MAX-PLANCK-INSTITUT FÜR INFORMATIK

A new ordering constraint solving method and its applications

Robert Nieuwenhuis

MPI–I–92–238 February 1993

Im Stadtwald
66123 Saarbrücken
Germany

A new ordering constraint solving method and its applications

Robert Nieuwenhuis

MPI–I–92–238 February 1993

A new ordering constraint solving method and its applications

Robert Nieuwenhuis*

February 3, 1993

Abstract

We show that it is possible to transform any given LPO ordering constraint C into a finite equivalent set of constraints S for which a special kind of solutions can be obtained. This allows to compute the equalities that follow from ordering constraints, and to decide e.g. whether an *ordering constrained equation* is a tautology. Another application we develop here is a method to check ordered rewrite systems for (ground) confluence.

*This work has been done during a half-year stay at the Max-Planck-Institut für Informatik, Im Stadtwald, D-W-6600 Saarbrücken, Germany. Author's Permanent address: Technical University of Catalonia, Pau Gargallo 5, 08028 Barcelona, Spain. E-mail: roberto@lsi.upc.es.

1 Introduction

It is well-known that rewriting and completion techniques, parameterized by some well-founded ordering, are one of the most successful approaches for reasoning with equations and more general equational formulae, cf. e.g. [DJ90, BG91], and a basic tool for solving (dis)unification problems [CHJ92].

The *lexicographic path ordering* (LPO) on terms is one of the standard orderings used for these purposes. It is obtained by lifting in a simple way an ordering $\succ_{\mathcal{F}}$ (called a *precedence*) on the function symbols $\mathcal{F}$ to an ordering on terms. If $\succ_{\mathcal{F}}$ is total on $\mathcal{F}$, then the corresponding LPO is a simplification ordering total on $\mathcal{T}(\mathcal{F})$. When necessary, the precedence can be extended to deal with new symbols. Note that totality is a needed requirement for most theorem proving purposes and that other general-purpose orderings like the recursive path ordering (RPO) are not total.

LPO-Ordering constraints are quantifier-free first-order formulae over the binary predicates $\succ$ and $=$, where $\succ$ denotes the LPO ordering and $=$ denotes syntactic equality. The satisfiability of such constraints was proved to be decidable by Comon [Com90], and satisfiability wrt. solutions in extended signatures was proved decidable in [NR92]. In section 4 of this paper we apply and extend both previous results for defining a new constraint solving algorithm which generates a particular kind of solutions and can *extract* the equalities that follow from constraints.

A constrained equations has the form $t = t' \,[\![C]\!]$, where $t = t'$ is an equation and C is an LPO-ordering constraint. Such an equation denotes all instances of $t = t'$ for which C is satisfied. If all these instances are of the form $s = s$ then $t = t' \,[\![C]\!]$ is a tautology. To decide this property is one application of the methods described here (cf. section 5).

Furthermore, in section 6 we define a *constrained* rewriting relation (similar to [KKR90] and [Pet90]) on ordering constrained equations. Suppose we have an equation $u = v$ and a constrained equation $e \,[\![C]\!]$ where $e|_p = u\sigma$. Then $e \,[\![C]\!]$ rewrites into $e[v\sigma]_p \,[\![C \wedge u\sigma \succ v\sigma]\!]$ and into the *complementary* equation $e \,[\![C \wedge u\sigma \not\succ v\sigma]\!]$. We apply such constrained rewrite steps whenever $C \wedge u\sigma \succ v\sigma$ is satisfiable in the sense of [NR92].

We also apply *equality extraction* steps, in which equations are instantiated with the equalities that follow from their constraints. Rewriting in this way produces *constrained rewrite trees,* since every constrained equation rewrites into several new ones. Constrained rewriting with equations having so-called *extra variables* is done in a special way.

Constrained rewriting is applied to checking *ordered rewriting systems* for confluence properties in section 7. An ordered rewrite system is a pair $(E, \succ)$ where E is a set of equations and $\succ$ is a reduction ordering on terms. Ordered rewriting is done by applying equations of E in whatever direction agrees with $\succ$. This allows one to deal with unorientable axioms (like commutativity) since it always terminates.

It generalizes "classical" rewriting with oriented rewrite rules in the sense that if $E = \{s_1 = t_1, \ldots, s_n = t_n\}$ and $s_i \succ t_i$ for $i = 1 \ldots n$, then ordered rewriting with $(E, \succ)$ is equivalent to rewriting with $R = \{s_1 \to t_1, \ldots, s_n \to t_n\}$. The standard choice for $\succ$ is

the lexicographic path ordering (LPO), since (ground) confluence requires $\succ$ to be total on ground terms.

