Metalevel Algorithms For Variant Satisfiability

Stephen Skeirik and José Meseguer

Department of Computer Science
University of Illinois at Urbana-Champaign, USA

Abstract. Variant satisfiability is a theory-generic algorithm to decide quantifier-free satisfiability in an initial algebra $T_{\Sigma/E}$ when the theory (Σ, E) has the finite variant property and its constructors satisfy a compactness condition. This paper: (i) gives a precise definition of several *meta-level sub-algorithms* needed for variant satisfiability; (ii) proves them correct; and (iii) presents a *reflective implementation* in Maude 2.7 of variant satisfiability using these sub-algorithms.

Keywords: finite variant property (FVP), folding variant narrowing, satisfiability in initial algebras, metalevel algorithms, reflection, Maude.

1 Introduction

SMT solving is at the heart of some of the most effective theorem proving and infinite-state model checking formal verification methods that can scale up to impressive verification tasks. A current limitation, however, is its *lack of extensibility*: current SMT solvers support a (typically small) library of decidable theories. Although these theories can be combined by the Nelson-Oppen (NO) [30, 31] or Shostak [33] methods under some conditions, only the theories in the SMT solver library and their combinations are available to the user: any other theories extending the tool must be implemented by the tool builders.

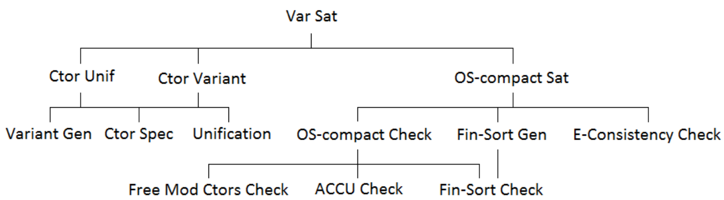
In practice, of course, the problem a user has to solve may not be expressible by the theories available in an SMT solver's library. Therefore, the goal of making SMT solvers *user-extensible*, so that a *user* can easily *define* new decidable theories and use them in the verification process is highly desirable.

For a well-known *subproblem* of SMT solving, such user extensibility has recently been achieved: *E-unifiability* is the subproblem of *satisfiability* defined by: (i) considering theories of the form $th(T_{\Sigma/E}(X))$, associated to equational theories (Σ, E), where $th(T_{\Sigma/E}(X))$ denotes the theory of the free (Σ, E)-algebra $T_{\Sigma/E}(X)$ on countably many variables X, and (ii) restricting ourselves to *positive* (i.e., negation-free) quantifier-free (QF) formulas. Lack of extensibility was the same: a unification tool supports a usually small library of theories (Σ, E), which can be combined by methods similar to the NO one (the paper [2] explicitly relates the NO algorithm and combination algorithms for unification). Again, the *user* could not extend such *decidable* unifiability/unification algorithms by defining new theories and using a *theory-generic* algorithm. This is now possible for theories (Σ, E) satisfying the *finite variant property* (FVP) [13] thanks to *variant unification* based on *folding variant narrowing* [18]. In fact, variant unification for user-definable FVP theories is already supported by Maude 2.7.

This suggests an obvious question: could variant unification be generalized to *variant satisfiability*, so that, under suitable conditions on and FVP theory (Σ, E), satisfiability of QF formulas in the initial algebra $T_{\Sigma/E}$ becomes *decidable* by a *theory-generic* satisfiability algorithm? This would then make satisfiability *user-extensible* as desired. This question has been positively answered in [27, 28] by giving general conditions under which satisfiability of QF formulas in the initial algebra $T_{\Sigma/E}$ of an FVP theory (Σ, E) is decidable. Section 3 summarizes the main results from [27, 28]; but the punchline is easy to summarize: Suppose that: (i) the convergent rewrite theory $\mathcal{R} = (\Sigma, B, R)$ is a so-called FVP decomposition of (Σ, E) (which is what it means for (Σ, E) to be FVP), (ii) B has a finitary B-unification algorithm, and (ii) $\mathcal{R}$ has an *OS-compact* constructor decomposition $\mathcal{R}_\Omega$ (definition in Section 3). Then satisfiability of QF formulas in $T_{\Sigma/E}$ is decidable by a *theory-generic* algorithm called variant satisfiability.

What this paper is about. The results in [27, 28] do not really provide an *algorithm* in the full sense of the word, but rather a theoretical *skeleton* on which such an algorithm can be fleshed out. Specifically, they *assume* that the constructor decomposition $\mathcal{R}_\Omega$ is *OS-compact*, but do not provide a way to *automate* both the checking of OS-compactness and the implementation of the various *auxiliary functions* needed for variant satisfiability based on OS-compactness. They also use the notions of *constructor variant* and *constructor unifier* (see Section 3), but give only their theoretical definitions instead of algorithms to compute them.

Main Contributions. A *theory-generic* algorithm such as variant satisfiability manipulates *metalevel* data structures such as theories, signatures, equations, disequations, rewrite rules, and the like. In this paper we provide for the first time: (i) a full-fledged algorithm for variant satisfiability with its sub-algorithms; (ii) a proof of its correctness; and (iii) a reflective Maude implementation of it. The algorithm uses the following *auxiliary functions*:



These functions automate the two main unsolved problems already mentioned: (a) checking and satisfiability in OS-compact theories; and (b) computing constructor variants and constructor unifiers. These sub-algorithms are defined and proved correct *at the metalevel of rewriting logic*. Since rewriting logic is *reflective* [10], the correctness-preserving passage from the metalevel description of the sub-algorithms to their implementations is very direct: we just *meta-represent* them at the logic's object level as suitable meta-level theories extending Maude's `META-LEVEL` module [8].

2 Preliminaries on Order-Sorted Algebra and Rewriting

The material is adapted from [25, 18, 28]. Due to space limitations the following elementary notions, which can be found in [25], are assume known: (i) order-sorted (OS) signature Σ; (ii) set $\widehat{S}$ of connected components (each denoted $[s] \in \widehat{S}$) of a poset of sorts $(S, \leqslant)$; (iii) sensible OS signature; (iv) order-sorted Σ-algebras and homomorphisms, and its associated category $\mathbf{OSAlg}_\Sigma$; and (v) the construction of the term algebra T_Σ and its initiality in $\mathbf{OSAlg}_\Sigma$ when Σ is sensible. Furthermore, for connected components $[s_1], \ldots, [s_n], [s] \in \widehat{S}$,

$$f_{[s]}^{[s_1]\ldots[s_n]} = \{f : s'_1 \ldots s'_n \to s' \in \Sigma \mid s'_i \in [s_i],\;\; 1 \leqslant i \leqslant n,\; s' \in [s]\}$$

denotes the family of "subsort polymorphic" operators f.

T_Σ will (ambiguously) denote: (i) the term algebra; (ii) its underlying S-sorted set; and (iii) the set $T_\Sigma = \bigcup_{s \in S} T_{\Sigma,s}$. For $[s] \in \widehat{S}$, $T_{\Sigma,[s]} = \bigcup_{s' \in [s]} T_{\Sigma,s'}$. An OS signature Σ is said to *have non-empty sorts* iff for each $s \in S$, $T_{\Sigma,s} \neq \emptyset$. *We will assume throughout that Σ has non-empty sorts.* An OS signature Σ is called *preregular* [19] iff for each $t \in T_\Sigma$ the set $\{s \in S \mid t \in T_{\Sigma,s}\}$ has a least element, denoted $ls(t)$. *We will assume throughout that Σ is preregular.*

An S-sorted set $X = \{X_s\}_{s \in S}$ of *variables*, satisfies $s \neq s' \Rightarrow X_s \cap X_{s'} = \emptyset$, and the variables in X are always assumed disjoint from all constants in Σ. The *Σ-term algebra* on variables X, $T_\Sigma(X)$, is the *initial algebra* for the signature $\Sigma(X)$ obtained by adding to Σ the variables X *as extra constants*. Since a $\Sigma(X)$-algebra is just a pair (A, α), with A a Σ-algebra, and α an *interpretation of the constants* in X, i.e., an S-sorted function $\alpha \in [X \to A]$, the $\Sigma(X)$-initiality of $T_\Sigma(X)$ can be expressed as the following theorem:

Theorem 1. *(Freeness Theorem). If Σ is sensible, for each $A \in \mathbf{OSAlg}_\Sigma$ and $\alpha \in [X \to A]$, there exists a unique Σ-homomorphism, $\_\alpha : T_\Sigma(X) \to A$ extending α, i.e., such that for each $s \in S$ and $x \in X_s$ we have $x\alpha_s = \alpha_s(x)$.*

In particular, when $A = T_\Sigma(X)$, an interpretation of the constants in X, i.e., an S-sorted function $\sigma \in [X \to T_\Sigma(X)]$ is called a *substitution*, and its unique homomorphic extension $\_\sigma : T_\Sigma(X) \to T_\Sigma(X)$ is also called a substitution. Define $dom(\sigma) = \{x \in X \mid x \neq x\sigma\}$, and $ran(\sigma) = \bigcup_{x \in dom(\sigma)} vars(x\sigma)$. A *variable specialization* is a substitution ρ that just renames a few variables and may lower their sort. More precisely, $dom(\rho)$ is a finite set of variables $\{x_1, \ldots, x_n\}$, with respective sorts $s_1, \ldots, s_n$, and ρ injectively maps the $x_1, \ldots, x_n$ to variables $x'_1, \ldots, x'_n$ with respective sorts $s'_1, \ldots, s'_n$ such that $s'_i \leqslant s_i$, $1 \leqslant i \leqslant n$.

The first-order language of *equational Σ-formulas* is defined in the usual way: its atoms are *Σ-equations* $t = t'$, where $t, t' \in T_\Sigma(X)_{[s]}$ for some $[s] \in \widehat{S}$ and each X_s is assumed countably infinite. The set $Form(\Sigma)$ of *equational Σ-formulas* is then inductively built from atoms by: conjunction ($\wedge$), disjunction ($\vee$), negation ($\neg$), and universal ($\forall x{:}s$) and existential ($\exists x{:}s$) quantification with sorted variables $x{:}s \in X_s$ for some $s \in S$. The literal $\neg(t = t')$ is denoted $t \neq t'$. Given a Σ-algebra A, a formula $\varphi \in Form(\Sigma)$, and an assignment $\alpha \in$

$[Y \to A]$, with $Y = fvars(\varphi)$ the free variables of φ, the *satisfaction relation* $A, \alpha \models \varphi$ is defined inductively as usual: for atoms, $A, \alpha \models t = t'$ iff $t\alpha = t'\alpha$; for Boolean connectives it is the corresponding Boolean combination of the satisfaction relations for subformulas; and for quantifiers: $A, \alpha \models (\forall x{:}s)\ \varphi$ (resp. $A, \alpha \models (\exists x{:}s)\ \varphi$) holds iff for all $a \in A_s$ (resp. some $a \in A_s$) we have $A, \alpha \uplus \{(x{:}s, a)\} \models \varphi$, where the assignment $\alpha \uplus \{(x{:}s, a)\}$ extends α by mapping $x{:}s$ to a. Finally, $A \models \varphi$ holds iff $A, \alpha \models \varphi$ holds for each $\alpha \in [Y \to A]$, where $Y = fvars(\varphi)$. We say that φ is *valid* (or *true*) in A iff $A \models \varphi$. We say that φ is *satisfiable* in A iff $\exists \alpha \in [Y \to A]$ such that $A, \alpha \models \varphi$, where $Y = fvars(\varphi)$. For a subsignature $\Omega \subseteq \Sigma$ and $A \in \mathbf{OSAlg}_\Sigma$, the *reduct* $A|_\Omega \in \mathbf{OSAlg}_\Omega$ agrees with A in the interpretation of all sorts and operations in Ω and discards everything in $\Sigma - \Omega$. If $\varphi \in Form(\Omega)$ we have the equivalence $A \models \varphi \ \Leftrightarrow\ A|_\Omega \models \varphi$.

An OS *equational theory* is a pair $T = (\Sigma, E)$, with E a set of Σ-equations. $\mathbf{OSAlg}_{(\Sigma,E)}$ denotes the full subcategory of $\mathbf{OSAlg}_\Sigma$ with objects those $A \in \mathbf{OSAlg}_\Sigma$ such that $A \models E$, called the (Σ, E)-*algebras*. $\mathbf{OSAlg}_{(\Sigma,E)}$ has an *initial algebra* $T_{\Sigma/E}$ [25]. Given $T = (\Sigma, E)$ and $\varphi \in Form(\Sigma)$, we call φ T-*valid*, written $E \models \varphi$, iff $A \models \varphi$ for each $A \in \mathbf{OSAlg}_{(\Sigma,E)}$. We call φ T-*satisfiable* iff there exists $A \in \mathbf{OSAlg}_{(\Sigma,E)}$ with φ satisfiable in A. Note that φ is T-*valid* iff $\neg\varphi$ is T-*unsatisfiable*. The inference system in [25] is *sound and complete* for OS equational deduction, i.e., for any OS equational theory (Σ, E), and Σ-equation $u = v$ we have an equivalence $E \vdash u = v \quad \Leftrightarrow \quad E \models u = v$. Deducibility $E \vdash u = v$ is abbreviated as $u =_E v$, called E-*equality*. An E-*unifier* of a system of Σ-equations, i.e., a conjunction $\phi = u_1 = v_1 \wedge \ldots \wedge u_n = v_n$ of Σ-equations is a substitution σ such that $u_i\sigma =_E v_i\sigma$, $1 \leqslant i \leqslant n$. An E-*unification algorithm* for (Σ, E) is an algorithm generating a *complete set* of E-unifiers $Unif_E(\phi)$ for any system of Σ equations ϕ, where "complete" means that for any E-unifier σ of ϕ there is a $\tau \in Unif_E(\phi)$ and a substitution ρ such that $\sigma =_E \tau\rho$, where $=_E$ here means that for any variable x we have $x\sigma =_E x\tau\rho$. The algorithm is *finitary* if it always terminates with a *finite set* $Unif_E(\phi)$ for any ϕ.

Given a set of equations B used for deduction modulo B, a preregular OS signature Σ is called B-*preregular*[1] iff for each $u = v \in B$ and variable specialization ρ, $ls(u\rho) = ls(v\rho)$.

In the above logical notions the lack of predicate symbols is only *apparent*: full order-sorted first-order logic can be *reduced* to order-sorted algebra and equational formulas. The essential idea is to view a predicate $p(x_1{:}s_1, \ldots, x_n{:}s_n)$ as a function symbol $p : s_1 \ldots s_n \to Pred$, with $Pred$, a new sort having a

[1] When the axioms B consist of a combination of associativity, commutativity, and (left and/or right) identity axioms, we can decompose B into the disjoint union $B = B_0 \uplus U$, where B_0 are associativity and/or commutativity axioms, and U are left and/or right identity axioms. The equations in U, of the general form $f(e, x) = x$ and/or $f(x, e) = x$, can be oriented as rewrite rules $R(U)$ of the form $f(e, x) \to x$ and/or $f(x, e) \to x$ to be applied *modulo* B_0. The B-preregularity notion can then be *broadened* by requiring only that: (i) Σ is preregular; (ii) Σ is B_0-preregular in the standard sense that $ls(u\rho) = ls(v\rho)$ for all $u = v \in B_0$ and sort specializations ρ; and (iii) the rules $R(U)$ are *sort-decreasing* in the sense of Definition 1. Maude automatically checks B-preregularity of an OS signature Σ in this broader sense [8].

constant tt. An atomic formula $p(t_1, \ldots, t_n)$ is then expressed as the equation $p(t_1, \ldots, t_n) = tt$. We refer the reader to [27, 28] for a detailed account of this reduction of predicate symbols to function symbols.

Recall the notation for term positions, subterms, and term replacement from [14]: (i) positions in a term viewed as a tree are marked by strings $p \in \mathbb{N}^*$ specifying a path from the root, (ii) $t|_p$ denotes the subterm of term t at position p, and (iii) $t[u]_p$ denotes the result of *replacing* subterm $t|_p$ at position p by u.

