Practical Algorithms for Deciding Path Ordering Constraint Satisfaction

Robert Nieuwenhuis and José Miguel Rivero[1]
Technical University of Catalonia, Dept. LSI
Jordi Girona 1, 08034 Barcelona, Spain
E-mail: roberto@lsi.upc.es rivero@lsi.upc.es

Version: November 8, 2001

We introduce new algorithms for deciding the satisfiability of constraints for the full recursive path ordering with status (RPO), and hence as well for other path orderings like LPO, MPO, KNS and RDO, and for all possible total precedences and signatures.

The techniques are based on a new notion of solved form, where fundamental properties of orderings like transitivity and monotonicity are taken into account.

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.

Key Words: Automated deduction, ordered strategies, symbolic constraints.

1. INTRODUCTION

An ordering constraint is a quantifier-free first-order formula built over the binary predicate symbols '$>$' and '$=$' which, respectively, denote a given 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 ordering $\succ$ and congruence $\equiv$. If C has a solution it is called satisfiable. Such constraints have many interesting applications like pruning the search space in automated theorem proving [10, 17, 18] or deciding the confluence of ordered rewrite systems [2]. 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 recursive path ordering with status (RPO), such $\succ_c$ subsume other path orderings like the lexicographic path ordering (LPO) [8], the recursive decomposition ordering (RDO) [12], or the KNS ordering [9], since all these path orderings coincide on ground terms (see [4]). 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 $\mathcal{F}$) and *extended signature* semantics (new symbols are allowed to appear in solutions). Satisfiability under extended signatures was considered in [16] as the adequate semantics for the computation of saturated sets of ordering constrained clauses

[1]Both authors partially supported by the Spanish Cicyt project HEMOSS, ref. TIC98-0949-C02-01.

that can be used for deduction with other clauses containing arbitrary new (e.g., Skolem) symbols.

The satisfiability problem for ordering constraints was first shown decidable for fixed signatures when $\succ$ is a total LPO [1] or a total RPO [7]. For extended signatures, decidability was shown for LPO in [17] and for RPO in [14]. Regarding complexity, NP algorithms for LPO (fixed and extended signatures) and RPO (extended ones) were given in [14]. More recently, an NP algorithm has been given as well for RPO under fixed signatures in [13]. NP-hardness of the satisfiability problem is known, even for one single inequation, for all these cases [3]. 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 [1], where for each i always s_i or t_i is a variable.

In algorithms like the ones of [1] and [17], 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 inequations considered independently. In fact any constraint $s > t$ can be expressed like the solved form $s > x \wedge x > t$, for some new variable x, which is equivalent w.r.t. satisfiability under extended signatures. 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 [14] and [13] 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 $\{s_n, \ldots, s_0\}$ is the set of *all subterms* of C. In [14] 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. For each simple system, 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 RPO decomposition rules, a restricted form of transitivity through variables is applied. It is proved that if C is a solved 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, ω, 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 of [7, 13] for dediding the satisfiability of multiset constraints on natural numbers.

An improvement with respect to the earlier short version of this work [15] is that here we introduce from the beginning the additional predicate $\geq$ in the constraint language, which leads to a better performance, since expressions $s \geq t$ need not be split into $s > t \vee s = t$.

In Section 7 we comment on some implementation issues. As we will show by experimental results from an implementation in the *Saturate* system [6], our methods outperform the best previous one (an improvement of [17], 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. We are not aware of the existence of any other (competitive or experimental) implementations by other researchers.

2. PATH ORDERINGS

Let $\succ$ be an ordering on terms and let $\equiv$ be a congruence relation. These relations induce relations on tuples and multisets of terms as follows.

The *lexicographic (left to right) extension of* $\succ$ *with respect to* $\equiv$ is the relation $\succ^{lex}$ on n-tuples of terms defined by:

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

for some k in $1 \ldots n$.

The *multiset extension of* $\equiv$ is defined as the smallest relation $==$ on multisets of terms such that $\emptyset == \emptyset$ and

$$M \cup \{s\} == N \cup \{t\} \quad \text{if} \quad M == N \text{ and } s \equiv t$$

The *multiset extension of* $\succ$ *with respect to* $\equiv$ is defined as the smallest ordering $\succ^{mul}$ on multisets of terms such that

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

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. By $=_{mul}$ we denote the equality of ground terms up to the permutation of direct arguments of symbols with multiset status: $f(s_1,\ldots,s_m) =_{mul} g(t_1,\ldots,t_n)$ iff $f = g$ and hence $m = n$, and $s_{\pi(i)} =_{mul} t_i$ for $1 \leq i \leq n$ and where π is a permutation of $1\ldots n$ which is the identity if $f \in lex$.

In this setting, the recursive path ordering (with status) (RPO) on ground terms is defined as follows: $s \succ_{rpo} x$ if x is a variable that is a proper subterm of s or else $s = f(s_1,\ldots,s_m) \succ_{rpo} g(t_1,\ldots,t_n) = t$ if at least one of the following conditions holds:

- $s_i \succ_{rpo} t$ or $s_i =_{mul} t$, for some $i \in \{1\ldots n\}$
- $f \succ_{\mathcal{F}} g$, and $s \succ_{rpo} t_j$, for all j in $\{1\ldots m\}$
- $f = g$ (and hence n=m) and $f \in mul$ and $\{s_1,\ldots,s_n\} \succ_{rpo}^{mul} \{t_1,\ldots,t_n\}$
- $f = g$ (and hence n=m) and $f \in lex$, $\langle s_1,\ldots,s_n\rangle \succ_{rpo}^{lex} \langle t_1,\ldots,t_n\rangle$, and $s \succ_{rpo} t_j$, for all j in $\{1\ldots n\}$

where $\succ_{rpo}^{lex}$ and $\succ_{rpo}^{mul}$ are, respectively, the lexicographic and multiset extensions of $\succ_{rpo}$ with respect to $=_{mul}$.

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

EXAMPLE 1. If $f \in mul$ and $g \in lex$, and $f \succ_{\mathcal{F}} g \succ_{\mathcal{F}} a \succ_{\mathcal{F}} b$, then $f(b, g(a, b, b), b) \succ_{rpo} g(a, f(a, g(b, a, a), a), a)$.

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 $>$, $\geq$ 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 $>$, $\geq$ and $=$ are interpreted as $\succ_{rpo}$, $\succeq_{rpo}$ and $=_{mul}$ respectively. Then we say that σ satisfies C in the *fixed signature* $(\mathcal{F},\succ_{\mathcal{F}})$.

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, i.e., if solutions are substitutions σ with range $\mathcal{T}(\mathcal{F}')$, and the predicates are interpreted w.r.t. RPO over $\succ_{\mathcal{F}'}$.

Note that if σ is a solution of a constraint C, then this is the case for all substitutions σ' such that $x\sigma' =_{mul} x\sigma$ for all $x \in Dom(\sigma)$. Hence, in the following, we consider all such solutions as a single one. i.e., we work with equivalence classes (w.r.t. $=_{mul}$) of solutions.

EXAMPLE 2. Let $\mathcal{F}$ be precedence $f \succ_{\mathcal{F}} g \succ_{\mathcal{F}} a$, and let S be the constraint $\{f(g(x), y) > f(x, g(y)) \;\wedge\; y > x\}$. If $f \in lex$, S has the solution $\sigma = \{x \mapsto a,\ y \mapsto f(a, a)\}$. If $f \in mul$, S has no solution.

EXAMPLE 3. Let $\mathcal{F}$ be the precedence $f \succ_{\mathcal{F}} g \succ_{\mathcal{F}} a$, and let S be the constraint $\{f(x) > g(y) > x > y\}$. With fixed signature semantics S has no solution. In the extended signature $\mathcal{F}' = f \succ_{\mathcal{F}} g \succ_{\mathcal{F}} a \succ_{\mathcal{F}} 0$, we have the solution $\sigma = \{x \mapsto g(0),\ y \mapsto a\}$.

