_{E,\succ} \circ \leftarrow^*_{E,\succ} v\sigma$ for all $u = v \in E$ and all σ with range in $T(\mathcal{F}^e)$*
(2) *For all s, t in $T(\mathcal{F}^e)$, it holds that $s \leftrightarrow^*_E t$ if, and only if, $s \rightarrow^*_{E,\succ} \circ \leftarrow^*_{E,\succ} t$*

PROOF. The second statement trivially implies the first one. Conversely, for all $s, t \in T(\mathcal{F}^e)$, clearly $s \rightarrow^*_{E,\succ} \circ \leftarrow^*_{E,\succ} t$ implies $s \leftrightarrow^*_E t$, and, if $s \leftrightarrow^*_E t$, we have a derivation $s \equiv s_1 \leftrightarrow_E s_2 \leftrightarrow_E \ldots \leftrightarrow_E s_n \equiv t$ where, by 1 and the monotonicity of $\succ$, each step can be decomposed into zero or more steps of $\leftrightarrow_{E,\succ}$, and, by confluence and termination of $\rightarrow_{E,\succ}$, this implies $s \rightarrow^*_{E,\succ} \circ \leftarrow^*_{E,\succ} t$. □

The implication $1 \Rightarrow 2$ of this lemma says that, if $\rightarrow_{E,\succ}$ is confluent and terminating on $T(\mathcal{F}^e)$, and the ordering $\succ$ is strong enough to prove all instances of the equations of E, then the word problem for E is decidable by ordered rewriting with $(E, \succ)$. Therefore, the adequate notion of *confluence* of an ordered TRS is the following one:

Definition 2.3. Let E be a set of equations built over $T(\mathcal{F}, \mathcal{X})$.

—Let $\succ$ be a well-founded monotonic ordering on $T(\mathcal{F}^e)$. Then $(E, \succ)$ is called *confluent* if $\rightarrow_{E,\succ}$ is confluent on $T(\mathcal{F}^e)$.
—Let $\succ$ be a well-founded monotonic ordering on $T(\mathcal{F})$. Then the ordered rewrite system $(E, \succ)$ is called *ground confluent* if $\rightarrow_{E,\succ}$ is confluent on $T(\mathcal{F})$.

Note that in classical term rewriting confluence is defined for arbitrary extensions of $\mathcal{F}$ with new symbols, but this makes no sense in ordered rewriting without extending $\succ$ accordingly. But, as said, the extension to $T(\mathcal{F}^e)$ suffices for practice, and hence our definition of confluence for ordered TRS is the adequate counterpart of confluence of classical TRS.

As for classical term rewriting, in ordered rewriting there is a gap between confluence and ground confluence: given an ordering $\succ$ on $T(\mathcal{F}^e)$, confluence implies ground confluence, but the converse is false. An easy example is given by $\{f(x) \rightarrow x;\ f(x) \rightarrow 0\}$. This rewrite system is not confluent: e.g., $f(a)$ has the two distinct normal forms a and 0. However, if $\mathcal{F}$ only consists of $0, f$, then the system is ground confluent, because all terms rewrite into the normal form 0. This example also applies to ordered rewrite systems: introducing new symbols, that is, going from $\mathcal{F}$ to $\mathcal{F}^e$, allows to form critical peaks that cannot be reduced. For example, with the corresponding $\succ$, we would have $f(\mathtt{succ}(0)) \rightarrow_{E,\succ} \mathtt{succ}(0)$ and $f(\mathtt{succ}(0)) \rightarrow_{E,\succ} 0$, with distinct normal forms $\mathtt{succ}(0)$ and 0. In the next section, the new symbol `succ` will also be used to build solutions of ordering constraints, which is crucial in this respect.

2.2 The Recursive Path Ordering with Status (RPO)

Let $\succ$ be an ordering on terms and let $=$ be a congruence relation. The lexicographic (left to right) extension $\succ^{\text{lex}}$ of $\succ$ with respect to $=$ for n-tuples is defined:

$$\langle s_1, \ldots, s_n \rangle \succ^{\text{lex}} \langle t_1, \ldots, t_n \rangle \quad \text{iff} \quad s_1 = t_1, \ldots, s_{k-1} = t_{k-1} \text{ and } s_k \succ t_k$$

for some k in $1 \ldots n$. If $\succ$ is well founded, so is $\succ^{\text{lex}}$. The extension of $=$ to multisets, denoted by $=^{\text{mul}}$, is the smallest relation such that $\emptyset =^{\text{mul}} \emptyset$ and

$$S \cup \{s\} =^{\text{mul}} S' \cup \{t\} \text{ if } s = t \wedge S =^{\text{mul}} S'$$

The extension of $\succ$ to multisets with respect to $=$ is defined as the smallest ordering $\succ^{\text{mul}}$ such that

$$M \cup \{s\} \succ^{\text{mul}} N \cup \{t_1, \ldots, t_n\} \text{ if } M =^{\text{mul}} N \text{ and } s \succ t_i \text{ for all } i \in 1 \ldots n$$

If $\succ$ is well-founded on S, so is $\succ^{\text{mul}}$ on finite multisets over S [DM79].

Definition 2.4. Let $>_{\mathcal{F}}$ be a well-founded ordering on $\mathcal{F}$ (called the *precedence*), and let *stat* be the *status* function *stat*: $\mathcal{F} \to \{lex, mul\}$. Then the RPO ordering is defined as follows: $s >_{\text{rpo}} x$ if x is a variable that is a proper subterm of s or else $s \equiv f(s_1 \ldots s_n) >_{\text{rpo}} t \equiv g(t_1 \ldots t_m)$ if at least one of the following conditions hold:

—$s_i \geq_{\text{rpo}} t$, for some $i = 1 \ldots m$

—$f >_{\mathcal{F}} g$, and $s >_{\text{rpo}} t_j$, for all $j = 1 \ldots n$

—$f \equiv g$ (and hence n=m) and $stat(f) = mul$ and $\{s_1, \ldots, s_n\} >^{\text{mul}}_{\text{rpo}} \{t_1, \ldots, t_n\}$

—$f \equiv g$ (and hence n=m) and $stat(f) = lex$, $\langle s_1, \ldots, s_n \rangle >^{\text{lex}}_{\text{rpo}} \langle t_1, \ldots, t_n \rangle$, and $s >_{\text{rpo}} t_j$, for all $j = 1 \ldots n$

where $\geq_{\text{rpo}}$, $>^{\text{lex}}_{\text{rpo}}$ and $>^{\text{mul}}_{\text{rpo}}$ are defined with respect to $=^{\text{rpo}}$, equality up to permutation of arguments of function symbols with *mul* (multiset) status.