Definition 1. *A* rewrite theory *is a triple $\mathcal{R} = (\Sigma, B, R)$ with (Σ, B) an order-sorted equational theory and R a set of Σ-*rewrite rules*, i.e., sequents $l \to r$, with $l, r \in T_\Sigma(X)_{[s]}$ for some $[s] \in \widehat{S}$. In what follows it is always assumed that:*

1. *For each $l \to r \in R$, $l \notin X$ and $vars(r) \subseteq vars(l)$.*
2. *Each rule $l \to r \in R$ is* sort-decreasing*, i.e., for each variable specialization ρ, $ls(l\rho) \geqslant ls(r\rho)$.*
3. *Σ is B-*preregular *(if $B = B_0 \uplus U$, in the broader sense of Footnote 1).*
4. *Each equation $u = v \in B$ is* regular*, i.e., $vars(u) = vars(v)$, and* linear*, i.e., there are no repeated variables in u, and no repeated variables in v.*

The one-step *R, B-rewrite relation $t \to_{R,B} t'$, holds between $t, t' \in T_\Sigma(X)_{[s]}$, $[s] \in \widehat{S}$, iff there is a rewrite rule $l \to r \in R$, a substitution $\sigma \in [X \to T_\Sigma(X)]$, and a term position p in t such that $t|_p =_B l\sigma$, and $t' = t[r\sigma]_p$. Note that, by assumptions (2)–(3) above, $t[r\sigma]_p$ is always a well-formed Σ-term.*

$\mathcal{R}$ is called: (i) terminating *iff the relation $\to_{R,B}$ is well-founded; (ii)* strictly B-coherent *[26] iff whenever $u \to_{R,B} v$ and $u =_B u'$ there is a v' such that $u' \to_{R,B} v'$ and $v =_B v'$; (iii)* confluent *iff $u \to^*_{R,B} v_1$ and $u \to^*_{R,B} v_2$ imply that there are w_1, w_2 such that $v_1 \to^*_{R,B} w_1$, $v_2 \to^*_{R,B} w_2$, and $w_1 =_B w_2$ (where $\to^*_{R,B}$ denotes the reflexive-transitive closure of $\to_{R,B}$); and (iv)* convergent *if (i)–(iii) hold. If $\mathcal{R}$ is convergent, for each Σ-term t there is a term u such that $t \to^*_{R,B} u$ and $(\nexists v)\ u \to_{R,B} v$. We then write $u = t!_{R,B}$, and call $t!_{R,B}$ the* R, B-normal form *of t, which, by confluence, is unique up to B-equality.*

Given a set E of Σ-equations, let $R(E) = \{u \to v \mid u = v \in E\}$. A *decomposition* of an order-sorted equational theory (Σ, E) is a convergent rewrite theory $\mathcal{R} = (\Sigma, B, R)$ such that $E = E_0 \uplus B$ and $R = R(E_0)$. The key property of a decomposition is the following:

Theorem 2. *(Church-Rosser Theorem) [22, 26] Let $\mathcal{R} = (\Sigma, B, R)$ be a decomposition of (Σ, E). Then we have an equivalence:*

$$E \vdash u = v \quad \Leftrightarrow \quad u!_{R,B} =_B v!_{R,B}.$$

If $\mathcal{R} = (\Sigma, B, R)$ is a decomposition of (Σ, E), and X an S-sorted set of variables, the *canonical term algebra* $C_{\mathcal{R}}(X)$ has $C_{\mathcal{R}}(X)_s = \{[t!_{R,B}]_B \mid t \in T_\Sigma(X)_s\}$, and interprets each $f : s_1 \ldots s_n \to s$ as the function $C_{\mathcal{R}}(X)_f : ([u_1]_B, \ldots, [u_n]_B) \mapsto [f(u_1, \ldots, u_n)!_{R,B}]_B$. By the Church-Rosser Theorem we then have an isomorphism $h : T_{\Sigma/E}(X) \cong C_{\mathcal{R}}(X)$, where $h : [t]_E \mapsto [t!_{R,B}]_B$. In particular, when X is the empty family of variables, the canonical term algebra

$C_{\mathcal{R}}$ is an initial algebra, and is the most intuitive possible model for $T_{\Sigma/E}$ as an algebra of *values* computed by R, B-simplification.

Quite often, the signature Σ on which $T_{\Sigma/E}$ is defined has a natural decomposition as a disjoint union $\Sigma = \Omega \uplus \Delta$, where the elements of $C_{\mathcal{R}}$, that is, the *values* computed by R, B-simplification, are Ω-terms, whereas the function symbols $f \in \Delta$ are viewed as *defined functions* which are *evaluated away* by R, B-simplification. Ω (with same poset of sorts as Σ) is then called a *constructor subsignature* of Σ. Call a decomposition $\mathcal{R} = (\Sigma, B, R)$ of (Σ, E) *sufficiently complete* with respect to the *constructor subsignature* Ω iff for each $t \in T_\Sigma$ we have: (i) $t!_{R,B} \in T_\Omega$, and (ii) if $u \in T_\Omega$ and $u =_B v$, then $v \in T_\Omega$. This ensures that for each $[u]_B \in C_{\mathcal{R}}$ we have $[u]_B \subseteq T_\Omega$. Of course, we want Ω *as small as possible* with these properties. In Example 1 below, $\Omega = \{\top, \bot\}$ and $\Delta = \{\_ \wedge \_, \_ \vee \_\}$. Tools based on tree automata [11], equational tree automata [21], or narrowing [20], can be used to automatically check sufficient completeness of a decomposition $\mathcal{R}$ with respect to constructors Ω under some assumptions.

Sufficient completeness is closely related to the notion of a *protecting* theory inclusion.

Definition 2. *An equational theory* (Σ, E) protects *another theory* (Ω, E_Ω) *iff* $(\Omega, E_\Omega) \subseteq (\Sigma, E)$ *and the unique* Ω*-homomorphism* $h : T_{\Omega/E_\Omega} \to T_{\Sigma/E}|_\Omega$ *is an isomorphism* $h : T_{\Omega/E_\Omega} \cong T_{\Sigma/E}|_\Omega$.

A decomposition $\mathcal{R} = (\Sigma, B, R)$ protects *another decomposition* $\mathcal{R}_0 = (\Sigma_0, B_0, R_0)$ *iff* $\mathcal{R}_0 \subseteq \mathcal{R}$, *i.e.,* $\Sigma_0 \subseteq \Sigma$, $B_0 \subseteq B$, *and* $R_0 \subseteq R$, *and for all* $t, t' \in T_{\Sigma_0}(X)$ *we have: (i)* $t =_{B_0} t' \Leftrightarrow t =_B t'$, *(ii)* $t = t!_{R_0,B_0} \Leftrightarrow t = t!_{R,B}$, *and (iii)* $C_{\mathcal{R}_0} = C_{\mathcal{R}}|_{\Sigma_0}$.

$\mathcal{R}_\Omega = (\Omega, B_\Omega, R_\Omega)$ *is a* constructor decomposition *of* $\mathcal{R} = (\Sigma, B, R)$ *iff* $\mathcal{R}$ *protects* $\mathcal{R}_\Omega$ *and* Σ *and* Ω *have the same poset of sorts, so that by (iii) above* $\mathcal{R}$ *is sufficiently complete with respect to* Ω. *Furthermore,* Ω *is called a subsignature of* free constructors modulo B_Ω *iff* $R_\Omega = \varnothing$, *so that* $C_{\mathcal{R}_0} = T_{\Omega/B_\Omega}$.

3 Variants and Variant Satisfiability

The notion of *variant* answers two questions: (i) how can we best describe symbolically the elements of $C_{\mathcal{R}}(X)$ that are *reduced substitution instances* of a given *pattern term* t? and (ii) when is such a symbolic description *finite*?

Definition 3. *Given a decomposition* $\mathcal{R} = (\Sigma, B, R)$ *of an OS equational theory* (Σ, E) *and a* Σ*-term* t, *a* variant[2] *[13, 18] of* t *is a pair* (u, θ) *such that: (i)* $u =_B (t\theta)!_{R,B}$, *(ii) if* $x \notin vars(t)$, *then* $x\theta = x$, *and (iii)* $\theta = \theta!_{R,B}$, *that is,* $x\theta = (x\theta)!_{R,B}$ *for all variables* x. (u, θ) *is called a* ground variant *iff* $u \in T_\Sigma$. *Note that if* (u, θ) *is a ground variant of some* t, *then* $[u]_B \in C_{\mathcal{R}}$. *Given variants* (u, θ) *and* (v, γ) *of* t, (u, θ) *is called* more general *than* (v, γ), *denoted* $(u, \theta) \sqsupseteq_{R,B} (v, \gamma)$, *iff there is a substitution* ρ *such that: (i)* $\theta\rho =_B \gamma$, *and (ii)* $u\rho =_B v$. *Let*

[2] For a discussion of similar but not exactly equivalent versions of the variant notion see [7]. Here we follow the formulation in [18].

$[\![t]\!]_{R,B} = \{(u_i, \theta_i) \mid i \in I\}$ *denote a* most general complete set of variants *of* t, *that is, a set of variants such that: (i) for any variant* (v, γ) *of* t *there is an* $i \in I$, *such that* $(u_i, \theta_i) \sqsupseteq_{R,B} (v, \gamma)$; *and (ii) for* $i, j \in I$, $i \neq j \Rightarrow ((u_i, \theta_i) \not\sqsupseteq_{R,B} (u_j, \theta_j) \wedge (u_j, \theta_j) \not\sqsupseteq_{R,B} (u_i, \theta_i))$. *A decomposition* $\mathcal{R} = (\Sigma, B, R)$ *of* (Σ, E) *has the* finite variant property *[13] (FVP) iff for each* Σ*-term* t *there is a* finite *most general complete set of variants* $[\![t]\!]_{R,B} = \{(u_1, \theta_1), \ldots, (u_n, \theta_n)\}$.

If B has a finitary unification algorithm, the *folding variant narrowing* strategy described in [18] provides an effective method to generate $[\![t]\!]_{R,B}$. Furthermore, $[\![t]\!]_{R,B}$ is finite for each t, so that the strategy *terminates* iff $\mathcal{R}$ is FVP.

Example 1. Let $\mathcal{B} = (\Sigma, B, R)$ with Σ having a single sort, say *Bool*, constants $\top, \bot$, and binary operators $\_ \wedge \_$ and $\_ \vee \_$, B the associativity and commutativity (AC) axioms for both $\_ \wedge \_$ and $\_ \vee \_$, and R the rules: $x \wedge \top \to x$, $x \wedge \bot \to \bot$, $x \vee \bot \to x$, and $x \wedge \top \to \top$. Then $\mathcal{B}$ is FVP. For example, $[\![x \wedge y]\!]_{R,B} = \{(x \wedge y, id), (y, \{x \mapsto \top\}), (x, \{y \mapsto \top\}), (\bot, \{x \mapsto \bot\}), (\bot, \{y \mapsto \bot\})\}$.

FVP is a *semi-decidable* property [7], which can be easily verified (when it holds) by checking, using folding variant narrowing, that for each function symbol f the term $f(x_1, \ldots, x_n)$, with the sorts of the $x_1, \ldots, x_n$ those of f, has a finite number of most general variants.

Folding variant narrowing provides also a method for generating a *complete set of E-unifiers* when (Σ, E) has a decomposition $\mathcal{R} = (\Sigma, B, R)$ with B having a finitary B-unification algorithm [18]. To express systems of equations, say, $u_1 = v_1 \wedge \ldots \wedge u_n = v_n$, as *terms*, we can extend Σ to a signature $\Sigma^\wedge$ by adding:

1. for each connected component $[s]$ that does not already have a top element, a fresh new sort $\top_{[s]}$ with $\top_{[s]} > s'$ for each $s' \in [s]$. In this way we obtain a (possibly extended) poset of sorts $(S_\top, \geqslant)$;
2. fresh new sorts *Lit* and *Conj* with a subsort inclusion *Lit* < *Conj*, with a binary conjunction operator $\_ \wedge \_ : \mathit{Lit}\ \mathit{Conj} \to \mathit{Conj}$, and
3. for each connected component $[s] \in \widehat{S_\top}$ with top sort $\top_{[s]}$, binary operators $\_ = \_ : \top_{[s]}\ \top_{[s]} \to \mathit{Lit}$ and $\_ \neq \_ : \top_{[s]}\ \top_{[s]} \to \mathit{Lit}$.

Theorem 3. *[28] Under the above assumptions on* $\mathcal{R}$, *let* $\phi = u_1 = v_1 \wedge \ldots \wedge u_n = v_n$ *be a system of* Σ*-equations viewed as a* $\Sigma^\wedge$*-term of sort Conj. Then*

$$\{\theta\gamma \mid (\phi', \theta) \in [\![\phi]\!]_{R,B} \wedge \gamma \in \mathit{Unif}_B(\phi') \wedge (\phi'\gamma, \theta\gamma) \text{ is a variant of } \phi\}$$

is a complete set of E*-unifiers for* ϕ, *where* $\mathit{Unif}_B(\phi')$ *denotes a complete set of most general* B*-unifiers for each variant* $\phi' = u'_1 = v'_1 \wedge \ldots \wedge u'_n = v'_n$.

Since if $\mathcal{R} = (\Sigma, B, R)$ is FVP, then $\mathcal{R}^\wedge = (\Sigma^\wedge, B, R)$ is also FVP, Theorem 3 shows that if a finitary B-unification algorithm exists and $\mathcal{R}$ is an FVP decomposition of (Σ, E), then E has a finitary E-unification algorithm.

The key question asked and answered in [27, 28] is: given an FVP decomposition $\mathcal{R} = (\Sigma, B, R)$ of an equational theory (Σ, E), under what conditions is satisfiability of QF equational Σ-formulas in the canonical term algebra $C_{\mathcal{R}}$ decidable? It turns out that: (i) $\mathcal{R}$ having a constructor decomposition

$\mathcal{R}_\Omega = (\Omega, B_\Omega, R_\Omega)$, and (ii) the associated notions of *constructor variant* and *constructor unifier* [28] play a crucial role in answering this question.

Definition 4. *Let $\mathcal{R} = (\Sigma, B, R)$ be a decomposition of (Σ, E), and let $\mathcal{R}_\Omega = (\Omega, B_\Omega, R_\Omega)$ be a constructor decomposition of $\mathcal{R}$. Then an R, B-variant (u, θ) of a Σ-term t is called a* constructor R, B-variant *of t iff $u \in T_\Omega(X)$.*

Suppose, furthermore, that B has a finitary B-unification algorithm, so that, given a unification problem $\phi = u_1 = v_1 \wedge \ldots \wedge u_n = v_n$, Theorem 3 allows us to generate the complete set of E-unifiers

$$\{\theta\gamma \mid (\phi', \theta) \in [\![\phi]\!]_{R,B} \wedge \gamma \in \mathit{Unif}_B(\phi') \wedge (\phi'\gamma, \theta\gamma) \text{ is a variant of } \phi\}$$

Then a constructor E-unifier[3] *of ϕ is either: (1) a unifier $\theta\gamma$ in the above set with $\phi'\gamma \in T_{\Omega^\wedge}(X)$; or otherwise, (2) a unifier $\theta\gamma\alpha$ such that: (i) $\theta\gamma$ belongs the above set, (ii) α is a substitution of the variables in $ran(\theta\gamma)$ such that $\phi'\gamma\alpha \in T_{\Omega^\wedge}(X)$, and (iii) $(\phi'\gamma\alpha, \theta\gamma\alpha)$ is a variant of ϕ. $mgu^\Omega_\mathcal{R}(\phi)$ denotes a set of* most general *constructor E-unifiers of ϕ, i.e., for any constructor E-unifier μ of ϕ there is another one $\eta \in mgu^\Omega_\mathcal{R}(\phi)$ and a substitution ν such that $\mu =_B \eta\nu$.*

Note that if (v, δ) is a ground variant of t, then $[v]_B \in C_\mathcal{R}$, so that v is an Ω-term. Therefore, any ground variant (v, δ) of t is "covered" by some constructor variant (u, θ) of t, i.e., $(u, \theta) \sqsupseteq_{R,B} (v, \delta)$. If (Σ, E) has a decomposition $\mathcal{R} = (\Sigma, B, R)$, B has a finitary B-unification algorithm and we are only interested in characterizing the *ground solutions* of an equation in the initial algebra $T_{\Sigma/E}$, only constructor E-unifiers are needed, since they completely cover all such solutions. Likewise, if we are only interested in *unifiability* of a system of equations only constructor E-unifiers are needed.

Theorem 4. *[27, 28] Let (Σ, E) have a decomposition $\mathcal{R} = (\Sigma, B, R)$ with B having a finitary B-unification algorithm. Then, for each system of Σ-equations $\phi = u_1 = v_1 \wedge \ldots \wedge u_n = v_n$, where $Y = vars(\phi)$, we have:*

1. *(Completeness for Ground Unifiers). If $\delta \in [Y \to T_\Sigma]$ is a ground E-unifier of ϕ, then there is a constructor E-unifier $\eta \in mgu^\Omega_\mathcal{R}(\phi)$ and a substitution β such that $\delta =_E \eta\beta$, i.e., $x\delta =_E x\eta\beta$ for each variable $x \in Y$.*
2. *(Unifiability). $T_{\Sigma/E} \models (\exists Y)\ \phi$ iff ϕ has a constructor E-unifier.*

Given an OS equational theory (Σ, E), call a Σ-equality $u = v$ *E-trivial* iff $u =_E v$, and a Σ-disequality $u \neq v$ *E-consistent* iff $u \neq_E v$. Likewise, call a conjunction $\bigwedge D$ of Σ-disequalities *E-consistent* iff each $u \neq v$ in D is so.

Theorem 4 is a key step to find conditions for the decidable satisfiability of QF equational Σ-formulas in $C_\mathcal{R}$ for $\mathcal{R} = (\Sigma, B, R)$ an FVP decomposition of (Σ, E), where B has a finitary B-unification algorithm and $\mathcal{R}$ has a constructor decomposition $\mathcal{R}_\Omega = (\Omega, B_\Omega, R_\Omega)$. The key idea is to reduce the problem to one of satisfiability of a conjunction of *Ω-disequalities* in the simpler canonical term algebra $C_{\mathcal{R}_\Omega}$. By $C_\mathcal{R}|_\Omega = C_{\mathcal{R}_\Omega}$, Theorem 4, and the Descent Theorems in [27, 28] (see [27, 28] for full details), we can apply the following algorithm to a conjunction of literals $\phi = \bigwedge G \wedge \bigwedge D$, with G equations and D disequations:

[3] [27, 28] give examples of constructor variants and constructor unifiers.