4. SOLVED FORMS

An ordering constraint C can be equivalently expressed without negation since $s \not> t$ and $s \not\geq t$ can be written as $t \geq s$ and $t > s$ respectively, and $s \neq t$ as $t > s \vee s > t$, while preserving the set of solutions. This is true since, for any pair of ground terms s and t, either $s \succ_{rpo} t$ or $t \succ_{rpo} s$ or $s =_{mul} t$.

After eliminating negation, C can be put into disjunctive normal form in the usual way, and hence satisfiability has to be checked only for conjunctive constraints without negation.

In the following we will deal with such conjunctions expressed as sets of *atoms*, that is, equations and inequations between terms.

The (non-confluent) rewrite system R given below operates on such sets of atoms. In its definition, the following notation is used. If s is $f(s_1, \ldots, s_n)$ and t is $f(t_1, \ldots, t_n)$, then $grmul(s,t)$ expresses, roughly, all the ways in which the multiset $\{s_1, \ldots, s_n\}$ can be greater than $\{t_1, \ldots, t_n\}$: $grmul(s,t)$ 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 $j \in i+1 \ldots n$. It illustrates the fact that, after removing the proper subset of common (w.r.t. $=_{mul}$) elements on both sides, for each element in $\{t_1, \ldots, t_n\}$ there is a bigger one in $\{s_1, \ldots, s_n\}$.

The set R consists of the following rules:

1.	$S \cup \{ s > f(t_1 \ldots t_n) \}$	$\longrightarrow \; S \cup \{ s > t_1, \; \ldots, \; s > t_n \}$ **if** $top(s) >_{\mathcal{F}} f$
2.	$S \cup \{ s > t \}$	$\longrightarrow \; S \cup S'$ **if** $top(s) = top(t) \in mul$ and $S' \in grmul(s,t)$
3.	$S \cup \{ f(s_1 \ldots s_n) > f(t_1 \ldots t_n) \}$	$\longrightarrow \; S \cup \{ s_1 = t_1, \; \ldots, \; s_{i-1} = t_{i-1}, \; s_i > t_i, \; f(s_1 \ldots s_n) > t_{i+1}, \; \ldots, \; f(s_1 \ldots s_n) > t_n \}$ for $1 \leq i \leq n$ **if** $f \in lex$
4.	$S \cup \{ f(s_1 \ldots s_n) > t \}$	$\longrightarrow \; S \cup \{ s_i \geq t \}$ for $2 \leq i \leq n$ **if** $top(t) = f \in lex$
5.	$S \cup \{ a > t \}$	$\longrightarrow \; \perp$ **if** a is a constant and $top(t) \geq_{\mathcal{F}} a$
6.	$S \cup \{ f(s_1 \ldots s_n) > t \}$	$\longrightarrow \; S \cup \{ s_i \geq t \}$ for $1 \leq i \leq n$ **if** $top(t) >_{\mathcal{F}} f$
7.	$S \cup \{ s = t \}$	$\longrightarrow \; S\sigma$ **if** $\sigma = mgu(s,t')$ for some t' with $t' =_{mul} t$
8.	$S \cup \{ s = t \}$	$\longrightarrow \; \perp$ **if** s unifies with no t' with $t' =_{mul} t$
9.	$S \cup \{ s \geq t \}$	$\longrightarrow \; S \cup \{ s > t \}$ **if** s and t are both non-variable terms
10.	$S \cup \{ s \geq t \}$	$\longrightarrow \; S \cup \{ s = t \}$ **if** s and t are both non-variable terms

Note that the case for $i = 1$ in rule 4 is not needed, since it is covered by rule 3 with also $i = 1$: $s_1 \geq_{rpo} t$ implies $s_1 \geq_{rpo} t_1$ and $\forall j : 2 \geq j \geq m : s > t_j$. Rules 4 and 6 could be merged into a single rule with condition $top(t) >_{\mathcal{F}} f$, but then this optimization of the case $i = 1$ for the case $top(t) =_{\mathcal{F}} f$ is lost.

Let us also remark that rules 9 and 10 split atoms with $\geq$ into $=$ or $>$, only if both sides of the atom are non-variable terms. For simplicity of presentation, here we have chosen for this approach rather than reproducing rules 1 to 6 for atoms with $\geq$, although the latter approach could lead to more efficiency in practice for specific cases like $f(a,x) \geq f(a,y)$, which would be dealt with as $x \geq y$, instead of having two branches with $x = y$ and $x > y$. But, like in most NP-complete problems, there is a large number of optimizations for specific cases and describing all of them at once would deteriorate readability of this article.

As said, the rewrite system R is not confluent. This is due to several reasons. One source of non-confluence is that the application of a single rule to a given atom can produce several (although finitely many) different results (this happens with rules 2, 3, 4, 6, and 7). Another source of non-confluence is that two rules may be applicable to a given atom (rules 3 and 4, rules 9 and 10). These two sources of non-confluence cannot be avoided: as we will see, in fact a given σ is a solution of an atom A if, and only if, σ is a solution of at least one of the results of applying a rewrite step by R to A. Hence the different results can be seen as a disjunction. Here we have chosen to avoid disjunctions in our constraint language, for simplicity, and because in practice it is better to deal with don't know choices and backtracking rather than with the memory consuming breadth-first search methods caused by explicit disjunctions. In what follows we will hence consider that the rules of R will be applied don't know non-deterministically in this sense.

Finally, the rewrite steps can take place on different atoms of the given set S (the operator $\cup$ is associative and commutative). But this amounts to different ways of obtaining essentially the same result, and hence in practice a *strategy* can be applied where in each set exactly one atom is selected for being rewritten. For instance, one could apply a *first fail* principle, where the apparently most restrictive atom (according to some heuristic) is selected for being rewritten first, in order to prune the search space as much as possible.

In the following, we will assume the use of such a strategy selecting the next atom to be rewritten, and hence we will consider only the *normal forms* obtained using this strategy (normal forms are sets of atoms to which no more rewrite steps are applicable). Note that such normal forms have the following properties:

LEMMA 1. *Let S be a set of atoms. Then every normal form of S with respect to R is either $\perp$ or a set of atoms of the form $s > t$ or $s \geq t$ where at least one of s and t is a variable.*

LEMMA 2. *The rewrite system R is terminating for any strategy.*

Proof. Let $>_R$ be the well-founded ordering on sets of atoms S that consists of the lexicographic combination of the following three components:

1. the number of different variables in S
2. the two-fold multiset of the pairs of sizes (the number of symbols) of both sides of atoms in S; for example, for $t_1 > t_2 \wedge t_3 = t_4$ the two-fold multiset would be $\{\{s_1, s_2\}, \{s_3, s_4\}\}$, where each s_i denotes the size of t_i.
3. the number of atoms $s \geq t$ in S.

where the first and third components are compared by the standard ordering $>_N$ on natural numbers, and the second one by its two-fold multiset extension. It is easy to check that, for each rule, its application decreases S w.r.t. $>_R$: all rules except 7, 9, and 10 decrease w.r.t. the second component without changing the first one; for rule 7 this is also the case whenever it does not decrease the first component; finally, rules 9 and 10 decrease the third component without changing the first two ones. ∎

LEMMA 3. *The rules given in R are correct and complete for any strategy, that is, for every set of atoms S, a ground substitution σ is a solution of S if, and only if, it is a solution of at least one of the normal forms of S with respect to the given strategy.*

Proof. Let $s = f(s_1, \ldots, s_m)$ and $t = g(t_1, \ldots, t_n)$ be two terms, let A be an atom with sides s and t, and let σ be a ground substitution. We show that σ is a solution of A iff it is a solution of one of the constraints obtained by rewriting A in one step. Soundness and completeness then follow from termination by induction on the number of steps.

