Orderings, AC-Theories and Symbolic Constraint Solving (Extended Abstract)*

H. Comon R. Nieuwenhuis A. Rubio

LRI & CNRS
Bat. 490, Univ. Paris-Sud
91405 ORSAY cedex, France
comon@lri.fr

Technical Univ. of Catalonia
Pau Gargallo 5
08028 Barcelona, Spain
{roberto,rubio}@lsi.upc.es

Abstract

We design combination techniques for symbolic constraint solving in the presence of associative and commutative (AC) function symbols. This yields an algorithm for solving AC-RPO constraints (where AC-RPO is the AC-compatible total reduction ordering of [16]), which was a missing ingredient for automated deduction strategies with AC-constraint inheritance [15, 19]. As in the AC-unification case (actually the AC-unification algorithm of [9] is an instance of ours), for this purpose we first study the pure case, *i.e. we show how to solve AC-ordering constraints built over a single AC function symbol and variables. Since AC-RPO is an interpretation-based ordering, our algorithm also requires the combination of algorithms for solving* interpreted *constraints and non-interpreted constraints.*

1 Introduction

Symbolic constraints are formulas which are interpreted in a fixed term structure. They are used in many areas of mathematics and computer science, in particular in automated deduction and logic programming.

A typical example of symbolic constraint solving is *E-unification.* Given a set E of equational axioms, it consists in solving equations in the quotient of the term algebra by the E-generated congruence. A general way of building E-unification algorithms is to decompose E into smaller components E_i and to combine E_i-unification algorithms. Such *combination techniques* have been applied, for example, to obtain unification algorithms in Boolean rings and Abelian groups [3], to arbitrary disjoint unions of equational theories [17, 2, 1] and to the important special case of Associative-Commutative (AC) unification [18, 10, 9]. In the latter, each AC symbol is associated with an E_i consisting of the AC axioms for that symbol. Then guessing beforehand in which theory each variable will be instantiated and the equalities between all (sub)terms, AC-unification is reduced to pure AC-unification and free term unification.

Ordering constraints are another example of symbolic constraints. They are used to express ordered strategies in automated deduction at the formula level [11]. This allows to reduce the search space by inheriting the ordering restrictions while keeping completeness [14]. In this case, atomic formulae not only consist of equations, but also of *inequations* $s \succ t$ where $=$ is freely interpreted and $\succ$ is interpreted as a well-founded monotonic term ordering which is total on ground terms. Typical examples of such orderings are the total *recursive path orderings* (RPO) introduced by N. Dershowitz [6]. The existential fragment of the theory of each RPO was shown decidable [4, 8] and actually NP-complete [13].

Ordered strategies and ordering constraints inheritance can be used in presence of AC symbols without loosing completeness [15, 19]. This requires first an ordering which, in addition to former requirements, is also AC-compatible. Then we need a constraint solving algorithm for such an ordering. AC-compatible (monotonic, well-founded, total on ground equivalence classes) orderings were unknown until recently, when P. Narendran and M. Rusinowitch exhibited one in [12], based on interpretations in polynomial rings. A conceptually simpler ordering in the spirit of recursive path orderings, and therefore called AC-RPO was later proposed by some of us [16]. Roughly, to order two terms s and t, AC-RPO compares with RPO

*An extended version of the paper can be obtained by anonymous ftp on `bloom.upc.es` (147.83.20.18) `pub/ac-constraints.ps`.

their interpretations (their normal forms wrt. a rewrite rule). In case of equal interpretations, it compares arguments of s and t using again AC-RPO. It remained however to study the AC-ordering constraints.

Our aim here is to design combination techniques yielding an AC-RPO constraint solving algorithm. As in the AC-unification case (actually the AC-unification algorithm of [9] will be an instance of ours), we first study *pure* AC-ordering constraints, i.e. built over a single AC function symbol and variables. More precisely, given any total ordering on a set S, we show how to solve ordering and equality constraints on the flattened multisets over S, thus extending pure AC-unification.

Then we use the classical variable abstraction rule, and we guess in which theory each variable will be instantiated. The relations between interpretations of all terms in a constraint C are also guessed and kept in an *interpreted* constraint C^i for C. We also guess in advance all identical subterms, which allows us to ensure that, afterwards, non indentified terms are distinct. As in the unification case, this eliminates possible interactions due to equalities between terms of distinct pure subsystems. Unfortunately, in our case there are also interactions between pure subsystems in the form of inequations between terms with different head symbols. However, these additional interactions are taken care of by the interpreted constraint C^i, since such terms must have different interpretations.

This reduces the general constraint solving problem to pure subsystems and interpreted systems, which can be solved, thanks again to pure constraint solving. However, solving both systems is not sufficient since the relationship between them is expressed using annotations of the variables which have to be considered while reconstructing the solution of the whole problem. Nevertheless, we prove that, if no inconsistency has been detected so far, there is at least one solution.

This shows the decidability of AC-RPO constraints. With respect to efficiency issues, the decision algorithm that can be derived from this paper is quite complex: at several points we need non-deterministic choices. Solving pure systems is also quite complex in itself. However, on one hand it is not possible to expect an efficient (complete) procedure as AC-unification, which is a particular case of AC-RPO contraints, is doubly exponential [9]. On the other hand, the complete satisfiability test needs to be performed only once (at the end of the deduction), partial satisfiability tests, that can also be derived from our procedure, being sufficient in intermediate deduction steps.

The paper is organized as follows. Section 2 contains the basic definitions, in particular the definition of AC-RPO. Section 3 explains how to reduce the AC-RPO constraints to linear constraints together with their interpreted part, including all aforementioned guesses. In section 4, we show how to solve pure AC ordering constraints. In section 5, we show how interactions between pure systems are reduced to the interpreted part. In section 6, we show how to solve the interpreted part. Finally, we show in section 7 how to build a solution of the (solved) constraint.

2 Basic Definitions

2.1 Terms

$\mathcal{F}$ is a finite set of *function symbols* together with their arity. AC is a subset of binary symbols of $\mathcal{F}$ which are *associative and commutative.* $\mathcal{T}(\mathcal{F})$ is the set of terms built on $\mathcal{F}$ and $=_{\mathrm{AC}}$ is the congruence generated on $\mathcal{T}(\mathcal{F})$ by the associativity and commutativity axioms for the symbols in AC. Representatives of terms in $\mathcal{T}(\mathcal{F})/_{=_{\mathrm{AC}}}$ will be written in the so-called *flattened form*: if $+ \in AC$ we write e.g. $+(a, b, c)$ instead of $+(a, +(b, c))$. $+$ may also be used in infix notation $a + b + c$.

2.2 The (AC)-ordering

Let us first recall the definition of RPO, the *recursive path ordering (with status)* (see e.g. [7]). This definition allows variadic symbols (and hence flattened terms).

