Solved Forms for Path Ordering Constraints

Robert Nieuwenhuis and José Miguel Rivero*

Technical University of Catalonia, Dept. LSI,
Jordi Girona 1, 08034 Barcelona, Spain
roberto@lsi.upc.es rivero@lsi.upc.es

Abstract. A usual technique in symbolic constraint solving is to apply transformation rules until a *solved form* is reached for which the problem becomes simple. Ordering constraints are well-known to be reducible to (a disjunction of) solved forms, but unfortunately no polynomial algorithm deciding the satisfiability of these solved forms is known.
Here we deal with a different notion of solved form, where fundamental properties of orderings like transitivity and monotonicity are taken into account. This leads to a new family of constraint solving algorithms for the full recursive path ordering with status (RPOS), and hence as well for other path orderings like LPO, MPO, KNS and RDO, and for all possible total precedences and signatures. Apart from simplicity and elegance from the theoretical point of view, the main contribution of these algorithms is on efficiency in practice. Since guessing is minimized, and, in particular, no linear orderings between the subterms are guessed, a practical improvement in performance of several orders of magnitude over previous algorithms is obtained, as shown by our experiments.

1 Introduction

An ordering constraint is a quantifier-free first-order formula built over the binary predicate symbols '$>$' and '$=$' which, respectively, denote a given path ordering $\succ$ and a congruence $\equiv$ on ground terms. A *solution* of a constraint C is a ground substitution σ such that $C\sigma$ evaluates to true under the given $\succ$ and $\equiv$. If C has a solution it is called satisfiable. Such constraints have many interesting applications like pruning the search space in automated theorem proving [KKR90,NR95] or deciding the confluence of ordered rewrite systems [CNNR98]. They also provide powerful decidable constraint-based termination orderings $\succ_c$ for term rewriting, defined $s \succ_c t$ if $s\sigma \succ t\sigma$ for all ground σ. If $\succ$ is the lexicographic path ordering (LPO) or the recursive path ordering with status (RPOS), such $\succ_c$ subsume other path orderings like the ones of [KNS85,Les90] since all these path orderings coincide on ground terms (see [Der87]). For example, if s is $g(f(x), f(y))$ and t is $g(g(x, y), g(x, y))$, and $f \succ_{\mathcal{F}} g$ in the precedence, then $s \not\succ_{rpo} t$, but $s \succ_c t$.

The first practical applications of ordering constraints gave rise to the distinction between *fixed signature* semantics (solutions are built over a given signature

* Both authors partially supported by the ESPRIT Basic Research WG CCL-II.

$\mathcal{F}$) and *extended signature* semantics (new symbols are allowed to appear in solutions) [NR95]. The satisfiability problem for ordering constraints was first shown decidable for fixed signatures when $\succ$ is a total LPO [Com90] or a total RPOS [JO91]. For extended signatures, decidability was shown for LPO in [NR95] and for RPO in [Nie93]. Regarding complexity, NP algorithms for LPO (fixed and extended signatures) and RPO (extended ones) were given in [Nie93]. Very recently, an NP algorithm has been given as well for RPO under fixed signatures in [NRV98]. NP-hardness of the satisfiability problem is known, even for one single inequation, for all these cases [CT94]. All these decision procedures use at some point the fact that a constraint C can be effectively expressed as an equivalent disjunction of expressions $s_1 > t_1 \wedge \ldots \wedge s_n > t_n$, called *solved forms* in [Com90], where for each i always s_i or t_i is a variable.

In algorithms like the ones of [Com90] and [NR95], the computation of solved forms is only a first step that is followed by other exponential phases. This is not surprising, since this notion of solved form only involves a local analysis of the inequalities considered independently. In fact any constraint $s > t$ can trivially be put into the equivalent (under extended sigantures) solved form $s > x \wedge x > t$, for some new variable x. This gives some intuition why this notion of solved form needs to be refined and, in particular, why transitivity through variables needs to be considered.

On the other hand, the NP algorithms of [Nie93] and [NRV98] are not very useful in practice, since they are based on a first very expensive guess of a *simple system* for C, a particular constraint S of the form $s_n \#_n s_{n-1} \#_{n-1} \ldots \#_1 s_0$, where each $\#_i$ is either $=$ or $>$, and $\{s_n, \ldots, s_1\}$ is the set of *all subterms* of C. In [Nie93] it is shown that, roughly, C is satisfiable under extended signatures if, and only if, some simple system contains one of its own solved forms and entails C. This can be checked in polynomial time, but the number of simple systems to be considered is far too large for practical usefulness. For fixed signature semantics in both LPO and RPO, this notion of simple systems is still insufficient and more guesses are needed.

In this paper we introduce some new notions of solved form, where, in addition to the closure under the classical RPOS decomposition rules, a restricted form of transitivity through variables is applied. It is proved that if C is a normal form in this sense, then it is satisfiable under extended signatures if, and only if, it has no cycle (Section 5).

For fixed signatures (Section 6) a slightly different transitivity rule is used. First, several particular cases of signatures are considered for which more efficient methods than the general one apply. The cases depend on whether (1) the smallest non-constant symbol f is unary and (2) there is at most one constant smaller than f. The following table summarizes the results. For instance, if (1) and (2) are true, then satisfiability is again equivalent to the absence of cycles. An entry 0 in the table denotes that, for some variables x, its relation with the smallest constant 0 needs to be guessed, that is, whether $x = 0$ or $x > 0$. Similarly, an ω denotes that for some variables its relation with the smallest limit ordinal term ω has to be guessed.

Results for fixed signatures		
precondition	$f \in lex$	$f \in mul$
1,2	no cycle (Section 6.1)	no cycle (Section 6.1)
1	no cycle, ω (Section 6.3)	no cycle, ω (Section 6.3)
2	no cycle (Section 6.2)	no cycle, ω (Section 6.4)
-	no cycle, $\omega, 0$ (Section 6.4)	no cycle, ω (Section 6.4)

For the cases marked with ω the problem is split into a natural and a non-natural part. The non-natural part is dealt with by cycle detection; the subproblem of natural number constraints, i.e., constraints where f is the only non-constant symbol and all terms and solutions correspond to the natural number fragment, can then be dealt with independently. This problem is solved for the case $f \in lex$ again by a transitivity closure, but now over the natural number ordering. For the case where f is not unary and has multiset status, we rely on the existing methods for solving multiset constraints on natural numbers.