1. Thanks to Theorem 4 we need only compute the *constructor* E-unifiers $mgu^{\Omega}_{\mathcal{R}}(\bigwedge G)$, and reduce to the case of deciding the satisfiability of some conjunction of disequalities $(\bigwedge D\alpha)!_{R,B}$, for some $\alpha \in mgu^{\Omega}_{\mathcal{R}}(\bigwedge G)$, discarding any $(\bigwedge D\alpha)!_{R,B}$ containing a B-inconsistent disequality.
2. For each remaining $(\bigwedge D\alpha)!_{R,B}$ we can then compute a finite, complete set of most general R, B-variants $[\![(\bigwedge D\alpha)!_{R,B}]\!]_{R,B}$ by folding variant narrowing, and obtain for each of them its B_{Ω}-consistent constructor variants $\bigwedge D'$.
3. Then by the Descent Theorems in [27, 28], ϕ will be satisfiable in $C_{\mathcal{R}}$ iff $\bigwedge D'$ is satisfiable in $C_{\mathcal{R}_{\Omega}}$ for some such $\bigwedge D'$ and some such α.

Therefore, the method hinges upon being able to decide when a conjunction of Ω-disequalities $\bigwedge D'$ is satisfiable in $C_{\mathcal{R}_{\Omega}}$. This is decidable if $\mathcal{R}_{\Omega}$ is the decomposition of an OS-*compact theory*, which generalizes the notion of *compact theory* in [12]:

Definition 5. *[27, 28] An equational theory (Σ, E) is called* OS-compact *iff: (i) for each sort s in Σ we can effectively determine whether $T_{\Sigma/E,s}$ is finite or infinite, and, if finite, can effectively compute a representative ground term $rep([u]) \in [u]$ for each $[u] \in T_{\Sigma/E,s}$ (ii) $=_E$ is decidable and E has a finitary unification algorithm; and (iii) any E-consistent finite conjunction $\bigwedge D$ of Σ-disequalities whose variables all have infinite sorts is satisfiable in $T_{\Sigma/E}$.*

The reason why satisfiability of a conjunction of disequalities in the initial algebra of an OS-compact theory is decidable [27, 28] is fairly obvious: by (iii) it is decidable when all variables have infinite sorts; and we can always reduce to a disjunction of formulas in that case by instantiating each variable with a finite sort s by all the possible representatives in $T_{\Sigma/E,s}$. Therefore we have:

Corollary 1. *For $\mathcal{R} = (\Sigma, B, R)$ an FVP decomposition of (Σ, E), where B has a finitary B-unification algorithm and $\mathcal{R}$ has an OS-compact constructor decomposition $\mathcal{R}_{\Omega}$, satisfiability of QF equational Σ-formulas in $C_{\mathcal{R}}$ is decidable.*

The papers [27, 28] contain many examples of commonly used theories that have FVP specifications whose constructor decompositions are OS-compact. This can be established by one of the two methods discussed below.

A first method to show OS-compactness is both very simple and widely applicable to constructor decompositions of FVP theories. It applies to OS equational theories of the form $(\Omega, ACCU)$, where $ACCU$ stands for any combination of associativity and/or commutativity and/or left- or right-identity axioms, except combinations where the same operator is associative but not commutative. We also assume that if any typing for a binary operator f in a subsort-polymorphic family $f^{[s]\ [s]}_{[s]}$ satisfies some axioms in $ACCU$, then any other typing in $f^{[s]\ [s]}_{[s]}$ satisfies the *same* axioms. The following theorem generalizes to the order-sorted and $ACCU$ case a similar result in [12] for the unsorted and AC case:

Theorem 5. *[27, 28] Under the above assumptions $(\Omega, ACCU)$ is OS-compact. Furthermore, satisfiability of QF Ω-formulas in $T_{\Omega/ACCU}$ is decidable.*

The range of FVP theories whose initial algebras have decidable QF satisfiability is greatly increased by a second method of *satisfiability-preserving FVP parameterized theories*. For our present purposes it suffices to summarize the basic general facts and assumptions for the case of FVP parameterized data types with a *single parameter* X. That is, we can focus on parameterized FVP theories of the form $\mathcal{R}[X] = (\mathcal{R}, X)$, where $\mathcal{R} = (\Sigma, B, R)$ is an FVP decomposition of an OS equational theory (Σ, E), and X is a sort in Σ (called the *parameter sort*) such that: (i) is empty, i.e., $T_{\Sigma,X} = \varnothing$; and (ii) X is a minimal element in the sort order, i.e., there is no other sort s' with $s' < X$.

Consider an FVP decomposition $\mathcal{G} = (\Sigma', B', R')$ of a finitary OS equational theory (Σ', E'), which we can assume without loss of generality is disjoint from (Σ, E), and additionally let s be a sort in Σ'. Then the *instantiation* $\mathcal{R}[\mathcal{G}, X \mapsto s] = (\Sigma[\Sigma', X \mapsto s], B \cup B', R \cup R')$ is the decomposition of a theory $(\Sigma[\Sigma', X \mapsto s], E \cup E')$, extending (Σ', E'), where the signature $\Sigma[\Sigma', X \mapsto s]$ is defined as the union $\Sigma[X \mapsto s] \cup \Sigma'$, with $\Sigma[X \mapsto s]$ just like Σ, except for X renamed to s. Its set of sorts is $(S - \{X\}) \uplus S'$, and the poset ordering combines those of $\Sigma[X \mapsto s]$ and Σ'. Furthermore, $\mathcal{R}[\mathcal{G}, X \mapsto s]$ is also FVP under mild assumptions [27].

Suppose B, B' and $B \cup B'$ have finitary unification algorithms and both $\mathcal{R}[X] = (\mathcal{R}, X)$ and $\mathcal{G}$ protect, respectively, the two constructor theories, say $\mathcal{R}_\Omega[X] = (\Omega, B_\Omega, R_\Omega)$ and $\mathcal{G}_{\Omega'} = (\Omega', B_{\Omega'}, R_{\Omega'})$. Then $\mathcal{R}[\mathcal{G}, X \mapsto s]$ will protect $\mathcal{R}_\Omega[\mathcal{G}_{\Omega'}, X \mapsto s]$. Suppose, further, that B_Ω, $B_{\Omega'}$, and $B_\Omega \cup B_{\Omega'}$ have decidable equality. The general satisfiability-preserving method of interest is then as follows: (i) assuming that $\mathcal{G}_{\Omega'}$ is the decomposition of an OS-compact theory, then (ii) under some assumptions about the cardinality of the sort s, prove the OS-compactness of $\mathcal{R}_\Omega[\mathcal{G}_{\Omega'}, X \mapsto s]$. It then follows from our earlier reduction of satisfiability in initial FVP algebras to their constructor decompositions that satisfiability of QF formulas in the initial model of the instantiation $\mathcal{R}[\mathcal{G}, X \mapsto s]$ is decidable.

In [27] the following parameterized data types have been proved satisfiability-preserving following the just-described pattern of proof: (i) $\mathcal{L}[X]$, *parameterized lists*, which is just an example illustrating the general case of any constructor-selector-based [29] parameterized data type; (ii) $\mathcal{L}^c[X]$, *parameterized compact lists*, where any two identical contiguous list elements are identified [16, 15]; (iii) $\mathcal{M}[X]$, *parameterized multisets*; (iv) $\mathcal{S}[X]$, *parameterized sets*; and (v) $\mathcal{H}[X]$, *parameterized hereditarily finite sets*.

4 Metalevel Algorithms for Variant Satisfiability

For $\mathcal{R} = (\Sigma, B, R)$ an FVP decomposition of (Σ, E), where B has a finitary B-unification algorithm and $\mathcal{R}$ has a constructor decomposition $\mathcal{R}_\Omega$, the issue of the decidable satisfiability of QF equational Σ-formulas in $C_{\mathcal{R}}$ has been condensed in Section 3 to two key sub-issues: (i) steps (1)–(3) in the high-level algorithm, which reduce satisfiability of a conjunction of Σ-literals in $C_{\mathcal{R}}$ to satisfiability of a conjunction of Ω-disequalities in $C_{\mathcal{R}_\Omega}$; and (ii) decidable satisfiability of conjunctions of Ω-disequalities in $C_{\mathcal{R}_\Omega}$ when $\mathcal{R}_\Omega$ is OS-compact (Corollary 1).

At a theoretical level this gives the *skeleton* of a high-level algorithm for variant satisfiability. But at a concrete, algorithmic level several important questions, essential for having an actual satisfiability *algorithm*, remain unresolved, including: (1) how can we *automatically check* that the constructor decomposition $\mathcal{R}_\Omega$ is OS-compact using the two methods for OS-compactness outlined in Section 3? (2) how can we *compute* constructor variants and constructor unifiers? (3) how can we *prove* that the auxiliary algorithms answering questions (1) and (2) are *correct*? and (4) how can we *implement* both the main algorithm and the auxiliary algorithms in a correctness-preserving manner?

Let us begin with question (3). The algorithm skeleton sketched in Section 3 manipulates metalevel entities like operators, signatures, terms, equations, and theories. Likewise, the checks for OS-compactness and the computation of constructor variants and constructor unifiers (questions (1)–(2)) are problems fully expressible in terms of such metalevel entities. Therefore, both for mathematical clarity and for simplicity of the needed correctness proofs, the definitions of the auxiliary algorithms should be carried out at the metalevel of rewriting logic.

This brings us to question (4), which has a simple answer: since rewriting logic is *reflective* [10], once we have defined and proved correct at the metalevel the auxiliary algorithms solving questions (1) and (2), we can derive correct implementations for them by *meta-representing* them at the logic's object level as equational or rewrite theories. In fact, this can be carried out in Maude by defining suitable meta-level theories extending the `META-LEVEL` module [8].

The previous paragraphs lead us to the main contributions of the present paper. We answer questions (1) and part of (3) by defining and proving correct at the metalevel a method to check OS-compactness, including: (a) checking which sorts s satisfy $|T_{\Omega/B_\Omega,s}| < \aleph_0$, and (b) computing for each such s a unique representative $rep([t]_{B_\Omega})$ for each $[t]_{B_\Omega} \in T_{\Omega/B_\Omega,s}$. We answer question (2) and the other part of (3) by defining and proving correct at the meta-level a method to compute constructor unifiers and constructor variants. And we answer question (4) by meta-representing both the auxiliary algorithms and the main algorithm (already proved correct at the meta-level in [27, 28]) in Section 5.

To help guide the discussion, the reader may refer to the tree diagram in the Introduction, which describes the dependencies among different subalgorithms. We first present a high-level description of the algorithms with some details omitted for readability; all remaining details, together with full proofs of correctness, can be found in the appendices.

4.1 OS-Compact Satisfiability

E_Ω-*consistency* of a conjunction of Ω-disequalities $\bigwedge D'$ in a constructor decomposition $\mathcal{R}_\Omega = (\Omega, B_\Omega, R_\Omega)$ is easy to check: we may assume $\bigwedge D'$ in R_Ω, B_Ω-normal form and just need to check that $u \neq_{B_\Omega} v$ for each $u \neq v$ in $\bigwedge D'$.

Checking that the constructor subtheory $\mathcal{R}_\Omega$ of $\mathcal{R}$ is *OS-compact* breaks into two cases: (1) when $\mathcal{R}$ is an *unparameterized theory*; and (2) when $\mathcal{R}$ is the instantiation of a possibly *nested* collection of satisfiability-preserving *parameterized theories* such as, for, example, *sets of lists of natural numbers*. In case (2) it is enough (for the parameterized theories described in Section 3) to check that: (i) the unparameterized theory $\mathcal{G}$ in the innermost instantiation (in our example the theory $\mathcal{N}_+$ of naturals with addition) is *OS-compact*, and the chosen sort (in our example the sort *Nat*) is *infinite*; and (ii) that the sorts chosen to instantiate each remaining parameter is the *principal sort* of the parameterized module immediately below in the nesting. In our example this is just checking that the parameter sort X for the *set* parameterized module is instantiated to the principal sort, namely *List*, of the *list* parameterized module immediately below. In this way, checking OS-compactness of $\mathcal{R}_\Omega$ in the, nested, parameterized case is reduced to checking OS-compactness of the unparameterized inner argument, plus a check of an infinite sort. All checks for the unparameterized case (1), including the two needed in case (2), are described below.

OS-Compactness Check (Unparameterized Case). As shown in Theorem 5, a sufficient condition for an unparameterized constructor decomposition $\mathcal{R}_\Omega = (\Omega, B_\Omega, R_\Omega)$ to be OS-compact is for $\mathcal{R}_\Omega$ to be of the form $\mathcal{R}_\Omega = (\Omega, ACCU, \varnothing)$. Thus, a sufficient condition is to require: (1) B_Ω to be a set of ACCU axioms, and (2) Ω to be a signature of *free constructors* modulo B_Ω. Fortunately, both of these subgoals are quite simple to check. Goal (1) can be solved by iterating over each axiom and applying a case analysis against its structure. Goal (2) can be solved by an application of propositional tree automata (PTA). In particular, if the rules R in $\mathcal{R}$ are linear and unconditional, then constructor freeness modulo B is translatable into a PTA emptiness problem; see [32] for further details.

Finite Sort Classification. Another needed algorithm takes as input a signature Ω and a sort s and checks if $|T_{\Omega/B_\Omega,s}| < \aleph_0$. We solve this problem in two phases: (1) we devise an algorithm to check $|T_{\Omega,s}| < \aleph_0$ (2) we use this as a subroutine in an *approximate* algorithm to check $|T_{\Omega/B_\Omega,s}| < \aleph_0$ when $B_\Omega = ACCU$. If the approximate algorithm fails to classify some s as either infinite or finite, s returned to the user as a proof obligation (Appendix C, Corollary 7).

If Ω is finite and has non-empty sorts, we show that $|T_{\Omega,s}| = \aleph_0$ iff there exists a *cycle* in the relation $(\prec) \subseteq S^2$ reachable from s where $s \prec s'$ iff the formula $\exists f : s_1 \cdots s_n \to s'' \in \Omega\, \exists i \in \mathbb{N}[s'' \leqslant s \wedge s \leqslant s_i] \vee [s' < s]$ holds. We construct a rewrite theory R_F over S such that $s \to_{R_F} s'$ iff $s \prec s'$. If $cy(S) = \{s \in S \mid s \to^+_{R_F} s\}$, then $s \to^*_{R_F} s'$ with $s' \in cy(S)$ implies $|T_{\Omega,s}| = \aleph_0$. Then $\bigvee_{s' \in cy(S_{\supset\varnothing})} R_F \vdash s \to^* s'$ holds iff there is a cycle in the relation $(\prec)$ reachable from s (Appendix C, Theorem 10).

We now lift the algorithm above to phase (2). We can show that for ACC axioms B_Ω there is an exact correspondence $|T_{\Omega/B_\Omega,s}| < \aleph_0$ iff $|T_{\Omega,s}| < \aleph_0$. The tricky case is when B_Ω contains unit axioms, since they may break this happy correspondence. For example, consider the unsorted signature $\Omega = (0, \_+\_)$ where 0 is a unit element for $\_+\_$. For the ACCU case, shows two simple checks that apply in most cases. Failing that, the classification of sort s is returned to the user as a proof obligation (Appendix C, Lemmas 10, 11, and 12).

Finite Sort Representative Generation. Here we require a method to do two things: (1) when $|T_{\Omega/B_\Omega,s}| < \aleph_0$, we can compute each $[t]_{B_\Omega} \in T_{\Omega/B_\Omega,s}$ (2) for each such $[t]_{B_\Omega}$, we can compute a unique representative $rep([t]_{B_\Omega})$. We first show how to generate $T_{\Omega,s}$. Recall that any order-sorted signature Ω can be viewed as a tree automaton such that the tree automaton accepts a term t in final state s iff $t \in T_{\Omega,s}$. Note also that tree automata are very simple *ground* rewrite theories. Let R_P be the ground rewrite rules for Ω's tree automaton over $T_{\Omega \cup S}$, so that $t \in T_{\Omega,s}$ iff $t \to^{+}_{R_P} s$. Let $R_G = R_P^{-1}$ then $T_{\Omega,s} = \{t \in T_\Omega \mid s \to^{!}_{R_G} t\}$ (Appendix C, Corollary 6). Furthermore, if $|T_{\Omega,s}| < \aleph_0$ and Ω has no empty sorts, this process will always terminate. Note that we can apply the rules R_G modulo B_Ω. Then the set $\mathrm{Rep}(T_{\Omega/B_\Omega,s}) = \{\mathrm{rep}([t]) \mid [t] \in T_{\Omega/B_\Omega,s}\}$ is exactly the set $\mathrm{Rep}(T_{\Omega/B_\Omega,s}) = \{t \mid s \to^{!}_{R_G,B_\Omega} t\}$.

4.2 Constructor Variants and Constructor Unifiers

We first show how to compute a set of *most general* constructor variants of a term t (i.e. a set of constructor variants $[\![t]\!]^{\Omega}_{R,B}$ such that for any constructor variant (t', θ), we have $\exists (t'', \psi) \in [\![t]\!]^{\Omega}_{R,B}[(t'', \psi) \sqsupseteq_{R,B} (t', \phi)]$) and then show how to use this method to compute a set of *most general* constructor unifiers $mgu^{\Omega}_{\mathcal{R}}(\phi)$. Recall that a constructor variant is just an variant (t, θ) such that $t \in T_\Omega(X)$. Thus, $[\![t]\!]^{\Omega}_{R,B}$ can be computed in two steps: (1) computing a set of most general variants $[\![t]\!]_{R,B}$ (2) for each most general variant (t', θ), compute the set of its *most general* constructor instances, i.e. a set of instances $mgci_B(t') = \{t'\eta_1, \cdots, t'\eta_n\}$ where for any other instance $t'\alpha$, there exists a substitution γ and η_i with $\alpha =_B \eta_i\gamma$. Note that (1) can be solved via folding variant narrowing, so we tackle (2) by a reduction to a B-unification problem via a signature transformation $\Sigma \mapsto \Sigma^c$. In this transformed signature, the instances $mgci_B(t')$ correspond exactly to the solutions of a *single* B-unification problem.