We assume given a *precedence* $\succ_{\mathcal{F}}$ on $\mathcal{F}$, and, for each $f \in \mathcal{F}$, a *status* which is either *multiset* or *lexicographic.* All variadic symbols are assumed to have the multiset status. Below, the equality $=$ has to be understood modulo permutations of the direct subterms of any symbol whose status is multiset. More precisely, for every permutation π, if status(f) = multiset, then, for every terms $t_1, \ldots, t_n$, $f(t_1, \ldots, t_n) = f(t_{\pi(1)}, \ldots t_{\pi(n)})$. $\succeq$ stands for $\succ \vee =$. Then RPO is defined as follows:

$s = f(s_1, \ldots, s_n) \succ_{rpo} g(t_1, \ldots, t_m) = t$ iff

(i) $\exists i \in \{1, \ldots, n\}$, $s_i \succeq_{rpo} t$

or (ii) $f >_{\mathcal{F}} g$ and $s \succ_{rpo} t_i$ for all $i = 1, \ldots, m$

or (iii) $f = g$ and status(f) = multiset, and $\{s_1, \ldots, s_n\} \ggcurly_{rpo} \{t_1, \ldots, t_m\}$ where $\ggcurly_{rpo}$ is the multiset extension of $\succ_{rpo}$

or (iv) $f = g$ and status(f) = lexicographic, and $(s_1, \ldots, s_n) \succ_{rpo}^{lex} (t_1, \ldots, t_n)$ where $\succ_{rpo}^{lex}$ is the lexicographic extension of $\succ_{rpo}$.

RPO has all desired properties on $\mathcal{T}(\mathcal{F})$, but it is not monotonic when there is at least one AC-symbol: if $f \succ_{\mathcal{F}} g$ and f is an AC symbol, then $f(a,a) \succ_{rpo} g(a)$ but under the context $f(a,[])$ we have $f(a,a,a) \not\succ_{rpo} f(a,g(a))$. The AC-RPO ordering $\succ_{acrpo}$ solves this problem. It is defined as follows.

Let R_I be the set of *interpretation* rules $f(\ldots,g(t_1,\ldots,t_n),\ldots) \rightarrow f(\ldots,t,\ldots)$ for all symbols g different from the smallest constant symbol 0, all symbols f with $f \succ_{\mathcal{F}} g$ and such that t is the maximal term (with the ordering $\succ_{acrpo}$ applied recursively) of $\{t_1,\ldots,t_n,0\}$. Now let $I(u)$ denote the *interpretation* of a ground term u: the normal form (which exists and is unique) of u wrt. R_I. Let s and t be ground terms of the form $f(s_1,\ldots,s_m)$ and $g(t_1,\ldots,t_n)$ resp.

Definition 2.1 *$s \succ_{acrpo} t$ iff $I(s) \succ_{rpo} I(t)$ or else $I(s) = I(t)$ (note that then $f = g$) and either $f \in AC$ and $\{s_1,\ldots,s_m\} \succ\!\!\succ_{acrpo} \{t_1,\ldots,t_n\}$ or $f \notin AC$ and $(s_1,\ldots,s_m) \succ_{acrpo}^{lex} (t_1,\ldots,t_n)$, where $\succ\!\!\succ_{acrpo}$ and $\succ_{acrpo}^{lex}$ are the multiset and lexicographic extensions of $\succ_{acrpo}$ respectively.*

Example 2.2 Suppose we have a precedence $* \succ_{\mathcal{F}} + \succ_{\mathcal{F}} s \succ_{\mathcal{F}} 0$ with AC operators $*$ and $+$, and where s and 0 are unary and constant respectively. Then figure 1 gives some terms in increasing order with their corresponding ordinal notations.

Theorem 2.3 ([16]) *$\succ_{acrpo}$ is a well-founded ordering on $\mathcal{T}(\mathcal{F})$ which is monotonic, compatible with $=_{\mathrm{AC}}$, has the subterm property and is total on $\mathcal{T}(\mathcal{F})/_{=_{\mathrm{AC}}}$.*

2.3 Ordering constraints

An *ordering constraint* is a first-order formula $\exists X.\phi$ where ϕ is quantifier-free and built over terms in $\mathcal{T}(\mathcal{F},\mathcal{X})$ and binary predicate symbols '$\succ$' and '$=$'. X (as well as Y, Z in the following) stands for a set of variables. A *solution* of a constraint C is a substitution σ with range $\mathcal{T}(\mathcal{F})$ and domain the set of free variables of C such that $C\sigma$ evaluates to true when interpreting $=$ as $=_{\mathrm{AC}}$ and $\succ$ as $\succ_{acrpo}$. Then σ *satisfies* C (written $\sigma \models C$).

Example 2.4 Suppose we have a precedence $f \succ_{\mathcal{F}} a \succ_{\mathcal{F}} b$ with $f \in AC$. Then $f(x,y,y,y) \succ f(z,z) \succ z \succ x \succ y$ is a satisfiable constraint (written in a compressed form) with a solution $\sigma = \{x \mapsto a, y \mapsto b, z \mapsto f(b,b)\}$.

Note the additional complications appearing here due to the presence of AC symbols: unlike in orderings like LPO or RPO, we can have $f(x,y,y,y) \succ f(z,z)$ even if the both x and y are smaller than z.

3 Simple systems

In this section we reduce the satisfiability problem for a constraint C to the satisfiability problem for finitely many *simple systems*. Such systems come into two parts: the system S itself and an *interpreted part* S^i. The simple systems are obtained by first guessing the relations between all (sub)terms occurring in C. Then guessing relations between their interpretations, we get S^i ("interpretation" may be understood so far as the result of applying the rules R_I).

Let C be a constraint built over $\mathcal{T}(\mathcal{F},\mathcal{X})$. A *linear constraint* for C is a constraint S of the form $t_1 \# \ldots \# t_n$ where $T_C = \{t_1,\ldots,t_n\}$ is the set of all (sub)terms appearing in C and each relation $\#$ is $\succ$ or $=$.

Lemma 3.1 ([5, 13]) *Each constraint C can be expressed as an equivalent finite disjunction of linear constraints for C.*

This can be done by considering all possible linear orderings on T_C with $\succ$ and $=$. Note that, by totality of $\succ_{acrpo}$ up to $=_{\mathrm{AC}}$, for all solutions σ of C and all pairs of terms s and t in T_C either $s\sigma \succ_{acrpo} t\sigma$ or $t\sigma \succ_{acrpo} s\sigma$ or $s\sigma =_{\mathrm{AC}} t\sigma$. Each relation $s \# t$ of C is either included in such a linear constraint S or in contradiction with S, so only those S that are compatible in this sense with C are considered. These S in fact imply C, and the remaining S are incompatible with C.