In Section 7 we comment on some implementation issues. As we will show by experimental results from an implementation in the *Saturate* system [GNN95], our methods outperform the best previous one (an improvement of [NR95], as implemented in Saturate) by several orders of magnitude for extended signatures and for fixed signatures fulfilling the requirements (1) and (2). For other fixed signatures, apart from the prohibitive methods guessing linear orderings on all subterms of the constraint, no previous algorithms were known.

2 Path Orderings

Let $\mathcal{F}$ and $\mathcal{X}$ be sets of function symbols and variables respectively, and let $\succ_{\mathcal{F}}$ be a total ordering on $\mathcal{F}$ (the *precedence)*. Furthermore let $\mathcal{F}$ be the disjoint union of two sets lex and mul, the symbols with lexicographic and multiset status, respectively, and let $=_{mul}$ denote the equality of terms up to the permutation of direct arguments of symbols with multiset status.

The recursive path ordering (with status) (RPO) on ground terms is defined as follows: $s = f(s_1, \ldots, s_m) \succ_{rpo} g(t_1, \ldots, t_n) = t$ iff:

1. $s_1 \succ_{rpo} t$ or $s_1 =_{mul} t$, for some i with $1 \leq i \leq m$ or
2. $f \succ_{\mathcal{F}} g$, and $s \succ_{rpo} t_j$, for all j with $1 \leq j \leq n$ or
3. $f = g$, $f \in lex$, $\langle s_1, \ldots, s_n \rangle \succ_{rpo}^{lex} \langle t_1, \ldots, t_n \rangle$, and $s \succ_{rpo} t_j$, for all j with $1 \leq j \leq n$
4. $f = g$, $f \in mul$, $\{s_1, \ldots, s_n\} \succ_{rpo}^{mul} \{t_1, \ldots, t_n\}$

where $\langle s_1, \ldots, s_n \rangle \succ_{rpo}^{lex} \langle t_1, \ldots, t_n \rangle$ if $\exists j \leq n$ s.t. $s_j \succ_{rpo} t_j$ and $\forall i < j \;\; s_i =_{mul} t_i$. Furthermore, $\succ_{rpo}^{mul}$ is the multiset extension of $\succ_{rpo}$, defined as the smallest ordering such that $S \cup \{s\} \succ_{rpo}^{mul} S' \cup \{t_1, \ldots, t_n\}$ whenever S is equal to S' up to $=_{mul}$ and $s \succ_{rpo} t_i$ for all i in $1 \ldots n$.

The *lexicographic path ordering* is the particular case of RPO where $\mathcal{F} = lex$, and the *multiset path ordering* (or RPO without status) is the particular case where $\mathcal{F} = mul$.

3 Ordering Constraints

An *RPO-ordering constraint* is a quantifier-free first-order formula built over terms in $\mathcal{T}(\mathcal{F},\mathcal{X})$ and over the binary predicate symbols '$>$' and '$=$'. 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 if $>$ and $=$ are interpreted as the RPO defined by $\succ_{rpo}$ and $=_{mul}$ respectively. Then we say that σ satisfies C in $(\mathcal{F},\succ_{\mathcal{F}})$.

By an *extension* $(\mathcal{F}',\succ_{\mathcal{F}'})$ of $(\mathcal{F},\succ_{\mathcal{F}})$ we mean a set of function symbols $\mathcal{F}'$ with $\mathcal{F}' \supseteq \mathcal{F}$ and a total precedence $\succ_{\mathcal{F}'}$ extending $\succ_{\mathcal{F}}$. We will call a constraint C satisfiable under *extended signatures* if there exists some extension $(\mathcal{F}',\succ_{\mathcal{F}'})$ of $(\mathcal{F},\succ_{\mathcal{F}})$ in which C is satisfiable.

4 Solved Forms

An ordering constraint C can be equivalently expressed without negation since $s \not> t$ is equivalent to $t > s \vee t = s$, and $s \neq t$ is equivalent to $t > s \vee s > t$. Then C can be put into disjunctive normal form, and hence satisfiability has to be checked only for conjunctive constraints without negation. In the following we will deal with such conjunctions expressed by sets of equalities and inequalities between terms. The following (non-confluent) term rewrite system R operates on such sets:

$S \cup \{\ s > f(t_1 \ldots t_n)\ \}$	$\longrightarrow$	$S \cup \{\ s > t_1,\ \ldots,\ s > t_n\ \}$ if $top(s) >_{\mathcal{F}} f$
$S \cup \{\ s > t\ \}$	$\longrightarrow$	$S \cup S'$ if $top(s) = top(t) \in mul$ and $S' \in mul(s,t)$
$S \cup \{\ f(s_1 \ldots s_n) > f(t_1 \ldots t_n)\ \}$	$\longrightarrow$	$S \cup \{\ s_1 = t_1,\ \ldots,\ s_{i-1} = t_{i-1},\ s_i > t_i\ f(s_1 \ldots s_n) > t_{i+1}, \ldots, f(s_1 \ldots s_n) > t_n\ \}$ for $1 \leq i \leq n$ if $f \in lex$
$S \cup \{\ f(s_1 \ldots s_n) > t\ \}$	$\longrightarrow$	$S \cup \{\ s_i = t\ \}$ for $2 \leq i \leq n$ if $top(t) = f \in lex$
$S \cup \{\ f(s_1 \ldots s_n) > t\ \}$	$\longrightarrow$	$S \cup \{\ s_i > t\ \}$ for $2 \leq i \leq n$ if $top(t) = f \in lex$
$S \cup \{\ a > t\ \}$	$\longrightarrow$	$\bot$ if a is a constant and $top(t) \geq_{\mathcal{F}} a$
$S \cup \{\ f(s_1 \ldots s_n) > t\ \}$	$\longrightarrow$	$S \cup \{\ s_i = t\ \}$ for $1 \leq i \leq n$ if $top(t) >_{\mathcal{F}} f$
$S \cup \{\ f(s_1 \ldots s_n) > t\ \}$	$\longrightarrow$	$S \cup \{\ s_i > t\ \}$ for $1 \leq i \leq n$ if $top(t) >_{\mathcal{F}} f$
$S \cup \{\ s = t\ \}$	$\longrightarrow$	$S\sigma$ if $\sigma \in mul_unifiers(s,t)$
$S \cup \{\ s = t\ \}$	$\longrightarrow$	$\bot$ if $mul_unifiers(s,t) = \emptyset$