RPO is a reduction ordering on terms that fulfills the subterm property ($s >_{\text{rpo}} t$ 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. MPO is the particular case of an RPO with only *mul* (multiset) status symbols. LPO is the particular case of an RPO with only *lex* (lexicographic) status symbols. It is a simplification ordering that is total on $T(\mathcal{F})$ if $>_{\mathcal{F}}$ is total on $\mathcal{F}$. RPO is total on $T(\mathcal{F})$ up to $=^{\text{rpo}}$ if $>_{\mathcal{F}}$ is total on $\mathcal{F}$.

LPO with a total precedence is a standard choice for ordered TRS, since it fulfills the first statement of of Lemma 2.2 if the precedence is total on $\mathcal{F}$. For RPO these requirements have to be ensured by making f lexicographic whenever there are "permuting" equations $f(t_1, \ldots, t_n) \simeq f(t_{\pi(1)} \ldots t_{\pi(n)})$ (possibly with some context) where π is a permutation, thus avoiding any non-orientable $\mathcal{F}^e$-instances.

Hence in the following we will assume $>_{\text{rpo}}$ to be such an RPO on $T(\mathcal{F}^e)$, with a total 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 w.l.o.g.) and the unary symbol $\texttt{succ}$ is the smallest function symbol, i.e., we have $0 >_{\mathcal{F}^e} \texttt{succ}$. Furthermore, we will consider ordered rewrite systems $(E, >_{\text{rpo}})$, but all our results apply equivalently to ordered TRS $(E, \succ)$ where $\succ$ is any ordering such that $s \succ t$ iff $s >_{\text{rpo}} t$ for all ground terms s and t, which is in particular the case for the aforementioned path orderings KNS and RDO. The ordering $>_{\text{lpo}}$ will be used several times to denote an RPO where all symbols have lexicographic status.

Example 2.5. *Let E be the set $\{x + y = y + x, \quad x + (y + z) = (x + y) + z\}$. If $s \equiv f(0, \texttt{succ}(f(\texttt{succ}(0), 0)))$ and $t \equiv \texttt{succ}(f(0, \texttt{succ}(\texttt{succ}(0))))$, then $s >_{lpo} t$*

and hence $s + t \rightarrow_{E,>_{lpo}} t + s$*. We also have* $(\mathtt{succ}(0) + 0) + 0 \rightarrow_{E,>_{lpo}} (0 + \mathtt{succ}(0)) + 0 \rightarrow_{E,>_{lpo}} 0 + (\mathtt{succ}(0) + 0) \rightarrow_{E,>_{lpo}} 0 + (0 + \mathtt{succ}(0))$*, the latter term being irreducible.* □

Example 2.6. *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* $f(a,a) \rightarrow_{E,>_{lpo}} g(a,a)$*, but* $f(a,a)$ *also rewrites into* $g(a,b)$ *or* $g(a,0)$ *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 with respect to* $>_{lpo}$*.*
Of course, if $\rightarrow_{E,>_{lpo}}$ *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* $\rightarrow_{E',>_{lpo}}$ *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*.* □

2.3 RPO constraints

For more details on the definitions and notations of this section, as well as for the main results, we refer to [Com90; Nie93; NR95; NR01].

Definition 2.7. An *RPO 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. $s = t$) if, and only if, $s\sigma >_{\mathrm{rpo}} t\sigma$ (resp. $s\sigma =^{\mathrm{rpo}} t\sigma$). 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 2.8. *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 independently of the (lexicographic or multiset) status of* f*.* □

Definition 2.9. A *simple system* S is an RPO constraint of the form

$$s_n \ \#_n \ s_{n-1} \ \#_{n-1} \ \ldots \ \#_1 \ s_0$$

where:

—each $\#_i$ is either $=$ or $>$
—every strict subterm of an s_i is $=^{\mathrm{rpo}}$-equivalent to some s_j with $i > j$ and
—$s_i \neq^{\mathrm{rpo}} s_j$ whenever $i \neq j$.

By $=_S$ we denote the least equivalence relation on $\{s_0, \ldots, s_n\}$ which contains all pairs (t_i, t_{i-1}) such that $\#_i$ is $=$. Similarly, $>_S$ 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 $s >_S v$ whenever $s =_S t$, $t >_S u$ and $u =_S v$. Intuitively, $=_S$ and $>_S$ are the equality and ordering relations that can be deduced from S.

In what follows, we will assume that all terms occurring in a simple system are written in a *sorted* way, that is, such that $t_i \geq_S t_j$ whenever $f(t_1, \ldots, t_n)$ occurs in S for some f with multiset status and $1 \leq i < j \leq n$. This allows us to consider relations between such terms $f(t_1, \ldots, t_n)$ and $f(s_1, \ldots, s_n)$ as if f had a lexicographic status; for example, if $f(t_1, \ldots, t_n) >_S f(s_1, \ldots, s_n)$ for some satisfiable simple system S, then necessarily $s_1 =_S t_1 \ \wedge \ldots \wedge \ s_{i-1} =_S t_{i-1} \wedge s_i >_S t_i$ for some i. Similarly, if $f(t_1, \ldots, t_n) =_S f(s_1, \ldots, s_n)$ for some satisfiable simple system S, then necessarily $s_1 =_S t_1 \ \wedge \ \ldots \ \wedge \ s_n =_S t_n$.

The *equational part* of a simple system S, denoted *eqpart*(S), is $\{s = t \mid s =_S t\}$. If σ is the most general simultaneous unifier of *eqpart*(S), then the *inequational part* of S, denoted *ineqpart*(S), is $\{s\sigma > t\sigma \mid s >_S t\}$. Furthermore, *ineqpart*$(S) = \perp$ if *eqpart*(S) is not unifiable. It is not difficult to see that *ineqpart*(S) is again a simple system (if it is not $\perp$). S is called *purely inequational* if $S \equiv$ *ineqpart*(S).