So from now on we only need to consider the satisfiability problem of such linear constraints S. In fact we can write S like

$$\exists X.\, x_1 = t_{1,1} = \ldots = t_{1,k_1} \succ x_2 = t_{2,1} = \ldots \\ \ldots = t_{2,k_2} \succ \ldots \succ x_n = t_{n,1} = \ldots = t_{n,k_n}$$

and denote by $=_S$ the equivalence relation on $\mathcal{T}(\mathcal{F},\mathcal{X})$ generated by the equalities in S and by $\succ_S$ the smallest strict ordering relation on $\mathcal{T}(\mathcal{F},\mathcal{X})$ compatible with $=_S$ and containing the inequalities of S. For instance, if S is the formula $f(x,y',y,y) \succ f(z,z) \succ z \succ x \succ y' = y$, we have $y' =_S y$ and $f(x,x,y,y,y) \succ_S f(x,z,z)$.

Wlog. we can require $\{x_1,\ldots,x_n\} = X = \mathcal{V}ars(S)$, i.e. there is exactly one variable in each $=_S$-equivalence class. Indeed it is sufficient to insert a new (existentially quantified) variable if necessary, or to merge two equal variables into one if necessary (merging of equal variables, which will be done more often in this paper, can be recorded separately if one wants to reconstruct a solution for the original constraint rather than to decide its satisfiability).

ordinal	0	n	ω	$\omega+1$	$\omega\times n$	ω^2	ω^2+1	$\omega^2+\omega$
$I(t)$	0	$s^n(0)$	$0+0$	$0+0$	$s^n(0+0)$	$0+0+0$	$0+0+0$	$0+0+0$
t	0	$s^n(0)$	$0+0$	$s(0)+0$	$s^n(0+0)$	$0+0+0$	$s(0)+0+0$	$s(0+0)+0$

ordinal	$\omega^2+\omega+1$	$\omega^2+\omega\times 2$	$\omega^2\times 2$	ω^3	ω^ω
$I(t)$	$0+0+0$	$0+0+0$	$0+0+0$	$s(0+0+0)$	$0*0$
t	$s(0+0)+s(0)$	$s(s(0)+0)+0$	$s(s(0+0))+0$	$s(0+0+0)$	$0*0$

Figure 1: The AC-ordering, an example of translation in ordinal notation

For each variable we guess its top symbol (note that guessing a constant top symbol amounts to completely determine its solution). This guess will also be assumed each time a new variable is introduced. Furthermore, wlog. we may assume two additional properties of S: all top-level arguments of non-variable terms in S are variables, and for terms $f(\dots x_i \dots)$ with $f \in AC$ there is no term $f(y_1,\dots,y_m)$ in the equivalence class of x_i. This is accomplished by keeping S normalized wrt. the following two solution-preserving rewrite rules where x denotes a variable:

(standardize)
$f(\dots t \dots) \to f(\dots x \dots)$ if $x =_S t$ and $x \neq t$

(replace)
$f(\dots x \dots) \to f(\dots y_1 \dots y_m \dots)$ if $f \in AC$ and $x =_S f(y_1,\dots,y_m)$

Definition 3.2 *A* simple system *is a linear constraint S of the form*

$$x_1 = t_{1,1} = \dots = t_{1,k_1} \succ x_2 = t_{2,1} = \dots \\ \dots = t_{2,k_2} \succ \dots \succ x_n = t_{n,1} = \dots = t_{n,k_n}$$

where all top-level arguments of non-variable terms in S are in $\{x_1,\dots x_n\} = X = \mathcal{V}ars(S)$, and for each term $f(\dots x_i \dots)$ in S with $f \in AC$ there is no term $f(y_1,\dots,y_m)$ in the equivalence class of x_i.

Lemma 3.3 *For every constraint C there exists a finite set $\{S_1,\dots,S_m\}$ of simple systems (that can be obtained as above) such that C is satisfiable iff one of $S_1,\dots,S_m$ is satisfiable.*

Now for each non-interpreted simple system we are going to construct finitely many interpreted simple systems, corresponding to all possible ways the interpretations of the terms occurring in S can be related. To this end, we need to introduce, for each $f \in \mathcal{F}$ and each $x \in \mathcal{V}ars(S)$, an annotated version x^f of the variable x. These additional notations allow us to lift the definition of the interpretation I to non-ground terms and introduce interpretations I_f (for "interpretation below f"):

$$\begin{array}{rcl} I(f(t_1,\dots,t_n)) & \to & f(I_f(t_1),\dots,I_f(t_n)) \\ I(x) & \to & x^f \\ & & \text{If } top(x) = f \\ I_f(g(t_1,\dots,t_n)) & \to & I_f(t_i) \\ & & \text{If } f \succ_{\mathcal{F}} g \text{ and } t_i = \max_{\succ_S}\{t_1,\dots,t_n,0\} \\ I_f(g(t_1,\dots,t_n)) & \to & g(I_g(t_1),\dots,I_g(t_n)) \\ & & \text{If } g \succeq_{\mathcal{F}} f \\ I_f(x) & \to & x^g \\ & & \text{If } g = \max\{f, top(x)\} \\ I_f(x^g) & \to & x^{\max(f,g)} \end{array}$$

Definition 3.4 *Let C be a conjunctive constraint (a conjunction of equalities and inequalities $s \# t$) whose set of variables is X. Assume that* top(x) *has been guessed for all $x \in X$ as before.*

An interpretation *of C is a conjunctive constraint C^i obtained by replacing all terms t occurring in C with $I(t)$ and each $\succ$ symbol with either $=$ or $\succ$ (if $s \succ_C t$ and $s =_C s'$ s.t. $I(s') = s'$ is ground, then the only possible guess is $I(s) \succ_{C^i} I(t)$). Moreover, assume all top symbols of variables in C^i have been guessed.*

In the following, $I(S)$ denotes the set of all interpretations S^i of a simple system S.

Example 3.5 Consider $C \stackrel{\text{def}}{=} f(x,y,y,y) \succ f(z,z) \succ x \succ z \succ y$ and assume that we guessed $\top(x) = a$, $\top(y) = b$ and $\top(z) = f$. Then the two following formulas belong to $I(C)$:

$$C_1^i \stackrel{\text{def}}{=} f(x^f,y^f,y^f,y^f) \succ f(z^f,z^f) \succ z^f \succ x^a \succ y^b$$
$$C_2^i \stackrel{\text{def}}{=} f(x^f,y^f,y^f,y^f) = f(z^f,z^f) \succ z^f \succ x^a \succ y^b$$

Definition 3.6 *Let C be a conjunctive constraint and C^i an interpretation of C. A* solution *of a pair (C,C^i) is a pair of substitutions (σ,σ^i) such that*

- *$\sigma \models C$ and $\sigma^i \models C^i$*

- *for every variable x in $\mathcal{V}ars(C)$, $top(x) = top(x\sigma)$*
- *for every variable x in $\mathcal{V}ars(C^i)$ $top(x) = top(x\sigma^i)$*
- *$I(t\sigma) = I(t)\sigma^i$ for every term t*
- *for all variables x^f and x^g in C^i if $g \succeq_{\mathcal{F}} f$ then $I_g(x^f\sigma^i) = x^g\sigma^i$*

Example 3.7 Following example 3.5 (with the precedence of example 2.4), if σ is the assignment $\{x \mapsto a; y \mapsto b; z \mapsto f(b,b)\}$ and $\sigma^i = \{x^f \mapsto b; y^f \mapsto b; z^f \mapsto f(b,b); x^a \mapsto a; y^b \mapsto b\}$, then the pair (σ, σ^i) is a solution of (C, C_2^i).