In R the following notation is used:

1. If s is $f(s_1, \ldots, s_n)$ and t is $f(t_1, \ldots, t_n)$, then $mul(s,t)$ is the set of all constraints of the form
   $\{\ s_{\pi(1)} = t_{\rho(1)},\ \ldots,\ s_{\pi(i)} = t_{\rho(i)},\ u_{i+1} > t_{\rho(i+1)},\ \ldots,\ u_n > t_{\rho(n)}\ \}$
   for permutations π and ρ of $1 \ldots n$ and $i \neq n$ and $u_j \in \{s_{\pi(i+1)}, \ldots, s_{\pi(n)}\}$ for all $j \in i+1 \ldots n$. Note that hence the second rule of R illustrates the fact that if $S_1 \succ_{rpo}^{mul} S_2$ then, after removing the proper subset of common (w.r.t. $=_{mul}$) elements on both sides, for each element in S_2 there is a bigger one in S_1.
2. Given two terms s and t, we denote by $mul_unifiers(s,t)$ the set of all unifiers modulo $=_{mul}$ of s and t. The term $f(t'_{\pi(1)}, \ldots, t'_{\pi(n)})$ is called a *permutation* of $f(t_1, \ldots, t_n)$ if t'_i is a permutation of t_i for $1 \leq i \leq n$ and π is a permutation of $1 \ldots n$ where $\pi = id$ if $f \in lex$. Then $mul_unifiers(s,t) = \{\ \sigma \mid \sigma = mgu(s,t')$ and t' is a permutation of $t\ \}$.

The termination of R can be shown by a well-founded ordering on sets S based on (i) the number of different variables in S, and (ii) on the multiset of sizes (in number of symbols) of the (in)equalities in S. See, e.g., [Com90] for details for the LPO case of the folowing lemma:

Lemma 1. *R is terminating, and, for each set of equalities and inequalities S, a normal form of S with respect to R is either $\bot$ or a set of inequalities of the form $s > t$ where at least one of s and t is a variable. Furthermore, S is satisfiable if, and only if, at least one of its normal forms is satisfiable.*

We now introduce our new notion of *solved form*, which, apart from being a normal form with respect to R, is also closed (in some sense) under transitivity through variables:

Definition 1. *A* solved form *S is a set of inequalities (and hence $S \neq \bot$) of the form $s > t$ where at least one of s and t is a variable, and such that if $s > x \in S$ and $x > t \in S$ for some variable x and non-variable s, then some normal form with respect to R of $\{s > t\}$ is a subset of S.*

Note that deciding whether some normal form of $\{s > t\}$ is a subset of S can be done in polynomial time. It suffices to check whether $s \succ_{rpo}^{S} t$, where $\succ_{rpo}^{S}$ is defined as the usual RPO extended with the case $s \succ_{rpo}^{S} t$ if $s > t \in S$.

Definition 2. *Let S be a set of equalities and inequalities. A* solved form *S' is called a solved form of S if it contains a normal form with respect to R of S.*

Lemma 2. *Let S be a set of equalities and inequalities. Then S is satisfiable if, and only if, at least one of its solved forms is satisfiable.*

For a given S, its set of solved forms can be computed by rewriting with R and closing under the transitivity rule $s > x \wedge x > t \Longrightarrow s > t$. Note that this process terminates when repeated work is avoided, since at any stage the sets contain only relations between subterms of S and the number of different such sets that can be obtained is finite. In Section 7 we will comment on some non-trivial implementation issues.

5 Cycles and Satisfiability over Extended Signatures

Definition 3. *A set of inequalities S* has a cycle *if* $\{x > t_1[x_1], x_1 > t_2[x_2], \ldots, x_{n-1} > t_n[x]\} \subseteq S$.

Lemma 3. *Let S be a set of inequalities with a cycle. Then S is unsatisfiable.*

Definition 4. *Let S be a solved form. We define $>_v$ as the smallest transitive relation on $\mathcal{V}ars(S)$ such that $x >_v y$ whenever $x > t[y] \in S$.*

Lemma 4. *Let S be a solved form with no cycles. Then $>_v$ is a well-founded strict partial ordering on $\mathcal{V}ars(S)$.*

Lemma 5. *Let S be a solved form with no cycles and let x be a variable occurring in S. Furthermore, let S' be the set obtained by removing from S all inequalities where x occurs, i.e., $S' = \{s > t \mid s > t \in S \wedge x \notin \mathcal{V}ars(s > t)\}$. Then S' is a solved form with no cycles.*

Proof. Clearly S' is still in normal form w.r.t. R and has no cycles. We now show that S' is still closed under transitivity. Suppose $s > y \in S'$ and $y > t \in S'$ for some variable $y \neq x$. Then x does not occur in $s > t$. But then the normal form of $s > t$ that belonged to S is still in S' since x does not occur in this normal form (the rules of R do not introduce any variables). □

Definition 5. *Let S be a solved form with no cycles, and let $(\mathcal{F}', \succ_{\mathcal{F}'})$ be the extension of $(\mathcal{F}, \succ_{\mathcal{F}})$ where $\mathcal{F}' = \mathcal{F} \cup \{f, 0\}$ and where $\succ_{\mathcal{F}'}$ extends $\succ_{\mathcal{F}}$ such that $g \succ_{\mathcal{F}'} f \succ_{\mathcal{F}'} 0$ for all $g \in \mathcal{F}$.*

The minimal substitution *σ for S is defined by induction on $>_v$ as follows. Let $x \in \mathcal{V}ars(S)$ and let σ_x be the partial substitution defined for the variables of S that are smaller than x w.r.t. $>_v$. Then*

- *$x\sigma = 0$ if there is no inequation $x > t$ in S and*
- *$x\sigma = f(t)$ if $t = max_{rpo}\{s\sigma_x \mid x > s \in S\}$.*

Theorem 1. *Let S be a solved form over $(\mathcal{F}, \succ_{\mathcal{F}})$. Then S is satisfiable over some extension of $(\mathcal{F}, \succ_{\mathcal{F}})$ if, and only if, S has no cycle.*

Proof. By Lemma 3, S is unsatisfiable if it has some cycle. Now we show that if S has no cycle the minimal substitution σ in the extension $(\mathcal{F}', \succ_{\mathcal{F}'})$ given in Definition 5 is a solution of S.