The satisfiability problem for RPO constraints (with solutions over $\mathcal{F}^e$) can be solved in NP time following the methods of [Nie93]. 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 if, and only if, 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 S is satisfiable if, and only if, every relation between non-variable terms $s >_S t$ follows from a conjunction of relations in S between s' and t' that are subterms of s and t respectively, at least one of them being a strict subterm. For example, $f(a, x, b) >_S f(a, y, c)$ follows from $x >_S y \wedge f(a, x, b) >_S c$; similarly, if $f >_{\mathcal{F}^e} g$, the relation $f(x) >_S g(a, a)$ follows from $f(x) >_S a$, and if $g >_{\mathcal{F}^e} f$, the relation $f(x) >_S g(a, b)$ follows from $x >_S g(a, b)$.
Let us give some intuition for this. On the one hand, if some $s >_S t$ does not follow from such relations between smaller terms in S, then S is unsatisfiable since S is closed under subterms and hence the relations between the smaller terms contradict $s >_S t$.
Reversely, if all relations $s >_S t$ between non-variable terms follow from relations between smaller terms, then one can build a solution σ from right to left in S as follows. Assume S is of the form $s_n > \ldots > s_0$. For each s_i that is a variable, let $s_i\sigma$ be $\mathtt{succ}(0)$ if $i = 0$ and $\mathtt{succ}(s_{i-1}\sigma)$ otherwise. It is not hard to see that this σ is indeed a solution of S (see [Nie93] for details).

Definition 2.10. Let S be a satisfiable purely inequational simple system of the form $s_n > \ldots > s_0$. Then the *minimal* solution of S is defined as follows. For each s_i that is a variable, $s_i\sigma$ is $\mathtt{succ}(0)$ if $i = 0$, and $s_i\sigma$ is $\mathtt{succ}(s_{i-1}\sigma)$ otherwise.

A solution σ is called *alien* if for every variable x, the solution $x\sigma$ is headed by the new symbol $\mathtt{succ}$. Note that minimal solutions are alien.

3. DECIDABILITY OF CONFLUENCE OF ORDERED REWRITING

In the following, $(E, >_{\mathrm{rpo}})$ will be an ordered TRS. According to the previous section, we assume that terms of E and of constraints belong to $T(\mathcal{F}, \mathcal{X})$ and that substitutions have range in $T(\mathcal{F}^e, \mathcal{X})$. 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 (and ordered rewriting) reduces to the joinability of critical pairs. The following result is essentially due to J. Hsiang and M. Rusinowitch [HR87] (see also [BDH86]):

LEMMA 3.1. *$\rightarrow_{E,>rpo}$ is confluent if, and only if, 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 $s\sigma \rightarrow^*_{E,>rpo} u \leftarrow^*_{E,>rpo} t\sigma$.*

Definition 3.2. A *confluence tree* for $(E, >_{\text{rpo}})$ 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 $\{e \mid S_1, \dots, e \mid S_n\}$ if c is satisfiable and not a simple system and $\{S_1, \dots, S_n\}$ 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 $\mathit{eqpart}(c)$.

Let us remark that being able to replace extra variables by 0 in *constrained rewriting* is a key ingredient in the decidability proof.

EXAMPLE 3.3. *Consider again the set $E \equiv \{x + y = x + z,\ \ (x + y) + z = x + (y + z)\}$ where $\mathit{stat}(+) = $ lex. 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 [NN93], 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 $\rightarrow_{E,>_{\text{rpo}}}$ is confluent if, and only if, for any critical pair one (any) such tree has

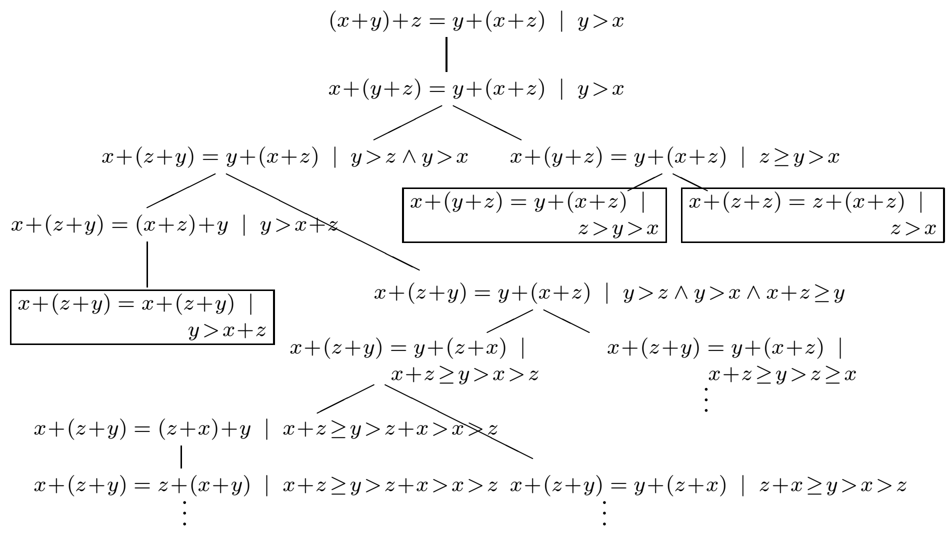

$(x+y)+z = y+(x+z) \mid y>x$
$x+(y+z) = y+(x+z) \mid y>x$
$x+(z+y) = y+(x+z) \mid y>z \wedge y>x$
$x+(y+z) = y+(x+z) \mid z \geq y>x$
$x+(y+z) = y+(x+z) \mid z>y>x$
$x+(z+z) = z+(x+z) \mid z>x$
$x+(z+y) = (x+z)+y \mid y>x+z$
$x+(z+y) = x+(z+y) \mid y>x+z$
$x+(z+y) = y+(x+z) \mid y>z \wedge y>x \wedge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

Fig. 1. An example of a confluence tree

only tautology leaves. For this purpose, we first show that the trees are finite (Lemma 3.4), and that it is easy to decide whether a leaf $s = t \mid c$ is a tautology (Lemma 3.5). Then, in Lemma 3.6 we show that every instance of the critical pair at the root can be rewritten into some leaf, and hence $\rightarrow_{E,>_{\mathrm{rpo}}}$ is confluent if all leaves are tautologies, since then all instances of the critical pairs are joinable. Finally, for the (harder) reverse implication (Lemma 3.8), 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 3.4. *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 instantiation steps can be applied on a branch since they reduce the number of variables of the descendant nodes, and no two consecutive decomposition steps can be applied 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 $e_j \mid c_j$, 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 $e_i\sigma >_{\mathrm{rpo}} e_j\sigma$, which contradicts $e_i \equiv e_j$ or $e_j >_{emb} e_i$. □