Lemma 3.8 *For every simple system S, σ is a solution of S iff there is some pair (S, S^i) with $S^i \in I(S)$ having a solution (σ, σ^i).*

This lemma is a consequence of the construction of $I(S)$: choose $x^f\sigma^i$ as $I_f(x\sigma)$.

Definition 3.9 *Let S be a simple system together with one of its interpretations S^i, and let f be a function symbol in $\mathcal{F}$. Then the* pure *subsystem P_f for f of S wrt. S^i is the subsystem of S consisting of:*

- *all equations $s = t$ such that $s =_S t$ and* $\mathrm{top}(s) = \mathrm{top}(t) = f$ *and*
- *all inequations $s \succ t$ such that $s \succ_S t$, $I(s) =_{S^i} I(t)$ and* $\mathrm{top}(s) = \mathrm{top}(t) = f$.

The pure *part $P(S)$ of S is the conjunction of P_f for all $f \in \mathcal{F}$.*

Example 3.10 Following example 3.7, the pure subsystem P_f for f w.r.t. (C, C_2^i) consists of the single inequality $f(x,y,y,y) \succ f(z,z)$. For (C, C_1^i), the pure subsystem P_f is empty.

In classical methods for the combination of unification algorithms, after variable abstraction (which in our case is done while building simple systems), the interaction between the different pure subsystems is limited to equalities between variables. Unfortunately, in our case there are also interactions in the form of relations (inequalities) in S between terms with different head symbols. However, these additional interactions are taken care of by the interpreted system S^i. More precisely, we have the following lemma:

Lemma 3.11 *Let S be a simple system together with one of its interpretations S^i. Then the pair (S, S^i) is logically equivalent to $(P(S), S^i)$.*

4 Solving pure AC-ordering constraints

Definition 4.1 *A* pure system *for a function symbol $f \in AC$ is a conjunction of equalities and inequalities $t \# u$ where each of t, u is either a variable whose top symbol is f or a term $f(y_1, \ldots, y_n)$ where $y_1, \ldots, y_n$ are variables*[1].

In this section, which can be read independently of the others, we show how to solve such pure systems. Its results will be used several times along this paper, both for non-interpreted and interpreted systems For example, the pure subsystems P_f of S are a particular case of the pure systems solved here. As we aim here at abstracting from the particular context in which the technique will be used, we consider an interpretation of pure systems in which the ordering between subterms is a parameter $\succ_p$. More precisely, let S be a pure system for f and $\succ_p$ any linear ordering on terms. Then σ is a solution of $s \succ t$ where $s\sigma = f(s_1, \ldots, s_n)$ and $t\sigma = f(t_1, \ldots, t_m)$ if and only if $\{s_1, \ldots, s_n\} \gg_p \{t_1, \ldots, t_m\}$ where $\gg_p$ is the multiset extension of $\succ_p$. Note also that solutions σ are assignments of the variables whose top symbol is f.

We are going to show how to solve pure systems, whatever $\succ_p$ is. By solving we mean here express it as an equivalent finite disjunction of solved forms. Solved forms are defined as in [9] for the equality part, but, in order to cope with inequalities (which is our main concern) we need an additional syntactic construction $atomic_f(z)$ which, roughly, selects the maximal subterm of z which is not headed with f.

More precisely, a (pure) *solved form* of P is a formula

$$\exists Z.\ x_1 = t_1 \wedge \ldots \wedge x_n = t_n \bigwedge_j atomic_f(z_j) \succ atomic_f(z'_j)$$

where $x_1, \ldots, x_n$ are the variables of P and occur only once, Z is the set of variables of $t_1, \ldots, t_n$ and $z_j, z'_j \in Z$.

Given a ground term t, $atomic_f(t)$ is the term t itself if the top symbol of t is not f. Otherwise $t = f(t_1, \ldots, t_n)$ where each t_i is not headed with f and $atomic_f(t) = \max_{\succ_p}\{t_1, \ldots, t_n\}$.

Let $+$ be an AC symbol. Then every pure system P for $+$ with variables $X = \{x_1 \ldots x_n\}$ can be written as a set of m inequalities

$$Ineq(P) \stackrel{\text{def}}{=} \bigwedge_{i=1}^{m} a_{i,1}x_1 + \ldots + a_{i,n}x_n \succ a'_{i,1}x_1 + \ldots + a'_{i,n}x_n$$

[1] In such systems, all variables that do not have f as top symbol are considered as distinct constants; this is similar to what is done in [9]

and a set of equalities

$$Eq_0(P) \stackrel{\text{def}}{=} \bigwedge_j b_{j,1}x_1 + \ldots + b_{j,n}x_n = b'_{j,1}x_1 + \ldots + b'_{j,n}x_n$$

where the (sub-indexed) natural coefficients a and b are such that in no side of a relation all coefficients are zero. Let $Eq_1(P)$ denote the following system of m equalities obtained from $Ineq(P)$ by replacing $\succ$ by $=$ and adding $2m$ new variables y_i and y'_i, one at each side of every inequality:

$$Eq_1(P) \stackrel{\text{def}}{=} \bigwedge_{i=1}^{m} a_{i,1}x_1 + \ldots + a_{i,n}x_n + y_i = a'_{i,1}x_1 + \ldots + a'_{i,n}x_n + y'_i$$

The system $Eq(P) = Eq_0(P) \cup Eq_1(P)$ can be solved (like in AC-unification) as linear diophantine equations. Let

$$(c_{i,1}, \ldots, c_{i,n}, d_{i,1}, \ldots, d_{i,m}, e_{i.1}, \ldots, e_{i,m})_{1 \leq i \leq N}$$

be a basis of the solutions of the linear diophantine system. Here the coeficients $c_{i,j}$ correspond to variables x_j, the coefficients $d_{i,j}$ to variables y_j and the coefficients $e_{i,j}$ to variables y'_j. Each solution of the linear diophantine system is a linear combination of elements in the basis. As in the unification case, we guess non-zero coefficients of the linear combination: for each $A \subseteq \{1, \ldots, N\}$, we get a solution $Sol(Eq(P))$ obtained of the form

$$\begin{array}{ll} \exists Z. & \bigwedge_{j=1}^{n} x_j = \sum_{k \in A} c_{k,j} z_k \\ \wedge & \bigwedge_{i=1}^{m} y_i = \sum_{k \in A} d_{k,i} z_k \\ \wedge & \bigwedge_{i=1}^{m} y'_i = \sum_{k \in A} e_{k,i} z_k \end{array}$$

where $Z = \{z_1, \ldots z_k\}$. We discard solutions in which the coefficients c, d and e are such that some variable x_j or y'_i is empty. However, for the variables y_i we do admit empty solutions. Furthermore, we also discard solutions in which the coefficients for some z_j are non-zero for both y_i and for y'_i, i.e. we require $d_{j,i} \times e_{j,i} = 0$.