We proceed by induction on the number k of variables in S. If $k = 0$, then S is empty and trivially satisfiable. For the induction step, let x be a variable that is maximal w.r.t. $>_v$ in S, and let S' be the conjunctive constraint obtained by removing from S all inequalities where x occurs, i.e., $S' = \{s > t \mid s > t \in S \wedge x \notin \mathcal{V}ars(s > t)\}$. Then by Lemma 5 S' is a solved form with no cycles.

Now let σ' be σ with its domain restricted to $\mathcal{V}ars(S) \setminus \{x\}$. Then σ' is the minimal substitution of S' and hence, by the induction hypothesis, σ' is a solution for S'.

To prove that σ is indeed a solution of S, it remains to be checked that $s\sigma \succ_{rpo} t\sigma$ for the relations $s > t$ where x appears in s or t.

There are six cases:

1. $x > t[x]$. No such inequation exists since this is a cycle.
2. $t[x]_p > x$ with $p \neq \lambda$. Then clearly $t\sigma[x\sigma]_p \succ_{rpo} x\sigma$ by the subterm property of RPO.
3. $y > t[x]$. No such inequation exists since x is maximal.
4. $x > t$ where $x \notin \mathcal{V}ars(t)$. Then, since $x\sigma = f(t)$ where $t = max_{rpo}\{s\sigma' \mid x > s \in S\}$, clearly $x\sigma \succ_{rpo} t\sigma$.
5. $s > x$ where $x \notin \mathcal{V}ars(s)$ and s is not a variable.
   If there is no inequation $x > t$ in S, then $x\sigma = 0$, and hence $s\sigma \succ_{rpo} x\sigma$.
   If there is at least one inequation $x > t$ in S, then $x\sigma = f(u)$ where $u = max_{rpo}\{t\sigma' \mid x > t \in S\}$. By definition of solved form, then S contains a normal form with respect to R of $s > t$ for all $x > t \in S$. Since $x \notin \mathcal{V}ars(s > t)$, in fact S' contains this solved form and hence $s\sigma' \succ_{rpo} t\sigma'$, and $s\sigma \succ_{rpo} t\sigma$, and, in particular, $s\sigma \succ_{rpo} u$. Since $top(s) \succ_{\mathcal{F}} f$, this implies $s\sigma \succ_{rpo} f(u)$ and $s\sigma \succ_{rpo} x\sigma$.
6. $s[x]_p > y$ where $p \neq \lambda$ and $x \neq y$. We proceed by contradiction. Let $s > y$ be an inequality in S of this form such that $s\sigma \not\succ_{rpo} y\sigma$ and where y is minimal in $>_v$. Clearly $y\sigma \neq 0$; hence $y\sigma = f(u)$ where $u = max_{rpo}\{t\sigma \mid y > t \in S\}$. We now show that $s\sigma \succ_{rpo} u$, contradicting $s\sigma \not\succ_{rpo} y\sigma$, since $top(s) \in \mathcal{F}$ and hence $top(s) \succ_{\mathcal{F}'} f$.
   By definition of solved form, S contains a normal form with respect to R of $s > t$ for all $y > t \in S$. For the inequalities in this solved form, either x does not occur in them, or else they fall under one of the five previous cases, and hence σ is a solution for them, or they have the form $s|_q > z$ for some variable z with $y >_v z$, and hence σ is a solution for them by the minimality assumption for y. Hence σ is a solution of this solved form and hence $s\sigma \succ_{rpo} u$. □

6 Fixed Signatures

Let f be the smallest non-constant symbol in $\mathcal{F}$, and let 0 be the smallest constant symbol (and hence the smallest term). In this section we will first consider two restrictions:

1. there is at most one constant symbol smaller than f, and
2. f is unary

If $(\mathcal{F}, \succ_{\mathcal{F}})$ satisfies both restrictions, it will be called *well-ended*. In several further subsections we will show how these restrictions can be dropped at the expense of adding some new rules to R.

6.1 Well-ended Signatures

Note that in this setting for every ground term t, its *successor*, the smallest term bigger than t, is $f(t)$. We sometimes write $f^n(t)$ to denote the n-th successor of t.

Example 1. Consider the constraint S of the form $f(f(0)) > x \ \wedge \ f(x) > y > z > 0$. It is a solved form in the sense of the previous section, and it has no cycle. However, it is unsatisfiable over fixed signatures, since it amounts to diophantine inequations over the natural numbers: we have $2 > x \wedge x + 1 > y > z > 0$. and there is no space for both y and z between 2 and 0.

The previous example shows us that we need to reconsider the definition of solved form, since the notion of closure under transitivity used in the previous section is too weak for fixed signatures.

Definition 6. *A* solved form S *is a set of inequalities of the form* $s > t$ *where at least one of* s *and* t *is a variable, and such that if* $s > x \in S$ *and* $x > t \in S$ *for some non-variable* s *and variable* x*, then some normal form with respect to* R *of* $\{s > f(t)\}$ *is a subset of* S*.*

Example 2. (Example 1 continued) Closing S under the new notion of transitivity gives the following. From $f(x) > y > z$ we get $f(x) > f(z)$, which simplifies into $x > z$. This, together with $f(f(0)) > x$ gives $f(f(0)) > f(z)$ which simplifies into $f(0) > z$. This, together with $z > 0$ gives $f(0) > f(0)$ whose unique normal form is $\bot$.

Clearly, like in the previous section, a set of (in)equalities S is satisfiable if, and only if, one of its solved forms in this new sense is satisfiable.