We prove the following results, which together provide our confluence test for ordered rewrite systems:

1. constrained rewrite trees are finite.
2. a critical pair $t = t'$ between the equations of ordered rewrite systems is joinable if all leafs of a rewrite tree with root $t = t'$ are tautologies (the paths are ordered rewrite proofs for *all* possible instances of the critical pair at the root).

Many times a non-tautology leaf of the tree is a counterexample to the confluence of E. We show that this is the case if the constraint solving algorithm provides an irreducible solution, since for the corresponding instance $s = s'$ of the leaf s and s' are different E-equivalent terms.

2 Acknowledgements

The author wishes to thank Leo Bachmair, Harald Ganzinger and Wayne Snyder (who suggested applying Kruskal's theorem for lemma 6.1) for a very helpful joint discussion.

3 Basic terminology

We assume the reader to be familiar with the basic concepts of term rewrite systems, as described e.g. by Dershowitz and Jouannaud [DJ90], whose notation we follow.

Along this paper we will suppose that we are given a set of function symbols $\mathcal{F}$ and a total ordering $\succ_{\mathcal{F}}$ over $\mathcal{F}$ (the *precedence)*. We sometimes write pairs $(\mathcal{F}, \succ_{\mathcal{F}})$ or $(E, \succ_{\mathcal{F}})$, where E is a set of equations. The LPO ordering generated by $\succ_{\mathcal{F}}$, denoted $\succ^{\mathcal{F}}_{lpo}$, is a total simplification ordering on $\mathcal{T}(\mathcal{F})$. It is defined as follows: $s = f(s_1, \ldots, s_m) \succ^{\mathcal{F}}_{lpo} g(t_1, \ldots, t_n) = t$ if

1. $s_i \succeq^{\mathcal{F}}_{lpo} t$, for some i with $1 \leq i \leq m$ or
2. $f \succ_{\mathcal{F}} g$, and $s \succ^{\mathcal{F}}_{lpo} t_j$, for all j with $1 \leq j \leq n$ or
3. $f = g$, $(s_1, \ldots, s_m) \gg^{\mathcal{F}}_{lpo} (t_1, \ldots, t_n)$, and $s \succ^{\mathcal{F}}_{lpo} t_j$, for all j with $1 \leq j \leq n$

where $(s_1, \ldots, s_n) \gg^{\mathcal{F}}_{lpo} (t_1, \ldots, t_n)$ if $\exists j \leq n$ s.t. $\forall i < j \;\; s_i = t_i$ and $s_j \succ^{\mathcal{F}}_{lpo} t_j$.

By an *extension* $(\mathcal{F}', \succ_{\mathcal{F}'})$ of $(\mathcal{F}, \succ_{\mathcal{F}})$ we mean a set of function symbols $\mathcal{F}'$ such that $\mathcal{F}' \supseteq \mathcal{F}$ and a total precedence $\succ_{\mathcal{F}'}$ extending $\succ_{\mathcal{F}}$.

An *LPO-ordering constraint* is a quantifier-free first-order formula built over the binary predicate symbols '$\succ$' and '$=$' relating terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$, where '$=$' denotes syntactic equality of terms, and '$\succ$' denotes a lexicographic path ordering.

A *solution* in $(\mathcal{F}', \succ_{\mathcal{F}'})$ of a constraint C is a substitution σ with range $\mathcal{T}(\mathcal{F}')$ and whose domain is a set of variables containing the variables of C, such that $C\sigma$ evaluates to true under $\succ_{lpo}^{\mathcal{F}'}$ in the usual sense. Then we say that C *satisfies* σ in $(\mathcal{F}', \succ_{\mathcal{F}'})$. We will use the symbols $\top$ (resp. $\bot$) to denote the constraint that satisfies all (resp. no) σ. Cf. Comon's paper [Com90] for more details and a decision procedure for satisfiability in $(\mathcal{F}, \succ_{\mathcal{F}})$ itself.

In the following, if $(\mathcal{F}, \succ_{\mathcal{F}})$ contains a smallest constant symbol (which we will denote by 0), then we will denote by $(\mathcal{F}_0, \succ_{\mathcal{F}_0})$ the particular extension of $(\mathcal{F}, \succ_{\mathcal{F}})$ s.t. $\mathcal{F}_0$ is $\mathcal{F} \cup \{f\}$ for some unary function symbol f that does not belong to $\mathcal{F}$ and where $\succ_{\mathcal{F}_0}$ is the extension of $\succ_{\mathcal{F}}$ where $g \succ_{\mathcal{F}_0} f$ for every symbol g in $\mathcal{F}$. If $\mathcal{F}$ does not contain any constant symbol, then $(\mathcal{F}_0, \succ_{\mathcal{F}_0})$ will be $\mathcal{F} \cup \{f, 0\}$ for some new constant 0 and then $g \succ_{\mathcal{F}_0} f \succ_{\mathcal{F}_0} 0$ for every symbol g in $\mathcal{F}$.