Now for each i in $1 \ldots m$ guess a $p_i \in A$ such that $e_{p_i,i} \neq 0$. For each possible such guess we obtain one *solved* pure system for P:

$$\begin{array}{ll} \exists Z. & \bigwedge_{j=1}^{n} x_j = \sum_{k \in A} c_{k,j} z_k \\ \wedge & \bigwedge_{i=1}^{m} \bigwedge_{q \in A, d_{q,i} \neq 0} atomic_+(z_{p_i}) \succ atomic_+(z_q) \end{array}$$

Example 4.2 Consider the system

$$\begin{cases} x_1 + x_2 & \succ & x_3 \\ 2x_3 & \succ & x_1 + x_2 \end{cases}$$

We have to solve the following system of linear diophantine equations:

$$\begin{cases} x_1 + x_2 + y_1 & = & x_3 + y'_1 \\ 2x_3 + y_2 & = & x_1 + y'_2 \end{cases}$$

A basis of solutions is

$$\begin{array}{lll} (0,0,0,0,1,0,1), & (0,0,0,1,0,1,0), & (0,0,1,1,0,0,2), \\ (0,1,0,0,0,1,0), & (0,1,1,0,0,0,2), & (1,0,1,0,0,0,1), \\ (1,0,0,0,1,1,0), & (2,0,1,0,0,1,0) & \end{array}$$

Now, we have to guess a non-empty subset of this eight elements set (this corresponds to $2^8 - 1$ possibilities) and remove those which do not satisfy the criteria. For example, the set of the last 6 basis vectors yields a solution

$$\begin{array}{ll} x_1 = z_6 + z_7 + 2z_8 & x_2 = z_4 + z_5 \\ x_3 = z_3 + z_5 + z_6 + z_8 & \\ y_1 = z_3 & y_2 = z_1 + z_7 \\ y'_1 = z_4 + z_7 + z_8 & y'_2 = 2z_3 + 2z_5 + z_6 \end{array}$$

satisfying the criteria (whereas, e.g. the whole set of 8 vectors doesn't satisfy the criterion $d_{j,i} \times e_{j,i} = 0$)

Assume now we guess $p_1 = 4$ and $p_2 = 3$, we get the solved pure system:

$$\begin{array}{lll} \exists Z. & x_1 = z_6 + z_7 + 2z_8 & \wedge x_2 = z_4 + z_5 \\ \wedge & x_3 = z_3 + z_5 + z_6 + z_8 & \\ \wedge & atomic_+(z_4) \succ atomic_+(z_3) & \\ \wedge & atomic_+(z_3) \succ atomic_+(z_1) & \\ \wedge & atomic_+(z_3) \succ atomic_+(z_7) & \end{array}$$

The disjunction Solved(P) of all such solved pure systems is logically equivalent to P:

Lemma 4.3 *Solved(P) is logically equivalent to P.*

Proof: We prove that σ is a solution of the original system P iff it is a solution of some system S in Solved(P).

$\Longrightarrow$): Let σ be a solution of P, i.e. the domain of σ is $X = \{x_1 \ldots x_n\}$. Then for each x_j we have $x_j\sigma = \alpha_1^j t_1 + \ldots + \alpha_N^j t_N$ where $t_1, \ldots, t_N$ are not headed with $+$ and α_i^j is a non-negative integer. For each of the m inequalities $s_i \succ s'_i$ of P, we can view $s_i\sigma$ and $s'_i\sigma$ as multisets of terms in $t_1, \ldots, t_N$, and $s_i\sigma \gg_p s'_i\sigma$ (and of course $s_i\sigma = s'_i\sigma$ for the equalities $s_i = s'_i$).

Let θ be the substitution with domain $X \cup \{y_1, y'_1 \ldots y_m, y'_m\}$ defined by $x_j\theta = x_j\sigma$ and $y_i\theta = s'_i\sigma \setminus s_i\sigma$ and $y'_i\theta = s_i\sigma \setminus s'_i\sigma$. Note that $y'_i\theta \gg_p y_i\theta$

(and that $y_i\theta$ may be the empty multiset; in that case we consider that y_i is not in the domain of θ).

Now it is easy to verify that θ is a solution of $Eq(P)$. This means that θ is an instance of one of the solutions $Sol(Eq(P))$, so for some substitution θ' extending θ to include values for the new variables Z for each i we have $y_i\theta' = (d_{1,i}z_1 + \ldots + d_{k,i}z_k)\theta'$ and $y'_i\theta' = (e_{1,i}z_1 + \ldots + e_{k,i}z_k)\theta'$. Since $y_i\theta'$ and $y'_i\theta'$ are disjoint, the maximal term in $y'_i\theta'$ is greater wrt. $\succ_P$ than all terms in $y_i\theta'$, so for each i there is some non-null coefficient $e_{p_i,i}$ such that for all non-null $d_{q,i}$ we have $atomic_+(z_{p_i}\theta') \succ atomic_+(z_q\theta')$, which implies that σ is a solution of some of the systems in Solved(P) (and where the existentially quantified variables Z are given values according to θ').

$\Longleftarrow$): Let σ be a solution of some system S in Solved(P) and let θ be σ extended to the existentially quantified variables Z (according to their values for which S is valid) and to the y_i and y'_i according to the solution $Sol(Eq(P))$ from which S was built. Now the construction is the reversed version of the one above: for each i we have $y'_i\theta \gg_P y_i\theta$ which means that since θ is a solution of $Sol(Eq(P))$ we have $s_i\sigma \succ s'_i\sigma$ for all inequalities $s_i \succ s'_i$ in P, and $s_i\sigma = s'_i\sigma$ for all equalities $s_i = s'_i$ in P. □

5 Solved Forms of Uninterpreted Systems

Now, for each $f \in AC$, the pure subsystem P_f can be solved using the method of section 4, yielding a set of solved forms Solved(P_f), from which we remove all constraints that contain, e.g., equalities between terms that do not have the same top symbol. Furthermore, we replace each expression $atomic_f(x)$ with x if x is not headed with f, and with a new variable x_a otherwise (this is for convenience of notations).

If $f \notin AC$, either one of the properties of figure 1 is not satisfied and the constraint is unsatisfiable, or else, we let Solved(P_f) be a subset of P_f containing for each variable x in P_f one equation $x = t$ if $x =_{P_f} t$ and if there is no such t the inequalities $s \succ x$ and $x \succ u$ where s (resp. u) is minimal (resp. maximal) w.r.t. $\succ_{P_f}$.

