Simple LPO constraint solving methods

Robert Nieuwenhuis*
Technical University of Catalonia
Pau Gargallo 5, 08028 Barcelona, Spain.
E-mail: roberto@lsi.upc.es.

Abstract

We present simple techniques for deciding the satisfiability of lexicographic path ordering constraints under two different semantics: solutions built over the given signature and solutions in extended signatures. For both cases we give the first NP algorithms, which is optimal as we prove the problems to be NP-complete.

We discuss the efficient applicability of the techniques in practice, where, as far as we know, their simply exponential bound improves upon the existing methods, and describe some optimizations.

Keywords: Automatic theorem proving.

1 Terminology

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*). We sometimes write pairs $(\mathcal{F}, \succ_{\mathcal{F}})$. The lexicographic path ordering (LPO) generated by $\succ_{\mathcal{F}}$, denoted $\succ_{lpo}^{\mathcal{F}}$, is a total simplification ordering on $\mathcal{T}(\mathcal{F})$. It is defined as follows: $s = f(s_1, \ldots, s_m) \succ_{lpo}^{\mathcal{F}} g(t_1, \ldots, t_n) = t$ if

1. $s_i \succeq_{lpo}^{\mathcal{F}} t$, for some i with $1 \leq i \leq m$ or
2. $f \succ_{\mathcal{F}} g$, and $s \succ_{lpo}^{\mathcal{F}} t_j$, for all j with $1 \leq j \leq n$ or
3. $f = g$, $\langle s_1, \ldots, s_m \rangle \succ_{lpo}^{\mathcal{F}\ lex} \langle t_1, \ldots, t_n \rangle$, and $s \succ_{lpo}^{\mathcal{F}} t_j$, for all j with $1 \leq j \leq n$

where $\langle s_1, \ldots, s_n \rangle \succ^{lex} \langle t_1, \ldots, t_n \rangle$ for a given $\succ$ if $\exists j \leq n$ s.t. $\forall i < j \;\; s_i = t_i$ and $s_j \succ t_j$.

An *LPO-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 '$\succ$' and '$=$' denoting LPO and syntactic equality respectively. Solving such constraints has interesting applications in rewriting and ordered theorem proving methods in first-order logic.

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 $\succ_{\mathcal{F}}$.

*This work is partially supported by the ESPRIT Basic Research working group 6028, CCL.

A *solution* in $(\mathcal{F}', \succ_{\mathcal{F}'})$ of a constraint C is a substitution σ with range $\mathcal{T}(\mathcal{F}')$ and whose domain is a set of variables containing the variables of C, such that $C\sigma$ evaluates to true under $\succ_{lpo}^{\mathcal{F}'}$. Then we say that C *satisfies* σ in $(\mathcal{F}', \succ_{\mathcal{F}'})$.

The satisfiability of LPO-ordering constraints was first proved to be decidable by Comon [Com90], under the given signature, that is, for solutions in $(\mathcal{F}, \succ_{\mathcal{F}})$.

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. This type of satisfiability was proved decidable in [NR92], where it is applied as the adequate semantics in complete refutation procedures for ordering constrained clauses.

A *simple system* is an expression S of the form $t_1 \ \# \ \dots \ \# \ t_n$, where each $\#$ is either $\succ$ or $=$, where every proper subterm of each t_i is some t_j with $i < j \leq n$, and where t_i and t_j are different terms if $i \neq j$.

A simple system S is a compact representation of a constraint $C(S)$ that is a conjunction of equalities and inequalities. For example, if S is $t_1 \succ t_2 = t_3$ then $C(S)$ is $t_1 \succ t_2 \wedge t_1 \succ t_3 \wedge t_2 = t_3$. If S is a simple system, then we write $t \succ_s t'$ ($t =_s t'$) if $t \succ t'$ (resp. $t = t'$) is a conjunct in $C(S)$, and we write $t \succeq_s t'$ iff $t \succ_s t'$ or $t =_s t'$.

Our simple systems are different from the ones in [Com90] in that we also allow equality relations, but we still require that all subterms of each term appear to the right of it. For example, the following expression is a simple system:

$$g(h(a,x),y) \succ h(a,x) = h(y,b) \succ y = a \succ x = b$$

The above simple system is satisfiable if $a \succ_{\mathcal{F}} b$: then it has the solution $\{x \mapsto b, y \mapsto a\}$.