If A is $s > t$, then we have the following cases:

1. $f \succ_{\mathcal{F}} g$.

   Then indeed $s\sigma \succ_{rpo} t\sigma$ iff $s\sigma \succ_{rpo} t_i\sigma$ for all i. This corresponds to rule 1.

2. $f = g \in lex$.

   Then $s\sigma \succ_{rpo} t\sigma$ iff $s_i\sigma \succeq_{rpo} t\sigma$ for some i or $\langle s_1\sigma, \ldots, s_n\sigma\rangle \succ_{rpo}^{lex} \langle t_1\sigma, \ldots, t_n\sigma\rangle$, and $s\sigma \succ_{rpo} t_j\sigma$ for all j with $1 \leq j \leq n$. This corresponds to rules 3 and 4.

3. $f = g \in mul$.

   This case is treated in a similar way by rule 2.

4. $g \succ_{\mathcal{F}} f$.

   Then indeed $s\sigma \succ_{rpo} t\sigma$ iff $s_i\sigma \succeq_{rpo} t\sigma$ for some i. This corresponds to rules 5 and 6.

If A is $s \geq t$ the result is trivial, since such atoms can only be rewritten into $s > t$ and $s = t$. Finally, for the case where A is $s = t$, we have to check that indeed $s\sigma =_{mul} t\sigma$ implies $s\sigma = t'\sigma$ for some t' with $t' =_{mul} t$ (the reverse implication is obvious). This follows by induction on the size of t. For example, if t is a variable (with $t \neq s$ and hence t does not occur in s) and $s\sigma = f(a, b) =_{mul} f(b, a) = t\sigma$, then the solution σ', where $s\sigma' = f(a, b)$ and $t\sigma' = f(a, b)$ that is equivalent (w.r.t. $=_{mul}$) to σ is preserved. ∎

Now we have reduced the satisfiability problem of general constraints to the one of constraints that are normal forms w.r.t. R. Unfortunately, this problem is still not easy (in fact it is still NP-complete). Our aim is hence to further transform such normal forms in order to obtain real *solved forms* for which deciding their satisfiability is simple. According to the notion we introduce below, a solved form is a normal form with respect to R where moreover also a form of transitivity through variables has been considered.

DEFINITION 1. Let S be a set of atoms. An atom $s > t$ is a *consequence by transitivity* of S if for some variable x and non-variable term s, either $\{s > x, x > t\} \subseteq S$ or $\{s \geq x, x > t\} \subseteq S$ or $\{s > x, x \geq t\} \subseteq S$. Similarly, $s \geq t$ is a consequence by transitivity of S if $\{s \geq x, x \geq t\} \subseteq S$ for some variable x and non-variable term s.

But we do not require such consequences by transitivity always to be really added to S. It suffices that they become *redundant* w.r.t. S. Roughly, an atom is redundant w.r.t. S if it follows from S using the RPO rules. The following formal definition of redundancy amounts to the usual definition of RPO for terms with variables, extended with the use of information coming from S (case 1):

DEFINITION 2. Let s and t be terms and let S be a set of atoms.

- The atom $s > t$ *is redundant w.r.t.* S, written $s \succ^S_{rpo} t$ if
  1. $s > u \in S$ and t is a subterm of u or $s \geq u \in S$ and t is a proper subterm of u or
  2. $s|_p \succeq^S_{rpo} t$ for some $p \neq \lambda$ or
  3. s and t are $f(s_1, \ldots, s_m)$ and $g(t_1, \ldots, t_n)$ resp. and
     (a) $f \succ_{\mathcal{F}} g$, and $s \succ^S_{rpo} t_j$, for all j with $1 \leq j \leq n$ or
     (b) $f = g$, $f \in lex$, $\langle s_1, \ldots, s_n\rangle \succ^{S\ lex}_{rpo} \langle t_1, \ldots, t_n\rangle$, and $s \succ^S_{rpo} t_j$, for all j with $1 \leq j \leq n$ or
     (c) $f = g$, $f \in mul$, $\{s_1, \ldots, s_n\} \succ^{S\ mul}_{rpo} \{t_1, \ldots, t_n\}$
- The atom $s = t$ *is redundant with respect to* S, written $s =^S_{mul} t$ if exists $s' = t' \in S$ such that $s =_{mul} s'$ and $t =_{mul} t'$

where $s \succeq^S_{rpo} t$ denotes $s \succ^S_{rpo} t \vee s =^S_{mul} t$ and $\succ^{S\ lex}_{rpo}$ and $\succ^{S\ mul}_{rpo}$ are the standard lexicographic and multiset extensions of $\succ^S_{rpo}$ as in the ones defined before for RPO[2].

It is not difficult to see that, as for the usual RPO, for given s, t and S it can be checked in quadratic time whether $s \succ^S_{rpo} t$ by using a dynamic programming scheme (see also [11]).

We now define another relation $\Rightarrow$ on sets of atoms. Apart from the rewrite steps with R, it also includes adding (non-redundant) consequences by transitivity and removing redundant atoms.

DEFINITION 3. Let S and S' be sets of atoms and let A be an atom. The relation $\Rightarrow$ is the smallest relation on sets of atoms such that

1. $S \cup \{A\} \Rightarrow S$ if A is redundant w.r.t. S
2. $S \Rightarrow S'$ if $S \rightarrow_R S'$ and no atom A in S is redundant w.r.t. $S \setminus \{A\}$
3. $S \Rightarrow S \cup \{A\}$ if A is a consequence by transitivity of S and no atom in $S \cup \{A\}$ is redundant w.r.t. S.

Furthermore, S' is a *solved form of* S if it is a normal form of S w.r.t. $\Rightarrow$ (i.e. $S \Rightarrow \ldots \Rightarrow S'$ and $S' \Rightarrow S''$ for no S'').

[2]By means of a specific definition for $\succeq^S_{rpo}$ a slightly stronger notion of redundancy can be obtained. For example, if $x \geq x'$ and $y \geq y'$ are in S then one would like to have $f(x, y) \succeq^S_{rpo} f(x', y')$ (independently of whether $f \in lex$ or $f \in mul$). Such stronger redundancy notions can be rather straightforwardly included in our framework but are not needed for our results (like termination of $\Rightarrow$) and are hence left out here for brevity of presentation.

We remind the reader that the second kind of steps (the ones with R) are applied on the selected atom w.r.t. the given strategy, and may be subject to backtracking. Backtracking is not necessary for the other two. Note that according to this definition, redundant atoms are eliminated eagerly. This is crucial for efficiency in practice. It also ensures that no atom is added more than once in a sequence of steps by $\Rightarrow$, which is needed for proving termination of $\Rightarrow$. In section 7 we will comment on some further non-trivial implementation issues.

Since the definition of redundancy is just an extension of the usual RPO with variables, the following two results are not surprising:

LEMMA 4. *If by one step of R without rule 7 the atom A rewrites into a set of atoms S then A is redundant w.r.t. S.*

LEMMA 5. *Let S be a set of atoms and let A be an atom that is redundant in S. If σ is a solution of S, then σ is a solution of A.*

The following lemma states that redundancy is preserved when removing a redundant atom. Intuitively this is clear, and indeed its proof is a simple case analysis and induction w.r.t. the size of A:

LEMMA 6. *Let A and B be atoms and let S be a set of atoms. If B is redundant w.r.t. S and A is redundant w.r.t. $S \cup \{B\}$ then A is redundant w.r.t. S.*

LEMMA 7. *Let A be an atom and let S and S' be sets of atoms such that A is redundant w.r.t. S. If $S \Rightarrow S'$ such that no step with rule 7 of R is applied, then A is redundant w.r.t. S'.*

Proof. If $S \Rightarrow S'$ by the first case, i.e., by removing a redundant atom, then the result holds by Lemma 6.