The signature transformation $\Sigma \mapsto \Sigma^c$ splits into two steps: (i) we extend the sort poset $(S, <)$ of Σ and Ω and (ii) likewise extend the operator sets F and F_Ω, as specified by the definitions below, respectively. Recall we assume Σ (and thus Ω) are finite; otherwise these transformations would not be effective.

Definition 6. *A* constructor sort refinement *of* $(S, <)$ *is defined by the following: (a) a set* $S^c = S \uplus S^{\downarrow}$ *with* $c : S \to S^{\downarrow}$ *a bijection, (b) a relation* $(<^c)$ *the smallest strict order where: (i)* $\forall s, s' \in S\, [s < s' \Leftrightarrow [s <^c s' \wedge c(s) <^c c(s')]]$ *and (ii)* $\forall s \in S\, [c(s) <^c s]$*, and (c) functions* $(^\bullet) : S^c \to S$ *and* $(_\bullet) : S^c \to S^{\downarrow}$ *defined by* $s^\bullet = s$ `if` $s \in S$ `else` $c^{-1}(s)$ *and* $s_\bullet = s$ `if` $s \in S^{\downarrow}$ `else` $c(s)$*.*

We let $(<^c)$ also ambiguously denote its extension to strings $(S^c)^*$. Similarly, let $(^\bullet)$ and $(_\bullet)$ ambiguously denote their extensions to $(S^c)^*$ and $\mathcal{P}(S^c)$.

Definition 7. *Given* $\Sigma = ((S,<),F)$ *and* $\Omega = ((S,<),F_\Omega)$ *where* $\Omega \subseteq \Sigma$ *and* $(S^c,<^c,(^\bullet),(_\bullet))$ *is a constructor sort refinement of* $(S,<)$*, we define signatures* $\Omega^\downarrow = ((S^c,<^c),F^\downarrow_\Omega)$ *and* $\Sigma^c = ((S^c,<^c),F^c)$ *such that* $F^c = F \uplus F^\downarrow_\Omega$ *and* $F^\downarrow_\Omega = \{f : w_\bullet \to s_\bullet \mid f : w \to s \in F_\Omega\}$*. Then let* $X^\downarrow = \{X_s\}_{s \in S^\downarrow}$ *and* $X^c = X \uplus X^\downarrow$*.*

In particular, we refer to signatures $\Sigma^c(X^c)$ and $\Omega^\downarrow(X^c)$ as the constructor sort refinement of $\Sigma(X)$ and $\Omega(X)$. It is in these signatures where we will perform unification. Also note that $(^\bullet)$ and $(_\bullet)$ extend naturally to signature morphisms $(^\bullet) : \Sigma^c \to \Sigma$ and $(_\bullet) : \Omega \to \Omega^\downarrow$. On ground terms $(^\bullet)$ and $(_\bullet)$ are the identity, but variables $x \in X^c$ are mapped either into X or $X^\downarrow$ respectively. They further extend into substitution mappings where $(x,t) \in \theta$ is mapped into $(x^\bullet,t^\bullet) \in \theta^\bullet$ and $(x_\bullet,t_\bullet) \in \theta_\bullet$ respectively.

In practice, for our unification algorithm to be efficiently used modulo a set of rewrite rules R, we want our transformed signature to be sensible and B-preregular. In general, sensibility is preserved, but preregularity (and thus B-preregularity) is not. Thus, we give a relatively mild condition which ensures B-preregularity is preserved. If $\Omega \subseteq \Sigma$, then we write $\Omega \prec \Sigma$ and say Ω is *preregular below* Σ iff Ω and Σ are preregular and $\forall t \in T_\Sigma[t \in T_\Omega \Rightarrow ls_\Omega(t) = ls_\Sigma(t)]$. Intuitively this means whenever a constructor typing is possible for a term, we need only examine its constructor typings to find its least possible typing.

Corollary 3. *Let* $R = (\Sigma,B,R)$ *be convergent with constructor decomposition* $R_\Omega = (\Omega,B_\Omega,R_\Omega)$ *and* $\Omega \prec \Sigma$*. Then* Σ^c *and* $\Omega^\downarrow$ *are sensible and* B*-preregular.*

Now we can derive the most general constructor instances via unification.

Theorem 8. *Suppose* $\Sigma(X)$ *and* $\Omega(X)$ *are sensible and* B*-preregular,* $\Omega \prec \Sigma$*, and* B *respects constructors. Then (a)* $\forall t \in T_\Sigma(X)_s\, \forall t' \in T_\Omega(X)_{s'}$ *with* $s \equiv_< s'$ *and* $x \notin vars(t)$*,* $t\alpha =_B t'$ *iff there are* $\eta \in mgu_B(t = x : c(s'))$ *and* θ *such that* $\eta^\bullet\theta|_{vars(t)} =_B \alpha$ *where* $\alpha \in [vars(t) \to T_\Omega(X)]$ *and* $\theta \in [X \to T_\Omega(X)]$ *and (b) the set of most general constructor instances of* t *modulo* B *is defined by* $mgci^\Omega_B(t) = \{t(\eta^\bullet) \mid \eta \in mgu_B(t = x : ls_{\Sigma(X)}(t)_\bullet)\}$*.*

Now that we can obtain constructor instances, we just need to show how to compute constructor variants. But this is now straightforward, since we already know we can compute every most general variant by folding variant narrowing.

Corollary 4. *Let* (Σ,B,R) *be convergent and protect constructor decomposition* $(\Omega,B_\Omega,\varnothing)$ *and* $\Omega \prec \Sigma$*. The most general constructor variants of* $t \in T_\Sigma(X)$ *are* $[\![t]\!]^\Omega_{R,B} = \{(t'\eta^\bullet,\theta\eta^\bullet) \mid (t',\theta) \in [\![t]\!]_{R,B} \wedge \eta \in mgu_B(t',x : ls_{\Sigma(X)}(t')_\bullet)\}$*.*

The reduction of constructor unifiers to constructor variants is simple. Recall any unification problem ϕ is a $\Sigma^\wedge$-term $\phi \in T_{\Sigma^\wedge}(X)_{Conj}$. Let $\{\alpha_i\}_{i \in I}$ denote the finite set of most general R,B-variant unifiers of ϕ obtained as explained in Theorem 3. Then the set of most general constructor unifiers of ϕ is the set $\{\alpha_i\eta^\bullet \mid \eta \in mgu_B((\phi\alpha_i)!_{R,B}, x : Conj_\bullet)\}$.

We finish with an example of constructor variants and unifiers, which illustrates some issues relating to subsort-overloading that need to be considered.

Consider the theory INT of integers with addition In our example, we have four sorts: *Int*, *Nat*, *NzNat*, and *NzNeg* where $NzNat < Nat < Int$ and $NzNeg < Int$. There are five constructors $\_+\_ : NzNat\ NzNat \to NzNat$, $\_+\_ : Nat\ Nat \to Nat$, $0 :\to Nat$, $1 :\to NzNat$, and $- : NzNat \to NzNeg$, and one defined operator $\_+\_ : Int\ Int \to Int$, where the addition operators all satisfy associativity, commutativity, and identity axioms with unit element 0. Let $n, m : NzNat$ and $i : Int$. Then the operators satisfy four equations: $i + -(n) + -(m) = i + -(n + m)$, $i + n + -(n) = i$, $i + n + -(n + m) = i + -(m)$, $i + n + m + -(n) = i + m$.

Note that this theory is FVP and protects its constructor subtheory. Suppose that using this signature we wish to compute the constructor variants of term $i + n$ where $i : Int$ and $n : NzNat$. We start computing the most general variants of the term $i + n$ using finite variant narrowing and obtain four variants: i, $i + n$, $i + -(n)$, and $i + n + -(m)$, where $i : Int$ and $n, m : NzNat$.

We then construct the extended signatures according to Definition 7. Figure 1 below illustrates how this is done, where for each sort s, we let $s_\bullet$ denote its lowered sort. Then, for each variant t above, we just compute and apply substitutions $mgu_B(t = x : ls_{\Sigma(X)}(t)_\bullet)\}$. Thus, we obtain the four constructor variants: i, $k + n$, $0 + n$, and $0 + -(n)$ where $i : Int$, $k : Nat$, and $n : NzNat$. Now recall $(+)$ is a defined operator over *Int* but a constructor over *Nat*; therefore, for each $(+)$ variant, in order to obtain the corresponding constructor variants, we instantiate subterm $i : Int$ so the typing of the whole term lowers into *Nat*.

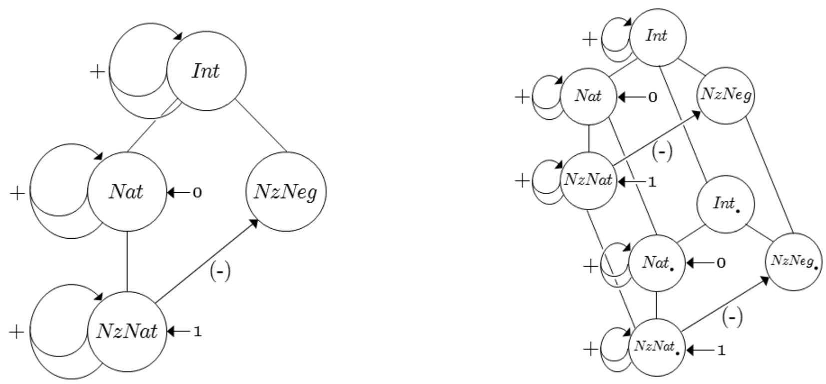


Fig. 1. INT signature Σ and its refinement Σ^c

5 Implementation and Example

Now we describe our implementation of the metalevel algorithms using Maude. Thanks to the reflective nature of rewriting logic and the fact that Maude directly implements rewriting logic, we can directly represent metalevel concepts in Maude as terms in a theory. In fact, such a library already exists in Maude's `META-LEVEL` module. By using `META-LEVEL`, we can directly write functions over meta-level constructs to implement our algorithms. Essentially, the algorithm follows the outline sketched in Section 4 and shown in the diagram in the Introduction, except that the finite sort checks for theories with axioms have not been implemented yet. The algorithm takes as input a reflected theory M and a formula $\phi = \bigwedge G \wedge \bigwedge D$ and returns a boolean indicating if the formula is satisfiable in M (for more details, see Appendix D).

Let us see how our algorithm can be applied to a concrete example theory NatList of lists of natural numbers with Presburger arithmetic. It has four sorts: *Bool*, *Nat*, *NeList*, and *List* such that $NeList < List$, seven constructors $0 :\to Nat$, $1 :\to Nat$, $\_ + \_ : Nat\ Nat \to Nat$, $\_ : \_ : Nat\ List \to NeList$, $\mathtt{nil} :\to List$, $\mathtt{true} :\to Bool$, and $\mathtt{false} :\to Bool$, and three defined operators $\_ < \_ : Nat\ Nat \to Bool$, $hd : NeList \to Nat$, and $tl : NeList \to List$ where $\_ + \_$ satisfies associativity, commutativity, and identity axioms for element 0. The theory has four equations: $m + 1 + n > n = \mathtt{true}$, $n > n + m = \mathtt{false}$, $hd(n : l) = n$ and $tl(n : l) = l$ where $n, m : Nat$ and $l : List$.

Suppose we want to show $\phi = \forall l, l' : NeList\, [hd(l) > hd(l') = true \Rightarrow l \neq l']$ is a theorem of the initial algebra of NatList. Usually, to solve equations in this combined theory, we would need a separate solver for each subtheory and use the Nelson-Oppen combination method to reason in the combined theory, but here, since the theory NatList is FVP and protects an OS-compact subtheory, we can directly reason in the combined theory. Thus, we proceed by proving the negation of ϕ $\exists l, l' : NeList\, [hd(l) > hd(l') = true \wedge l = l']$ is unsatisfiable. But we immediately find that the formula has no variant unifiers, proving unsatisfiability, and thus, the original formula is a theorem, as claimed.

6 Conclusions and Related Work

We have presented the meta-level sub-algorithms needed to obtain a full-fledged variant satisfiability algorithm, proved them correct, and derived a Maude reflective implementation. Correctness has been the main concern, but efficiency has also been taken into account. Much work remains ahead. We plan to experimentally evaluate and optimize the performance of our algorithm by means of representative satisfiability case studies. We also plan to use the algorithm itself in various infinite-state model checking and theorem proving applications.

The most closely-related work is [27, 28], for which it provides the first full-fledged algorithm and implementation. Other related topics include folding variant narrowing [18], the FVP [13], and unsorted compactness [12]. Of course, this work occurs in the larger context of decidable satisfiability algorithms and the vast literature on SMT solving, e.g., [6, 23, 3, 5, 4, 6, 24, 1, 17], and additional references in [27, 28]. Finally, the literature on Maude's reflective algorithms and tools, e.g., [9, 8] is also closely related.

Acknowledgements. Partially supported by NSF Grant CNS 13-19109.

References

1. Armando, A., Ranise, S., Rusinowitch, M.: A rewriting approach to satisfiability procedures. Inf. Comput. 183(2), 140–164 (2003)
2. Baader, F., Schulz, K.U.: Combining constraint solving. In: Constraints in Computational Logics CCL'99, International Summer School. vol. 2002, pp. 104–158. Springer LNCS (1999)
3. Barrett, C., Sebastiani, R., Seshia, S., Tinelli, C.: Satisfiability modulo theories. In: Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, vol. 185, chap. 26, pp. 825–885. IOS Press (February 2009)
4. Barrett, C., Shikanian, I., Tinelli, C.: An abstract decision procedure for satisfiability in the theory of inductive data types. Journal on Satisfiability, Boolean Modeling and Computation 3, 21–46 (2007)
5. Barrett, C., Tinelli, C.: Satisfiability modulo theories. In: Clarke, E., Henzinger, T., Veith, H. (eds.) Handbook of Model Checking. Springer (2014), (to appear)
6. Bradley, A.R., Manna, Z.: The calculus of computation - decision procedures with applications to verification. Springer (2007)
7. Cholewa, A., Meseguer, J., Escobar, S.: Variants of variants and the finite variant property. Tech. rep., CS Dept. University of Illinois at Urbana-Champaign (February 2014), available at `http://hdl.handle.net/2142/47117`
8. Clavel, M., Durán, F., Eker, S., Meseguer, J., Lincoln, P., Martí-Oliet, N., Talcott, C.: All About Maude. Springer LNCS Vol. 4350 (2007)
9. Clavel, M., Durán, F., Eker, S., Meseguer, J., Stehr, M.O.: Maude as a formal meta-tool. In: Wing, J., Woodcock, J. (eds.) FM'99 — Formal Methods. Springer LNCS, vol. 1709, pp. 1684–1703. Springer-Verlag (1999)
10. Clavel, M., Meseguer, J., Palomino, M.: Reflection in membership equational logic, many-sorted equational logic, Horn logic with equality, and rewriting logic. Theoretical Computer Science 373, 70–91 (2007)
11. Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata (2007), Release October, 12th 2007
12. Comon, H.: Complete axiomatizations of some quotient term algebras. Theor. Comput. Sci. 118(2), 167–191 (1993)
13. Comon-Lundth, H., Delaune, S.: The finite variant property: how to get rid of some algebraic properties, in Proc *RTA'05*, Springer LNCS 3467, 294–307, 2005
14. Dershowitz, N., Jouannaud, J.P.: Rewrite systems. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Vol. B, pp. 243–320. North-Holland (1990)
15. Dovier, A., Piazza, C., Rossi, G.: A uniform approach to constraint-solving for lists, multisets, compact lists, and sets. ACM Trans. Comput. Log. 9(3) (2008)
16. Dovier, A., Policriti, A., Rossi, G.: A uniform axiomatic view of lists, multisets, and sets, and the relevant unification algorithms. Fundam. Inform. 36(2-3), 201–234 (1998)
17. Dross, C., Conchon, S., Kanig, J., Paskevich, A.: Adding Decision Procedures to SMT Solvers using Axioms with Triggers. Journal of Automated Reasoning (2016), https://hal.archives-ouvertes.fr/hal-01221066, accepted for publication
18. Escobar, S., Sasse, R., Meseguer, J.: Folding variant narrowing and optimal variant termination. J. Algebraic and Logic Programming 81, 898–928 (2012)
19. Goguen, J., Meseguer, J.: Order-sorted algebra I. Theoretical Computer Science 105, 217–273 (1992)

20. Hendrix, J., Clavel, M., Meseguer, J.: A sufficient completeness reasoning tool for partial specifications. In: Proc. RTA 2005. vol. 3467, pp. 165–174. Springer LNCS (2005)
21. Hendrix, J., Meseguer, J., Ohsaki, H.: A sufficient completeness checker for linear order-sorted specifications modulo axioms. In: Automated Reasoning, Third International Joint Conference, IJCAR 2006. pp. 151–155 (2006)
22. Jouannaud, J.P., Kirchner, H.: Completion of a set of rules modulo a set of equations. SIAM Journal of Computing 15, 1155–1194 (November 1986)
23. Kroening, D., Strichman, O.: Decision Procedures - An Algorithmic Point of View. Texts in Theoretical Computer Science. An EATCS Series, Springer (2008)
24. Krstic, S., Goel, A., Grundy, J., Tinelli, C.: Combined satisfiability modulo parametric theories. In: Proc. TACAS 2007. vol. 4424, pp. 602–617. Springer LNCS (2007)
25. Meseguer, J.: Membership algebra as a logical framework for equational specification. In: Proc. WADT'97. pp. 18–61. Springer LNCS 1376 (1998)
26. Meseguer, J.: Strict coherence of conditional rewriting modulo axioms. Tech. Rep. `http://hdl.handle.net/2142/50288`, C.S. Department, University of Illinois at Urbana-Champaign (August 2014)
27. Meseguer, J.: Variant-based satisfiability in initial algebras. Tech. Rep. `http://hdl.handle.net/2142/88408`, University of Illinois at Urbana-Champaign (November 2015)
28. Meseguer, J.: Variant-based satisfiability in initial algebras. In: Artho, C., Ölveczky, P. (eds.) Proc. FTSCS 2015. pp. 1–32. Springer CCIS 596 (2016), in press.
29. Meseguer, J., Goguen, J.: Order-sorted algebra solves the constructor-selector, multiple representation and coercion problems. Information and Computation 103(1), 114–158 (1993)
30. Nelson, G., Oppen, D.C.: Simplification by cooperating decision procedures. ACM Trans. Program. Lang. Syst. 1(2), 245–257 (1979)
31. Oppen, D.C.: Complexity, convexity and combinations of theories. Theor. Comput. Sci. 12, 291–302 (1980)
32. Rocha, C., Meseguer, J.: Constructors, sufficient completeness, and deadlock freedom of rewrite theories. In: Proc. LPAR 2010. Lecture Notes in Computer Science, vol. 6397, pp. 594–609. Springer (2010)
33. Shostak, R.E.: Deciding combinations of theories. Journal of the ACM 31(1), 1–12 (Jan 1984)

A Constructor Variants and Unifiers: An Example

The notions of constructor variant and constructor unifier become more subtle when, due to order-sortedness, a same subsort-polymorphic operator f has some typings that are constructors and some other typings that are defined functions. The following examples illustrates the issues involved.