Furthermore, in the following, we will call a constraint C *satisfiable* if there exists some extension $(\mathcal{F}', \succ_{\mathcal{F}'})$ of $(\mathcal{F}, \succ_{\mathcal{F}})$ in which C is satisfiable. This notion of satisfiability is the one needed for refutation completeness of inference systems for ordering constrained clauses, studied in [NR92], where it is also shown that this notion is equivalent to satisfiability in $(\mathcal{F}_0, \succ_{\mathcal{F}_0})$.

Below we denote variables by $x, y \ldots$ and terms (with variables) by s and t. Every ordering constraint can be expressed by an equivalent set (disjunction) of *solved forms* [Com90], by keeping it in disjunctive normal form, eliminating negations with $(t \not\succ t') \equiv (t' \succ t \vee t = t')$ and $(t \neq t') \equiv (t' \succ t \vee t \succ t')$, and applying the definition of LPO for decomposing all inequalities $s \succ s'$ with non-variable s and s', like by

$$s \succ f(t_1, \ldots, t_n) \implies s \succ t_1 \wedge \ldots \wedge s \succ t_n \quad \text{if } top(s) \succ_{\mathcal{F}} f$$

A *solved form* F is either $\top$, $\bot$ or a formula

$$x_1 \succ t_1 \wedge \ldots \wedge x_n \succ t_n \quad \wedge \quad t'_1 \succ x'_1 \wedge \ldots \wedge t'_m \succ x'_m \quad \wedge \quad y_1 = s_1 \wedge \ldots \wedge y_r = s_r$$

where y_k appears only once in F for $k = 1 \ldots r$.

The solutions of a solved form F are composed by the equality part $eqpart(F) = (y_1 = s_1 \wedge \ldots \wedge y_r = s_r)$ (the values of the already *solved* variables) and the solutions of the inequality part $ineqpart(F)$ (which do not depend on $eqpart(F)$). A constraint C has the same solutions as its set (disjunction) of solved forms.

A solved form F can be transformed into a finite set (disjunction) of *simple systems* [Com90] which again has the same solutions. A *simple system* S is a formula

$$t_1 \succ \ldots \succ t_n \quad \wedge \quad y_1 = s_1 \wedge \ldots \wedge y_r = s_r$$

where y_k appears only once in S for $k = 1 \ldots r$, and for $1 \leq i \leq n$ every subterm of t_i is some t_j with $i \leq j \leq n$.

Like in solved forms, the solutions of a simple system S are composed by the values of the solved variables $eqpart(S) = (y_1 = s_1 \wedge \ldots \wedge y_r = s_r)$ and the solutions of the inequality part $ineqpart(S) = (t_1 \succ \ldots \succ t_n)$ (which do not depend on $eqpart(S)$), and S is satisfiable iff $ineqpart(S)$ is.

Lemma 3.1 (cf. [Com90]). Let S be a simple system. If *ineqpart*(S) does not contain any of its solved forms then its only solved form is $\bot$, and *ineqpart*(S) is unsatisfiable.

This lemma follows from the fact that solved forms are computed by decomposing inequalities into a combination F of (in)-equalities between subterms of it. Since *ineqpart*(S) contains inequalities between all its (sub)terms, *ineqpart*(S) must be either incompatible with such F, or else contain F; if it is incompatible with all such F then its only solved form is $\bot$.

4 Computing particular solutions of ordering constraints

An $\mathcal{NF}_0$-*solution* (normalized $\mathcal{F}_0$-solution) of a simple system S is a solution σ in $(\mathcal{F}_0, \succ_{\mathcal{F}_0})$, in which, for each pair of different variables x and y of *ineqpart*(S), $x\sigma$ is of the form $f(\ldots^{k)} f(t)\ldots)$, $y\sigma$ is of the form $f(\ldots^{k'} f(t')\ldots)$, where $k \neq k'$, and the topmost symbols of t and t' are different from f.

Theorem 4.1 Let S be a simple system. Then the following statements are equivalent:

1. S is satisfiable
2. *ineqpart*(S) contains one of the solved forms of *ineqpart*(S)
3. S has an $\mathcal{NF}_0$-solution

Proof The implication 1. $\Longrightarrow$ 2. is the previous lemma. Moreover, 3. trivially implies 1. We now prove that 1. and 2. imply 3. by induction on the number k of variables in $T = \textit{ineqpart}(S) = t_1 \succ \ldots \succ t_n$.