If $S \Rightarrow S'$ by the second case, i.e., by one step by R without rule 7, of the form $\{B\} \rightarrow_R S''$ then by Lemma 4 we have that B is redundant in S'' and hence in S' (which is a superset of S''). Then, by Lemma 6, we have that A is redundant in S'.

If $S \Rightarrow S'$ by the third case, i.e., by adding a consequence by transitivity, then the result trivially holds. ∎

LEMMA 8. *The rewrite relation $\Rightarrow$ is terminating.*

Proof. Each time rule 7 of R is applied (in a non-trivial way) the number of variables strictly decreases, which cannot happen infinitely many times since no new variables are ever introduced. Hence it suffices to show termination of the remaining steps. All other steps by R add a finite number of relations between (not necessarily proper) subterms of terms occurring in S. This is also the case when consequences by transitivity are added. Since the number of relations between subterms occuring in a given finite S is finite, termination follows if repeated work is avoided, that is, if no atom is added that has already belonged to some ancestor. By Lemma 7 such repeated atoms are redundant and hence cannot be added. ∎

LEMMA 9. *Let S be a set of atoms. The rewrite relation $\Rightarrow$ is correct and complete for any strategy for R, that is, a ground substitution σ is a solution of S if, and only if, it is a solution of at least one of the normal forms of S with respect to the given strategy for R.*

Proof. Removing a redundant atom, as well as adding a consequence by transitivity preserves the set of solutions. Soundness and completeness for R has already been proved in Lemma 3. Hence the result follows by termination of $\Rightarrow$. ∎

5. CYCLES AND SATISFIABILITY OVER EXTENDED SIGNATURES

DEFINITION 4. If s and t are terms, we denote by $t[s]$ ($t[s]_p$) that s is a subterm of t (at position p), see, e.g., [5]. A set of inequalities S *has a cycle* of length n if

$$\{\ x \#_1 t_1[x_1]_{p_1},\quad x_1 \#_2 t_2[x_2]_{p_2},\ \ldots,\quad x_{n-1} \#_n t_n[x]_{p_n}\ \} \subseteq S$$

where $\#_i \in \{>, \geq\}$ for $i \in 1 \ldots n$, and for some i either $p_i \neq \lambda$ or $\#_i$ is $>$.

EXAMPLE 4. Let $\mathcal{F}$ be $\{f,\ 0\}$, and let S be $\{x > y,\ \ y \geq x,\ \ x \geq f(y)\}$. S contains two cycles: $\{x > y,\ y \geq x\}$ where $\#_1$ is $>$, and $\{y \geq x,\ x \geq f(y)\}$ where $p_2 \neq \lambda$.

LEMMA 10. *Let S be a set of atoms with a cycle. Then S is unsatisfiable.*

DEFINITION 5. Let S be a solved form. We define $\geq_v$ as the smallest reflexive transitive relation on $\mathcal{V}ars(S)$ such that $x \geq_v y$ whenever $x > t[y] \in S$, or $x \geq t[y] \in S$. Furthermore, let $>_v$ be its strict part, that is, $x >_v y$ whenever $x \geq_v y$ but not $y \geq_v x$, and let $=_v$ denote the equivalence relation induced by $\geq_v$, that is, $x =_v y$ whenever $x \geq_v y$ and $y \geq_v x$.

LEMMA 11. *Let S be a solved form. Then $>_v$ is a well-founded strict partial ordering on $\mathcal{V}ars(S)$. Furthermore, if S has no cycles and $x =_v y$ then there are atoms of the form $x_1 \geq x_2, x_2 \geq x_3, \ldots, x_{n-1} \geq x_n, x_n \geq x_1$ in S where all x_i are variables, and $\{x, y\} \subseteq \{x_1, \ldots, x_n\}$, and hence $x\sigma =_{mul} y\sigma$ for all solutions σ of S.*

LEMMA 12. *Let S be solved form with no cycles such that $x =_v y$. If $\{\ \ s \#_1 x,\ \ y \#_2 t\ \ \} \subseteq S$ where s is not a variable and $\#_1, \#_2 \in \{>, \geq\}$, then the corresponding consequence by transitivity $s \#_3 t$ inferred if x and y were the same variable is redundant in S.*

Proof. Assume $\{x, y\} \subseteq \{x_1, \ldots, x_n\}$ as in the previous lemma. We show something slightly stronger: $s \#_3 t$ is redundant whenever
$\{x_1 \geq x_2, x_2 \geq x_3, \ldots, x_{n-1} \geq x_n, x_n \#_2 t\} \subseteq S$
and $s \#_1 x_1$ is redundant in S (i.e., we do not require $s \#_1 x_1$ to belong to S). We proceed by induction on n. It suffices to show that if $s \#_1 x_n$ is redundant in S, then $s \#_3 t$ is redundant in S.

The atom $s \#_1 x_n$ can only be redundant in S for two reasons. If $x_n \in \mathcal{V}ars(s)$ then $s \#_3 t$ is redundant in S due to $x_n \#_2 t$. Otherwise, $s' \#_4 x_n$ is in S, for some (not necessarily proper) subterm s' of s and an appropriate $\#_4$. Since S is a solved form, then the consequence $s' \#_5 t$ of $s' \#_4 x_n$ and $x_n \#_2 t$ is redundant in S. It is easy to check that in all cases from this it follows that $s \#_3 t$ is redundant as well. ∎

LEMMA 13. *Let S be a solved form with no cycles, let x be maximal w.r.t. $>_v$ in $\mathcal{V}ars(S)$, and let V be the set of variables equivalent w.r.t. $=_v$ to x, that is, $V = \{x' \mid x' \in \mathcal{V}ars(S) \wedge x' =_v x\}$.*

Furthermore, let S' be the set obtained by removing from S all atoms where some variable of V occurs, i.e., $S' = \{A \mid A \in S \wedge \mathcal{V}ars(A) \cap V = \emptyset\}$.

Then: (i) if some atom A such that $\mathcal{V}ars(A) \cap V = \emptyset$ is redundant in S, then it is also redundant in S', and (ii) S' is a solved form with no cycles.

Proof. For part (i), it is easy to see by case analysis on the definition of redundancy, that the redundancy of such an atom A can only follow from atoms without variables of V: the only case where atoms with some variable x from V can be useful is case 1: from an atom like $y > f(x,z)$, relations without x like $y > z$ can be proved redundant, but this cannot happen if x is a maximal variable.

Regarding part (ii), since S' is obtained by removing atoms from S, clearly (a) S' is still in normal form w.r.t. R; (b) S' has no cycles, and (c) no redundant atoms remain to be removed from S', since such an atom would already have been redundant in S. By part (i), every consequence by transitivity A of S' (which was redundant in S) is still redundant in S'. ∎

DEFINITION 6. 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 be maximal w.r.t. $>_v$ in $\mathcal{V}ars(S)$, and let V be the set of variables equivalent w.r.t. $=_v$ to x, that is, $V = \{x' \mid x' \in \mathcal{V}ars(S) \wedge x' =_v x\}$. Let σ_x be the partial substitution defined for the variables of S that are smaller than x w.r.t. $>_v$ (and hence smaller than all $x' \in V$).