Example 2. (Integers with Addition). The FVP decomposition $\mathcal{Z}_+$ for integers with addition has sorts Nat, $NzNat$, $NzNeg$, and Int, and subsorts $NzNat < Nat$ and $Nat\ NzNeg < Int$, where $NzNat$ (resp. $NzNeg$) denotes the non-zero naturals (resp. negatives). The constructor signature Ω has constants 0 of sort Nat and 1 of sort $NzNat$, and operators $\_ + \_ : Nat\ Nat \rightarrow Nat$, $\_ + \_ : NzNat\ NzNat \rightarrow$

$NzNat$, and $- : NzNat \to NzNeg$. The only defined function symbol is: $\_+\_ : Int\ Int \to Int$, also ACU. The rewrite rules R defining $+$ and making $(\Omega, ACU, \varnothing)$ an ACU-free constructor decomposition of $\mathcal{Z}_+$ are the following (with i a variable of sort Int, and n, m variables of sort $NzNat$): $i + n + -(n) \to i$, $i + -(n) + -(m) \to i + -(n+m)$, $i + n + -(n+m) \to i + -(m)$, and $i + n + m + -(n) \to i + m$.

Consider now the term $x + y$ with x, y variables of sort Int. Then $(x + y, id)$ with id the identity substitution is a variant, but *not* a constructor variant in $\mathcal{Z}_+$, but there are variants that are *less general* than $(x + y, id)$ and are constructor variants. The *most general* constructor variants less general than $(x + y, id)$ are: (i) $(x, \{y \mapsto 0\})$, (ii) $(y, \{x \mapsto 0\})$, and (iii) $(x' + y', \{x \mapsto x' : Nat, y \mapsto y' : Nat\})$. Likewise, let ϕ be the equation $z = x + y$, with x, y, z of sort Int. Then $\{z \mapsto x + y\}$ is a trivial $\mathcal{Z}_+$-unifier of ϕ, but *not* a constructor unifier. A complete set $mgu^{\Omega}_{\mathcal{R}}(\phi)$ of most general constructor $\mathcal{Z}_+$-unifiers of ϕ is given by the unifiers: (i) $\{z \mapsto x, y \mapsto 0\}$, (ii) $\{z \mapsto y, x \mapsto 0\}$, and (iii) $\{z \mapsto x' + y', x \mapsto x' : Nat, y \mapsto y' : Nat\}$.

For other examples of constructor variants and constructor unifiers we refer the reader to Examples 3–4 in [28].

B Correctness Proofs for Constructor Variant Generation

Here we design an algorithm to solve the most general constructor instance problem and then prove our algorithm is correct. Specifically, we use a signature transformation to reduce the most general constructor instance problem into a B-unification problem. In the transformed signature, the instances $mgci_B(t')$ correspond exactly to the solutions of a *single* B-unification problem. We then use it as a subalgorithm for computing constructor variants.

We assume throughout two signatures, $\Sigma = ((S, <), F)$ and $\Omega = ((S, <), F_\Omega)$, with $\Omega \subseteq \Sigma$ and a possibly empty set of ACCU axioms B, where Ω and Σ are sensible and B-preregular. We recall the following definitions.

Definition 6. *A* constructor sort refinement *of $(S, <)$ is defined by the following: (a) a set $S^c = S \uplus S^{\downarrow}$ with $c : S \to S^{\downarrow}$ a bijection, (b) a relation $(<^c)$ the smallest strict order where: (i) $\forall s, s' \in S\, [s < s' \Leftrightarrow [s <^c s' \wedge c(s) <^c c(s')]]$ and (ii) $\forall s \in S\, [c(s) <^c s]$, and (c) functions $(^{\bullet}) : S^c \to S$ and $(_{\bullet}) : S^c \to S^{\downarrow}$ defined by $s^{\bullet} = s$ `if` $s \in S$ `else` $c^{-1}(s)$ and $s_{\bullet} = s$ `if` $s \in S^{\downarrow}$ `else` $c(s)$.*

We let $(<^c)$ also ambiguously denote its extension to strings of sorts $(S^c)^*$. Also, note that $(<) \subseteq (<^c)$ by definition and functions $(^{\bullet})$ and that $(_{\bullet})$ have unique homomorphic extensions to free monoid homomorphisms denoted by: $(^{\bullet}) : (S^c)^* \to S^*$ and $(_{\bullet}) : (S^c)^* \to (S^{\downarrow})^*$. Likewise, $(^{\bullet})$ and $(_{\bullet})$ have unique extensions to powersets, $(^{\bullet}) : \mathcal{P}(S^c) \to \mathcal{P}(S)$ and $(_{\bullet}) : \mathcal{P}(S^c) \to \mathcal{P}(S^{\downarrow})$. Lastly, $(^{\bullet})|_{(S^{\downarrow})^*}$ and $(_{\bullet})|_{S^*}$ are bijective by definition and lift into poset and powerset isomorphisms.

Definition 7. *Given $\Sigma = ((S, <), F)$ and $\Omega = ((S, <), F_\Omega)$ where $\Omega \subseteq \Sigma$ and $(S^c, <^c, (^{\bullet}), (_{\bullet}))$ is a constructor sort refinement of $(S, <)$, we define:*

1. $\Sigma^+ = ((S^c, <^c), F)$ *and* $\Omega^+ = ((S^c, <^c), F_\Omega)$
2. $\Sigma^\downarrow = ((S^c, <^c), F^\downarrow)$ *and* $\Omega^\downarrow = ((S^c, <^c), F^\downarrow_\Omega)$
3. $\Sigma^c = ((S^c, <^c), F^c)$ *and* $\Omega^c = ((S^c, <^c), F^c_\Omega)$
4. $\Omega^\downarrow_\bullet = ((S^\downarrow, <^c|_{S^\downarrow}), F^\downarrow_\Omega)$

where $F^\downarrow = (F/F_\Omega) \uplus F^\downarrow_\Omega$*,* $F^\downarrow_\Omega = \{f : w_\bullet \to s_\bullet \mid f : w \to s \in F_\Omega\}$*,* $F^c = F \uplus F^\downarrow_\Omega$*, and* $F^c_\Omega = F_\Omega \uplus F^\downarrow_\Omega$*. Similarly, we also define* $X^\downarrow = \{X_s\}_{s \in S^\downarrow}$*. Then* $X^c = X \uplus X^\downarrow$*.*

We can summarize the definition above with the figure below:

$$\begin{array}{cccccccccccc}
\Sigma(X) & \hookrightarrow & \Sigma^+(X^c) & \hookrightarrow & \Sigma^c(X^c) & \hookleftarrow & \Sigma^\downarrow(X^c) & \hookleftarrow & \Sigma^\downarrow(X^\downarrow) & & \\
\uparrow & & \uparrow & & \uparrow & & \uparrow & & \uparrow & & \\
\Omega(X) & \hookrightarrow & \Omega^+(X^c) & \hookrightarrow & \Omega^c(X^c) & \hookleftarrow & \Omega^\downarrow(X^c) & \hookleftarrow & \Omega^\downarrow(X^\downarrow) & \hookleftarrow & \Omega^\downarrow_\bullet(X^\downarrow)
\end{array}$$

where each arrow is a signature inclusion. The signature decorations are intended to be suggestive of the transformation: Σ^+ extends the subsort relation; Σ^c copies each constructor; $\Sigma^\downarrow$ shifts constructors below; and finally $\Omega^\downarrow_\bullet$ shifts constructors below and discards sorts S by applying $(_\bullet)$. In this section, we will primarily consider $\Sigma^c(X^c)$ and $\Omega^\downarrow(X^c)$ which we refer to as the *constructor sort refinements* of Σ and Ω. The other signatures will be referenced as needed.

Note that $(^\bullet)$ and $(_\bullet)$ naturally extend into signature morphisms. The sort mapping is either $(^\bullet)$ or $(_\bullet)$. If $t \in T_{\Sigma^c}(X^c)$, then the term mapping is given by: (a) if $t = x : s \in X^c$, then $(x : s)^\bullet = x : (s^\bullet)$ and $(x : s)_\bullet = x : (s_\bullet)$, (b) if $t = a : \to s \in F^c$, then $a^\bullet = a_\bullet = a$ (c) if $t = f(t_1, \cdots, t_n)$, then $t^\bullet = f(t_1^\bullet, \cdots, t_n^\bullet)$ and $t_\bullet = f(t_{1_\bullet}, \cdots, t_{n_\bullet})$. The term mappings $(^\bullet)$ and $(_\bullet)$ also naturally extend to substitutions $\theta \in [X^c \to T_{\Sigma^c}(X^c)]$. Then for each $(x, t) \in \theta$, we have $(x^\bullet, t^\bullet) \in \theta^\bullet$ and $(x_\bullet, t_\bullet) \in \theta_\bullet$. In particular, we note three facts: (i) $(_\bullet) : \Omega(X) \to \Omega^\downarrow_\bullet(X^\downarrow)$ is a signature isomorphism with inverse $(^\bullet)$ (ii) $(^\bullet) : \Sigma^c(X^c) \to \Sigma(X)$ is a signature morphism (iii) as sets of terms, $T_{\Omega^\downarrow}(X^\downarrow) = T_{\Omega^\downarrow_\bullet}(X^\downarrow)$ and $T_\Omega = T_{\Omega^\downarrow} = T_{\Omega^\downarrow_\bullet}$.

Our first goal in this section is to show that term sorting, sensibility, and preregularity are all preserved by constructor sort refinement, i.e., refinement in the sense that all existing sort information is preserved and only new sort information is added. Note that we trivially have preservation of term sorts by facts (i)-(iii) above since $\forall s \in S^c\ \forall t \in T_{\Sigma^c}(X^c)_s[t^\bullet \in T_\Sigma(X)_{s^\bullet} \wedge s \leqslant^c s^\bullet]$, $(^\bullet)$ specializes to the identity when $t \in T_\Sigma(X)$, and $\forall s \in S[t \in T_{\Omega^\downarrow_\bullet, s_\bullet} \Leftrightarrow t \in T_{\Omega,s}]$. Thus, it is enough to prove preservation of sensibility and preregularity. However, the example below shows our current assumptions are not strong enough.

Example 3. Consider sort poset $(S, <) = (\{a, b\}, \{(a, b)\})$ and signatures $\Sigma = ((S, <), \{f : a \to a, f : b \to b\})$ and $\Omega = ((S, <), \{f : b \to b\})$. The ctor sort refinement $(S^c, <^c) = (S \uplus \{a_\bullet, b_\bullet\}, (<) \uplus \{(a_\bullet, a), (b_\bullet, b), (a_\bullet, b_\bullet), (a_\bullet, b)\})$ where $\Sigma^c = ((S^c, <^c), \{f : a \to a, f : b \to b, f : b_\bullet \to b_\bullet\})$ violates preregularity for sort $a_\bullet$ where $(a_\bullet \leqslant^c a \wedge a_\bullet \leqslant^c b_\bullet)$ but $(a \nleqslant^c b_\bullet \wedge b_\bullet \nleqslant^c a)$ even though Σ and Ω are both preregular by construction.

Note that in the previous example the violation occurred when a constructor had a subsort-overloaded defined operator below. However, just restricting subsort-overloading does not fix the problem.

Example 4. Let $(S,<) = (\{a,b,c\},\{(a,b),(a,c)\})$, $\Sigma = ((S,<),\{f : b \to a, f : c \to c\})$, and $\Omega = ((S,<),\{f : c \to c\})$. Then $(S^c,<^c) = (S \uplus \{a_\bullet, b_\bullet\}, (<) \uplus \{(a_\bullet,a),(b_\bullet,b),(c_\bullet,c),(a_\bullet,b_\bullet),(a_\bullet,c_\bullet),(a_\bullet,b),(a_\bullet,c)\})$. But now note $\Sigma^c = ((S^c,<^c),\{f : b \to a, f : c \to c, f : c_\bullet \to c_\bullet\})$ violates preregularity for sort $a_\bullet$ where $(a_\bullet \leqslant^c b \wedge a_\bullet \leqslant^c c_\bullet)$ holds but $(a \nleqslant^c c_\bullet \wedge c_\bullet \nleqslant^c a)$.

Essentially, the invariant violated by both examples was that Ω was not *preregular below* Σ, in the sense that, given a symbol and arity with a constructor typing, it's minimal typing was not a constructor. In order to formally specify this invariant, we will need some auxiliary notation.

Let $\Sigma = ((S,<),F)$ be an arbitrary signature and $(P,\lhd)$ an arbitrary poset. Let $ty_\Sigma : T_\Sigma \to F$ be defined by the two equations $ty_\Sigma(c) = \{c :\to s \in F\}$ and $ty_\Sigma(f(t_1,\cdots,t_n)) = \{f : s_1 \cdots s_n \to s \in F \mid t_i \in T_{\Sigma_{s_i}}\}$. Also let ty_Σ denote the function $ty_\Sigma(f,w) = \{f : w' \to s \in F \mid w \leqslant w'\}$. Further let $min_\lhd : \mathcal{P}(P) \to P \uplus \{\varnothing\}$ be $min_\lhd(I) = \bigwedge I$ if $(\exists \bigwedge I) \wedge \bigwedge I \in I$ else $\varnothing$ where $\bigwedge I$ denotes the greatest lower bound of I in $(P,\lhd)$ if it exists.

Definition 8. *Let $\Sigma = ((S,<),F)$ have subsignature $\Omega = ((S,<),F_\Omega)$. Then Ω is* preregular below Σ *(written $\Omega \prec \Sigma$) iff Ω and Σ are preregular and for any f we have $\forall w \in S^*[ty_\Omega(f,w) \neq \varnothing \Rightarrow min_<(ty_\Sigma(f,w)) \in ty_\Omega(f,w)]$ where $(F,<)$ is the poset where $f : w \to s < g : w' \to s' \Leftrightarrow s < s'$.*

We now prove that the constructor sort refinements $\Omega^\downarrow(X^c)$ and $\Sigma^c(X^c)$ preserve sensibility and preregularity iff Ω and Σ are sensible and $\Omega \prec \Sigma$. Note that, by definition, for any signature Σ, we have $ls_\Sigma(t) = min_<(ty_\Sigma(t))$ for the poset $(F,<)$ and to prove Σ is preregular it is enough to show $\forall t \in T_\Sigma[ls_\Sigma(t) \neq \varnothing]$. To complete the proof, we will need three lemmas. To preserve the logical flow of the argument, we will state them here as assumptions to be used in the main argument and then discharge them later.

Lemma 1. $\forall t \in T_\Sigma\,[t \in T_\Omega \Rightarrow ls_\Omega(t) = ls_\Sigma(t)]$
Lemma 2. $\forall t \in T_{\Omega^\downarrow}(X^c)/X^c\,[ty_{\Omega^\downarrow(X^c)}(t) = ty_{\Omega^\downarrow(X^\downarrow)}(t) = ty_{\Omega^\downarrow_\bullet(X^\downarrow)}(t)]$
Lemma 3. $\forall t \in T_{\Sigma^c}(X^c)/X^c\,[ty_{\Sigma^+(X^c)}(t) = ty_{\Sigma(X)}(t^\bullet)]$

Theorem 6. *If $\Omega \prec \Sigma$ and Ω and Σ are sensible, then the constructor sort refinements $\Sigma^c(X^c) = ((S^c,<^c),F^c \uplus X^c)$ and $\Omega^\downarrow(X^c) = ((S^c,<^c),F^\downarrow_\Omega \uplus X^c)$ are both sensible and preregular.*

Proof.
We first prove $\Sigma^c(X^c)$ is sensible (which implies $\Omega^\downarrow(X^c)$ is sensible). Since Σ sensible implies $\Sigma(X)$ sensible for any signature Σ, it is sufficient to prove Σ^c is sensible. Recall the signature morphism $(^\bullet) : \Sigma^c \to \Sigma$. Then suppose that $f : w \to s, f : w' \to s' \in F^c$ where $w \equiv_{<^c} w'$. Then $w^\bullet \equiv_{<^c} w'^\bullet$ and, by sensibility of Σ, $s^\bullet \equiv_{<^c} s'^\bullet$, which implies $s \equiv_{<^c} s'$ by Corollary 5.