Note that, by definition of simple system and because T is satisfiable, every variable of T appears exactly once as some t_i in T. Let $x_k, x_{k-1}, \ldots, x_1$ be the variables appearing in this way from left to right in T, i.e. T is of the form $\ldots \succ x_k \succ \ldots \succ x_{k-1} \succ \ldots \succ x_1 \ldots$ We prove that there exists an $\mathcal{NF}_0$-solution σ in which each $x_j\sigma$ is of the form $f(\ldots^{j)} f(t)\ldots)$, where the topmost symbol of t is different from f.

The case $k = 0$ holds trivially. If $k \neq 0$, then let T' be the expression obtained by deleting from T all terms containing the variable x_k. Now T' is still a simple system. Moreover, T' is satisfiable, since it is contained in T. Therefore T' contains one of its solved forms. It has one variable less than T and by the induction hypothesis T' has an $\mathcal{NF}_0$-solution θ of the desired form.

Let σ be defined as follows. If x_k is t_n then $x_k\sigma$ is $f(0)$. Otherwise, x_k is some t_i with $i \neq n$. Then $x_k\sigma$ is $f(\ldots^{k)} f(t_{i+1}\theta)\ldots)$ if t_{i+1} is not the variable x_{k-1}, and $x_k\sigma$ is $f(t_{i+1}\theta)$ if t_{i+1} is x_{k-1}. Finally, $y\sigma = y\theta$ for all other variables y in T. Below we show that σ is a solution (and therefore the needed $\mathcal{NF}_0$-solution) of T.

Each solution of a solved form of a constraint C is a solution of C, and T contains one of its solved forms. Therefore we only have to prove that σ is solution of the solved part of T, i.e. that $s\sigma \succ^{\mathcal{F}_0}_{lpo} s'\sigma$ for all inequalities $s \succ s'$ in T where s or s' is a variable. If x_k

does not appear in $s \succ s'$ then $s \succ s'$ is in T' and $s\sigma = s\theta \succ^{\mathcal{F}_0}_{lpo} s'\theta = s'\sigma$. The inequalities in which x_k may appear are the following:

1. $x_k \succ t$: By construction of σ, $x_k\sigma = t_i\sigma \succ^{\mathcal{F}_0}_{lpo} t_{i+1}\sigma$. The case when t is to the right of t_{i+1} in T is covered by transitivity and because θ is solution of T'.

2. $s \succ x_k$: If x_k appears in s, then $s\sigma \succ^{\mathcal{F}_0}_{lpo} x_k\sigma$. Otherwise, $s\sigma = s\theta$, and $s\theta$ is some term with a topmost symbol g with $g \succ_{\mathcal{F}_0} f \succ_{\mathcal{F}_0} 0$. ($s$ is not a variable, because x_k is the leftmost variable, nor is s the constant 0, since T is satisfiable). Now if $x_k\sigma$ is $f(0)$ then $s\sigma \succ^{\mathcal{F}_0}_{lpo} x_k\sigma$. Otherwise, $x_k\sigma$ is $f(\ldots f(t_{i+1})\ldots)\theta$, and $s\theta \succ^{\mathcal{F}_0}_{lpo} f(\ldots f(t_{i+1})\ldots)\theta$ iff $s\theta \succ^{\mathcal{F}_0}_{lpo} t_{i+1}\theta$ because $g \succ_{\mathcal{F}_0} f$. But $s\theta \succ^{\mathcal{F}_0}_{lpo} t_{i+1}\theta$, since θ is solution of T'.

3. $s[x_k] \succ y$: Here y is some t_j with $j > i$, and $s[x_k]\sigma \succ^{\mathcal{F}_0}_{lpo} x_k\sigma \succ^{\mathcal{F}_0}_{lpo} y\sigma$ by case 1.

□

The previous result (a constraint is satisfiable iff it has a simple system whose solved form is not $\bot$) is related to [NR92], where another method, with a much more complicated proof, is given for finding solutions in $(\mathcal{F}_0, \succ_{\mathcal{F}_0})$.

However, the above technique is not only interesting because of its simplicity, but also for the special $\mathcal{NF}_0$-solutions, which are crucial for the results of the following section.

5 Ordering constrained equations

An *ordering constrained equation* is a pair formed by an equation $t = t'$ and an LPO-ordering constraint C, written $t = t' \,[\![C]\!]$, denoting all instances of $t = t'$ for which C is satisfied. If all these instances are of the form $s = s$ then $t = t' \,[\![C]\!]$ is a tautology.

Lemma 5.1 It is decidable whether a constrained equation $t = t' \,[\![C]\!]$ is a tautology.