Theorem 2. *Let* $\mathcal{F}$ *be a well-ended signature, and let* S *be a solved form (in the sense of Definition 6) over* $(\mathcal{F}, \succ_{\mathcal{F}})$*. Then* S *is satisfiable in* $(\mathcal{F}, \succ_{\mathcal{F}})$ *if, and only if,* S *has no cycles and* $0 > x \notin S$*.*

Proof. The proof follows the same ideas as the one of Theorem 1. Note that Lemma 5 still applies for the new notion of solved form. Of the six cases, the first four ones remain equal. The remaining two cases become:

5. $s > x$ where $x \notin \mathcal{V}ars(s)$ and s is not a variable.
   If there is no inequation $x > t$ in S, then $x\sigma = 0$, and hence $s\sigma \succ_{rpo} x\sigma$, since $0 > x$ is not in S.
   If there is at least one inequation $x > t$ in S, then $x\sigma = f(u)$ where $u = max_{rpo}\{t\sigma' \mid x > t \in S\}$. By the new definition of solved form, then S contains a normal form with respect to R of $s > f(t)$ for all $x > t \in S$ and we conclude as in Theorem 1.
6. $s[x]_p > y$ where $p \neq \lambda$ and $x \neq y$. We proceed by contradiction. Let $s > y$ be an inequality in S of this form such that $s\sigma \not\succ_{rpo} y\sigma$ and where y is minimal in $>_v$. Clearly $y\sigma \neq 0$, since s is a non-constant term and for any such terms $s\sigma \succ_{rpo} 0$. Hence $y\sigma = f(u)$ where $u = max_{rpo}\{t\sigma \mid y > t \in S\}$. We now show that $s\sigma \succ_{rpo} f(u)$, contradicting $s\sigma \not\succ_{rpo} y\sigma$. By the new definition of solved form, S contains a normal form with respect to R of $s > f(t)$ for all $y > t \in S$, and we can conclude as in Theorem 1. □

6.2 At Most One Constant below a Non-unary f $\in$ lex

If f is non-unary and lexicographic, and there is at most one constant below f, then the successor of every ground term t is $f(0,\ldots,0,t)$. And then the satisfiability of solved forms can be decided by cycle detection as in the previous subsection, if everywhere $f(t)$ is replaced by $f(0,\ldots,0,t)$: if $s > x \in S$ and $x > t \in S$ for some non-variable s, a variable x, and term t, then now it is required that some normal form with respect to R of $\{s > f(0,\ldots,0,t)\}$ is a subset of S. Similarly, the minimal substitution σ for a solved form with no cycles S is now defined by taking successors like this: $x\sigma = f(0,\ldots,0,t)$ if $t = max_{rpo}\{s\sigma_x \mid x > s \in S\}$. From this we get the following:

Theorem 3. *Let $(\mathcal{F},\succ_\mathcal{F})$ be such that the smallest non-constant symbol f is in lex, and there is at most one constant symbol smaller than f. Let S be a solved form over $(\mathcal{F},\succ_\mathcal{F})$. Then S is satisfiable in $(\mathcal{F},\succ_\mathcal{F})$ if, and only if, S has no cycles and $0 > x \notin S$.*

6.3 More than One Constant below a Unary f

Assume the signature is ended by $g \succ_\mathcal{F} f \succ_\mathcal{F} a_1 \succ_\mathcal{F} \ldots \succ_\mathcal{F} a_n \succ_\mathcal{F} 0$, with $n > 0$ and where the smallest non-constant symbol f is unary. Then 0 is the smallest term, but no longer for every ground term t, its successor is $f(t)$. We have the following increasing sequence of ground terms:

$$0,\ a_n,\ \ldots,\ a_1,\ f(0),\ f(a_n),\ \ldots,\ f(a_1),\ f(f(0)),\ \ldots,\ g(0,\ldots,0) = \omega,\ f(\omega),\ldots$$

where ω is the first limit ordinal term, that is, the smallest term ω such that $w \succ_{rpo} t$ for infinitely many terms t. Terms below ω are called *natural terms*. Clearly the transitivity notion $s > x \wedge x > t \Longrightarrow s > f(t)$ applied in the Subsection 6.1 is now correct only if t is known to be non-natural. In order to know whether t is natural or not at the non-ground level, we now add three new rules to the rewrite system R that *guess* for each variable how it is related to ω. Let R_ω be the set of the following three rules:

$$\begin{array}{lcll} S & \longrightarrow & S\{x \mapsto \omega\} & \text{if } x \in vars(S) \text{ and } x > \omega \notin S \text{ and } \omega > x \notin S \\ S & \longrightarrow & S \cup \{\ x > \omega\ \} & \text{if } x \in vars(S) \text{ and } x > \omega \notin S \text{ and } \omega > x \notin S \\ S & \longrightarrow & S \cup \{\ \omega > x\ \} & \text{if } x \in vars(S) \text{ and } x > \omega \notin S \text{ and } \omega > x \notin S \end{array}$$

Now it is clear for every normal form S with respect to R_ω whether a term t with variables is natural or not: t is natural if, and only if, it contains only symbols smaller than g and variables x for which $\omega > x \in S$. Note that in practice it is not necessary to guess the relations with ω for *all* variables, as long as for all sides of an inequality it is known whether they are natural or not. For example, t is known to be natural if it contains no symbol greater than f, or $s > t \in S$ for some natural term s, etc.

Now we can adapt the notion of solved form to include transitivity only for non-natural terms, and again a set of (in)equalities S is satisfiable if, and only if, one of its solved forms in this new sense is satisfiable.

Definition 7. *A* solved form S *is a set of inequalities in normal form w.r.t.* $R \cup R_\omega$, *such that if* $s > x \in S$ *and* $x > t \in S$ *for some non-variable term* s, *a variable* x *and a non-natural term* t, *then some normal form with respect to* R *of* $\{s > f(t)\}$ *is a subset of* S.