Lemma 3.5. *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 if, and only if, 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 $\mathtt{succ}$), 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. $\square$

LEMMA 3.6. *Let $\mathcal{T}$ be a confluence tree rooted by $s = t \mid c$. If all leaves of $\mathcal{T}$ 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 an instance of the child and joinability follows by the induction hypothesis. $\square$

LEMMA 3.7. *Let S be an inequational simple system $s_n > \ldots > s_1$ and let T be a constraint over $\mathcal{T}(\mathcal{F})$ of the form $t_1 > s_{i_1} \wedge \ldots \wedge t_k > s_{i_k}$, with $i_j \in 1 \ldots n$, such that $\mathit{vars}(T) \subseteq \mathit{vars}(S)$. Then, if $S \wedge T$ is satisfiable, the minimal solution σ of S (see Definition 2.10) is also a solution of T.*

Proof: Let α be a solution of $S \wedge T$. We show that for every subterm t of some t_j we have $t\sigma >_{\mathrm{rpo}} s_i\sigma$ whenever $t\alpha >_{\mathrm{rpo}} s_i\alpha$. If t is itself one of the s_j, or if t contains some variable x occurring in S to the left of s_i then the result trivially holds. Otherwise, t is headed with a symbol g of $\mathcal{F}$ with $g >_{\mathcal{F}^e} 0$ and we proceed by induction on (i) the subindex i of s_i, and (ii) on the size of t.

Base step. If s_1 is a variable, then $s_1\sigma$ is $\mathtt{succ}(0)$, and $t\sigma >_{\mathrm{rpo}} s_i\sigma$ for all t headed with a symbol g of $\mathcal{F}$ with $g >_{\mathcal{F}^e} 0$. If s_1 is a constant, then, either $t\alpha >_{\mathrm{rpo}} s_1$ because t is a constant larger than s_1 or t is a non-constant term containing some symbol g of $\mathcal{F}$ with $g \geq_{\mathcal{F}^e} s_1$ and again we have $t\sigma >_{\mathrm{rpo}} s_i\sigma$ (note that if t contains no such g then $t\alpha >_{\mathrm{rpo}} s_1$ can only hold because t contains some variable x occurring in S to the left of s_1).

Induction step. If s_i is a variable, then $s_i\sigma$ is $\mathtt{succ}(s_{i-1}\sigma)$, and $t\sigma >_{\mathrm{rpo}} \mathtt{succ}(s_i\sigma)$ since by induction hypothesis $t\sigma >_{\mathrm{rpo}} s_{i-1}\sigma$ and t is headed with a symbol g of $\mathcal{F}$ with $g >_{\mathcal{F}^e} \mathtt{succ}$. If s_i is not a variable, by induction hypothesis we may assume that $t'\sigma >_{\mathrm{rpo}} s'\sigma$ whenever $t'\alpha >_{\mathrm{rpo}} s'\alpha$ if t' and s' are subterms of t and s respectively, and at least one of them is a proper subterm (note that s' is an s_j in S with $i \geq j$). But then $t\alpha >_{\mathrm{rpo}} s_i\alpha$ is due to the relation with respect to

$>_{\mathcal{F}^e}$ between their topmost symbols (and in case of equal top symbols, due to their lexicographic or multiset status), and due to the relations between such $t'\alpha$ and $s'\alpha$, and hence in each case, if $t\alpha >_{\mathrm{rpo}} s_i\alpha$, then for the same reason also $t\sigma >_{\mathrm{rpo}} s_i\sigma$ holds. □

Lemma 3.8. *Let $(E, >_{rpo})$ be an ordered TRS and let $\mathcal{T}$ be a confluence tree for some critical pair between two equations in E. If $\mathcal{T}$ has some non-tautology leaf then $\rightarrow_{E,>_{rpo}}$ 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. $\rightarrow_{E,>_{\mathrm{rpo}}}$, which contradicts confluence, since $E \models s = s'$ for every leaf $s = s' \mid c$.

We first build a satisfiable constraint $S \wedge T$ 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 $=^{\mathrm{rpo}}$-different subterms of s and s', so there is a simple system S of the form $s_n > \ldots > s_1$ containing all subterms of s and s' such that α is a solution of S. Now consider the constraint T consisting of all relations $r\theta > s_i$ such that $l\theta \equiv s_i$ for some i, some θ, and some $l = r \in E$ with $x\theta = 0$ for all $x \in vars(r) \setminus vars(l)$, and $l\theta \neq^{\mathrm{rpo}} r\theta$. Hence $r\theta$ does not contain any variables that are not in S, and all variables of T are in S.

Since α is a solution of c we have $r\theta\alpha \geq_{\mathrm{rpo}} l\theta\alpha$ (otherwise $s = s'|c$ would be further reducible and would not be a leaf). But $r\theta\alpha =^{\mathrm{rpo}} l\theta\alpha$ is impossible because α is alien and $l\theta \neq^{\mathrm{rpo}} r\theta$, so we have $r\theta\alpha >_{\mathrm{rpo}} l\theta\alpha$. Hence α satisfies $S \wedge T$. Now by the previous lemma, the minimal solution σ of the S satisfies $S \wedge T$.

Now we complete the proof by showing that all $s_i\sigma$ are distinct and irreducible with respect to $\rightarrow_{E,>_{\mathrm{rpo}}}$. Since σ satisfies $s_n > \ldots > s_1$, clearly all $s_i\sigma$ are distinct. Now we prove that they are also irreducible with respect to $\rightarrow_{E,>_{\mathrm{rpo}}}$. We proceed by induction on the sub-indices 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 $s_i\sigma \equiv l\gamma$ for some $l = r \in E$. We show that for all such $l = r$ it is the case that $r\gamma \geq_{\mathrm{rpo}} l\gamma$ even if $x\gamma = 0$ 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 $s_i \equiv l\theta$ for some θ. If $l\theta =^{\mathrm{rpo}} r\theta$ then $r\gamma =^{\mathrm{rpo}} l\gamma$ and we are done. If $l\theta \neq^{\mathrm{rpo}} r\theta$ then $r\theta > l\theta$ has been added in the construction of T and hence $r\theta\sigma >_{\mathrm{rpo}} l\theta\sigma$ which implies $r\gamma \geq_{\mathrm{rpo}} l\gamma$. □

Theorem 3.9. *The confluence of ordered TRS $(E, >_{rpo})$ 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 3.10. *We continue Example 3.3: we consider the axioms of associa-*

tivity and commutativity of the binary (lexicographic status) symbol $+$. These axioms cannot be handled by standard completion as commutativity cannot be oriented without loosing termination.