Proof Let $\{S_1, \ldots, S_n\}$ be the simple systems of C, and $\sigma_i = \{y_1 \mapsto s_1, \ldots, y_r \mapsto s_r\}$ if $eqpart(S_i) = (y_1 = s_1 \wedge \ldots \wedge y_r = s_r)$, for $1 \leq i \leq n$. Now $t = t' \,[\![C]\!]$ is equivalent to $EE = \{\ t\sigma_1 = t'\sigma_1 \,[\![ineqpart(S_1)]\!],\ \ldots,\ t\sigma_n = t'\sigma_n \,[\![ineqpart(S_n)]\!]\ \}$. Therefore $t = t' \,[\![C]\!]$ is a tautology iff all constrained equations in EE are tautologies.

We now prove that a constrained equation $s = s' \,[\![T]\!]$ in EE is a tautology iff s and s' are the same term or T is unsatisfiable. The if-part of this statement is obvious. For the only-if part, suppose that $s \neq s'$ and T is satisfiable. By the previous theorem we know that T has an $\mathcal{NF}_0$-solution θ, and $s\theta \neq s'\theta$, because θ cannot be a unifier of s and s': it instantiates different variables with different terms whose topmost symbol f does not appear in $s = s'$. This means that $s = s' \,[\![T]\!]$ is not a tautology.

□

6 Constrained rewriting

A constrained equation $e\,[\![C]\!]$ can be rewritten by *constrained rewriting* with $u = v$ into $e[v\sigma]_p\,[\![C \wedge u\sigma \succ v\sigma]\!]$ and into the *complementary* equation $e\,[\![C \wedge u\sigma \not\succ v\sigma]\!]$ iff

1. $e|_p = u\sigma$ and
2. $C \wedge u\sigma \succ v\sigma$ is satisfiable and
3. $x\sigma = 0$ for every (so-called *extra*) variable x in v that is not contained in u.

By *equality extraction*, $e\,[\![C]\!]$ can be rewritten into
$\{\ e\sigma_1\,[\![ineqpart(S_1)]\!], \ldots, e\sigma_n\,[\![ineqpart(S_n)]\!]\ \}$ iff

1. the set $\{S_1, \ldots, S_n\}$ of *satisfiable* simple systems of C is non-empty and
2. the equality parts $eqpart(S_i)$ are non-empty for $1 \leq i \leq n$ and
3. $\sigma_i = \{y_1 \mapsto s_1, \ldots, y_r \mapsto s_r\}$ if $eqpart(S_i) = (y_1 = s_1 \wedge \ldots \wedge y_r = s_r)$ for $1 \leq i \leq n$.

Note that the set of constrained equations obtained by equality extraction is equivalent to $e\,[\![C]\!]$ (it has the same instances). We instantiate the equation with the equalities that follow from its constraint. Furthermore, each constrained equation obtained contains a strictly smaller number of variables than $e\,[\![C]\!]$.

By equality extraction and constrained rewriting we can compute *confluence trees*. A *confluence tree* (using E) for an equation $t = t'$ is a tree T such that

1. the nodes of T are constrained equations
2. the root of T is $t = t'\,[\![\top]\!]$
3. the children of each inner node $e\,[\![C]\!]$ are the constrained equations obtained by one step of constrained rewriting on $e\,[\![C]\!]$ with an equation in E, or by one step of equality extraction on $e\,[\![C]\!]$.
4. no leaf of T can be rewritten by constrained rewriting with E or by equality extraction.

Lemma 6.1 There is no infinite confluence tree using (a finite) E.

Proof The number of children of each inner node is finite (two in the case of constrained rewrite steps, and n in equality extraction steps, where n is the finite number of satisfiable solved forms of a constraint C).

We now derive a contradiction from the existence of some infinite branch B. Let B be an infinite sequence of constrained equations of the form $e_1\,[\![C_1]\!]$, $e_2\,[\![C_2]\!], \ldots$ where each constrained equation $e_{i+1}\,[\![C_{i+1}]\!]$ is obtained from $e_i\,[\![C_i]\!]$ by constrained rewriting or by equality extraction, and where no C_j is insatisfiable.

First, note that there can only be a finite number of equality extraction steps in B, because they strictly reduce the number of variables, and in constrained rewriting steps the number of variables does not increase. This means that in B there must be an infinite contiguous subsequence CR of only constrained rewrite steps.

Second, in CR there is no infinite contiguous subsequence of only complementary steps $e\,[\![C_k]\!]$, $e\,[\![C_{k+1}]\!], \ldots$ since the number of possible applications of equations to a finite e is finite, and no equation can be applied twice at the same position (then the non-complementary constraint becomes insatisfiable).