Some simple systems can very easily be shown to be unsatisfiable, because they contradict properties of LPO or of syntactic equality. These are the ones we will call *trivially bottom*. More formally, a simple system S is trivially bottom if

- $s =_s t$ with $top(s) \neq top(t)$ or
- $f(s_1, \dots, s_p) =_s f(s'_1, \dots, s'_p)$ and $\exists \ i \in 1 \dots p$ s.t. $s_i \neq_s s'_i$ or
- $s =_s t$ and t is a proper subterm of s or vice versa, or
- $f(s_1, \dots, s_p) \succ_s t$ with $top(t) \succ_{\mathcal{F}} f$ and $\nexists \ i \in 1 \dots p$ s.t. $s_i \succeq_s t$ or
- $f(s_1, \dots, s_p) \succ_s f(s'_1, \dots, s'_p)$ and $\langle s_1, \dots, s_p \rangle \not\succ_s^{lex} \ \langle s'_1, \dots, s'_p \rangle$.

It can be decided in polynomial time whether a simple system S is trivally bottom. Note that a simple system contains relations between all its subterms. Therefore, for instance, if $f(s_1, \dots, s_p) =_s f(s'_1, \dots, s'_p)$ in a satisfiable simple system S, it must be the case that $s_i =_s s'_i$ for all i in $1 \dots p$, because otherwise $s_i \succ_s s'_i$ or $s'_i \succ_s s_i$ for some i, which would make S unsatisfiable.

In other words, if S is not trivially bottom, each $s \ \#_s \ t$ for non-variable s and t follows from other relations in S between subterms of s and t. This is also the reason that the satisfiability of such S only depends on the relations $s \#_s t$ where at least one of s or t is a variable (cf. lemma 1.2 below).

It turns out, as we prove in section 2, that in extended signatures *a simple system is satisfiable if and only if it is not trivially bottom.* In section 3 we prove the same result with a slightly different notion of trivially bottom simple systems for solutions over the given signature. Therefore, to decide the satisfiability of an arbitrary constraint C, one only has to express it as a disjunction of simple systems and check whether these are all trivially bottom or not. Reducing constraints into disjunctions of simple systems is not difficult, as we discuss later on.

The following lemmas hold for solutions under extended signatures and also for solutions under the given signature:

Lemma 1.1 A simple system S that is trivially bottom is unsatisfiable.

Lemma 1.2 Let S be a simple system that is not trivially bottom. Then a substitution σ is a solution of S iff it is a solution of all relations $s \#_s t$ such that at least one of s or t is a variable.

2 Satisfiability under extended signatures

In this section we consider satisfiability under extended signatures[1]. In particular, we prove that every simple system which is not trivially bottom has a solution in the extension $(\mathcal{F}_0, \succ_{\mathcal{F}_0})$, where $\mathcal{F}_0$ is $\mathcal{F} \cup \{f, 0\}$ for some unary function symbol f and constant 0 that do not belong to $\mathcal{F}$, and where $\succ_{\mathcal{F}_0}$ is the extension of $\succ_{\mathcal{F}}$ s.t. $g \succ_{\mathcal{F}_0} f \succ_{\mathcal{F}_0} 0$ for every symbol g in $\mathcal{F}$. This also means that a simple system is satisfiable under extended signatures iff it has some solution in $(\mathcal{F}_0, \succ_{\mathcal{F}_0})$.

Theorem 2.1 A simple system S is satisfiable under extended signatures iff it is not trivially bottom.

Proof If S is trivially bottom, then it is not satisfiable. Now suppose S is not trivially bottom. We proceed by induction on the number k of variables in S. If $k = 0$, i.e. S is ground, then by lemma 1.2 S is satisfiable. For the induction step, let x be the leftmost variable in S, i.e. S is of the form $t_1 \# \dots \# t_{k-1} \# x \# t_{k+1} \# \dots \# t_n$ where x is t_k, and no t_j in $t_1 \dots t_{k-1}$ is a variable. We will construct a solution σ in $(\mathcal{F}_0, \succ_{\mathcal{F}_0})$ for S.