We assume here that there are no equations between variables (they can be removed eagerly, as before). A *solved form* will consist in any conjunction $\bigwedge_{f\in\mathcal{F}} C_f$ where $C_f \in$ Solved(P_f). Note that solved forms do not contain any more the relations between terms having different head symbols. Now, for each such solved form $\mathcal{SF}$, we insert in its corresponding interpreted constraint $\mathcal{SF}^i$ (containing the relations between interpretations of all its terms) for each variable $x_a \in \mathcal{SF}$ with $top(x) = f$, the relations $I(x) \succ I(x_a)$ and $x^f = f(x_a^f, x'^f)$ where x' is a new variable. We also insert the equalities $x^g = x_a^g$ for all $g \succ_{\mathcal{F}} f$. This yields several possible interpreted systems $\mathcal{SF}^i$ for $\mathcal{SF}$ such that:

Lemma 5.1 *σ is a solution of the simple system S iff there is a pair (σ, σ^i) which is a solution of some $(\mathcal{SF}, \mathcal{SF}^i)$ for some solved form $\mathcal{SF}$ of S.*

6 Solved Forms of Interpreted Systems

In this section we solve interpreted systems S^i obtained as $\mathcal{SF}^i$ in the previous section. We assume here that S^i is an interpreted system that is initially in simple system form like for lemma 3.3 and that is kept closed under I_f for all f. Assume $f_n \succ_{\mathcal{F}} \ldots \succ_{\mathcal{F}} f_1$. For $i = 1$ to n we iterate the following:

1. Let P_{f_i} be the pure subsystem consisting of inequalities $s \succ t$ (resp. equalities $s = t$) such that $top(s) = top(t) = f_i \in AC$, $s \succ_{S^i} t$ (resp. $s =_{S^i} t$)

2. Solve the pure system P_{f_i} according to the method of section 4. Here, expressions $atomic_{f_i}(x)$ are replaced with $I_{f_{i+1}}(x)$ if $i < n$ and they are replaced with $\perp$ (unsatisfiable formula) if $i = n$.[2] Let Z_{f_i} be the set of new variables introduced at this stage. All top symbols of variables in Z_{f_i} must be guessed among $\{g \mid g \succeq_{\mathcal{F}} f_i\}$. All variables in $\{z^g \mid z \in Z_{f_i} \wedge g \succeq_{\mathcal{F}} top(z) \wedge g \neq f_i\}$ are inserted in S^i. Also all new equations are added to the system.

3. Remove the systems (i.e. derive a failure) which do not satisfy one of the properties of figure 1 (where S is replaced with S^i) or one of the properties of figure 2.

Note that new equalities and inequalities introduced at step i can only involve terms with a head symbol greater than (or equal to) f_i, because atomics of an interpreted term headed with f_i are always headed with a symbol strictly larger than f_i (or 0). For this reason, at step $i+1$, we are not undoing what was done at step i: roughly, after step i, the pure system P_{f_i} is solved and remains solved at further steps.

Every system $\mathcal{SF}^i$ obtained in the previous process is closed under I_f for all f and is a linear ordering on all terms occurring in it except the new variables of Z_{f_i}

[2] Indeed, relations of the form $atomic_{f_n}(x) \succ atomic_{f_n}(y)$ are never satisfiable since both sides have to be assigned to 0, the only possible subterm of an interpreted term headed with f_n.

1.	$t\|_p \not\succeq_S t$ if $p \neq \Lambda$
2.	$f(s_1,\ldots,s_n) =_S f(t_1,\ldots,t_n) \Rightarrow s_1 =_S t_1 \wedge \ldots \wedge s_n =_S t_n$ if $f \notin AC$
3.	$s = f(s_1,\ldots,s_n) \succ_S t = f(t_1,\ldots,t_n) \Rightarrow \langle s_1,\ldots,s_n\rangle \succ_S^{lex} \langle t_1,\ldots,t_n\rangle$ if $f \notin AC$

Figure 2: Properties that have to be fulfilled by all systems S

with top symbol f_i generated when solving P_{f_i}. Let Solved($\mathcal{SF}^i$) be any subset of $\mathcal{SF}^i$ containing for each variable x in $\mathcal{SF}^i$ one equation $x = t$ if $x =_{\mathcal{SF}^i} t$ and, if there is no such t, all inequalities $s \succ x$ and $x \succ u$ in $\mathcal{SF}^i$ except the ones such that $top(s) = top(x) \in AC$ and $top(u) = top(x) \in AC$. Then we have the following:

Lemma 6.1 *$\mathcal{SF}^i$ is logically equivalent to Solved($\mathcal{SF}^i$).*

7 Building a Solution of Solved Constraints

Let us develop an example for all stages of the process:

Example 7.1 Suppose $f \succ_\mathcal{F} g \succ_\mathcal{F} h \succ_\mathcal{F} 0$ with $f, g \in AC$ and h and 0 unary and constant respectively. Let S be a simple system $f(x,x,x,x) \succ f(y,y) \succ y \succ x$, where we have guessed $top(x) = g$ and $top(y) = f$ and $I(f(x,x,x,x)) = I(f(y,y)) \succ I(y) \succ I(x)$.

Then P_f consists of the relation $f(x,x,x,x) \succ f(y,y)$, and one system in Solved(P_f) (corresponding to the case where x and y do not share any maximal subterms not headed with f) is $x = z_1 \wedge y = z_2 \wedge atomic_f(z_1) \succ atomic_f(z_2)$. The new variables z_1 and z_2 are equal to x and y resp., so they disappear and the relation $atomic_f(x) \succ atomic_f(y)$ is written $x \succ y_a$, since $top(x) = g$ and $top(y) = f$. Suppose we guess $top(y_a) = g$.

One possible corresponding interpreted system is e.g.: $f(x^f,x^f,x^f,x^f) = f(y^f,y^f) \succ y^f \succ x^g = y_a^g \succ x^f = y_a^f$. Note that for variables like x only interpretations below top symbols bigger or equal than $top(x) = g$ need to be considered, since e.g. x^h would be necessarily equal to x^g. Furthermore, we have to add $y_f = f(y_a^f, y'^f)$ and guess top symbols (in both cases among f and 0) for the new variables: $top(x^f) = f$ and $top(y'^f) = 0$. Inserting $y'^f = 0$ and merging equal variables, the only possible system we can get is: $f(x^f,x^f,x^f,x^f) = f(y^f,y^f) \succ y^f = f(x^f,0) \succ x^g \succ x^f \succ 0$.

Now we solve the interpreted system. From $f(x^f,x^f,x^f,x^f) = f(y^f,y^f)$ we get $y_f = f(x^f,x^f)$. We obtain a solution by taking always the smallest possible interpreted terms and respecting the relations between variables x^f and x^g (if $g \succeq_\mathcal{F} f$ then $I_g(x^f\sigma^i) = x^g\sigma^i$):

$\sigma^i = \{\ x^f \mapsto f(0,0),\quad x^g \mapsto g(f(0,0),0),\quad y^f \mapsto f(0,0,0,0)\ \}$.