Then, for each $x \in V$ we define $x\sigma$ to be the maximal element w.r.t. $\succ_{rpo}$ of the set of ground terms:

$$\{0\} \ \cup \ \{f(s\sigma_x) \mid x' > s \in S \ \wedge \ x' \in V\} \ \cup \ \{s\sigma_x \mid x' \geq s \in S \ \wedge \ x' \in V\}$$

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 10, 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 6 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. Let S' be obtained be removing all atoms with variables of the set V of variables equivalent to x w.r.t. $=_v$. Then by Lemma 13 S' is a solved form with no cycles.

Now let σ' be σ with its domain restricted to $\mathcal{V}ars(S) \setminus V$. 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 $s\sigma \succ_{rpo} t\sigma$ ($s\sigma \succeq_{rpo} t\sigma$) for the relations $s > t$ ($s \geq t$) where some variable of V appears in s or t. In the following, all these variables of V will be denoted by x (this can be done since they are assigned the same solution). Other variables not in V will be denoted by y and z.

There are six cases:

1. $x > t[x]$, or $x \geq t[x]_p$ with $p \neq \lambda$. No such inequations exist since these atoms form cycles (if the x denote different equivalent variables x' and x'', they form cycles together with the relations making x' and x'' equivalent).

2. $t[x]_p > x$ with $p \neq \lambda$ or $t[x] \geq x$. Then clearly $t\sigma[x\sigma]_p \succ_{rpo} x\sigma$ and $t\sigma[x\sigma] \succeq_{rpo} x\sigma$ by the subterm property of RPO.

3. $y > t[x]$, or $y \geq t[x]$. No such inequation exists since x is maximal.

4. $x > t$ or $x \geq t$ where $x \notin \mathcal{V}ars(t)$. Then $x\sigma \succ_{rpo} t\sigma$ ($x\sigma \succeq_{rpo} t\sigma$) holds by the construction of σ.

5. $s > x$ or $s \geq x$ with non-variable s and where $x \notin \mathcal{V}ars(s)$.

   Let us show $s\sigma \succ_{rpo} x\sigma$ for the atoms $s > x$ (the case $s \geq x$ is similar). We know $x\sigma$ is the maximal term of $\{0\} \cup \{f(t\sigma) \mid x > t \in S\} \cup \{t\sigma \mid x \geq t \in S\}$. By definition of solved form, for all $x > t$ and $x \geq t$ in S, the atom $s > t$ is a consequence by transitivity and hence redundant in S and, by Lemma 13, also in S' so by Lemma 5 $s\sigma' \succ_{rpo} t\sigma'$. Futhermore, since $top(s) \succ_{\mathcal{F}} f$ this implies $s\sigma \succ_{rpo} f(t\sigma)$ for all $x > t \in S$, from which $s\sigma \succ_{rpo} x\sigma$ follows.

6. $s[x]_p > y$ or $s[x]_p \geq y$ where $p \neq \lambda$.

   Let y be the minimal variable w.r.t. $>_v$ such that $s\sigma \not\succ_{rpo} y\sigma$ for some atom $s[x]_p > y$ or $s\sigma \not\succeq_{rpo} y\sigma$ for some atom $s[x]_p \geq y$. If we denote by y all variables equivalent w.r.t. $=_v$ to y, then we know $y\sigma$ is the maximal term of $\{0\} \cup \{f(t\sigma) \mid y > t \in S\} \cup \{t\sigma \mid y \geq t \in S\}$. By definition of solved form, for all $y > t$ and $y \geq t$ in S, the atom $s > t$ (or $s \geq t$) is a consequence by transitivity and hence redundant in S. In the atoms $s' > t'$ or $s' \geq t'$ of S from which this redundancy follows, s' is a subterm of s and t' contains some subterm of t, as can be seen in Definition 2. Moreover, S is a solved form and hence either s' or t' is a variable. Hence, the redundancy of $s > t$ (or $s \geq t$) follows from (i) atoms without x or atoms analysed in one of the previous cases of this proof, or (ii) atoms $s'[x] > z$ or $s'[x] \geq z$ where $y >_v z$ because z is a variable occurring in t and $y > t$ or $y \geq t$ belongs to S.

   For the atoms of type (i) we already know that σ is a solution. For the atoms of type (ii), $s'[x]\sigma \succ_{rpo} z\sigma$ (or $s'[x]\sigma \succeq_{rpo} z\sigma$) follows by the induction hypothesis since $y >_v z$. Therefore, by Lemma 5, we have $s\sigma \succ_{rpo} t\sigma$ (or $s\sigma \succeq_{rpo} t\sigma$), and since $top(s) \succ_{\mathcal{F}} f$ this implies $s\sigma \succ_{rpo} f(t\sigma)$ for all $y > t \in S$, from which $s\sigma \succ_{rpo} y\sigma$ follows.

∎

EXAMPLE 5. Let $\mathcal{F}$ be the signature $g \succ_{\mathcal{F}} f \succ_{\mathcal{F}} 0$ where $\{g\} = mul$, and let S_0 be the constraint $\{g(x, f(f(y))) > f(f(z)),\quad z \geq g(f(x), f(x)),\quad x \geq y\ \}$. Below a derivation from S_0 to one of its solved forms is shown. Note that the strategy applies eager redundancy detection.

$$
\begin{array}{l}
\{\ \boxed{g(x, f(f(y))) > f(f(z))}\quad z \geq g(f(x), f(x)), \quad x > y\ \} \\
\qquad\qquad\downarrow \textit{ Rule } 1 \\
\{\ \boxed{g(x, f(f(y))) > f(z)}\quad z \geq g(f(x), f(x)), \quad x > y\ \} \\
\qquad\qquad\downarrow \textit{ Rule } 1 \\
\{\ \boxed{g(x, f(f(y))) > z, \quad z \geq g(f(x), f(x))}\quad x > y\ \} \\
\qquad\qquad\qquad\qquad\swarrow \textit{ Transitivity} \\
\{\ \boxed{g(x, f(f(y))) > g(f(x), f(x))}\quad g(x, f(f(y))) > z, \quad z \geq g(f(x), f(x)), \quad x > y\ \} \\
\qquad\qquad\swarrow \textit{ Rule } 2 \\
\{\ \boxed{f(f(y)) > f(x)}\quad g(x, f(f(y))) > z, \quad z \geq g(f(x), f(x)), \quad x > y\ \} \\
\qquad\swarrow \textit{ Rule } 3 \\
\{\ \boxed{f(y) > x}\quad g(x, f(f(y))) > z, \quad z \geq g(f(x), f(x)), \quad \boxed{x > y}\ \} \\
\qquad\swarrow \textit{ Transitivity} \\
\{\ f(y) > y, \quad f(y) > x, \quad g(x, f(f(y))) > z, \quad z \geq g(f(x), f(x)), \quad x > y\ \}
\end{array}
$$

In this example, there is only one step where a rule application could have introduced a disjunction, namely the application of rule 2. But note that the other possible applications of this rule, making the multiset $\{x, f(f(y))\}$ larger than the multiset $\{f(x), f(x)\}$ would immediately lead to an unsatisfiable atomic constraint (in this case, $x = f(x)$ or $x > f(x)$).

In any case, since the solved form obtained contains no cycle, the constraint is satisfiable, and hence no other aplications of rule 2 need to be considered.

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 6. 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 (because of the isomorphism between terms and natural numbers where the term 0 corresponds to the natural number 0, and $f(t)$ corresponds to the successor of the natural number correponding to t), and 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 7. Let S be a set of atoms, let x be a variable and let s be a non-variable term.

The atom $s > f(t)$ is a *consequence by transitivity w.r.t. fixed signatures* of S if $\{s > x, x > t\} \subseteq S$. Similarly, $s > t$ is a consequence by transitivity of S if $\{s \geq x, x > t\} \subseteq S$ or $\{s > x, x \geq t\} \subseteq S$. Finally, $s \geq t$ is a consequence by transitivity of S if $\{s \geq x, x \geq t\} \subseteq S$.
In this subsection, the notions of consequence by transitivity are assumed to be w.r.t. fixed signatures, and the rewrite relation $\Rightarrow$, as well as the notion of solved form are considered w.r.t. this adapted notion.