Let S' be the simple system obtained by deleting in S all terms t_i with i in $1 \dots n$ such that x appears in t_i. Then it is easy to see that S' is still a non trivially bottom simple system. Therefore, by the induction hypothesis, S' has a solution σ'. Now let σ be defined as follows: if $x =_s t_{k+1}$ then $x\sigma$ is $t_{k+1}\sigma'$; if $t_{k-1} =_s x$ then $x\sigma$ is $t_{k-1}\sigma'$ (note that if $t_{k-1} =_s x =_s t_{k+1}$ then x does not appear in t_{k-1} nor in t_{k+1} and $t_{k-1}\sigma'$ is $t_{k+1}\sigma'$);

[1] It can easily be extended to constraints over the *recursive path ordering with status* (RPOS) by treating everywhere $=$ as equality up to permutations of arguments of operators with multiset status.

otherwise, $x\sigma$ is $f(t_{k+1})\sigma'$ if $k \neq n$ and $x\sigma$ is 0 if $k = n$. Furthermore, $y\sigma$ is $y\sigma'$ for any other variable y in S.

Now we prove that σ is indeed a solution of S. By the previous lemma, we only have to check the relations $s \#_s t$ where either s or t is a variable. Furthermore, σ is a solution for all $s \#_s t$ where s and t do not contain x since σ' is solution of S'. It remains to check $s\sigma \# t\sigma$ for the relations $s \#_s t$ where x appears in s or t and where at least one of s and t is a variable.

By construction, σ is a solution of $x \# t_{k+1}$, and therefore, by transitivity of $\succ_{lpo}^{\mathcal{F}_0}$ and $=$, also of $x \# t_{k+1} \# \ldots \# t_n$. The only remaining cases are:

1. $t[x]_p \succ_s x$ with $p \neq \lambda$: then $t[x]\sigma \succ_{lpo}^{\mathcal{F}_0} x\sigma$ by the subterm property of LPO.

2. $t =_s x$ where x does not appear in t: then $t\sigma$ is $t_{k-1}\sigma$ which is $x\sigma$.

3. $t \succ_s x$ where x does not appear in t. Here, if x is t_n then $x\sigma$ is the new smallest constant symbol 0, and $t \succ_{lpo}^{\mathcal{F}_0} 0$. Otherwise $t \succ_s t_{k+1}$ and $t\sigma \succ_{lpo}^{\mathcal{F}_0} t_{k+1}\sigma$ because neither t nor t_{k+1} contain any x, and σ' is a solution of S'. We analize two cases: If $t \succ_s x =_s t_{k+1}$ then since $t\sigma \succ_{lpo}^{\mathcal{F}_0} t_{k+1}\sigma$ also $t\sigma \succ_{lpo}^{\mathcal{F}_0} x\sigma$. If $t \succ_s x \succ_s t_{k+1}$ then $x\sigma$ is $f(t_{k+1}\sigma')$. The term t is not a variable, so $top(t) \succ_{\mathcal{F}_0} f$, and by definition of LPO $t\sigma \succ_{lpo}^{\mathcal{F}_0} f(t_{k+1}\sigma')$, since $t\sigma' \succ_{lpo}^{\mathcal{F}_0} t_{k+1}\sigma'$. □

3 Satisfiability in the given signature

Let f and 0 be the smallest non-constant and constant function symbols in $(\mathcal{F}, \succ_{\mathcal{F}})$, respectively (if there are no such symbols then the problem is trivial), and let K be the set of constants smaller (wrt. $\succ_{\mathcal{F}}$) than f, and let k be $max\{\ |K|,\ 1\ \}$. The terms in $\mathcal{T}(\mathcal{F})$ can be seen as ordinal numbers. For example. if $f \succ_{\mathcal{F}} a \succ_{\mathcal{F}} 0$, and f is binary, then the terms are, in increasing order wrt. $\succ_{lpo}^{\mathcal{F}}$:

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

We will denote the first fragment (corresponding to the natural numbers) by $\mathcal{N}$, and the first limit ordinal ($f(a,0)$ in the example) by ω. The value of a term t in $\mathcal{N}$ as a natural number is denoted by $|t|$. The ground term ω is easy to identify in every $(\mathcal{F}, \succ_{\mathcal{F}})$: if f is unary, then ω is $g(0,\ldots,0)$ where g is the smallest symbol greater than f; if f is not unary then ω is $f(0,\ldots,0,t,0)$ where t is the second smallest ground term. Note that always $f(0,\ldots,0,t)$ is the k-th successor term of a ground term t if t is in $\mathcal{N}$, and otherwise $f(0,\ldots,0,t)$ is the first successor of t. (cf. [Com90] or [JO91] for more details).