We now split the satisfiability problem for a solved form S in two parts: the *natural part* S_N, and the *non-natural* one S_ω, i.e., $S_N = \{s > t \mid s > t \in S$ and s and t are natural $\}$ and $S_\omega = S \setminus S_N$. The satisfiability of S_N can be easily decided. It suffices to express all ground terms as their corresponding natural number and terms $f^k(x)$ as $x+k*(n+1)$. In fact, the resulting problem of satisfiability of diophantine inequations is in P, since it can be solved by closure under transitivity. Note that this is precisely what was done in the previous section at the symbolic level (and hence we will not prove its correctness again here): close under the rule $x + k > y \wedge y > z + k' \Longrightarrow x + k > z + k' + 1$, simplifying the conclusion to get a variable at one of both sides; if no cycle or contradiction of the form $n > n + k$ appears, then the problem is satisfiable.

Lemma 6. *Let* $(\mathcal{F}, \succ_\mathcal{F})$ *be such that* $g \succ_\mathcal{F} f \succ_\mathcal{F} a_1 \succ_\mathcal{F} \ldots \succ_\mathcal{F} a_n \succ_\mathcal{F} 0$, *where the smallest non-constant symbol* f *is unary. Let* S *be a solved form (in the sense of Definition 7)* S *over* $(\mathcal{F}, \succ_\mathcal{F})$. *Then* $s > t \in S$ *for no* s *that is natural and* t *that is non-natural.*

Proof. Assume such an $s > t$ is in S.

If s is not a variable, then it is $s > x$ where $x > \omega \in S$. Then, by transitivity, a solved form of $s > f(\omega)$ is in S. If s is ground, this leads to $\bot$ by R, and then S would be $\bot$, contradicting $s > t \in S$. If s is non-ground, it is of the form $f^n(y)$ for some $n > 0$, where $\omega > y \in S$, and the unique normal form of $s > f(\omega)$ is $y > \omega$, which leads to $\bot$ by transitivity from $\omega > y$.

If s is a variable x, then $\omega > x \in S$. Then, by transitivity, a solved form of $\omega > f(t)$ is in S. If t is ground, this leads to $\bot$. If t is non-ground, it either leads to $\bot$ or to a solved form where $\omega|_p > y$ for some variable y in t such that $y > \omega \in S$, which leads to $\bot$ as before. □

Theorem 4. *Let* $(\mathcal{F}, \succ_\mathcal{F})$ *be such that* $g \succ_\mathcal{F} f \succ_\mathcal{F} a_1 \succ_\mathcal{F} \ldots \succ_\mathcal{F} a_n \succ_\mathcal{F} 0$, *where the smallest non-constant symbol* f *is unary. Let* S *be a solved form (in the sense of Definition 7) over* $(\mathcal{F}, \succ_\mathcal{F})$. *Then* S *is satisfiable in* $(\mathcal{F}, \succ_\mathcal{F})$ *if, and only if,* S_N *is satisfiable and* S *has no cycles.*

Proof. If S is satisfiable then S_N is also satisfiable and S has no cycles. Now assume that S_N is satisfiable and S has no cycles. Let σ_N a solution of S_N, and we will show that it can be extended to a solution σ for the whole S, by building the minimal solution σ for S_ω, starting from σ_N.

First, note that for the inequalities $s > t$ where s is non-natural and t is natural, we trivially have $s\sigma \succ_{rpo} t\sigma$, since $t\sigma$ is natural, and $s\sigma$ will be non-natural: since s either contains some symbol g with $g \succ_\mathcal{F} f$, or else some non-natural variable x and $x\sigma \succ_{rpo} \omega$ for all such x. Hence by Lemma 6, it only remains to check the inequalities between non-natural terms.

Note that Lemma 5 still applies for the new notion of solved form. Of the six cases of Theorem 1, again the first four ones remain equal. The last two cases are:

5. $s > x$ where $x \notin \mathcal{V}ars(s)$ and s is not a variable.
   There is at least one inequation $x > t$ in S, since $x > \omega \in S$. Then $x\sigma = f(u)$ where $u = max_{rpo}\{t\sigma' \mid x > t \in S\}$. By the new definition of solved form, then S contains a normal form with respect to R of $s > f(t)$ for all $x > t \in S$ and we conclude as in Theorem 1.
6. $s[x]_p > y$ where $p \neq \lambda$ and $x \neq y$. We proceed by contradiction. Let $s > y$ be an inequality in S of this form such that $s\sigma \not> _{rpo}\ y\sigma$ and where y is minimal in $>_v$. Since y is non-natural, $y\sigma = f(u)$ where $u = max_{rpo}\{t\sigma \mid y > t \in S\}$, and $s\sigma \succ_{rpo} f(u)$ follows as in Theorem 2. □

6.4 When the Smallest Non-constant Symbol f Is Non-unary

We now also eliminate the restriction that f is unary. We will continue with the same methodology as before, using the rules of R_ω that guess for each variable its relation to ω, and splitting the solved forms into the two independent parts: the natural and non-natural ones.

Now different approaches are needed depending on whether f has multiset or lexicographic status.

The Multiset Case If $f \in mul$, then clearly a term t is natural if, and only if, it is built from the smallest non-constant symbol f, constants smaller than f and natural variables (i.e., variables x with $\omega > x \in S$). In this sense, the multiset case is simpler than the lexicographic one, as we will see. E.g., if f is binary, and $f \succ_\mathcal{F} a_1 \succ_\mathcal{F} \ldots \succ_\mathcal{F} a_n \succ_\mathcal{F} 0$, we have the following increasing sequence of natural ground terms:

$$0,\ a_n,\ \ldots,\ a_1,\quad f(0,0),\quad f(0,a_n),\quad f(a_n,a_n),\quad f(0,a_{n-1}),\quad f(a_n,a_{n-1}),$$
$$f(a_{n-1},a_{n-1}),\quad f(0,a_{n-2}),\quad \ldots,\quad f(a_1,a_1),\quad f(0,f(0,0)),\ldots$$

and if there is at most one constant below f, we have

$$0,\quad f(0,0),\quad f(0,f(0,0)),\quad f(f(0,0),f(0,0)),$$
$$f(0,f(0,f(0,0))),\quad f(f(0,0),f(0,f(0,0))),\quad \ldots$$

Here 0 is still the smallest term, and for every non-natural ground term t, its successor is $f(0,\ldots,0,t)$. The smallest non-natural term ω is $g(0,\ldots,0)$, where g is the smallest symbol bigger than f and than 0. Solving the non-natural part can hence be done as for the case of unary f, if everywhere $f(t)$ is replaced by $f(0,\ldots,0,t)$.