EXAMPLE 7. (Example 6 continued) Rewriting S w.r.t. the new notion of $\Rightarrow$ gives the following: from $y > z > 0$ we get $y > f(0)$, and $f(x) > y \wedge y > f(0)$ gives $f(x) > f(f(0))$ which simplifies by R into $x > f(0)$. From $f(f(0)) > x \wedge x > f(0)$ we get $f(f(0)) > f(f(0))$ whose unique normal form w.r.t. R is $\perp$.

EXAMPLE 8. Consider S to be $f(0) \geq x \geq y > z > 0$. From $f(0) \geq x \geq y$ we get $f(0) \geq y$ which with $y > z$ gives $f(0) > z$. From $f(0) > z$ and $z > 0$ we get $f(0) > f(0)$ whose unique normal form w.r.t. R is $\perp$.

EXAMPLE 9. (Example 5 continued) Note that the initial constraint of Example 5 with a fixed signature would lead to a cycle (in all branches), because the second application of transitivity would produce the atom $f(y) > f(y)$, and hence the initial constraint would be proved unsatisfiable.

Clearly, since only the notion of consequence by transitivity has changed, the analogous results of Lemmas 8 and 10 of the previous section still hold, and again a set of atoms S is satisfiable if, and only if, one of its solved forms in this new sense is satisfiable (this corresponds to Lemma 9). The following is the analogous of Lemma 8:

LEMMA 14. *Let $\Rightarrow$ be the rewrite relation of Definition 3 with the transitivity notion of Definition 7. Then the relation $\Rightarrow$ is terminating.*

Proof. The proof is similar to the one of Lemma 8. Again it is based onthe fact that the set of atoms that can be generated is finite, but here it is the set of atoms $s > t$, $s > f(t)$, $s \geq t$, and $s \geq f(t)$ where s and t are subterms occurring in the initial constraint, i.e., now there can be (at most) one additional f on top of the right hand sides of atoms. To see that no new terms with two f's, like $f(f(t))$, can be generated, note that from $s > x$ and $x > t$ one generates $s > f(t)$ only if s is a non-variable term. Hence the atom $s > f(t)$ can only be applied in further transitivity inferences if it is decomposed by the rules of R. Since f is the smallest non-constant symbol, this decomposition always removes the topmost f. The rest of the proof is exactly like the one of Lemma 8. ∎

THEOREM 2. *Let $\mathcal{F}$ be a well-ended signature, and 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 $0 > x \notin S$.*

Proof. The proof follows the same ideas as the one of Theorem 1, and the solution is built as before (but now f and 0 are in $\mathcal{F}$). Note that Lemma 13 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. $s > x$ or $s \geq x$ with non-variable s and where $x \notin \mathcal{V}ars(s)$.

   Let us show $s\sigma \succ_{rpo} x\sigma$ for the atoms $s > x$ (the case $s \geq x$ is similar). We know $x\sigma$ is the maximal term of $\{0\} \cup \{f(t\sigma) \mid x > t \in S\} \cup \{t\sigma \mid x \geq t \in S\}$. By definition of solved form, for all $x > t$ ($x \geq t$) in S, the atom $s > f(t)$ ($s > t$) is a consequence by transitivity and hence redundant in S and, by Lemma 13, also in S'. Since $s > 0$ is not in S, by Lemma 5 $s\sigma' \succ_{rpo} f(t)\sigma'$ ($s\sigma' \succ_{rpo} t\sigma'$) and also $s\sigma \succ_{rpo} f(t\sigma)$ ($s\sigma \succ_{rpo} t\sigma$), from which $s\sigma \succ_{rpo} x\sigma$ follows.

6. $s[x]_p > y$ or $s[x]_p \geq y$ where $p \neq \lambda$.

   Let y be the minimal variable w.r.t. $>_v$ such that $s\sigma \not\succ_{rpo} y\sigma$ for some atom $s[x]_p > y$ or $s\sigma \not\succeq_{rpo} y\sigma$ for some atom $s[x]_p \geq y$. If we denote by y all variables equivalent w.r.t. $=_v$ to y, then we know $y\sigma$ is the maximal term of $\{0\} \cup \{f(t\sigma) \mid \ y > t \in S\} \cup \{t\sigma \mid \ y \geq t \in S\}$. By definition of solved form, for all $y > t$ and $y \geq t$ in S, the atom $s > f(t)$ (or $s > t$ or $s \geq t$) is a consequence by transitivity and hence redundant in S. The atoms of S from which this redundancy follows can be (i) atoms without x or atoms analysed in one of the previous cases of this proof, or (ii) atoms $s'[x] > z$ or $s'[x] \geq z$ where $y >_v z$. For the atoms of type (i) we already know that σ is a solution. For the atoms of type (ii), $s'[x]\sigma \succ_{rpo} z\sigma$ (or $s'[x]\sigma \succeq_{rpo} z\sigma$) follows by the induction hypothesis since $y >_v z$. Therefore, by Lemma 5, we have $s\sigma \succ_{rpo} f(t\sigma)$ (or $s\sigma \succ_{rpo} t\sigma$ or $s\sigma \succeq_{rpo} t\sigma$), from which $s\sigma \succ_{rpo} y\sigma$ (or $s\sigma \succeq_{rpo} y\sigma$) follows.

∎

6.2. At Most One Constant below a Non-unary $\mathbf{f} \in$ 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 $f(0,\ldots,0,t)$: for example, in the definition of consequence by transitivity, if $s > x \in S$ and $x > t \in S$ for some non-variable s, a variable x, and term t, then $s > f(0,\ldots,0,t)$ is a consequence by transitivity of S. Similarly, the minimal substitution σ for a solved form with no cycles S is now defined by taking successors in that way: $x\sigma$ is the maximal element w.r.t. $\succ_{rpo}$ of the set of ground terms:

$$\{0\} \ \cup \ \{f(0,\ldots,0,s\sigma_x) \mid x' > s \in S \ \wedge \ x' \in V\} \ \cup \ \{s\sigma_x \mid x' \geq s \in S \ \wedge \ x' \in V\}$$

where V is as before. 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. For each signature, it is easy to identify ω.

Ground terms below ω are called *natural terms*; the remaining ones (including ω) are called *non-natural.* In a given set of atoms S, a (possibly non-ground) term t is called natural if, for every solution σ of S, the ground term $t\sigma$ is natural; it is non-natural if $t\sigma$ is non-natural for every solution σ.

Clearly, notions of consequence by transitivity like $s > f(t)$ from $s > x$ and $x > t$ applied in the Subsection 6.1 are now correct and complete (the set of solutions is preserved) if t is non-natural, but not necessarily if for some solution σ the term $t\sigma$ is natural. For example, if $f \succ_{\mathcal{F}} a \succ_{\mathcal{F}} b \succ_{\mathcal{F}} c$, then from $a > x$ and $x > y$ we cannot infer $a > f(y)$, since for the solution where $x\sigma = b$ and $y\sigma = c$, the term $y\sigma$ is natural.

In order to classify all non-ground terms as either natural or non-natural, 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 for every normal form S with respect to R_ω it is easy to see that all terms t are either natural or non-natural: t is natural if, and only if, it contains only symbols smaller than g and variables x for which $\omega > x \in S$. Otherwise, it is non-natural. 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 non-natural. This suffices for adding the consequences by transitivity:

DEFINITION 8. Let S be a set of atoms, let x be a variable, s a non-variable term, and t a non-natural term.