We first prove that $\Omega^{\downarrow}(X^c)$ is preregular. By abuse of language, let X also denote the signature $((S,<),X)$. Then note that $\forall t \in T_{\Omega^{\downarrow}}(X^c)[ty_{\Omega^{\downarrow}(X^c)}(t) = ty_{\Omega^{\downarrow}(X^{\downarrow})}(t) \uplus ty_X(t)]$ and $\Omega^{\downarrow}(X^{\downarrow}) \cap X = \varnothing$. Thus, by Lemma 2, we obtain that $\forall t \in T_{\Omega^{\downarrow}}(X^c)/X^c\ [ty_{\Omega^{\downarrow}(X^{\downarrow})}(t) = ty_{\Omega^{\downarrow}_{\bullet}(X^{\downarrow})}(t)]$. Thanks to the facts above, $ls_{\Omega^{\downarrow}(X^c)} = ls_{\Omega^{\downarrow}_{\bullet}(X^{\downarrow})} \uplus ls_X$. By the signature isomorphism $\Omega^{\downarrow}_{\bullet}(X^{\downarrow}) \cong \Omega(X)$, this is equivalent to $ls_{\Omega^{\downarrow}(X^c)} = (^{\bullet}); ls_{\Omega(X)}; (_{\bullet}) \uplus ls_X$, where semicolon denotes function composition in diagrammatic order. Since X is preregular by definition and $\Omega(X)$ by assumption, $ls_{\Omega^{\downarrow}(X^c)}$ satisfies $\forall t \in T_{\Sigma}[ls_{\Sigma}(t) \neq \varnothing]$, as required.

We now prove $\Sigma^c(X^c)$ is preregular. First let $t \in X^c$. Then $t \in X \uplus X^{\downarrow}$. If $t = x{:}s \in X$ then $ls_{\Sigma^c(X^c)}(x{:}s) = ls_{\Sigma(X)}(x{:}s^{\bullet}) = s$. Similarly, if $t = x : s \in X^{\downarrow}$, $ls_{\Sigma^c(X^c)}(x{:}s) = ls_{\Omega(X)}(x{:}s^{\bullet})_{\bullet} = s$.

Now let $t \in T_{\Sigma^c}(X^c)/X^c$. Note $ty_{\Sigma^c(X^c)}(t) = ty_{\Omega^{\downarrow}(X^c)}(t) \uplus ty_{\Sigma^+(X^c)}(t)$, i.e., the type of non-variable t is from $F^{\downarrow}_{\Omega}$ or F and $ls_{\Sigma^c(X^c)}(t) = min_<(ty_{\Omega^{\downarrow}(X^c)}(t) \uplus ty_{\Sigma^+(X^c)}(t))$. Suppose $t \in T_{\Omega^{\downarrow}}(X^{\downarrow})/X^{\downarrow}$. By Lemma 2 and $\Omega^{\downarrow}_{\bullet}(X^{\downarrow}) \cong \Omega(X)$, we obtain $ty_{\Omega^{\downarrow}(X^c)}(t) = ty_{\Omega^{\downarrow}_{\bullet}(X^{\downarrow})}(t) = ty_{\Omega(X)}(t^{\bullet})_{\bullet}$. By Lemmas 1 and 3, we have $min_<(ty_{\Omega(X)}(t^{\bullet})) = min_<(ty_{\Sigma(X)}(t^{\bullet})) = min_<(ty_{\Sigma^+(X^c)}(t))$. Then note $ls_{\Sigma^c(X^c)}(t) = min_<(ty_{\Omega(X)}(t^{\bullet})_{\bullet} \uplus ty_{\Omega(X))}(t^{\bullet})) = ls_{\Omega(X)}(t^{\bullet})_{\bullet}$. Finally, assume that $t \in T_{\Sigma^c}(X^c)/T_{\Omega^{\downarrow}}(X^c)$. Then we obtain $ty_{\Omega^{\downarrow}(X^c)}(t) = \varnothing$ and $ls_{\Sigma^c(X^c)}(t) = min_<(ty_{\Sigma^+(X^c)}(t)) = min_<(ty_{\Sigma(X)}(t^{\bullet})) = ls_{\Sigma(X)}(t^{\bullet})$ by Lemma 3. Thus, we have $\forall t \in T_{\Sigma^c}(X^c)[ls_{\Sigma^c(X^c)}(t) \neq \varnothing]$, as required. □

Corollary 2. *The functions $ls_{\Omega^{\downarrow}(X^c)}$ and $ls_{\Sigma^c(X^c)}$ are defined by:*

(a) $\forall t \in T_{\Omega^c}(X^c)\ ls_{\Omega^{\downarrow}(X^c)}(t) = ls_{\Omega(X)}(t^{\bullet})_{\bullet}$ *if* $t \in T_{\Omega^{\downarrow}}(X^{\downarrow})$ *else* $ls_{\Sigma(X)}(t^{\bullet})$
(b) $\forall t \in T_{\Sigma^c}(X^c)\ ls_{\Sigma^c(X^c)}(t) = ls_{\Omega(X)}(t^{\bullet})_{\bullet}$ *if* $t \in T_{\Omega^{\downarrow}}(X^{\downarrow})$ *else* $ls_{\Sigma(X)}(t^{\bullet})$

We now extend the result above to show that B-preregularity is preserved under a weak assumption that is often satisfied in practice. We first state the required condition and then give the proof.

Definition 9. *Let B be a set of axioms, $t = t' \in B$ with $vars(t = t') = Y$ and $\alpha \in [Y \to X]$. We say B* respects constructors *iff $t\alpha \in T_{\Omega}(X) \Leftrightarrow t'\alpha \in T_{\Omega}(X)$.*

Theorem 7. *Assume $\Sigma(X)$ and $\Omega(X)$ are sensible and B-preregular and that $\Omega(X) \prec \Sigma(X)$ and B respects constructors. Then their respective constructor sort refinements $\Sigma^c(X^c)$ and $\Omega^{\downarrow}(X^c)$ are also B-preregular.*

Proof. We apply Theorem 6 to immediately show that $\Sigma^c(X^c)$ and $\Omega^{\downarrow}(X^c)$ are sensible and preregular. We first prove $\Sigma^c(X^c)$ is B-preregular. Thus, let $t = t' \in B$ with $Y = \text{vars}(t) = \text{vars}(t')$ and $\alpha \in [Y \to X^c]$. Note that the value of functions $ls_{\Omega^{\downarrow}(X^c)}$ and $ls_{\Sigma^c(X^c)}$ is completely determined by the input term t and functions $ls_{\Omega(X)}$ and $ls_{\Sigma(X)}$. In particular, if $ls_{\Sigma(X)}(t\alpha) = ls_{\Sigma(X)}(t'\alpha)$, then $ls_{\Sigma^c(X^c)}(t\alpha) = ls_{\Sigma^c(X^c)}(t'\alpha)$ iff $t\alpha \in T_{\Omega^{\downarrow}}(X^{\downarrow}) \Leftrightarrow t'\alpha \in T_{\Omega^{\downarrow}}(X^c)$ by Corollary 2 (the same holds true for $ls_{\Omega^{\downarrow}(X^c)}$). Since B respects constructors, it is enough to show $\forall t \in T_{\Omega}(X)[t\alpha \in T_{\Omega^{\downarrow}}(X) \Leftrightarrow \alpha \in [Y \to T_{\Omega^{\downarrow}}(X^{\downarrow})]]$ where $Y = \text{vars}(t) \neq \varnothing$. The base case where $t = x : s$ is trivial, so assume $t = f(t_1, \cdots, t_n)$. Then

$t\alpha = f(t_1\alpha, \cdots, t_n\alpha)$ and $t_i\alpha \in T_{\Omega^\downarrow}(X) \Leftrightarrow \alpha \in [Y \to T_{\Omega^\downarrow}(X^\downarrow)]$ for $1 \leqslant i \leqslant n$ by induction hypothesis. But then $f : s_1 \cdots s_n \to s \in F_\Omega$ with $t_1 \in T_\Omega(X)_{s_i}$ iff $f : s_{1_\bullet} \cdots s_{n_\bullet} \to s_\bullet \in F_\Omega$ and $f(t_1\alpha, \cdots, t_n\alpha) = t\alpha \in T_{\Omega^\downarrow}(X^\downarrow)$, as required. $\square$

The following corollary lifts the result above to decompositions.

Corollary 3. *Let $R = (\Sigma, B, R)$ be convergent with constructor decomposition $R_\Omega = (\Omega, B_\Omega, R_\Omega)$ and $\Omega < \Sigma$. Then Σ^c and $\Omega^\downarrow$ are sensible and B-preregular.*

Proof. Note that protecting a constructor decomposition implies B respects constructors (see pg. 6). Then apply Theorem 7.

We have now shown that our construction, under mild conditions, preserves sensibility and B-preregularity. Thus, B-unification will be well-defined in our new signature. We now move to prove the main theorem of this section which shows how most general constructor instances of a term modulo B may be obtained by a single unification problem in $\Sigma^c(X^c)$. We first collect a number of essential facts which relate $T_\Omega(X)$ to $T_{\Omega^\downarrow}(X^\downarrow)$ and will be used in the proof.

Lemma 4. *Suppose that $\alpha, \beta \in [X \to T_\Omega(X)]$, $\alpha', \beta' \in [X^\downarrow \to T_{\Omega^\downarrow}(X^\downarrow)]$, and $\theta, \gamma \in [X^c \to T_{\Sigma^c}(X^c)]$. Let $id^\downarrow \in [X^c \to X^\downarrow]$ where $id^\downarrow(x : s) = x : s_\bullet$. Then:*

(a) $\forall \alpha \in [X \to T_\Omega(X)]\ [(\alpha_\bullet)^\bullet = \alpha]$, $\forall \alpha \in [X^\downarrow \to T_{\Omega^\downarrow}(X^\downarrow)]\ [(\alpha^\bullet)_\bullet = \alpha]$
(b) $\forall t, t' \in T_\Omega(X)[t =_B t' \Leftrightarrow t_\bullet =_B t'_\bullet] \ \wedge\ \forall t, t' \in T_{\Omega^\downarrow}(X^\downarrow)[t =_B t' \Leftrightarrow t^\bullet =_B t'^\bullet]$
(c) $[\alpha =_B \beta \Leftrightarrow \alpha_\bullet =_B \beta_\bullet] \ \wedge\ [\alpha' =_B \beta' \Leftrightarrow \alpha'^\bullet =_B \beta'^\bullet]$
(d) $\forall t \in T_{\Sigma^c}(X^c)[t_\bullet = t(id^\downarrow)] \wedge (id^\downarrow)^\bullet = id$
(e) $\forall t \in T_{\Sigma^c}(X^c)[(t\theta)_\bullet = t_\bullet(\theta_\bullet) \wedge (t\theta)^\bullet = t^\bullet(\theta^\bullet) \wedge (\theta\gamma)_\bullet = \theta_\bullet(\gamma_\bullet) \wedge (\theta\gamma)^\bullet = \theta^\bullet(\gamma^\bullet)]$

Proof. Both (a) and (b) follow immediately since $T_{\Omega^\downarrow_\bullet}(X^\downarrow) = T_{\Omega^\downarrow}(X^\downarrow)$ and by isomorphism $(^\bullet) : \Omega^\downarrow_\bullet(X^\downarrow) \to \Omega(X)$. Then (c) is an immediate application of (b). Finally, (d) and (e) are easy structural induction proofs. $\square$

We now give a precise construction of $mgci^\Omega_B$ using B-unification in $\Sigma^c(X^c)$.

Theorem 8. *Suppose $\Sigma(X)$ and $\Omega(X)$ are sensible and B-preregular, $\Omega < \Sigma$, and B respects constructors. Then (a) $\forall t \in T_\Sigma(X)_s\, \forall t' \in T_\Omega(X)_{s'}$ with $s \equiv_< s'$ and $x \notin vars(t)$, $t\alpha =_B t'$ iff there are $\eta \in mgu_B(t = x : c(s'))$ and θ such that $\eta^\bullet\theta|_{vars(t)} =_B \alpha$ where $\alpha \in [vars(t) \to T_\Omega(X)]$ and $\theta \in [X \to T_\Omega(X)]$ and (b) the set of most general constructor instances of t modulo B is defined by $mgci^\Omega_B(t) = \{t(\eta^\bullet) \mid \eta \in mgu_B(t = x : ls_{\Sigma(X)}(t)_\bullet)\}$.*

Proof. We first prove (a). Let $\beta = \alpha_\bullet \uplus \{(x : s'_\bullet, t'_\bullet)\}$. Then observe:

$$\begin{aligned} t\alpha =_B t' &\Leftrightarrow (t\alpha)_\bullet =_B t'_\bullet \\ &\Leftrightarrow t_\bullet(\alpha_\bullet) =_B t'_\bullet \\ &\Leftrightarrow t_\bullet\beta =_B x : s'_\bullet\beta \\ &\Leftrightarrow \exists \eta' \in mgu_B(t_\bullet = x : s'_\bullet)\, \exists \theta' \in [X^\downarrow \to T_{\Omega^\downarrow}(X^\downarrow)]\, [\eta'\theta' =_B \beta] \end{aligned}$$

which follow by Lemma 4 and the fact B respects constructors so $t\alpha \in T_\Omega(X)$. Let id be the identity substitution and note $x\!:\!(s'_\bullet)_\bullet = x\!:\!s'_\bullet$. Then we obtain:

$$
\begin{aligned}
&\eta' \in mgu_B(t_\bullet = x\!:\!s'_\bullet)\\
&\Leftrightarrow \eta' \in mgu_B(t_\bullet = x\!:\!(s'_\bullet)_\bullet)\\
&\Leftrightarrow \eta' \in mgu_B(t(id^\downarrow) = x\!:\!s'_\bullet(id^\downarrow))\\
&\Leftrightarrow id^\downarrow \eta' \in mgu_B(t = x\!:\!s'_\bullet)
\end{aligned}
\qquad
\begin{aligned}
&\eta'\theta' =_B \beta\\
&\Leftrightarrow (\eta'\theta)^\bullet =_B \beta^\bullet\\
&\Leftrightarrow \eta'^\bullet(\theta'^\bullet) =_B \beta^\bullet\\
&\Leftrightarrow \eta'^\bullet(\theta'^\bullet)|_{vars(t)} =_B \alpha \;\wedge\\
&\qquad \eta'^\bullet(\theta'^\bullet)(x) =_B t'
\end{aligned}
$$

by Lemma 4. Now let $\eta = id^\downarrow\eta'$ and $\theta = \theta'^\bullet$. Then we can derive equalities $\eta^\bullet\theta = (id^\downarrow\eta')^\bullet\theta = (id^\downarrow)^\bullet(\eta'^\bullet)\theta = id(\eta'^\bullet)\theta = \eta'^\bullet(\theta'^\bullet)$ as required. Finally (b) is an immediate application of (a). ◻

In case the constructor decomposition has no rules (i.e., free constructors modulo B_Ω), Theorem 8 yields an easy method to compute constructor variants.

Corollary 4. *Let (Σ, B, R) be convergent and protect constructor decomposition $(\Omega, B_\Omega, \varnothing)$ and $\Omega \prec \Sigma$. The most general constructor variants of $t \in T_\Sigma(X)$ are $[\![t]\!]^\Omega_{R,B} = \{(t'(\eta^\bullet), \theta\eta^\bullet) \mid (t', \theta) \in [\![t]\!]_{R,B} \wedge \eta \in mgu_B(t', x : ls_{\Sigma(X)}(t')_\bullet)\}$.*

Proof. Apply Corollary 3. It is sufficient to prove: (a) each $(t'\eta\,\theta\eta) \in [\![t]\!]^\Omega_{R,B}$ is a constructor variant (b) for any constructor variant (t'', ψ), we obtain that $\exists(t'\eta, \theta\eta) \in [\![t]\!]^\Omega_{R,B}[(t'\eta, \theta\eta) \sqsupseteq_{R,B} (t'', \phi)]$. To see (a), suppose $(t', \theta) \in [\![t]\!]_{R,B}$. By definition of most general unifier and Theorem 8, $mgu_B(t', x : c(ls_{\Sigma(X)}(t')))$ is the set of most general substitutions η modulo B such that $t'\eta^\bullet \in T_{\Omega(X)}$. Since (Σ, B, R) protects $(\Omega, B_\Omega, \varnothing)$ and Ω is a signature of free constructors modulo B, we obtain $t'\eta^\bullet!_{R,B} = t'\eta^\bullet$, and $(t'\eta^\bullet, \theta\eta^\bullet)$ is a constructor variant. To see (b), note that, by definition, $[\![t]\!]$ covers every variant, and $mgci^\Omega_B(t')$ covers every constructor instance, as required. ◻

Finally, we can apply Corollary 4 directly to find the set of most general constructor B-unifiers of ϕ, by letting variant unifiers of ϕ be represented by terms $\phi' \in T_{\Sigma^\wedge(X)}$ and computing the most general constructor variants of ϕ'.