The definition of solved form again considers $R \cup R_\omega$, and if $s > x \in S$ and $x > t \in S$ for some non-variable s, a variable x, and a non-natural t, then now it is required that some normal form with respect to R of $\{s > f(0,\ldots,0,t)\}$ is a subset of S. Similarly, the minimal substitution σ for a solved form with no cycles S is now defined by taking successors like this: $x\sigma = f(0,\ldots,0,t)$ if $t = max_{rpo}\{s\sigma_x \mid x > s \in S\}$. From this we get the following result:

Theorem 5. *Let $(\mathcal{F}, \succ_\mathcal{F})$ be such that the smallest non-constant symbol f is in mul. Let S be a solved form over $(\mathcal{F}, \succ_\mathcal{F})$. Then S is satisfiable in $(\mathcal{F}, \succ_\mathcal{F})$ if, and only if, S_N is satisfiable, and S has no cycles.*

Deciding whether S_N is satisfiable amounts to solving purely natural RPO constraints, that is, constraints built only over f, 0, and possibly other constants smaller than f, and with solutions over this same signature. If $f \in lex$ this is a simple problem over the natural numbers, but for $f \in mul$ this seems not the case. Hence for the moment we propose to use the algorithm of [JO91] or the NP one of [NRV98] for S_N, which is normally a minor part of S.

The Lexicographic Case Let $(\mathcal{F}, \succ_\mathcal{F})$ be such that $f \succ_\mathcal{F} a_1 \succ_\mathcal{F} \ldots \succ_\mathcal{F} a_n \succ_\mathcal{F} 0$, where the smallest non-constant symbol f is in lex (the case with at most one constant below f has been treated already in Subsection 6.2). Then a term like $f(x, 0)$ can have non-natural instances even if x is instantiated with a natural term: e.g., $f(a_n, 0)$ is precisely ω, the first limit ordinal. If f is binary, we have:

$$0, \quad a_n, \quad \ldots, \quad a_1, \quad f(0,0), \quad f(0,a_n), \quad \ldots, \quad f(0,a_1), \quad f(0,f(0,0)),$$
$$f(0,f(0,a_n)), \quad \ldots, \quad f(0,f(0,f(0,0))), \quad \ldots, \quad f(a_n,0)\ldots$$

Therefore, in order to split the constraint into its natural and non-natural parts, we need not only to know the relation between some variables x and ω, but also whether x is 0 or not. Hence the three additional rules of R given in the previous subsection become now the following four ones, which we will call $R_{\omega,0}$:

$$\begin{array}{lcll} S & \longrightarrow & S\{x \mapsto 0\} & \text{if } x \in vars(S) \\ S & \longrightarrow & S\{x \mapsto \omega\} & \text{if } x \in vars(S) \\ S & \longrightarrow & S \cup \{\, x > \omega \,\} & \text{if } x \in vars(S) \text{ and } x > \omega \notin S \\ S & \longrightarrow & S \cup \{\, \omega > x, x > 0 \,\} & \text{if } x \in vars(S) \text{ and } \{\, \omega > x, x > 0 \,\} \not\subseteq S \end{array}$$

As for the remaining rules of R given before, sometimes the application of a rule will produce a reduction to $\bot$, and in many case these situations can be foreseen. Also, guessing whether a variable x is 0 is in fact needed only if x appears in some term t that is otherwise unknown to be natural or not. Similar efficiency issues will be discussed later on.

For normal forms with respect to $R_{\omega,0}$, clearly a term is natural if, and only if, it is of the form $f(0,\ldots,0,f(0,\ldots,0,f(\ldots f(0,\ldots,0,t))))$ where t is a constant smaller than f or a natural variable (i.e., a variable x with $\omega > x \in S$), and for every non-natural ground term t, its successor is $f(0,\ldots,0,t)$. Again the results of the previous section go through if everywhere $f(t)$ is replaced by $f(0,\ldots,0,t)$, and we get the following:

Theorem 6. *Let $(\mathcal{F}, \succ_\mathcal{F})$ be such that the smallest non-constant symbol f is in lex. Let S be a solved form over $(\mathcal{F}, \succ_\mathcal{F})$. Then S is satisfiable in $(\mathcal{F}, \succ_\mathcal{F})$ if, and only if, S_N is satisfiable and S has no cycles.*

And here again the satisfiability of the natural part S_N can be decided in polynomial time by translation into diophantine inequations of the form $x+k > y$ or $x > y + k$, which can be handled by transitive closure as in the previous subsection.

7 Implementation Issues

Let us first consider a practical improvement obtained by using '$\geq$' as an additional predicate in constraints. This importantly reduces the number of solved forms under consideration. For example, the two rules of R for the case $top(t) >_{\mathcal{F}} f$, can now be covered by the single one $S \cup \{ f(s_1, \ldots, s_n) > t \} \longrightarrow S \cup \{ s_i \geq t \}$. Additional rules for decomposing atoms $s \geq t$ are defined analogously to the ones for '$>$'. If a new rule $S \cup \{ s \geq t, t \geq s \} \longrightarrow S \cup \{ s = t \}$ is added as well, and the notions of cycle and transitivity, etc., are defined as expected, it is not difficult to check that all results of the previous sections go through.

7.1 Computing Solved Forms

We have seen that the constraint satisfiability problem roughly amounts to deciding the existence of a solved form with no cycle. Solved forms can be obtained by computing normal forms with respect to R, adding new inequations by application of the transitivity rule, which in turn have to be turned into normal form, etc. In order to avoid repeated work, standard methods from theorem proving or completion for closing under inference rules can, and have to be, used.

In our implementation we deal with three sets of inequalities: the *old* ones, that are in normal form and closed under transitivity, the *new* ones, that are in normal form, and an additional set T with the recently added consequences under transitivity. Initially *old* and *new* are empty, and T contains the set of input inequalities.