The atom $s > f(t)$ is a *consequence by transitivity* (of the non-natural part) of S if $\{s > x, x > t\} \subseteq S$. Similarly, $s > t$ is a consequence by transitivity of S if $\{s \geq x, x > t\} \subseteq S$ or $\{s > x, x \geq t\} \subseteq S$. Finally, $s \geq t$ is a consequence by transitivity of S if $\{s \geq x, x \geq t\} \subseteq S$.
In this subsection, the notions of consequence by transitivity are assumed to be of the non-natural part, and the rewrite relation $\Rightarrow$, as well as the notion of solved form are considered w.r.t. this adapted notion, and the rules of $R \cup R_\omega$.

This allows us to split every set of atoms S in two disjoint parts: the *natural part* S_N, and the remaining *non-natural* one S_ω, i.e., $S_N = \{s \# t \mid s \# t \in S \text{ and } s \text{ and } t \text{ are natural}\,\}$ 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 $x + k * (n+1)$, denoting n, as before, the number of constants a_i. 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.

The following lemma states what happens with one kind of atoms built from a natural and a non-natural term:

LEMMA 15. *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 8) over $(\mathcal{F}, \succ_{\mathcal{F}})$. Then there is no $s > t$ or $s \geq t$ in S for s that is natural and t that is non-natural.*

Proof. Assume such an $s > t$ is in S (the proof for atoms $s \geq t$ is similar).

If s is not a variable, then it is $s > x$ where $x > \omega \in S$. Then, by transitivity of the non-natural part, $s > f(\omega)$ is redundant in S. This cannot be the case if s is ground, since $f(\omega) \succ_{rpo} s$, and it could only be redundant by relations that would reduce to $\perp$ by R. If s is non-ground, it is of the form $f^n(y)$ for some $n > 0$, where $\omega > y \in S$, and $s > f(\omega)$ cannot be redundant in S either, since by Definition 2 it would mean that $y > \omega$ is redundant in S (i.e. $y > u[\omega] \in S$) which leads to $\perp$ by transitivity from $\omega > y$. If s is a variable x, then $\omega > x \in S$ and by transitivity $\omega > f(t)$ is redundant in S, which cannot be the case for the same reason as for $s > f(\omega)$. ∎

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 8) 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 be a solution of S_N. 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, in a similar way as it was done for Theorem 1.

First, note that for atoms $s > t$ or $s \geq t$ 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 15, it only remains to check the inequalities between non-natural terms.

Note that Lemma 13 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. $s > x$ or $s \geq x$ with non-variable s and where $x \notin \mathcal{V}ars(s)$.

   Let us show $s\sigma \succ_{rpo} x\sigma$ for the atoms $s > x$ (the case $s \geq x$ is similar). There is at least one inequation $x > t$ in S, since $x > \omega \in S$. Then $x\sigma$ is the maximal term w.r.t. $\succ_{rpo}$ of $\{f(t\sigma) \mid x > t \in S\} \cup \{t\sigma \mid x \geq t \in S\}$. By the definition of solved form, $s > f(t)$ $(s > t)$ is redundant in S for all $x > t$ $(x \geq t)$ in S and we conclude as in Theorem 2.

6. $s[x]_p > y$ or $s[x]_p \geq y$ where $p \neq \lambda$.

Let y be the minimal variable w.r.t. $>_v$ such that $s\sigma \not\succ_{rpo} y\sigma$ for some atom $s[x]_p > y$ or $s\sigma \not\succeq_{rpo} y\sigma$ for some atom $s[x]_p \geq y$. We know $y\sigma$ is the maximal term of $\{f(t\sigma) \mid y > t \in S\} \cup \{t\sigma \mid y \geq t \in S\}$ since y is non-natural. By definition of solved form, for all $y > t$ and $y \geq t$ in S, the atom $s > f(t)$ (or $s > t$ or $s \geq t$) is a consequence by transitivity and hence redundant in S, and $s\sigma \succ_{rpo} y\sigma$ or $s\sigma \succeq_{rpo} y\sigma$ 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.

6.4.1. 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:

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

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 we say that $s > f(0,\ldots,0,t)$ is a consequence by transitivity of S. Similarly, the minimal substitution σ for a solved form with no cycles S is now defined taking successors in that way: $x\sigma$ is the maximal element w.r.t. $\succ_{rpo}$ of the set $\{f(0,\ldots,0,t\sigma_x) \mid x' > t \in S \ \wedge\ x' \in V\} \cup \{t\sigma_x \mid x' \geq t \in S \ \wedge\ x' \in V\}$ where V is as before. 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 [7] or the NP one of [13] for S_N, which is normally a minor part of S.

6.4.2. 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:

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

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 $R_{\omega,0}$:

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

Again, 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.

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 subsection 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.

EXAMPLE 10. Let $\mathcal{F}$ be the signature $g \succ_\mathcal{F} f \succ_\mathcal{F} a \succ_\mathcal{F} 0$ where $\{f\} = lex$, and let S_0 be the constraint $\{\ f(x,a) > y,\ \ f(y,0) \geq g(z),\ \ f(x,0) > z,\ \ f(z,g(y)) > g(x)\ \}$. Below a derivation from S_0 to one of its solved forms is shown. Note that the strategy applies eager redundancy detection, and that we cut off trivially unsatisfiable branches.

$$\{\ f(x,a) > y,\ \ \boxed{f(y,0) \geq g(z)}\ \ f(x,0) > z,\ \ f(z,g(y)) > g(x)\ \}$$

$\downarrow$ *Rule* 6. Other branch with $0 \geq g(z)$: unsatisfiable

$$\{\ \boxed{f(x,a) > y,\ \ y \geq g(z)}\ \ f(x,0) > z,\ \ f(z,g(y)) > g(x)\ \}$$

$\downarrow$ Transitivity: all ground terms $g(z\sigma)$ are non-natural

$$\{\ \boxed{f(x,a) > f(0,g(z))}\ \ f(x,a) > y,\ \ y \geq g(z),\ \ f(x,0) > z,\ \ f(z,g(y)) > g(x)\ \}$$

$\downarrow$ *Rule* 3. Other branches: with $x = 0\ \wedge\ a > g(z)$: unsatisfiable; with Rule 6: $a \geq f(0,g(z))$: unsat.

$$\{\ \overbrace{x > 0,\ \ \boxed{f(x,a) > g(z)}}\ \ f(x,a) > y,\ y \geq g(z),\ \ f(x,0) > z,\ f(z,g(y)) > g(x)\ \}$$

$\swarrow$ *Rule* 6. Other branch with $a \geq g(z)$: unsatisfiable

$$\{\ x > 0,\ \ x \geq g(z),\ \ f(x,a) > y,\ \ y \geq g(z),\ \ f(x,0) > z,\ \ \boxed{f(z,g(y)) > g(x)}\ \}$$

Rule 6. Other branch with $z \geq g(x)$: cycle with $x \geq g(z)$. $\downarrow$

$$\{\ x > 0,\ \ x \geq g(z),\ \ f(x,a) > y,\ \ y \geq g(z),\ \ f(x,0) > z,\ \ \boxed{g(y) \geq g(x)}\ \}$$

Rules 9, 3. Branch with 10, 3: cycle $x = y, f(x,a) > y$. $\swarrow$

$$\{\ x > 0,\ \ x \geq g(z),\ \ f(x,a) > y,\ \ y \geq g(z),\ \ f(x,0) > z,\ \ y > x\ \}$$

Note that, with a slightly more powerful notion of redundancy, where an atom $x > t$ is redundant if $x > s$ is available for some s such that $s \succ_{rpo} t$, in the current constraint the atom $x > 0$ could be removed due to $x > g(z)$. Of course other similar redundancy notions are possible where one atom is subsumed in this way by another one, but checking for this kind of redundancy not always pays off in practice; extending the completeness proofs of our algorithms with such additional redundancy notions is not difficult but makes them a bit longer and more difficult to read.