The confluence tree of Example 3.3 shows that these two axioms alone are not confluent with respect to 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 with respect to 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 [MN90]. 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 $x_4\sigma >_{lpo} x_3\sigma \equiv x_2\sigma >_{lpo} x_1\sigma$, then no more information is needed to show that both sides of cpσ 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 $(E, >_{\text{rpo}})$, 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 if, and only if, 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: The main idea is that we can express, without any reference to $s, 0$ that a term cannot contain $s, 0$. Then we can reduce ground confluence of a terminating rewrite system to the confluence of ordered rewriting with respect to a set of constrained equations. On the other hand, ground confluence is undecidable, even for terminating rewrite systems.

We reduce the ground confluence problem for an lpo-terminating string rewriting to the confluence of ordered rewriting of a finite set of constrained equations. The former is undecidable [KNO90].

We consider the system $R_{u,v}$ of [KNO90], section 5. It only involves unary function symbols. No right hand side is reduced to a variable. This system is terminating; its termination can be proved using an lpo extending a total precedence on a finite alphabet of unary function symbols, plus a constant \$, which is the smallest constant in the signature. Then, we extend this signature $\mathcal{F}$ with 3 function symbols: $m > \texttt{succ} > 0$ which is at the lower end of the precedence. Let $\mathcal{F}_0 = \mathcal{F} \setminus \{\$\} \cup \{0\}$, $\mathcal{F}_1 = \mathcal{F} \cup \{m\}$ and $\mathcal{F}_1^e = \mathcal{F}_1 \cup \{\texttt{succ}, 0\}$. Our set of equations contains $R_{u,v}$ plus the following additional equations E_0:

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

and a copy of some rules in $R_{u,v}$; let E_1 be

$$\alpha(x) = \beta(x) \mid \$ > x \quad \text{for every rule } \alpha\$ \to \beta\$ \in R_{u,v}$$

Basically, the first equation expresses what we want: the solutions for y of $m(f(x)) > y > f(x)$ are the terms $\texttt{succ}^n(f(x))$ $(n > 1)$. Moreover, if E is the resulting set of constrained equations, we claim that for every term $s \in T(\mathcal{F}_1^e)$, s has a unique normal form $\hat{s}$ w.r.t. E_0 which belongs to $T(\mathcal{F}_0)$. Indeed,

$$\begin{array}{rl} [\![E_0]\!] = & \{\texttt{succ}^n(f(x)) \to f(x) \mid f \in \mathcal{F}, n \geq 1\} \\ & \cup\{\$ \to u \mid u \in T(\{m, \texttt{succ}, 0\})\} \\ & \cup\{u \to v \mid u, v \in T(\{m, Succ, 0\}) \ \text{ and } u > v\} \\ & \cup\{m(x) \to x\} \end{array}$$

According to the three last sets of rules, the only irreducible term in the set $T(\{\$, m, \texttt{succ}, 0\})$ is 0 and every term in this set reduces to 0. The first set of rules eliminates all occurrences of succ which are above a symbol $f > \$$. Moreover $[\![E_0]\!]$ is confluent since all critical pairs are trivially joinable.

Moreover, on $T(\mathcal{F}_1^e)$, $\to_{R_{u,v}}, \to_{!,E_0} \subseteq \to_{!,E_0} \to_{>,E_1}$ if $\to_{!,E_0}$ is the reduction relation with respect to E_0 until a normal form is reached. This can be shown by induction on the number of rewriting steps. This implies more generally that $\to_{>,E} \to_{!,E_0} \subseteq \to_{!,E_0} \to_{>,E_1}$

Now, we claim that $\to_{>,E}$ is confluent iff $R_{u,v}$ is ground confluent, which proves undecidability.

First assume that $R_{u,v}$ is ground confluent (on $T(\mathcal{F})$). Then E_1 is ground confluent on $T(\mathcal{F}_0)$. Since $\to_{>,E}$ is terminating, we only have to show the local confluence. Assume $s_1 \leftarrow_{>,E} s_0 \to_{>,E} s_2$ where $s_0, s_1, s_2 \in T(\mathcal{F}_1^e)$. Then by the above commutation property, $\hat{s_1} \leftarrow_{>,E_1} \hat{s_0} \to_{>,E_1} \hat{s_2}$ and by ground confluence of E_1, s_1 and s_2 are joinable by E_1: $s_1 \to_{*,E_0} \to_{*,E_1} \leftarrow_{*,E_1} \leftarrow_{*,E_0} s_2$, which shows the confluence.

Conversely, assume that $\to_{>,E}$ is confluent. Then $\to_{>,E_1}$ is confluent on $T(\mathcal{F}_0)$ since a term in $T(\mathcal{F}_0)$ can only be rewritten by rules in E_1. Replacing 0 with \$ we get the confluence of $R_{u,v}$ on $T(\mathcal{F})$. $\square$

5. UNDECIDABILITY OF GROUND REDUCIBILITY

Let us recall that a term t is *ground reducible* with respect to 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 (see [Pla85]). 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 with respect to $\rightarrow_{E,>}$?

is undecidable.

Proof: We reduce the halting problem for a two-counters machine. First, let us recall this computation model.

A (deterministic) two counter machines is a tuple (q_0, Q_f, Q, Δ) where Q is a finite set of states, $Q_f \subseteq Q$ is the set of final states, q_0 is the initial state and Δ is a *transition function* from Q to a finite set of *actions* A, consisting of

(1) couples $(1, q')$ where $q' \in Q$
(2) couples $(2, q')$ where $q' \in Q$
(3) triples $(1, q', q'')$ where $q', q'' \in Q$
(4) triples $(2, q', q'')$ where $q', q'' \in Q$

We assume that Δ is undefined on states $q \in Q_f$.