The working cycle consists of the following. *One* normal form w.r.t. R of each non-redundant element $s > t$ in T is added to *new*, where $s > t$ is redundant if some normal form of it is a subset of $old \cup new$. This can be checked in polynomial time by using $s \succ^S_{rpo} t$, where $\succ^S_{rpo}$ is defined as the usual RPO extended with the case $s \succ^S_{rpo} t$ if $s > t \in old \cup new$. Backtracking on the choice of normal form of $s > t$ occurs when a cycle is detected at some point. If T is empty, one inequality $u > v$ of *new* is moved to *old* and all transitivity consequences between $u > v$ and *old* are put in T. If both T and *new* are empty, and there is no cycle in *old* then the constraint is satisfiable.

7.2 Some First Practical Experiments

We experimented with a Prolog implementation based on the aforementioned procedure. In order to obtain objective problem sets, we ran *Saturate* on 10 problems in first-order theorem proving. For each problem, we kept the

set of all ordering constraints generated during the run. It turned out that for only three of the ten problems, on rings, abelian groups, and embedding, respectively, a statistically significant number of non-trivial constraints were generated. In the tables below we show the results for these three problem sets comparing our *New* method with the previously best known one *Old* (an improvement of [NR95]) as it was (quite carefully) implemented in the *Saturate* system, for extended signatures. Times are in miliseconds for Sicstus Prolog 3.7.1 on a SUN Ultra 5. The problem sets and test program are available from `http://www.lsi.upc.es/~roberto`. For well-ended fixed signatures very similar results are obtained. For other fixed signatures, apart from the prohibitive methods guessing linear orderings on all subterms of the constraint, no previous algorithms were known.

The leftmost column *Threshold* indicates the minimum time in miliseconds required for considering a problem. For instance, the first row considers *all* problems of the set, the second row only the ones where at least one of both algorithms takes 20 ms or more, etc. Note that for harder problems the improvement ratio is higher:

Constraints from problem Rings				
Threshold	*# Problems*	*Total Time Old*	*Total Time New*	*Improvement Ratio*
0	977	62230	1350	46.10
20	370	59980	700	85.69
50	186	55410	550	100.75
100	130	52050	490	106.22
200	64	41510	280	148.25
400	26	31450	100	314.50
1000	14	25200	50	504.00

Constraints from problem Abelian Groups				
Threshold	*# Problems*	*Total Time Old*	*Total Time New*	*Improvement Ratio*
0	246	35520	590	60.20
20	99	35010	390	89.77
50	80	34480	360	95.78
100	64	33560	330	101.70
200	35	29500	260	113.46
400	20	25560	170	150.35
1000	12	21680	120	180.67

Constraints from problem Embedding				
Threshold	*# Problems*	*Total Time Old*	*Total Time New*	*Improvement Ratio*
0	814	78000	1170	66.67
20	349	76470	500	152.94
50	185	72290	180	401.61
100	117	67660	120	563.83
200	53	59080	50	1181.60
400	27	52120	30	1737.33
1000	13	40960	10	4096.00

8 Conclusions and Further Work

We have shown that, for an adequate notion of solved form, simply based on RPO decomposition and transitivity, deciding the satisfiability of path ordering constraints roughly amounts to solved form computation and cycle detection.

This leads to new algorithms that, we believe, are currently the best choice for path ordering constraint solving under all possible precedences and semantics.

Although it is not very relevant from the practical point of view, it seems quite clear that, when more carefully formulated, our algorithms can be shown to be in NP. First one guesses a rewrite derivation with R (and R_ω or $R_{\omega,0}$ if needed) into a normal form. While doing this, in order to avoid the creation of terms of exponential size, a different treatment for the equality relation is needed (see [NRV98]). For the cases where S is split into S_N and S_ω, at some point also the satisfiability of S_N has to be checked (which is in P if $f \in lex$, and requires to apply the NP algorithm of [NRV98] if $f \in mul$).

We believe that more practical algorithms can be found for purely natural multiset constraints; this is also the subject of further work.

References

[CNNR98] Hubert Comon, Paliath Narendran, Robert Nieuwenhuis, and Michael Rusinowitch. Decision problems in ordered rewriting. In *13th IEEE Symp. Logic in Comp. Sc. (LICS)*, pages 410–422, Indianapolis, USA, 1998.

[Com90] Hubert Comon. Solving symbolic ordering constraints. *International Journal of Foundations of Computer Science*, 1(4):387–411, 1990.

[CT94] Hubert Comon and Ralf Treinen. Ordering Constraints on Trees. In *Proc. CAAP*, LNCS 787, Edinburgh, Scotland, Springer-Verlag.

[Der87] Nachum Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69–116, 1987.

[GNN95] Harald Ganzinger, Robert Nieuwenhuis, and Pilar Nivela. The Saturate System, 1995. (see `www.mpi-sb.mpg.de/SATURATE/Saturate.html`).

[JO91] J-P. Jouannaud and M. Okada. Satisfiability of systems of ordinal notations with the subterm property is decidable. In Proc *18th ICALP*, LNCS 510, Madrid, Spain, July 16–20 1991. Springer-Verlag.

[KKR90] Claude Kirchner, Hélène Kirchner, and Michaël Rusinowitch. Deduction with symbolic constraints. *Revue Française d'Intelligence Artificielle*, 4(3):9–52, 1990.

[KNS85] Deepak Kapur, Paliath Narendran, and G. Sivakumar. A path ordering for proving termination for term rewriting systems. In *Proc. of 10th ICALP*, LNCS 185, pages 173–185, Germany, 1985. Springer-Verlag.

[Les90] Pierre Lescanne. On the recursive decomposition ordering with lexicographical status and other related orderings. *J. Aut. Reasoning*, 6(1):39–49, 1990.

[Nie93] Robert Nieuwenhuis. Simple LPO constraint solving methods. *Information Processing Letters*, 47:65–69, August 1993.

[NR95] Robert Nieuwenhuis and Albert Rubio. Theorem Proving with Ordering and Equality Constrained Clauses. *J. Symbolic Comp.*, 19(4):321–351, 1995.

[NRV98] P. Narendran, M. Rusinowitch, and R. Verma. RPO constraint solving is in NP. In *CSL 98*, Brno, Czech Republic, August 23–28, 1998. Abstract at http://www.dbai.tuwien.ac.at/CSL98.