This means that, if we omit in CR the constrained equations obtained by complementary steps, we get an infinite sequence B' of the form $e'_1\,[\![C'_1]\!]$, $e'_2\,[\![C'_2]\!], \ldots$ where each $e'_{i+1}\,[\![C'_{i+1}]\!]$ is obtained from $e'_i\,[\![C'_i]\!]$ by a non-complementary constrained rewrite step (and by possibly adding some more conditions to C'_{i+1} corresponding to omitted complementary steps in between).

In B', if $j > i$, then $C'_j \wedge e'_j \succeq e'_i$ is insatisfiable, because in each step on $e'_i\,[\![C'_i]\!]$ obtaining $e'_{i+1}\,[\![C'_{i+1}]\!]$ a condition (equivalent to) $e'_i \succ e'_{i+1}$ is added to the constraint.

Furthermore, by Kruskal's theorem, for some node $e'_i\,[\![C'_i]\!]$ in B' the equation e'_i must be *embedded* in the equation e'_j of some node $e'_j\,[\![C'_j]\!]$ with $j > i$, since all terms appearing in the tree are built over a finite set of symbols. Then also $e'_j \succeq^{\mathcal{F}_0}_{lpo} e'_i$, because all simplification orderings contain the embedding relation.

But $C'_j \wedge e'_j \succeq e'_i$ is insatisfiable, and $C'_j \wedge e'_j \succeq e'_i$ is equivalent to C'_j since $e'_j \succeq^{\mathcal{F}_0}_{lpo} e'_i$, i.e. C'_j is also insatisfiable, which contradicts the initial assumptions. □

7 A confluence test for ordered rewrite systems

Rewrite methods have to be adapted in those cases in which any orientation of the axioms yields a nonterminating system, like in the presence of commutativity. *Ordered rewriting and completion* techniques overcome this problem. An *ordered rewrite system* is a pair formed by a set of equations E and a reduction ordering $\succ$ on terms. Ordered rewriting is done by applying an equation in whatever direction agrees with the given ordering, and therefore always terminates. Let $u = v$ (or $v = u$) be an equation in E, and let t be a term with $t|_p = u\sigma$ and $u\sigma \succ v\sigma$, for some substitution σ. Then t rewrites by ordered rewriting with $(E, \succ)$ into $t[v\sigma]_p$, denoted $t \to_{(E,\succ)} t[v\sigma]_p$ or simply $t \to_E t[v\sigma]_p$.

For instance, with $x + y = y + x$ the term $a + b$ rewrites into $b + a$ only if $a + b \succ b + a$. For example

$$\begin{aligned} x + y &= y + x \\ (x + y) + z &= x + (y + z) \\ x + (y + z) &= y + (x + z) \end{aligned}$$

is a confluent system for associativity and commutativity (if $\succ$ treats $+$ lexicographically). This means that terms like $x + (g(a) + b)$ and $(b + x) + g(a)$ are AC-equivalent for all x iff their (unique) normal forms by ordered rewriting with $(E, \succ)$ are equal. However, first $\succ$ has to be extended to some $\succ'$ that is able to deal with the new symbols g, a and b, and

also with variables like x (which are in fact treated as Skolem constants). Moreover, $\succ'$ must be *total* on all terms of the extended signature[1] and therefore the standard choice for $\succ$ is LPO.

So, given $(E, \succ)$, the required property for this purpose is the confluence of $(E, \succ')$ for all total extensions $\succ'$ of $\succ$.

There exist *unfailing* completion methods that generate a finite confluent ordered rewrite system for a given input $(E, \succ)$ whenever such a system exists (if $\succ$ is total on E-equivalent terms) [Bac87]. However, it is not always easy to check whether such a system has been obtained.

For terminating systems of oriented rewrite rules, a well-known result by Knuth and Bendix [KB70] states that such a system R is confluent if and only if all critical pairs between its rules are joinable by rewriting with R. In this paper we show that, surprisingly, it is possible do something similar for ordered rewrite systems: find ordered rewrite proofs for *all* instances (possibly with new symbols) of each critical pair.

A binary relation $\rightarrow$ on any set T is *confluent* if the relation $\leftarrow^* \circ \rightarrow^*$ is contained in the joinability relation $\rightarrow^* \circ \leftarrow^*$.

In the following, for a given $(E, \succ_{\mathcal{F}})$, by *confluence* of $(E, \succ_{\mathcal{F}})$ we mean the confluence, for every extension $(\mathcal{F}', \succ_{\mathcal{F}'})$ of $(\mathcal{F}, \succ_{\mathcal{F}})$, of the ordered rewrite relation $\rightarrow_{(E, \succ^{\mathcal{F}'}_{lpo})}$ on $\mathcal{T}(\mathcal{F}')$.