A *configuration* of the machine consists in two non-negative integers n, m and a state $q \in Q$. A *move* of the machine from configuration (n_1, m_1) to (n_2, m_2, q') (written $(n_1, m_1, q) \vdash (n_2, m_2, q')$) is possible iff there is a transition $\Delta(q) = a$ and

(1) Either $a = (i, q')$ and $p^i_2 = p^i_1 + 1$ and $r^i_2 = r^i_1$. (Increase counter i and move to q').
(2) Or $a = (i, q', q'')$ and $p^i_1 = p^i_2 = 0$ and $r^i_1 = r^i_2$. (If counter i is zero, move to q' without changing the counters values).
(3) Or $a = (1, q'', q')$ and $p^i_2 = p^i_1 - 1 \geq 0$ and $r^i_2 = r^1_i$ (If counter i is positive, decrement it and move to q').

where $p^1_j = n_j$, $r^1_j = m_j$, $p^2_j = m_j$ and $r^2_j = n_j$.

The *input* of such a machine is a non-negative integer n, corresponding to the initial configuration $(n, 0, q_0)$. The machine *halts* on the input n iff there is a finite sequence of transitions yielding a configuration (n', m', q_f) with $q_f \in Q_f$.

The following problem is undecidable [Min67]:

Input. : a two counter machine M and a non-negative integer n

Question. : Does M halt on n ?

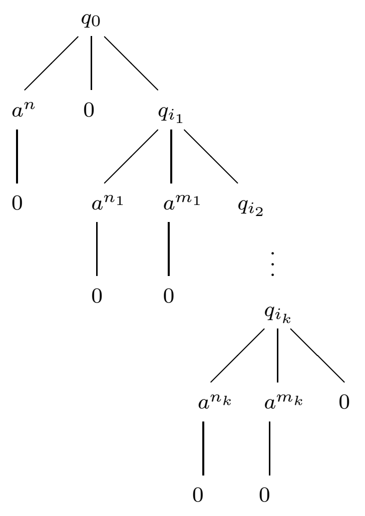


Fig. 2. Representation of a sequence of configurations

We may assume without loss of generality that $q' \neq q''$ whenever $\Delta(q) = (i, q', q'')$: if $q' = q''$, it suffices to introduce two new states q'_1 and q'_2 and let $\Delta(q) = (i, q', q'_1)$, $\Delta(q'_1) = (i, q'_2)$, $\Delta(q'_2) = (i, q'_1, q')$.

In order to encode two-counters 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.

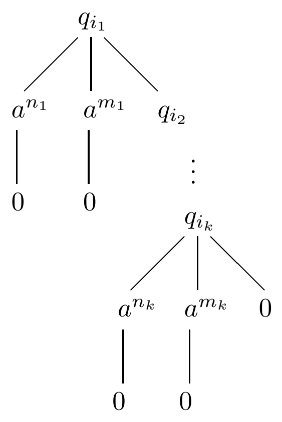
We let t_n be the term $q_0(a^n(0), 0, x)$ and we are going to show that t_n is not ground reducible (w.r.t. a set of equations which is defined below) iff M halts on n. Intuitively, we are going to design E in such a way that irreducible ground instances of t_n encode (halting) sequences of successive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

We divide the set of equations into 2 parts: the first part is independent of Δ and is designed in such a way that only sequences of possible configurations (not necessarily consecutive ones) are kept as irreducible terms. The second set (the main one) encodes the computations of the machine.

The first set of equations.

1.	$q(x, y, 0) = 0$	For every non final state q
2.	$q(x, y, a(z)) = 0$	for every state q
3.	$q(q'(x_1, x_2, x_3), x_4, x_5) = 0$	for every states q, q'
4.	$q(x_1, q'(x_2, x_3, x_4), x_5) = 0$	for every states q, q'
5.	$a(q(x, y, z)) = 0$	for every state q
6.	$b(x) = x$	

All these equations can actually be turned into rules from left to right. Irreducible ground terms w.r.t. these rules are the terms $a^k(0)$ and the terms of the form:



where q_{i_k} is a final state. Let S_1 be this set of ground terms. In what follows, we only have to consider the applicability of the rules on S_1.

The second set of equations.

Equation (7) simply remove configurations which cannot be consecutive because of the successive states.

7. $q(x_1, x_2, q'(x_3, x_4, x_5)) = 0$ If $q, q' \in Q$ and $\forall i, \forall q''.\ \Delta(q) \neq (i, q')$ $\&\Delta(q) \neq (i, q', q'')\&\Delta(q) \neq (i, q'', q')$

Now, we have to move the counters in the right way. The following array displays the equations in the case where $\Delta(q) = (1, q')$ and summarizes the constraints which are imposed by each rule on a the terms of S_1.

	equation	$\Delta(q)$	constraint on n_1, m_1, n_2, m_2 resulting from the irreducibility
8	$q(x, y, q'(x_1, y_1, z_1)) = q(x, y, q'(b(a(x)), y_1, z_1))$	$(1, q')$	$n_2 \leq n_1 + 1$
9	$q(x, y, q'(a(x_1), y_1, z_1)) = q(b(x_1), y, q'(a(x_1), y_1, z_1))$	$(1, q')$	$n_2 \geq n_1 + 1 \vee n_2 = 0$
10	$q(x, y, q'(0, y_1, z_1)) = 0$	$(1, q')$	$n_2 \neq 0$
11	$q(x, y, q'(x_1, y_1, z_1)) = q(x, b(y_1), q'(x_1, y_1, z_1))$	$(1, q')$	$m_2 \geq m_1$
12	$q(x, y, q'(x_1, y_1, z_1)) = q(x, y, q'(x_1, b(y), z_1))$	$(1, q')$	$m_1 \geq m_2$

(8), when applied from left to right reduces a term

$$q(a^{n_1}(0), a^{m_1}(0), q'(a^{n_2}(0), a^{m_2}(0), t)$$

when $a^{n_2}(0) > b(a(a^{n_1}(0)))$, i.e. when $n_2 > 1 + n_1$. (8) cannot be applied from right to left on terms of S_1 since b does not occur in any term of S_1.