Here we consider simple systems $t_1 \# \ldots \# t_{p-1} \succ \omega \succ t_{p+1} \# \ldots \# t_{n-1} \succ t_n$ where t_n is the smallest constant 0. Now $t_{p+1} \# \ldots \# t_n$ is called the *natural part.* and $t_1 \# \ldots \# t_{p-1} \succ \omega$ is called the *non-natural part* of the system. Furthermore, we require that between each t_i in the natural part of the form $f(0,\ldots,0,t)$ and t there are at least k operators $\succ$, and if some t_i is in $\mathcal{N}$, then there must be at least $|t_i|$ operators

$\succ$ between t_i and t_n. This can be assumed without loss of generality since every simple system can be expressed as an equivalent finite disjunction of simple systems of this form, by inserting ω, 0 and new variables. Note that no variable is 0 in any solution of such a system S.

Now we consider a simple system S to be trivially bottom if it is trivially bottom in the sense of the previous section or:

1. $\omega \succ_s f(0,\ldots,0,t)$ and there are strictly more than k operators $\succ$ between $f(0,\ldots,0,t)$ and t, or

2. $\omega \succ_s t$ with t in $\mathcal{N}$, and strictly more than $|t|$ operators $\succ$ between t and t_n, or

3. $f(0,\ldots,0,t) \succ_s t \succ_s \omega$ and there is strictly more than one operator $\succ$ between $f(0,\ldots,0,t)$ and t.

Lemma 3.1 Every simple system S of the form $\omega \succ t_1 \# \ \ldots \ \# \, t_n$ which is not trivially bottom has a solution in $\mathcal{N}$.

Proof Note that the only symbols in S are f and the constants of K (otherwise S would be trivially bottom). We proceed from right to left (i.e. by induction on $n-i$ for the subindices i) in S to build a solution σ in which each $|t_i\sigma|$ is the number of operators $\succ$ right of t_i in S.

For t_n, which is 0, the result holds. For the induction step, if t_i is a variable, then let $t_i\sigma$ be the successor of $t_{i+1}\sigma$ if $t_i \succ_s t_{i+1}$, and $t_{i+1}\sigma$ if $t_i =_s t_{i+1}$. If t_i is ground (or of the form $f(0,\ldots,0,t)$), then 0 (resp. t) are at the adequate distances to the right of t_i and the result holds by the induction hypothesis. □

Theorem 3.2 A simple system S is satisfiable under the given signature iff it is not trivially bottom.

Proof We proceed like in theorem 4.2, but here we apply induction on the number of variables in the non-natural part of S.

Suppose the non-natural part is ground. By the previous lemma there is a solution σ in $\mathcal{N}$ for the natural part. Then σ satisfies S (not only its natural part): the only relations $s \#_s t$ involving the non-natural part where at least one of s and t is a variable are of the form $t \succ y$ where t is ground. But then $t \succ_{lpo}^{\mathcal{F}} y\sigma$, because $y\sigma$ is in $\mathcal{N}$, and t is not in $\mathcal{N}$ (if t were in $\mathcal{N}$ then $t \succ_s \omega$ would make S trivially bottom).

Otherwise, there is some leftmost variable x in the non-natural part. Let S' be as in theorem 4.2, with a solution σ', and let σ be: if $x =_s t_{k+1}$ then $x\sigma$ is $t_{k+1}\sigma'$; if $t_{k-1} =_s x$ then $x\sigma$ is $t_{k-1}\sigma'$; otherwise, $x\sigma$ is $f(0,\ldots,0,t_{k+1})\sigma'$. Furthermore, $y\sigma$ is $y\sigma'$ for any other variable y in S.

To prove that σ is indeed a solution of S, the only case that is different wrt. theorem 4.2 is $t \succ_s x$ where x does not appear in t. Then $t \succ_s t_{k+1}$ and $t\sigma \succ_{lpo}^{\mathcal{F}} t_{k+1}\sigma$, and if $t \succ_s x =_s t_{k+1}$ then also $t\sigma \succ_{lpo}^{\mathcal{F}} x\sigma$.