Finally, we can reconstruct the solution σ for the non-interpreted system, respecting the relations $\succ$ and $=$, and the properties $I(t\sigma) = I(t)\sigma^i$ (compatibility with the interpreted part) and $y_a\sigma = atomic_f(y\sigma)$ (compatibility between atomics). For the relations $s \succ t$ we only have to take care when $I(s)\sigma^i = I(t)\sigma^i$, since otherwise $s \succ t$ follows from compatibility with the interpreted part. Again we take the smallest possible solutions, starting with the smallest terms: $y_a\sigma$ is its own interpretation (which is the one of $x\sigma$, since $x^g = y_a^g$ in the interpreted part); since $x \succ y_a$, for $x\sigma$ we take the successor of $y_a\sigma$, and finally for y we also take the smallest possible term that preserves the compatibility with the interpreted part, which is $f(y_a\sigma, y'^f\sigma^i)$:

$\sigma = \{\ y_a \mapsto g(f(0,0),0),\quad x \mapsto g(f(0,0),h(0)),\quad y \mapsto f(g(f(0,0),0),0)\ \}$. □

For some kinds of precedences, building solutions is more difficult than for others. For instance, the precedence of the previous example is among the simpler cases, because successor terms are easily defined. More precisely, if h and 0 denote the smallest non-constant and constant symbols respectively, for precedences in which $h \notin AC$ and where there is no constant $a \neq 0$ smaller than h, the successor (wrt. $\succ_{acrpo}$) of a term $f(t_1,\ldots,t_n)$ is given in figure 4.

Here we will only consider precedences of this kind. Note that, since in AC-RPO arbitrary precedences are allowed and are user-defined, this is not a real restriction.

It is also convenient to consider solutions in which $\mathcal{F}$ is in fact a signature extended with an additional symbol at the lower end, because in practical applications such additional (e.g. Skolem) symbols are often needed. For more details about the need of such *extended signature* semantics, in which solutions may contain new symbols not occurring in the constraint,

1. $f \succ_{\mathcal{F}} g \Rightarrow I_g(t) \succeq_{S^i} I_f(t)$
2. $x^f \not\succeq_{S^i} g(\ldots x^f \ldots)$
3. $s =_{S^i} t \Rightarrow top(s) = top(t)$
4. $s \succeq_{S^i} t \Rightarrow I_g(s) \succeq_{S^i} I_g(t)$
5. $s =_{S^i} t \Rightarrow I_g(s) =_{S^i} I_g(t)$
6. $top(x^f) = 0 \vee top(x^f) \succeq_{\mathcal{F}} f$
7. $s \succ_{S^i} t \wedge f = top(t) \succ_{\mathcal{F}} top(s) \Rightarrow I_f(s) \succeq_{S^i} t$
8. $s \succ_{S^i} t \wedge f = top(s) \succ_{\mathcal{F}} top(t) \Rightarrow s \succ_{S^i} I_f(t)$
9. $u \succ_{S^i} s \succ_{S^i} t \wedge top(s) = f \notin AC \Rightarrow u \succ_{S^i} f(0, \ldots, 0, I_f(t))$

Figure 3: Properties that have to be fulfilled by the interpreted systems

Precondition	Value of succ$(f(t_1, \ldots, t_n))$
f constant	$h(0, \ldots, 0, f)$
$f \notin AC$, f non-constant	$f(t_1, \ldots, t_{n-1}, \mathrm{succ}(t_n))$
$f \in AC$, $t_n \succeq \ldots \succeq t_{k+1} \succ t_k =_{\mathrm{AC}} \ldots =_{\mathrm{AC}} t_1$, $k \geq 1$	$f(\mathrm{succ}(t_1), I_f(t_2), \ldots, I_f(t_k), t_{k+1}, \ldots, t_n)$
$f \in AC$, $t_1 =_{\mathrm{AC}} \ldots =_{\mathrm{AC}} t_n$	$f(succ(t_1), I_f(t_2), \ldots, I_f(t_n))$

Figure 4: Computation of the successor function

see [14]. For this reason we will suppose that the symbol h does not occur in the original constraint, like in the example above.

We believe, however, that the following theorem also holds for the other cases, if one first detects and eliminates constraints that are unsatisfiable because of the lack of *room* between two terms, like it is done for RPO constraints in [5, 8, 13] (for instance in the previous example there is no value for x between two successor terms, like in $f(0, h(0)) \succ x \succ f(0, 0)$).

Theorem 7.2 *Let $\succ_{\mathcal{F}}$ be a precedence with smallest non-constant and constant h and 0 resp. such that $h \notin AC$ and there is no $g \neq 0$ with $h \succ_{\mathcal{F}} g$, and suppose C is a constraint over $\mathcal{F} \setminus \{h\}$. Then C is satisfiable if and only if at least one of the pairs $(\mathcal{SF}, \mathcal{SF}^i)$ built from C according to the methods of the previous sections satisfies the rules of the figures 2 and 3.*

Proof (sketch)
The main difficulty is to prove that any pair $(\mathcal{SF}, \mathcal{SF}^i)$ which satisfies the properties of figures 2 and 3 do have a solution pair (σ, σ^i).

First note that, if (σ, σ^i) is a solution, then replacing each h in the target of (σ, σ^i) with h^k (for any fixed $k \geq 1$) will yield another solution. (This is when h is unary, but a similar property can be stated for any non-zero arity).

We build σ^i by induction as follows: at each step we enlarge the substitution σ^i by giving a value to the rightmost variable x, appearing in $\mathcal{SF}^i$ that still has no value. Then we prove that σ^i is a solution of Solved$(\mathcal{SF}^i)$ and is consistent with the condition: for all variables x^f and x^g in $\mathcal{SF}^i$ if $g \succeq_{\mathcal{F}} f$ then $I_g(x^f\sigma^i) = x^g\sigma^i$.

Let x^{f_i} be the first (from right to left) variable that appears in $\mathcal{SF}^i$ s.t. $x^{f_i}\sigma^i$ is not ground:

1. $x^{f_i}\sigma^i = u\sigma^i$, if $x^{f_i} = u \in$ Solved$(\mathcal{SF}^i)$;
2. $x^{f_i}\sigma^i = f(0, \ldots, 0, I_{f_i}(t\sigma^i))$, if t is the larger (wrt. $\succ_{\mathcal{SF}^i}$) term s.t. $x^{f_i} \succ t \in$ Solved$(\mathcal{SF}^i)$, and $x^{f_{i+1}} =_{\mathcal{SF}^i} I_{f_{i+1}}(t)$;
3. $x^{f_i}\sigma^i = f(0, \ldots, 0, x^{f_{i+1}}\sigma^i)$, otherwise.

First we show that the final σ^i is a solution of Solved$(\mathcal{SF}^i)$ and hence, by lemma 6.1, it is also a solution of $\mathcal{SF}^i$. We prove that the ground part of Solved$(\mathcal{SF}^i)\sigma^i$ is true, by induction on the number of variables $x \in \mathcal{SF}^i$ s.t. $x\sigma^i$ is ground. Note that by the way we build σ^i, if $x\sigma^i$ is ground then $t\sigma^i$ is ground for all $x \# t \in \mathcal{SF}^i$.