(9), when applied from left to right reduces a term

$$q(a^{n_1}(0), a^{m_1}(0), q'(a^{n_2}(0), a^{m_2}(0), t)$$

when $a^{n_1}(0) > b(a^{n_2-1}(0))$, i.e. when $n_1 \geq n_2 > 0$. As before, (9) cannot be applied from right to left on terms from S_1. (10) forces $n_2 > 0$.

Putting together (8), (9) and (10), if we assume that there is a rule $\Delta(q) = (1, q')$, a term

$$q(a^{n_1}(0), a^{m_1}(0), q'(a^{n_2}(0), a^{m_2}(0), t))$$

is irreducible (at the root) iff $n_2 = n_1 + 1$.

Note that the role of b is twofold: each time it appears in a right member of a rule, we have only to consider possible applications of the equation from left to right, since b does not occur in any term of S_1. This forces to consider a particular orientation, which we will not recall in what follows. Then, because b is minimal in the precedence, it plays no role in the ordering constraint, except that it prevents some equalities.

In a similar way, (11) and (12) force the second counter to be remain constant (i.e. $m_2 = m_1$) for such transition rules.

Now, we have similar equations as (8–12) for transitions $\Delta(q) = (2, q')$:

	equation	$\Delta(q)$	constraint on n_1, m_1, n_2, m_2 resulting from the irreducibility
13	$q(x, y, q'(x_1, y_1, z_1)) = q(x, y, q'(x_1, b(a(y)), z_1))$	$(2, q')$	$m_1 + 1 \geq m_2$
14	$q(x, y, q'(x_1, a(y_1), z_1)) = q(x, b(y_1), q'(x_1, a(y_1), z_1))$	$(2, q')$	$m_2 - 1 \geq m_1$ $\vee m_2 = 0$
15	$q(x, y, q'(x_1, 0, z_1)) = 0$	$(2, q')$	$m_2 \neq 0$
16	$q(x, y, q'(x_1, y_1, z_1)) = q(b(x_1), y, q'(x_1, y_1, z_1))$	$(2, q')$	$n_2 \geq n_1$
17	$q(x, y, q'(x_1, y_1, z_1)) = q(x, y, q'(b(x), y_1, z_1))$	$(2, q')$	$n_1 \geq n_2$

Finally, we design similar rules for the other transitions, which are displayed in the next array.

	equation	$\Delta(q)$	constraint on n_1, m_1, n_2, m_2 resulting from the irreducibility
18	$q(a(x), y, q'(x_1, y_1, z_1)) = 0$	$(1, q', q'')$	$n_1 = 0$
19	$q(x, y, q'(a(x_1), y_1, z_1)) = 0$	$(1, q', q'')$	$n_2 = 0$
20	$q(x, y, q'(x_1, y_1, z_1)) = q(x, b(y_1), q'(x_1, y_1, z_1))$	$(1, q', q'')$	$m_2 \geq m_1$
21	$q(x, y, q'(x_1, y_1, z_1)) = q(x, y, q'(x_1, b(y), z_1))$	$(1, q', q'')$	$m_1 \geq m_2$
22	$q(0, y, q'(x_1, y_1, z_1)) = 0$	$(1, q'', q')$	$n_1 > 0$
23	$q(x, y, q'(x_1, y_1, z_1)) = q(b(a(x_1)), y, q'(x_1, y_1, z_1))$	$(1, q'', q')$	$n_2 + 1 \geq n_1$
24	$q(a(x), y, q'(x_1, y_1, z_1)) = q(a(x), y, q'(b(x), y_1, z_1))$	$(1, q'', q')$	$n_1 - 1 \geq n_2$ $\vee n_1 = 0$
25	$q(x, y, q'(x_1, y_1, z_1)) = q(x, b(y_1), q'(x_1, y_1, z_1))$	$(1, q'', q')$	$m_2 \geq m_1$
26	$q(x, y, q'(x_1, y_1, z_1)) = q(x, y, q'(x_1, b(y), z_1))$	$(1, q'', q')$	$m_1 \geq m_2$
27	$q(x, a(y), q'(x_1, y_1, z_1)) = 0$	$(2, q', q'')$	$m_1 = 0$
28	$q(x, y, q'(x_1, a(y_1), z_1)) = 0$	$(2, q', q'')$	$m_2 = 0$
29	$q(x, y, q'(x_1, y_1, z_1)) = q(b(x_1), y, q'(x_1, y_1, z_1))$	$(2, q', q'')$	$n_2 \geq n_1$
30	$q(x, y, q'(x_1, y_1, z_1)) = q(x, y, q'(b(x), y_1, z_1))$	$(2, q', q'')$	$n_1 \geq n_2$
31	$q(x, 0, q'(x_1, y_1, z_1)) = 0$	$(2, q'', q')$	$m_1 > 0$
32	$q(x, y, q'(x_1, y_1, z_1)) = q(x, b(a(y_1)), q'(x_1, y_1, z_1))$	$(2, q'', q')$	$m_2 + 1 \geq m_1$
33	$q(x, a(y), q'(x_1, y_1, z_1)) = q(x, a(y), q'(x_1, b(y), z_1))$	$(2, q'', q')$	$m_1 - 1 \geq m_2$ $\vee m_1 = 0$
34	$q(x, y, q'(x_1, y_1, z_1)) = q(b(x_1), y, q'(x_1, y_1, z_1))$	$(2, q'', q')$	$n_2 \geq n_1$
35	$q(x, y, q'(x_1, y_1, z_1)) = q(x, y, q'(b(x), y_1, z_1))$	$(2, q'', q')$	$n_1 \geq n_2$

For instance, if $\Delta(q) = (1, q', q'')$, the equations (18) and (22) force a transition to q' when the first counter is 0 and a transition to q'' when the counter is not 0. Then the new values of the counters are forced by equations (19), (20), (21) (in case of a transition to q') and (23–26) in case of a transition to q''.

In the end, we get what we were looking for: irreducible terms of S_1 are those in which every pattern $q(a^{n_1}(0), a^{m_1}(0), q'(a^{n_2}(0), a^{m_2}(0), t))$ corresponds to a move $(n_1, m_1, q) \vdash (n_2, m_2, q')$ of the machine. $\square$

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.

Regarding the complexity of our confluence test, it is not clear how to derive interesting bounds from our proof, since the finiteness of confluence trees is obtained from Kruskal's theorem.