Now the constraint is in normal form w.r.t. R, and transitivity (see Definition 8) has been applied in all cases where the rightmost term (the term t in Definition 8) is known to be non-natural. But in order to be able to split the constraint into its natural and non-natural parts, we now need to apply rules of $R_{\omega,0}$. A natural heuristic for minimizing the amount of guessing is to apply the four additional rules first for the variable that occurs most frequently, which in this case is x. Note that for this signature ω is $f(a,0)$.

- $x = 0$. Now we have
  $\{\ \boxed{0 > 0}\ \ 0 \geq g(z), \ldots\}$
  unsatisfiable
- $\omega > x > 0$. This leads to unsatisfiability from (transitivity applied to) $f(a,0) > x$ and $x \geq g(z)$, resulting in the unsatisfiable atomic constraint $f(a,0) > f(0,g(z))$.
- $x = \omega$. This again leads to unsatisfiability. From $f(a,0) \geq g(z)$ by rule 6 we get either $a \geq g(z)$ or $0 \geq g(z)$.

- $x > \omega$. Adding $x > f(a,0)$, the atom $x > 0$ becomes redundant and we get:

$$\{\ x > f(a,0),\quad x \geq g(z),\quad \boxed{f(x,a) > y}\quad y \geq g(z),\quad f(x,0) > z,\quad \boxed{y > x}\ \}$$

Transitivity: all ground $x\sigma$ with $x\sigma > \omega$ are non-natural $\downarrow$

$$\{\ \ldots,\quad f(x,a) > y,\quad y \geq g(z),\quad f(x,0) > z,\quad y > x,\quad \boxed{f(x,a) > f(0,x)}\ \}$$

Redundant because $x > 0$ and $f(x,a) > x$ are redundant

$$\{\ x > f(a,0),\quad x \geq g(z),\quad f(x,a) > y,\quad y \geq g(z),\quad f(x,0) > z,\quad y > x\ \}$$

Now we can guess on variable z. Since it now only occurs on small sides of inequations, a good guess may be $z = 0$ (in fact, it is easy to see that in this case, if there is no solution with $z = 0$, then there is no solution at all). Then the atom $f(x,0) > z$ becomes redundant, and we have

$$\{\ x > f(a,0),\quad x \geq g(0),\quad f(x,a) > y,\quad y \geq g(0),\quad y > x\ \}$$

Similarly, when guessing on variable y, only $y > \omega$ is possible, because we have the atom $y > g(0)$ in the constraint. Then only one new consequence $f(x,a) > f(0, f(a,0))$ by transitivity between $f(x,a) > y$ and $y > f(a,0)$ is generated. However, this atom is redundant from $x > f(a,0)$. Since the final constraint has no cycle, it is safisfiable, and hence also the initial constraint S_0 is satisfiable.

7. IMPLEMENTATION ISSUES

Here we discuss some aspects of the implementation of our techniques: an optimization detecting equivalent variables, a *saturation-based* implementation approach, and some experimental results.

7.1. Detecting Equivalent Variables

Since cycle detection is done frequently, at a very low additional cost one can also detect chains of equivalent variables. As stated in Lemma 11, if $x =_v y$ in S then there are atoms of the form $x_1 \geq x_2, x_2 \geq x_3, \ldots, x_{n-1} \geq x_n, x_n \geq x_1$ in S where all x_i are variables, and $\{x, y\} \subseteq \{x_1, \ldots, x_n\}$, and hence $x\sigma =_{mul} y\sigma$ for all solutions σ of S. It seems reasonable to have an additional rule in R collapsing all these variables into the same one, adding equalities $x_1 = x_2, \ldots, x_1 = x_n$ to S or directly unifying all these variables. Such a rule is obviously correct, and will probably help detecting failure situations earlier. In order to obtain more general results, in this paper we did not want to impose the existence of such a rule. In fact one could generalize the rule considering not only equivalence chains among variables, but also among arbitrary terms, and computing their unifiers up to $=_{mul}$.

7.2. 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 consequences by transitivity, 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 be (and probably have to be) used.

In our implementation we deal with three sets of atoms: the *old* ones, that are in normal form w.r.t. R and closed under transitivity, the *new* ones, that are in normal form w.r.t. R, 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 atoms.

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*. Backtracking on the choice of normal form of $s > t$ occurs when a cycle is detected at some point (from our experiments, we have seen that cycle detection should be done eagerly). If T is empty, one atom A of *new* is moved to *old* and all consequences by transitivity between A 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[3].

7.3. 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 our previous best one *Old* (an improvement of [17]) as it was (quite carefully) implemented in the *Saturate* system, for extended signatures. We are not aware of the existence of any other (competitive or experimental) implementations by other researchers.

Times are in miliseconds for Sicstus Prolog 3.7.1 on a SUN Ultra 5. The problem sets and test program are available from the authors. 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 T. 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

[3]Note however, that due to the application of rule 7 of R, it may happen that some atom A in *old* or *new* changes because some of its variables get instantiated. In those cases it may be necessary to move A to T again.

Constraints from problem Abelian Groups				
Threshold	*# Problems*	*Total Time Old*	*Total T. 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 T. 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 solving this problem 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 [13]). 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 [13] 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

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

[2] Hubert Comon, Paliath Narendran, Robert Nieuwenhuis, and Michaël Rusinowitch. Decision problems in ordered rewriting. In *13th IEEE Symposium on Logic in Computer Science (LICS)*, pages 410–422, Indianapolis, USA, June 27–30, 1998.

[3] Hubert Comon and Ralf Treinen. Ordering Constraints on Trees. In *Proc. of Colloquium on Trees in Algebra and Programming (CAAP)*, LNCS 787, pages 1–14, Edinburgh, Scotland, April 1994. Springer-Verlag.

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

[5] 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.

[6] Harald Ganzinger, Robert Nieuwenhuis, and Pilar Nivela. The Saturate System, 1999. Software and documentation available at: `http://www.mpi-sb.mpg.de/SATURATE/Saturate.html`.

[7] J-P. Jouannaud and M. Okada. Satisfiability of systems of ordinal notations with the subterm property is decidable. In *18th International Colloquium Automata, Languages and Programming (ICALP)*, LNCS 510, pages 455–468, Madrid, Spain, July 16–20 1991. Springer-Verlag.

[8] S. Kamin and J.-J. Levy. Two generalizations of the recursive path ordering. Unpublished note, Dept. of Computer Science, Univ. of Illinois, Urbana, IL, 1980.

[9] 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.

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

[11] M. S. Krishnamoorthy and P. Narendran. A note on recursive path ordering. *Theoretical Computer Science*, 40:323–328, 1985.

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

[13] P. Narendran, M. Rusinowitch, and R. Verma. RPO constraint solving is in NP. In G. Gottlob, E. Grandjean, and Katrin Seyr, editors, *12th Int. Conference of the European Association of Computer Science Logic (CSL'98)*, LNCS 1584, pages 385–398, Brno, Czech Republic, August 23–28, 1999. Springer-Verlag.

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

[15] Robert Nieuwenhuis and José Miguel Rivero. Solved forms for path ordering constraints. In P. Narendran and M. Rusinowitch, editors, *Tenth International Conference on Rewriting Techniques and Applications (RTA)*, LNCS 1631, pages 1–15, Trento, Italy, July 2–4, 1999. Springer-Verlag.

[16] Robert Nieuwenhuis and Albert Rubio. Theorem proving with ordering constrained clauses. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, LNAI 607, pages 477–491, Saratoga Springs, New York, 1992.

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

[18] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers (to appear), 2000.