B.1 Auxiliary Lemmas

In these proofs, we always assume that $(S^c, <^c)$ is a constructor sort refinement of $(S, <)$. In Lemma 1, we require two simple lemmas which are left as an exercise to the reader. Let Σ be an arbitrary signature. Then (1) if Σ is preregular and $f(t_1, \cdots, t_n) \in T_\Sigma$ then $ty_\Sigma(f(t_1, \cdots, t_n)) = ty_\Sigma(f, ls_\Sigma(t_1) \cdots ls_\Sigma(t_n))$ with $n \geqslant 0$ and (2) $t \in T_\Sigma \Leftrightarrow ty_\Sigma(t) \neq \varnothing$.

Lemma 1. *If $\Omega \prec \Sigma$ then $\forall t \in T_\Sigma[t \in T_\Omega \Rightarrow ls_\Omega(t) = ls_\Sigma(t)]$.*

Proof. Assume $\Omega \prec \Sigma$ and $t \in T_\Omega$. Suppose that $t = c \in T_\Omega$ is a constant. Then $ty_\Omega(c, nil) \neq \varnothing$ and $min_<(ty_\Sigma(c, nil)) \in ty_\Omega(c, nil)$. Since we have $ty_\Omega(c, nil) \subseteq ty_\Sigma(c, nil)$ then $min_<(ty_\Omega(c, nil)) = min_<(ty_\Sigma(c, nil))$ and $ls_\Omega(t) = ls_\Sigma(t)$. Now suppose $t = f(t_1, \cdots, t_n)$. Then $ty_\Omega(f(t_1, \cdots, t_n)) \neq \emptyset$ and $ty_\Omega(f, w) \neq \emptyset$ where $w = ls_\Omega(t_1) \cdots ls_\Omega(t_n)$. But $t_1 \cdots t_n \in T_\Omega$, so by induction hypothesis, $w = ls_\Sigma(t_1) \cdots ls_\Sigma(t_n)$. Since $min_<(ty_\Sigma(f, w)) \in ty_\Omega(f, w)$ and $ty_\Omega(f, w) \subseteq ty_\Sigma(f, w)$, then we have $min_<(ty_\Omega(f, w)) = min_<(ty_\Sigma(f, w))$ and $ls_\Omega(t) = ls_\Sigma(t)$, as required. □

Lemma 2. $\forall t \in T_{\Omega^\downarrow}(X^c)/X^c\, [ty_{\Omega^\downarrow(X^c)}(t) = ty_{\Omega^\downarrow(X^\downarrow)}(t) = ty_{\Omega^\downarrow_\bullet(X^\downarrow)}(t)]$

Proof. The base case where $t = c \in T_{\Omega^\downarrow}(X^c)/X^c$, a constant, is trivial, so suppose $t = f(t_1, \cdots, t_n)$. There are two cases: either for each $1 \leqslant i \leqslant n$, we have $\mathrm{vars}(t_i) \subseteq X^\downarrow$ or not. If not, $ty_{\Omega^\downarrow(X^c)}(t) = ty_{\Omega^\downarrow(X^\downarrow)}(t) = ty_{\Omega^\downarrow_\bullet(X^\downarrow)}(t) = \emptyset$ since these three signatures share the same non-variable operators $F^\downarrow_\Omega$ whose arity is contained in $(S^\downarrow)^*$. Otherwise, by induction hypothesis, for $1 \leqslant i \leqslant n$, we have $ty_{\Omega^\downarrow(X^c)}(t_i) = ty_{\Omega^\downarrow(X^\downarrow)}(t_i) = ty_{\Omega^\downarrow_\bullet(X^\downarrow)}(t_i)$, and since operators $F^\downarrow_\Omega$ are shared, we have $ty_{\Omega^\downarrow(X^c)}(t) = ty_{\Omega^\downarrow(X^\downarrow)}(t) = ty_{\Omega^\downarrow_\bullet(X^\downarrow)}(t)$. □

Lemma 3. $\forall t \in T_{\Sigma^c}(X^c)/X^c\, [ty_{\Sigma^+(X^c)}(t) = ty_{\Sigma(X)}(t^\bullet)]$

Proof. The case where $t = c \in T_{\Sigma^c}(X^c)/T_{\Omega^\downarrow}(X^c)$, a constant, is trivial, so suppose $t = f(t_1, \cdots, t_n)$. By definition, $\exists f : s_1 \cdots s_n \to s \in F$ with $s_i \in S$, $t_i : s'_i$, and $s'_i \leqslant^c s_i$ for $1 \leqslant i \leqslant n$. But $(^\bullet) : (S, <^c) \to (S, <)$—also $(^\bullet) : \Sigma^+(X^c) \to \Sigma(X) \subseteq \Sigma^+(X^c)$—is a poset/signature morphism, so $s'^\bullet_i \leqslant s^\bullet_i = s_i$, $t^\bullet_i \in T_{\Sigma(X)}$, and $ty_{\Sigma^+(X^c)}(t) = ty_{\Sigma^+(X^c)}(t^\bullet)$. Also note $ty_{\Sigma^+(X^c)}|_{T_{\Sigma(X)}} = ty_{\Sigma(X)}$, since $f : s_1 \cdots s_n \to s \in F \cup X^c$ with $s_1 \cdots s_n \in S^*$ iff $f : s_1 \cdots s_n \to s \in F \cup X$. But $t^\bullet \in T_{\Sigma(X)}$, thus $ty_{\Sigma^+(X^c)}(t) = ty_{\Sigma^+(X^c)}(t^\bullet) = ty_{\Sigma(X)}(t^\bullet)$, as required. □

Lemma 4. $\forall s, s' \in S^\downarrow [s \equiv_{\leqslant^c} s' \Leftrightarrow s^\bullet \equiv_\leqslant s'^\bullet]$

Proof. Note that $(\equiv_{\leqslant^c})$ is the smallest equivalence relation generated by $(\leqslant^c)$, i.e. $(\equiv_{\leqslant^c}) = (\leqslant^c \cup \geqslant^c)^*$. Likewise, $(\equiv_\leqslant) = (\leqslant \cup \geqslant)^*$. To see $(\Leftarrow)$, note since $(\leqslant) \subseteq (\leqslant^c)$, we have $s^\bullet \equiv_\leqslant s'^\bullet \Rightarrow s^\bullet \equiv_{\leqslant^c} s'^\bullet$. Since $s \leqslant^c s^\bullet$ and $s' \leqslant^c s'^\bullet$, by transitivity of $\equiv_{\leqslant^c}$ and since $(\leqslant^c \cup \geqslant^c) \subseteq (\equiv_{\leqslant^c})$, we have $s \equiv_{\leqslant^c} s'$. To see $(\Rightarrow)$, note $s \equiv_{\leqslant^c} s' \Leftrightarrow \exists n \in \mathbb{N}[s(\leqslant^c \cup \geqslant^c)^n s']$. For $n = 0$, $(\leqslant^c \cup \geqslant^c)$ is the equality relation and the result follows trivially. Now suppose we have $s(\leqslant^c \cup \geqslant^c)^n s''(\leqslant^c \cup \geqslant^c)s'$. By the induction hypothesis, $s^\bullet(\leqslant \cup \geqslant)^* s''^\bullet$. Suppose now that $s'' \leqslant^c s'$ (the case $s'' \geqslant^c s'$ is analogous). Since $(^\bullet)$ is monotonic, $s''^\bullet \leqslant s'^\bullet$. Thus, $s^\bullet(\leqslant \cup \geqslant)^* s'^\bullet$ giving $s^\bullet \equiv_\leqslant s'^\bullet$ as required. □

Corollary 5.

(a) $\forall n \in \mathbb{N}\ \forall w, w' \in (S^\downarrow)^n [w \leqslant^c w' \Rightarrow w^\bullet \leqslant w'^\bullet]$
(b) $\forall n \in \mathbb{N}\ \forall w, w' \in (S)^n\ [w \leqslant\ w' \Rightarrow w_\bullet \leqslant w'_\bullet]$
(c) $\forall n \in \mathbb{N}\ \forall w, w' \in (S^\downarrow)^n [w \equiv_{\leqslant^c} w' \Leftrightarrow w^\bullet \equiv_\leqslant w'^\bullet]$

Proof. (a) and (b) follow by monotonicity of $(^\bullet)$ and $(_\bullet)$ and since the homomorphic extension to strings preserves monotonicity. (c) follows by Lemma 4 and the fact that the homomorphic extension of $(^\bullet)$ preserves $(\leqslant)$ and thus $(\equiv_\leqslant)$. □

C Empty and Finite Sort Constructions

In this section, we present three algorithms and prove their correctness. Given an order-sorted signature, possibly with axioms, we define rewrite theories and sentences in rewriting logic which represent solutions to the: (i) sort emptiness, (ii) sort finiteness, and (iii) term generation problems by rewrite theories implementable in the Maude rewrite engine. In the following definitions we always assume that we are reasoning over an order-sorted, kind-complete[4] signature $\Sigma = ((S, <), F)$ where B is a set of associative/commutative/unit axioms over Σ. Before proceeding, we define some notation. For $f : s_1 \cdots s_n \to s$, let rags$(f) = \{s_1, \cdots, s_n\}$ and ran$(f) = s$. Let $S_{\supset\varnothing} = \{s \in S \mid T_{\Sigma/B,s} \neq \varnothing\}$, $F_{\supset\varnothing} = \{f \in F \mid \text{args}(f) \subseteq S_{\supset\varnothing}\}$, and $\Sigma_{\supset\varnothing} = ((S_{\supset\varnothing}, <|_{S_{\supset\varnothing}}), F_{\supset\varnothing})$. Given $F' \subseteq F$, let $\Sigma|_{F'} = ((S, <), F')$. Given binary relations $R_1 \subseteq S_1 \times S_1$, and $R_2 \subseteq S_2 \times S_2$, we write $R_1 \cong R_2$ iff R_1 and R_2 are bisimilar. Given $S \subseteq S_1 \cap S_2$, $R_1 \xleftrightarrow{S} R_2$ holds iff for all $s \in S$, (R_1, s) terminates iff (R_2, s) terminates where, by definition, (R, s) *terminates* iff there is no infinite R-path starting from s.

C.1 Sort Emptiness Check for General Signatures.

Here we develop an algorithm that checks if a sort $s \in S$ satisfies $T_{\Sigma,s} = \varnothing$ by performing unsorted rewriting over $\mathcal{P}(S)$. The initial state is the sort we wish to check for non-emptiness. We trace the operator declarations in reverse to see which sorts are needed to build operators inhabiting the argument sort.

Definition 10. *Let $\mathcal{R}_M(\Sigma) = (\Sigma_M, ACI, R_M)$ where:*

(1) $\Sigma_M = S \uplus \{*\} \uplus \{\_,\_\}$ *(an unsorted signature)*
(2) $ACI = \{\boldsymbol{x,y = y,x}\} \cup \{\boldsymbol{(x,y),z = x,(y,z)}\} \cup \{\boldsymbol{x,x = x}\}$
(3) R_M *is the smallest rewrite relation such that:*
 (a) $(s, s') \in (<) \Rightarrow s' \to s \in R_M$
 (b) $c : \to s \in F \Rightarrow s \to * \in R_M$
 (c) $f : s_1 \cdots s_k \to s \in F \wedge k \geqslant 1 \Rightarrow s \to s_1, \cdots, s_k \in R_M$

In the text below, let $(\to) \subseteq T_{\Sigma_M} \times T_{\Sigma_M}$ abbreviate $(=_{ACI}; \to_{R_M}; =_{ACI})$. We further let $(\to^0) = (=_{ACI})$, $(\to^{n+1}) = (\to); (\to^n)$, $(\to^*) = \bigcup_{n\geqslant 0}(\to^n)$, and also $(\to^+) = \bigcup_{n>0}(\to^n)$.

Lemma 5. *Let $a_1, \ldots, a_k$, $k \geqslant 1$ be a ground Σ_M-term, so that $a_i \in S \uplus \{*\}$, i.e., $a_1, \ldots, a_k$ is a* multiset. *If $a_1, \ldots, a_k \to^n *$, then for each nonempty submultiset $B \subseteq a_1, \ldots, a_k$ there is an $m \leqslant n$ such that $B \to^m *$.*

[4] Any signature can be easily extended to a *kind-complete* one by: (i) adding a top sort, named $[s]$, above each connected component $[s]$; and (ii) adding for each operator $f : s_1 \ldots s_n \to s$ in the original signature a new typing $f : [s_1] \ldots [s_n] \to [s]$. For the original sorts $s \in S$, the terms in the original signature and in its kind-completion are the *same*. Maude always perform this kind completion for any user-given signature.

Proof. By induction on n.
Base Case. If $n = 0$ we must have $a_i = *$, $1 \leqslant i \leqslant k$, and the result follows trivially.
Induction Step. Suppose the result true for n and let $a_1, \ldots, a_k \to^{n+1} *$. Since rewriting takes place modulo ACI we may assume without loss of generality that $i \neq j \Rightarrow a_i \neq a_j$. Then we must have some $a_i \in S$, a rule $a_i \to D$ in R_M, and rewrites

$$a_1, \ldots, a_k \to a_1, \ldots, a_{i-1}, D, a_{i+1}, \ldots, a_n \to^n *.$$

Note that $a_1, \ldots, a_{i-1}, D, a_{i+1}, \ldots, a_n$ may have repeated elements. We now reason by cases on $B \subseteq a_1, \ldots, a_k$. If $a_i \notin B$, then $B \subseteq a_1, \ldots, a_{i-1}, D, a_{i+1}, \ldots, a_n$ and the result follows trivially by the induction hypothesis. If $B = a_i, B'$ (where by convention B' could be empty), then $B \to D, B'$ and we have an inclusion $D, B' \subseteq a_1, \ldots, a_{i-1}, D, a_{i+1}, \ldots, a_n$ so the result follows again trivially by the induction hypothesis. □

Lemma 6. $\forall s \in S \ [T_{\Sigma,s} \neq \varnothing \Leftrightarrow s \to^+ *]$

Proof. ($\Rightarrow$). Let $s \in S$ with $T_{\Sigma,s} \neq \varnothing$. Pick any $t \in T_{\Sigma,s}$ and proceed by structural induction on t.
Base case. $[t = c]$: Suppose $c : \to s'' \in F$ is a constant. Since $c \in T_{\Sigma,s}$, we know $s'' \leqslant s$. If $s'' = s$, then directly apply rule $s \to *$ generated by declaration $c : \to s'' \in F$. If $s'' < s$, we will have an additional rule $s \to s''$, which we can apply followed by $s \to *$. In either case, obtain $s \to^+ *$.
Induction Step. $[t = f(t_1, \cdots, t_n)]$: Since $t = f(t_1, \cdots, t_n) \in T_{\Sigma,s}$, we have $\exists f : s_1 \cdots s_k \to s'' \in F$ with $s'' \leqslant s$ where $t_i \in T_{\Sigma,s_i}$ for $i \in k$. If $s'' = s$, then directly apply rule $s \to s_1 \, , \cdots , s_k$ generated by declaration $f : s_1 \cdots s_k \to s'' \in F$. Since $t_i \in T_{\Sigma,s_i}$ for $i \in k$, we know that $T_{\Sigma,s_i} \neq \varnothing$. Thus, by inductive hypothesis, obtain that $s_i \to^+ *$ for $i \in k$. By transitivity, we have $s'' \to^+ * \, , \cdots , *$. By idempotency, obtain $s'' \to^+ *$. If $s'' < s$, we will have an additional rule $s \to s''$ we can apply followed by $s'' \to^+ *$. In either case, obtain $s \to^+ *$.
($\Leftarrow$). Suppose towards a contradiction the set $S' = \{s \in S \mid T_{\Sigma,s} = \varnothing \wedge s \to^+ *\}$ is non-empty. For each $s \in S'$ these is an $m(s) \in \mathbb{N}$ with $s \to^{m(s)} *$ and $m(s)$ smallest possible with that property. Pick $s_0 \in S'$ with $m(s_0)$ smallest among such $m(s)$. We now have two cases to consider: $m(s_0) = 1$ or $m(s_0) > 1$. Suppose $m(s_0) = 1$. Then $s_0 \to *$. But this can only happen if there is a $c : \to s_0 \in F$. But then $c \in T_{\Sigma,s_0}$ and $T_{\Sigma,s_0} \neq \varnothing$, a contradiction. Thus, assume $m(s_0) > 1$. Again, there are two possibilities: $s_0 \to s' \to^{m(s_0)-1} *$ or $s_0 \to s_1 \, , \cdots , s_k \to^{m(s_0)-1} *$. If $s_0 \to s' \to^{m(s_0)-1} *$, since $m(s_0)$ is smallest possible in S', we must have $s' \notin S'$ and therefore $T_{\Sigma,s'} \neq \varnothing$. But this rewrite can only occur if $s' < s_0$. Thus, $T_{\Sigma,s'} \subseteq T_{\Sigma,s_0}$, so that $T_{\Sigma,s_0} \neq \varnothing$, a contradiction. If $s_0 \to s_1 \, , \cdots , s_k \to^{m(s_0)-1} *$, by Lemma 5 for each $1 \leqslant i \leqslant k$ we have $s_i \to^{m_i} *$ for some $m_i \leqslant m(s_0) - 1$. Therefore, $T_{\Sigma,s_i} \neq \varnothing$, $1 \leqslant i \leqslant k$. But the rewrite $s_0 \to s_1 \, , \cdots , s_k$ can only occur if there is an $f : s_1 \cdots s_k \to s_0 \in F$. But given any $t_i \in T_{\Sigma,s_i}$, $1 \leqslant i \leqslant k$, we can construct $f(t_1, \cdots, t_k) \in T_{\Sigma,s_0}$. Thus, $T_{\Sigma,s_0} \neq \varnothing$, a contradiction. □

There are two remaining questions: (i) is checking the sentence $s \to^{+} *$ decidable? and (ii) can this approach compute emptiness of equivalence classes of terms $T_{\Sigma/E}$ defined by a theory (Σ, E)? Fortunately, in this case, there is no extra work to be done. To answer (i), note that whenever $|S| + |F| < \aleph_0$, then $|\mathcal{P}(S)| + |R_M| < \aleph_0$ by construction. Thus, we have a finite number of states and rules, rendering the search problem decidable. To answer (ii), note that, $T_{\Sigma/E,s}$ is just an equivalence relation over $T_{\Sigma,s}$. Thus, $T_{\Sigma/E,s} = \varnothing$ iff $T_{\Sigma,s} = \varnothing$. As a result of this section, note that the set of sorts $S_{\supset\varnothing} \subseteq S$ is computable; thus, we obtain that $F_{\supset\varnothing}$ and $\Sigma_{\supset\varnothing}$ are computable as well.