By *ground confluence* of $(E, \succ_{\mathcal{F}})$ we mean the confluence of the ordered rewrite relation $\rightarrow_{(E, \succ^{\mathcal{F}}_{lpo})}$ on $\mathcal{T}(\mathcal{F})$.

In the following we will focus on general confluence, and not on ground confluence, since the latter property follows from the former, i.e. every confluent ordered rewrite system is ground confluent (the inverse implication is not true). The following lemma is well-known:

Lemma 7.1 $(E, \succ_{\mathcal{F}})$ is confluent iff for every instance $t = t'$ of a critical pair between equations in E and for every extension $(\mathcal{F}', \succ_{\mathcal{F}'})$ of $(\mathcal{F}, \succ_{\mathcal{F}})$ such that t and t' are in $\mathcal{T}(\mathcal{F}')$, $t = t'$ is joinable by ordered rewriting with $(E, \succ^{\mathcal{F}'}_{lpo})$.
Similarly, $(E, \succ_{\mathcal{F}})$ is ground confluent iff for every ground instance (over the given signature) $t = t'$ of a critical pair between equations in E, $t = t'$ is joinable by ordered rewriting with $(E, \succ^{\mathcal{F}}_{lpo})$.

Lemma 7.2 Let $t = t'$ be an equation in $\mathcal{T}(\mathcal{F}, \mathcal{X})$, and let T be a confluence tree (using E) for $t = t'$. Then, for every instance $t\sigma = t'\sigma$ of $t = t'$ and every extension $(\mathcal{F}', \succ_{\mathcal{F}'})$ of $(\mathcal{F}, \succ_{\mathcal{F}})$ with $t\sigma$ and $t'\sigma$ in $\mathcal{T}(\mathcal{F}', \mathcal{X})$, the instance $t\sigma = t'\sigma$ can be rewritten with $(E, \succ^{\mathcal{F}'}_{lpo})$ into some instance of a leaf of T.

Proof $t\sigma = t'\sigma$ can be rewritten (in 0 steps) into an instance of the root $t = t' [\![\top]\!]$. Therefore, it suffices to prove that if $t\sigma = t'\sigma$ can be rewritten into some instance $s = s'$ of a node $e [\![C]\!]$ then either $e [\![C]\!]$ is a leaf or else $t\sigma = t'\sigma$ can also be rewritten into some instance of one of the children of $e [\![C]\!]$.

[1] Theoretically, $(E, \succ')$ could also be convergent if $\succ'$ were total only on E-equivalent terms, but properties like totality on E-equivalent terms can only be guaranteed by requiring totality on all terms.

Let the children $e'[\![C \wedge u \succ v]\!]$ and $e[\![C \wedge u \not\succ v]\!]$ of $e[\![C]\!]$ be obtained by a constrained rewrite step. If $u\sigma \succ^{\mathcal{F}'}_{lpo} v\sigma$, then $s = s'$ can be rewritten into an instance of $e'[\![C \wedge u \succ v]\!]$. Otherwise $s = s'$ is an instance of $e[\![C \wedge u \not\succ v]\!]$.

If the children of $e[\![C]\!]$ are obtained by an equality extraction step, then $s = s'$ is also an instance of one of the children. □

Lemma 7.3 If there is a confluence tree for $t = t'$ using E in which all leafs are tautologies then, for every instance $t\sigma = t'\sigma$ of $t = t'$ and for every extension $(\mathcal{F}', \succ_{\mathcal{F}'})$ of $(\mathcal{F}, \succ_{\mathcal{F}})$ with $t\sigma$ and $t'\sigma$ in $\mathcal{T}(\mathcal{F}', \mathcal{X})$, the instance $t\sigma = t'\sigma$ is joinable wrt. $(E, \succ^{\mathcal{F}'}_{lpo})$.

Proof By the previous lemma, for every σ the equation $t\sigma = t'\sigma$ can be reduced by ordered rewriting with $(E, \succ^{\mathcal{F}'}_{lpo})$ into some instance of a leaf of the tree. But if all leafs have only instances of the form $s = s$, then $t\sigma = t'\sigma$ is joinable. □

The previous lemma provides a confluence test for ordered rewrite systems: if for each one of its critical pairs there exists a constrained rewrite tree with only tautologies as leafs, then the rewrite system is confluent. The lemma below states that in may cases a constrained rewrite tree with a non-tautology leaf in fact provides a counter-example to the confluence of E (the gap for obtaining decidability of confluence lies in that it is not known whether constraints can be manipulated in such a way that irreducible $\mathcal{NF}_0$-solutions can always be obtained):

Lemma 7.4 If $E \models t = t'$ and there is an confluence tree for $t = t'$ with a non-tautology leaf $e[\![C]\!]$ with an irreducible (wrt. $(E, \succ_{\mathcal{F}})$) $\mathcal{NF}_0$-solution for C, then $(E, \succ_{\mathcal{F}})$ is not confluent.