In the base case, i.e. when σ^i does not ground any variable, the ground part of Solved$(\mathcal{SF}^i)\sigma^i$ is empty. Therefore it is trivially true.

For the inductive step consider that we enlarge σ^i by defining $x^{f_i}\sigma^i$. Then there are six cases to be considered, corresponding to the relations in Solved$(\mathcal{SF}^i)$ that become ground because of the definition of $x^{f_i}\sigma^i$.

Then we prove that for all variables x^f and x^g in $\mathcal{SF}^i$ if $g \succeq_{\mathcal{F}} f$ then $I_g(x^f\sigma^i) = x^g\sigma^i$. It is enough to prove that $I_{f_{i+1}}(x^{f_i}\sigma^i) = x^{f_{i+1}}\sigma^i$.

Next, we build a solution σ which satisfies:

- the relations $\succ_{\mathcal{SF}}$ and $=_{\mathcal{SF}}$, i.e. $s\sigma \succ_{acrpo} t\sigma$ for all relations $s \succ_{\mathcal{SF}} t$ and $s\sigma =_{\mathrm{AC}} t\sigma$ for all relations $s =_{\mathcal{SF}} t$.
- $I(t\sigma) = I(t)\sigma^i$ for all t in $\mathcal{SF}$ (compatibility with the interpreted part) and
- $y_a\sigma = atomic_f(y\sigma)$ if $top(y) = f$ for all variables y and y_a in $\mathcal{SF}$ (compatibility between atomics).

As for σ^i, σ is built by induction on $\succ_{\mathcal{SF}}$ with the invariant that σ is a solution for the relations between ground terms of the system and fulfilling the three compatibility restrictions given above. Moreover, we build σ in such a way that there is some symbol h in $y\sigma$ for all variables y. In what follows, the *class* of t is the set of terms s such that $I(s) =_{\mathcal{SF}^i} I(t)$.

- If y has top symbol f and there is some corresponding y_a then we choose $y\sigma = f(y_a\sigma, y'^f\sigma^i)$. We also have $y^{f+1}\sigma^i = y_a^{f+1}\sigma^i$, which shows that $y_a\sigma$ is larger or equal than any direct subterm not headed by f of $y'^f\sigma^i$, thus preserving the compatibility between atomics and with the interpreted part.
- For the variables y such that $y =_{\mathcal{SF}} t$ for some compound term t, we let $y\sigma = t\sigma$. $t\sigma$ is defined at this point since the variables of t are in smaller classes. By construction, the compatibility with the interpreted part is preserved.
- For the remaining variables y that are minimal in their class we let $y\sigma = \mathrm{succ}(y^i\sigma^i)$ (except if $y\sigma^i$ is built over $\{h, 0\}$, in which case the successsor function does not preserve compatibility with the interpreted part). This ensures that there is at least one symbol h in $y\sigma$.

  If there is some term s such that $s \succ_{\mathcal{SF}} y$ and $s\sigma$ is already ground, then we have $s\sigma \succeq_{acrpo} I(s)\sigma^i =_{\mathrm{AC}} y^i\sigma^i$. Then, if $s\sigma =_{\mathrm{AC}} y^i\sigma^i$ or $s\sigma =_{\mathrm{AC}} \mathrm{succ}(y^i\sigma^i)$, before instantiating y we replace h with h^2, in order to create room for $y\sigma$.
- Finally, for the remaining variables y there is a maximal (wrt. $\succ_{acrpo}$) term $t\sigma$ with $y \succ_{\mathcal{SF}} t$. In this case we let $y\sigma = \mathrm{succ}(t\sigma)$.

  Again, if there is some term s such that $s \succ_{\mathcal{SF}} y \succ_{\mathcal{SF}} t$ and $s\sigma = \mathrm{succ}(t\sigma)$, i.e. there is no room for $y\sigma$, then we replace h with h^2.

$\square$

References

[1] F. Baader and K. Schultz. Combination techniques and decision problems for unification. In C. Kirchner, editor, *5th RTA*, LNCS 690, Montreal, June 16–18, 1993.

[2] A. Boudet. Unification in a combination of equational theories: an efficient algorithm. In M. E. Stickel, editor, *10th CADE*, LNAI 449, pages 292–307, Kaiserslautern, July 24–27, 1990.

[3] A. Boudet, J.-P. Jouannaud, and M. Schmidt-Schauss. Unification in Boolean rings and abelian groups. *Journal of Symbolic Computation*, 8:449–477, 1989.

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

[5] H. Comon. Solving symbolic ordering constraints. *Int. J. Found. Comp. Science*, 1(4):387–411, 1990.

[6] N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Comp. Sc.*, 17(3):279–301, 1982.

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

[8] 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.

[9] D. Kapur and P. Narendran. Double exponential complexity of computing complete sets of AC-unifiers. In *Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 11–21, Santa Cruz, California, USA, June 22–25, 1992. IEEE Computer Society Press.

[10] C. Kirchner. From unification in combination of equational theories to a new AC-unification algorithm. In H. Aït-Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures*, volume 2: Rewriting Techniques, chapter 6, pages 171–210. Academic Press, New York, 1989.

[11] C. Kirchner, H. Kirchner, and M. Rusinowitch. Deduction with symbolic constraints. *Revue Française d'Intelligence Artificielle*, 4(3):9–52, 1990.

[12] P. Narendran and M. Rusinowitch. Any ground associative commutative theory has a finite canonical system. In *Fourth int. conf. on Rewriting Techniques and Applications*, LNCS 488, pages 423–434, Como, Italy, Apr. 1991. Springer-Verlag.

[13] R. Nieuwenhuis. Simple LPO constraint solving methods. *Information Processing Letters*, 47:65–69, Aug. 1993.

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

[15] R. Nieuwenhuis and A. Rubio. AC-Superposition with constraints: No AC-unifiers needed. In A. Bundy, editor, *12th International Conference on Automated Deduction*, LNAI, Nancy, France, June 1994. Springer-Verlag.

[16] A. Rubio and R. Nieuwenhuis. A precedence-based total AC-compatible ordering. In C. Kirchner, editor, *5th International Conference on Rewriting Techniques and Applications*, LNCS 690, pages 374–388, Montreal, Canada, June 16–18, 1993. To appear in Theoretical Computer Science.

[17] M. Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *Journal of Symbolic Computation*, 8(1–2):51–99, 1989.

[18] M. E. Stickel. A unification algorithm for associative-commutative functions. *Journal of the ACM*, 28(3):423–434, 1981.

[19] L. Vigneron. Associative Commutative Deduction with constraints. In A. Bundy, editor, *12th International Conference on Automated Deduction*, LNAI, Nancy, France, June 1994. Springer-Verlag.