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

Also a related problem is to decide redundancy of critical pairs during completion, i.e. given a rewrite system and a pair of terms, to test whether the pair converges via ordered rewriting. This problem might be undecidable since instantiating extra variables by the minimal constant 0 is not sufficient in this case to show existence (or absence) of joinability proofs. For example, the rule $f(x) \to g(y)$ generates the critical pair $g(x) = g(y)$. But if we do not have this critical pair yet, then another critical pair $f(a) = g(a)$ is joinable with the rule if the extra variable is instantiated with a, and not if it is instantiated with 0. It may me possible to adapt the reduction used for constrained equations to show undecidability.

Finally H. Ganzinger has pointed to us a related *ordered resolution* problem that might be undecidable too (even if we have only Horn clauses with at most two literals): given an LPO and a set of clauses, decide whether the set is saturated up to redundancy ?

Acknowledgment

We thank R. Treinen and H. Ganzinger for their comments on an earlier version of the paper.

REFERENCES

Leo Bachmair. Proof by consistency in equational theories. In *Proceedings, Third Annual Symposium on Logic in Computer Science*, pages 228–233, Edinburgh, Scotland, 5–8 July 1988. IEEE Computer Society.

Leo Bachmair and Nachum Dershowitz. Equational inference, canonical proofs, and proof orderings. *J. of the Association for Computing Machinery*, 41(2):236–276, February 1994.

Leo Bachmair, Nachum Dershowitz, and Jieh Hsiang. Orderings for equational proofs. In *First IEEE Symposium on Logic in Computer Science (LICS)*, pages 346–357, Cambridge, Massachusetts, USA, June 16–18, 1986. IEEE Computer Society Press.

Leo Bachmair, Nachum Dershowitz, and David Plaisted. Completion Whitout Failure. In Hassan Aït-Kaci and Maurice Nivat, editors, *Resolution of Equations in Algebraic Structures*, volume 2: Rewriting Techniques, chapter 1, pages 1–30. Academic Press, New York, 1989.

Hubert Comon. Solving symbolic ordering constraints. *International Journal of Foundations of Computer Science*, 1(4):387–411, 1990.

H. Comon and R. Treinen. The first-order theory of lexicographic path orderings is undecidable. *Theoretical Computer Science*, 176(1-2):67–87, April 1997.

Max Dauchet. Simulation of Turing machines by a left-linear rewrite rule. In N. Dershowitz, editor, *Rewriting Techniques and Applications, 3th International Conference*, LNCS, pages 109–120. Springer-Verlag, 1989.

Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.

Nachum Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69–116, 1987.

N. Dershowitz, J-P. Jouannaud, and J-W. Klop. More problems in rewriting. In C. Kirchner, editor, *5th International Conference on Rewriting Techniques and Applications (RTA)*, LNCS 690, pages 468–487, Montreal, Canada, 1993.

Nachum Dershowitz and Zohar Manna. Proving termnation with multiset orderings. *Comm. of ACM*, 22(8), 1979.

Nachum Dershowitz and David Plaisted. Rewriting. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers and MIT Press (to appear), 2001.

Harald Ganzinger, Robert Nieuwenhuis, and Pilar Nivela. The Saturate System, 2001. Software and documentation available at: `http://www.mpi-sb.mpg.de/SATURATE/Saturate.html`.

J. Hsiang and M. Rusinowitch. On word problems in equational theories. In T. Ottmann, editor, *Proc. 14th Int. Colloquium Automata, Languages and Programming*, LNCS 267, pages 54–71, Berlin, Germany, 1987. Springer-Verlag.

Jean-Pierre Jouannaud and Emmanuel Kounalis. Automatic proofs by induction in equational theories without constructors. In *Proceedings, Symposium on Logic in Computer Science*, pages 358–366, Cambridge, Massachusetts, 16–18 June 1986. IEEE Computer Society.

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.

Claude Kirchner, Hélène Kirchner, and Michaël Rusinowitch. Deduction with symbolic constraints. *Revue Française d'Intelligence Artificielle*, 4(3):9–52, 1990.

Deepak Kapur and David R. Musser. Proof by consistency. *Artificial Intelligence*, 31(2):125–157, February 1987.

Deepak Kapur, Paliath Narendran, and Friedrich Otto. On ground confluence of term rewriting systems. *Inform. Comput.*, 86(1):14–31, 1990.

Deepak Kapur, Paliath Narendran, and G. Sivakumar. A path ordering for proving termination for term rewriting systems. In *Proc. of 10th Colloquium on Trees in Algebra and Programming*, LNCS 185, pages 173–185, Germany, 1985. Springer-Verlag.

Pierre Lescanne. On the recursive decomposition ordering with lexicographical status and other related orderings. *Journal of Automated Reasoning*, 6(1):39–49, 1990.

Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall, London, 1 edition, 1967.

Ursula Martin and Tobias Nipkow. Ordered rewriting and confluence. In Mark E. Stickel, editor, *10th International Conference on Automated Deduction (CADE)*, LNAI 449, pages 366–380, Kaiserslautern, FRG, July 24–27, 1990. Springer-Verlag.

Robert Nieuwenhuis. Simple LPO constraint solving methods. *Information Processing Letters*, 47:65–69, August 1993.

Pilar Nivela and Robert Nieuwenhuis. Practical results on the saturation of full first-order clauses: Experiments with the saturate system. (system description). In C. Kirchner, editor, *5th International Conference on Rewriting Techniques and Applications (RTA)*, LNCS 690, pages 436–440, Montreal, Canada, June 16–18, 1993. Springer-Verlag.

Robert Nieuwenhuis and Albert Rubio. Theorem Proving with Ordering and Equality Constrained Clauses. *Journal of Symbolic Computation*, 19(4):321–351, April 1995.

Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers and MIT Press(to appear), 2001.

Gerald E. Peterson. Complete sets of reductions with constraints. In Mark E. Stickel, editor, *10th International Conference on Automated Deduction (CADE)*, LNAI 449, pages 381–395, Kaiserslautern, FRG, July 24–27, 1990. Springer-Verlag.

David A. Plaisted. Semantic confluence tests and completion methods. *Information and Control*, 65(2/3):182–215, May/June 1985.

Received April 2001; revised September 2001; accepted September 2001