If $t \succ_s x \succ_s t_{k+1}$ then $x\sigma$ is $f(0,\ldots,0,t_{k+1}\sigma')$, i.e. the successor of $t_{k+1}\sigma'$. But then also $t\sigma \succ x\sigma$, because $t\sigma$ is not the successor of $t_{k+1}\sigma'$. (Otherwise, t would be of the form $f(0,\ldots,0,t')$, since $z\sigma'$ is 0 for no variable z, with $t'\sigma'$ equal to $t_{k+1}\sigma'$. But there would only be one operator $\succ$ between t and t', which implies $t' \succ_s t_{k+1}$, contradicting $t'\sigma'$ equal to $t_{k+1}\sigma'$, since these terms contain no x and σ' is a solution of S'.) $\square$

4 Practical considerations and complexity

It is not surprising that the implementations for LPO constraint solving we have developed are inefficient for very large constraints, since the problem is NP-complete, as we show now.

A simple system S *implies* a constraint C if C evaluates to *true* after replacing by *true* all its atoms $s \# t$ such that $s \#_s t$ (this can be checked in polynomial time). The following lemma holds for both kinds of satisfiability:

Lemma 4.1 A constraint C is satisfiable iff there exists a satisfiable simple system S implying C of the form $t_1 \# \ldots \# t_n$, where $\{t_1,\ldots,t_n\}$ is the set of all subterms of C (plus, in the case of solutions over the given signature, the additional terms ω, 0 and the additional variables for getting a simple system in the sense of section 3).

Theorem 4.2 Deciding the satisfiability of a constraint C (under extended signatures or under the given signature) is NP-complete.

Proof The problem is very easily shown NP-hard by reducing 3-SAT: encode the set N of clauses $(x \vee \neg y \vee z) \wedge \ldots$ by the constraint C of the form $(x \succ a \vee y \not\succ a \vee z \succ a) \wedge \ldots$ Then C is satisfiable (in any signature in which there are terms greater than a, e.g. $b \succ_{\mathcal{F}} a$) iff N is satisfiable. In fact, even for constraints consisting of one single inequation the problem is NP-hard[2].

Furthermore, the following algorithm decides satisfiability of C in non-deterministic polynomial time (wrt. the number n of subterms of terms in C):

1. Guess a simple system S of the form $t_1 \# \ldots \# t_n$, where $\{t_1,\ldots t_n\}$ is the set of all subterms of C (plus, in the case of solutions over the given signature, the terms ω, 0 and the additional variables to get a simple system in the sense of section 3).
2. If S implies C and S is satisfiable wrt. the given semantics then return "yes". $\square$

A naive deterministic version of the algorithm given above requires to check a large exponential number of simple systems. In practice, one can do much better by reducing constraints into an equivalent disjunction of simple systems as follows.

First, eliminate negations with $(t \not\succ t') \equiv (t' \succ t \vee t = t')$ and $(t \neq t') \equiv (t' \succ t \vee t \succ t')$, and put the result in disjunctive normal form. Then one can simplify each conjunction (and eliminate many unsatisfiable ones) by applying the definitions of LPO and equality.

[2] Also by reduction of 3-SAT. Hubert Comon, private communication.

Applying the definition of LPO may introduce new disjunctions which is not always efficient, but e.g. replacing $f(s,t) \succ f(s,t')$ by $t \succ t'$ is of course convenient.

Finally the equivalent set of simple systems is obtained by adding in all possible ways the relations with equality and inequality between all subterms.

At any stage, equalities can be eliminated by computing their simultaneous unifier σ and applying σ to the rest of the expression, but this may generate new subterms which have to be added (generating new equalities) when getting simple systems, etc. Although this iterative process terminates, as the number of variables decreases, it seems inconvenient to do this exhaustively in practice. A main advantage of the techniques given here is precisely that it is not necessary to eliminate equalities.

References

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

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

[NR92] Robert Nieuwenhuis and Albert Rubio. Theorem proving with ordering constrained clauses. In Deepak Kapur, editor, *11th CADE*, LNAI 607, pages 477–491, Saratoga Springs, New York, 1992. Extended version to appear in Journal of Symbolic Computation.