Proof Let $s = s'[\![C]\!]$ be a non-tautology leaf of the confluence tree, and let $\{S_1, \ldots, S_n\}$ be the set of *satisfiable* simple systems of C (this set is non-empty as $s = s'[\![C]\!]$ is not a tautology). Since $s = s'[\![C]\!]$ is in normal form wrt. equality extraction, there must be some i with $1 \leq i \leq n$ such that the equality part $eqpart(S_i)$ is empty. By the theorem XX we know that $ineqpart(S_i)$ (which is equal to S_i) has an $\mathcal{NF}_0$-solution θ, i.e. $s\theta = s'\theta$ is an instance in $\mathcal{T}(\mathcal{F}_0)$ of $s = s'[\![C]\!]$.

We now prove that $s\theta = s'theta$ is not joinable (in fact, it is irreducible) by ordered rewriting with $(E, \succ^{\mathcal{F}_0}_{lpo})$. This implies that $(E, \succ_{\mathcal{F}})$ is not confluent, because, by construction of the confluence tree, $E \models s\theta = s'\theta$.

First, note that $s\theta \neq s'\theta$, because θ cannot be a unifier of s and s': it instantiates different variables with different terms whose topmost symbol f does not appear in $s = s'$.

It remains show that $s\theta = s'\theta$ is also irreducible wrt. $(E, \succ^{\mathcal{F}_0}_{lpo})$ at non-variable positions of $s = s'$, i.e. at positions not inside the substitution *theta*: it would mean $(s = s')|_p = u\sigma$ for some σ, and $(s\theta = s'\theta)|_p = u\sigma\theta$. But the constraint $C \wedge u\sigma \succ v\sigma$ is insatisfiable, since $s = s'[\![C]\!]$ is in normal form wrt. constrained rewriting, and therefore $u\sigma\theta \not\succ^{\mathcal{F}_0}_{lpo} v\sigma\theta$, since θ is a solution of C.

□

Let us briefly explain why our treatment of extra variables works. Recall that the extra variables in a step applying an equation $u = v$ are the variables of the right hand

side v that do not belong to u. In such steps, the instantiation of these variables is not determined by the matching instantiation of u, i.e. one has to "guess" their value.

For computing normal forms with a confluent E, in practice one can always choose to instantiate the extra variables with the smallest constant 0: if $u = v$ can somehow be applied reductively, then it can also be applied reductively instantiating the extra variables with 0 (0 is the smallest constant, and therefore the smallest ground term) and therefore the same normal forms are computed in this way.

When building confluence trees we also instantiate extra variables with the smallest constant 0: if there is a non-tautology leaf, then E cannot be confluent; if all leafs are tautologies, then we have found a rewrite proof for all instances of the critical pair.

References

[Bac87] Leo Bachmair. Proof methods for equational theories, 1987.

[BG91] Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. Technical Report MPI-I-91-208, Max-Planck-Institut für Informatik, Saarbrücken, August 1991.

[CHJ92] Hubert Comon, Marianne Haberstrau, and Jean-Pierre Jouannaud. Decidable problems in shallow equational theories (extended abstract). In *Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 255–265, Santa Cruz, California, USA, June 22–25, 1992. IEEE Computer Society Press.

[Com90] Hubert Comon. Solving inequations in term algebras (extended abstract). In *Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 62–69, Philadelphia, Pennsylvania, USA, June 4–7, 1990. IEEE Computer Society Press, Los Alamitos, CA, USA.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 6, pages 244–320. Elsevier Science Publishers B.V., Amsterdam, New York, Oxford, Tokyo, 1990.

[KB70] 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.

[KKR90] Claude Kirchner, Hélène Kirchner, and Michaël Rusinowitch. Deduction with symbolic constraints. Draft, May 23, 1990.

[NR92] Robert Nieuwenhuis and Albert Rubio. Theorem proving with ordering constrained clauses. In Deepak Kapur, editor, *11th International Conference on Automated Deduction*, LNAI 607, pages 477–491, Saratoga Springs, New York, USA, June 15–18, 1992. Springer-Verlag.

[Pet90] Gerald E. Peterson. Complete sets of reductions with constraints. In Mark E. Stickel, editor, *10th International Conference on Automated Deduction*, LNAI 449, pages 381–395, Kaiserslautern, FRG, July 24–27, 1990. Springer-Verlag.