C.2 Term Generation for General Signatures.

In this section, we present an algorithm which, given an order-sorted signature Σ and a sort s, will generate all terms in $T_{\Sigma,s}$. We begin with a few opening remarks. Note that: (i) an order-sorted signature Σ can be modeled as a tree automaton so that $t \in T_{\Sigma,s}$ iff t is accepted by the corresponding automaton when the accepting state is s; and (ii) any tree automaton and its computations can be modeled as an unsorted ground rewrite theory. Clearly, an order-sorted ground rewrite theory will also work; here we prefer an order-sorted theory because it gives a simpler definition that preserves the original signature. Throughout this section, we let S_Σ denote the signature of constants s associated to sorts $s \in S$, where each sort s is declared a constant whose sort is the top sort $[s]$: $S_\Sigma = ((S,<), \{s : \to [s] \mid s \in S\})$.

Definition 11. *Let $\mathcal{R}_P(\Sigma) = (\Sigma_{\supset\varnothing} \uplus S_\Sigma, \varnothing, R_P)$ where R_P is the smallest rewrite relation $R_P = R_{P,S} \uplus R_{P,NC} \uplus R_{P,C}$ such that:*

(a) $(s, s') \in (<) \Rightarrow s \to s' \in R_{P,S}$
(b) $f : s_1 \cdots s_k \to s \in F_{\supset\varnothing} \wedge k \geqslant 1 \Rightarrow f(s_1, \cdots, s_k) \to s \in R_{P,NC}$
(c) $c : \to s \in F_{\supset\varnothing} \Rightarrow c \to s \in R_{P,C}$

Note that, even though $\Sigma_{\supset\varnothing} \subseteq \Sigma$, we do not lose completeness for parsing, since any sort in $s \in S/S_{\supset\varnothing}$ necessarily satisfies $T_{\Sigma,s} = \varnothing$. Furthermore, it is straightforward to show that $\Sigma \uplus S_\Sigma$ is sensible and preregular iff Σ is sensible and preregular and $\forall s \in S_{\supset\varnothing}\ [t \in T_{\Sigma,s} \Leftrightarrow t \to^{+}_{R_P} s]$. We now turn to term generation.

Definition 12. *Let $\mathcal{R}_G(\Sigma) = (\Sigma_{\supset\varnothing} \uplus S_\Sigma, \varnothing, R_G)$ with $R_G = R_P^{-1}$. Since $R_P = R_{P,S} \uplus R_{P,NC} \uplus R_{P,C}$ we will use the notation: $R_{G,S} = R_{P,S}^{-1}$, $R_{G,NC} = R_{P,NC}^{-1}$, and $R_{G,C} = R_{P,C}^{-1}$.*

Again, by only considering $\Sigma_{\supset\varnothing} \subseteq \Sigma$, we do not lose completeness for term generation. We immediately obtain the following corollary.

Corollary 6. $\forall s \in S_{\supset\varnothing}\ [t \in T_{\Sigma \uplus S_\Sigma} \Leftrightarrow s \to^{!}_{R_G} t]$

C.3 Finite Sort Detection for Finite Signatures.

Here we develop an algorithm which, given $s \in S$, checks if $|T_{\Sigma,s}| < \aleph_0$. Note that using $\mathcal{R}_G$ we already trivially obtain a semi-decidable algorithm for sort finiteness: compute $S_{\supset\varnothing}$ via $\mathcal{R}_M$; if $s \notin S_{\supset\varnothing}$, then return yes; otherwise compute $\{t \in T_{\Sigma,s} \mid s \to^!_{R_G} t\}$; if the process terminates, then return yes. Of course, an efficient, decidable algorithm would be preferable. Nevertheless, $\mathcal{R}_G$ is not too far from our desired decidable solution.

Our strategy is as follows: (i) give sufficient conditions so that termination of $\mathcal{R}_G$ corresponds to sort finiteness in Σ, (ii) define a rewrite system $\mathcal{R}_F$ and give sufficient conditions to prove termination of $\mathcal{R}_F$, (iii) show $\mathcal{R}_F$ terminates if and only if $\mathcal{R}_G$ terminates, (iv) and finally, present a decidable algorithm using LTL model checking to characterize when $\mathcal{R}_F$ terminates.

Lemma 7. *If $|S| + |F| < \aleph_0$ then $(\mathcal{R}_G, s)$ is non-terminating iff $|T_{\Sigma,s}| = \aleph_0$*

Proof. By construction of R_G, $|R_G| = |(<)| + |F| < |S|^2 + |F| < \aleph_0$. Viewing possible rewrite paths starting from s as forming a tree, observe that the tree branches finitely, since each term has finite positions and possible rewrites. Suppose $(\mathcal{R}_G, s)$ is terminating. Then, by K onig's Lemma, the tree of rewrites must be finite and therefore there is a finite number of final states, so that $|T_{\Sigma,s}| < \aleph_0$. Otherwise, if $(\mathcal{R}_G, s)$ is non-terminating, we have an infinite path $s \to_{R_G} t_1 \to_{R_G} t_2 \to_{R_G} \cdots t_n \to_{R_G} \cdots$. Since $|R_G| < \aleph_0$, $\exists R \subseteq R_G$ that repeats infinitely often. Since $R_G = R_{G,S} \uplus R_{G,C} \uplus R_{G,NC}$ and $R_{G,S} \uplus R_{G,C}$ terminates (because acyclicicty/finiteness of $<$ and only S-terms can be rewritten), we must have $R \cap R_{G,NC} \neq \varnothing$. But note that, if $|t|$ is the of t as viewed as a tree, then if $t \to_{R_{G,S} \uplus R_{G,C}} t'$, we must have $|t| = |t'|$, whereas if $t \to_{R_{G,NC}} t'$, we must have $|t| < |t'|$, so that $\{|t_i|\}_{i\in\mathbb{N}}$ is a sequence such that $|t_i| \to \infty$. Also note that by the definition of R_G, all sorts s' occurring as a subterm of t_i belong to $S_{\supset\varnothing} = \{s_1, \cdots, s_m\}$, so that we can choose terms $u_1 \in T_{\Sigma,s_1}, \cdots, u_m \in T_{\Sigma,s_m}$. We can then regard $S_{\supset\varnothing}$ as a set of variables and view $\sigma = \{s_1 \mapsto u_1, \cdots, s_m \mapsto u_n\}$ as a substitution. But, by definition of R_G, this gives us an infinite sequence $\{t_i\sigma\}_{i\in\mathbb{N}}$ of terms where for each $i \in \mathbb{N}$, $t_i\sigma \in T_{\Sigma,s}$ and $|t_i\sigma| \geqslant |t_i|$. Therefore, $|t_i\sigma| \to \infty$, and since $T_{\Sigma,s}$ contains terms of unbounded size, we have $|T_{\Sigma,s}| = \aleph_0$. □

Definition 13. *Let $\mathcal{R}_F(\Sigma) = (S_{\supset\varnothing}, \varnothing, R_F)$ where $R_F = R_{F,S} \cup R_{F,NC}$ is the smallest rewrite relation such that:*

(a) $s < s' \Rightarrow s' \to s \in R_{F,S}$
(b) $f : s_1 \cdots s_n \to s' \in F_{\supset\varnothing} \wedge \{s\} \subseteq \{s_1, \cdots, s_n\} \Rightarrow s' \to s \in R_{F,NC}$

Note that we only consider $S_{\supset\varnothing}$ and $F_{\supset\varnothing}$, because, implicitly, any sort $s \in S/S_{\supset\varnothing}$ trivially satisfies $|T_{\Sigma,s}| < \aleph_0$ and any operator $f \in F/F_{\supset\varnothing}$ cannot contribute meaningfully to building a term $t \in T_{\Sigma,s}$. Before we complete the main proof, we prove a lemma and add an additional definition.

Lemma 8. *Given $|S_{\supset\varnothing}| < \aleph_0$ and $s \in S_{\supset\varnothing}$, then the following are equivalent:*

1. *$(\mathcal{R}_F, s)$ is non-terminating*
2. $\exists s' \in S_{\supset\varnothing}[s \to^*_{R_F} s' \to^+_{R_F} s']$
3. *there is an infinite R_F-rewrite path $s \to_{R_F} s_1 \to_{R_F} s_2 \cdots \to_{R_F} s_n \to_{R_F} \cdots$ and $s' \in S_{\supset\varnothing}$ occurring infinitely often in the sequence*

Proof. Obviously, (3) implies (2), since if s' occurs infinitely often, we must have $s \to^*_{R_F} s' \to^+_{R_F} s'$. Also, (2) implies (1) since $s \to^*_{R_F} s' \to^+_{R_F} s' \to^+_{R_F} s' \to^+_{R_F} \cdots$ is a non-terminating sequence. Finally, (1) implies (3), since $|S_{\supset\varnothing}| \leqslant \aleph_0$, which forces some $s' \in S_{\supset\varnothing}$ to occur infinitely often in any infinite sequence. $\square$

Definition 14. *Given $\Sigma = ((S,<), NC \uplus C)$ with non-constants and constants NC and C respectively, let $\mathcal{R}^\star_G(\Sigma) = (\Sigma_{\supset\varnothing}|_{NC} \uplus S_\Sigma, \varnothing, R_{G,\star})$ such that $R_{G,\star} = R_{G,S} \uplus R_{G,NC}$.*

Observe that $\mathcal{R}^\star_G$ is identical to $\mathcal{R}_G$ except that $\mathcal{R}^\star_G$ contains neither constants nor rewrite rules over constants. Now we are ready to prove the main theorem.

Theorem 9. *$\mathcal{R}_F \cong \mathcal{R}^\star_G$ and $\mathcal{R}^\star_G \overset{S_{\supset\varnothing}}{\longleftrightarrow} \mathcal{R}_G$*

Proof. We first prove $\mathcal{R}_F \cong \mathcal{R}^\star_G$. Define a relation $H \subseteq (S_{\supset\varnothing} \times T_{\Sigma|_{NC} \uplus \hat{S}})$ where $(s,t) \in H$ iff $s \trianglelefteq t$. To prove $\mathcal{R}_F \cong \mathcal{R}^\star_G$, we show that given two arrows, we can find another two arrows to make the diagrams below commute.

$$\begin{tikzcd} s \arrow[r, "R_F"'] \arrow[d, "H"'] & s' \arrow[d, dashed, "H"] \\ t \arrow[r, dashed, "R_{G,\star}"'] & t' \end{tikzcd} \qquad \begin{tikzcd} s \arrow[r, dashed, "R_F"'] \arrow[d, "H"'] & s' \arrow[d, dashed, "H"] \\ t \arrow[r, "R_{G,\star}"'] & t' \end{tikzcd}$$

Suppose $s \trianglelefteq t$. If $(s,s') \in R_F$ then $(s,s') \in R_{F,S}$ or $(s,s') \in R_{F,NC}$. Assume $(s,s') \in R_{F,S}$. Then $s' < s$ in $\Sigma_{\supset\varnothing}$. But then, by definition, $(s,s') \in R_{G,S}$. Thus, $t[s] \to_{R_{G,\star}} t[s']$ and $s' \trianglelefteq t[s']$, as required. Alternatively, assume $(s,s') \in R_{F,NC}$. Then $\exists f : s_1 \cdots s_n \to s' \in F_{\supset\varnothing}$ with $\{s\} \subseteq \text{args}(f)$. But then, by definition, $(s', f(s_1,\cdots,s_n)) \in R_{G,NC}$. Thus, $t[s] \to_{R^{NC}_G} t[f(s_1,\cdots,s_n)]$ and $s' \trianglelefteq t[f(s_1,\cdots,s_n)]$. Since we used only definitional equivalences, the other direction follows symmetrically.

To prove $\mathcal{R}^\star_G \overset{S_{\supset\varnothing}}{\longleftrightarrow} \mathcal{R}_G$, given $s \in S_{\supset\varnothing}$, we must show $(\mathcal{R}^\star_G, s)$ terminates iff $(\mathcal{R}_G, s)$ terminates. To begin, note $R_G = R_{G,\star} \uplus R_{G,C}$. Thus, if $R_{G,\star}$ is non-terminating, R_G must also be non-terminating. To see the other direction, note $R_{G,C}$ always terminates since each rule has the form $s \to c \in C$ and constants cannot be rewritten. We proceed by proving the contrapositive. Thus, assume $R_{G,\star}$ terminates. By Lemma 9, $s \to^n_{R_G} t$ iff $s \to^i_{R_{G,\star}} t' \to^j_{R_{G,C}} t$ with $n = i + j$. Since $R_{G,\star}$ and $R_{G,C}$ are terminating and finitely branching, there are maximum bounds on the size of i and j, say, i_{max} and j_{max} respectively. But then any rewrite path $s \to^n_{R_G} t$ necessarily has $n \leqslant i_{max} + j_{max}$; thus (R_G, s) is terminating. $\square$

Lemma 9. $\forall n \in \mathbb{N}\,[[s \to^n_{R_G} t] \Leftrightarrow [\exists i,j \in \mathbb{N}\,[s \to^i_{R_{G,\star}} t' \to^j_{R_{G,C}} t \wedge n = i + j]]]$

Proof. To begin, recall $R_G = R_{G,\star} \uplus R_{G,C}$ and note the following equivalence for $s \in S_{\supset\varnothing}, n \in \mathbb{N}$, and $t \in T_\Sigma$:

$$s \rightarrow^n_{R_G} t$$
$$\Leftrightarrow$$
$$\exists l_1, l_2, m_1, m_2 \in \mathbb{N}\ \exists t', t'', t''', t^{iv} \in T_\Sigma$$
$$[[[s \rightarrow^{l_1}_{R_{G,\star}} t' \rightarrow^{l_2}_{R_{G,C}} t] \vee [s \rightarrow^{m_1}_{R_{G,\star}} t'' \rightarrow_{R_{G,C}} t''' \rightarrow_{R_{G,\star}} t^{iv} \rightarrow^{m_2}_{R_G} t]] \wedge$$
$$l_1 + l_2 = m_1 + m_2 + 2 = n]$$

That is, either all the applications of rules in $R_{G,C}$ occur at the end, or there is at least one such application *before* a rule in $R_{G,\star}$. Since the first case already fits the desired form, we need only consider the second case. Note all rules in R_G have the form $S \ni s \rightarrow t \in T_{\Sigma \uplus S_\Sigma}$. $R_{G,C}$ rules in particular have the form $s \rightarrow c$ for $c \in F$. Thus, if a $R_{G,C}$ rule is applied to $t[s]_p$ at position p, a $R_{G,\star}$ rule cannot later also be applied at p. Now suppose $s \rightarrow^{m_1}_{R_{G,\star}} t'' \rightarrow_{R_{G,C}} t''' \rightarrow_{R_{G,\star}} t^{iv} \rightarrow^{m_2}_{R_G} t$. Then, $t'' = t''[s', s'']_{p,q}$ with p, q disjoint positions and:

$$\begin{array}{ccccc} s & \xrightarrow[R_{G,\star}]{*} & t''[s', s''] & \xrightarrow[R_{G,C}]{} & t''[c, s''] \\ & & {\scriptstyle R_{G,\star}}\downarrow & & \downarrow{\scriptstyle R_{G,\star}} \\ & & t''[s', u] & \xrightarrow{R_{G,C}} & t''[c, u] \end{array}$$

for any $c \in C$ and $u \in T_{\Sigma \uplus S_\Sigma}$, the diagram above commutes. We complete the proof by induction on m_2, the number of rewrites occurring after the first $R_{G,C}$ rule followed by a $R_{G,\star}$ rule. Suppose $m_2 = 0$. Then we can commute the $R_{G,\star}$ and $R_{G,C}$ arrows as above, to obtain a rewrite chain of the form $s \rightarrow^{m_1+1}_{R_{G,\star}} v \rightarrow_{R_{G,C}} t$, for some $v \in T_{\Sigma \uplus S_\Sigma}$, as required. Now suppose $m_2 > 0$. Again, we commute the two arrows to obtain $s \rightarrow^{m_1+1}_{R_{G,\star}} v_1 \rightarrow_{R_{G,C}} v_2 \rightarrow^{m_2}_{R_G} t$. We apply our induction hypothesis to obtain $s \rightarrow^{m_1+1}_{R_{G,\star}} v_1 \rightarrow^{k_1}_{R_{G,\star}} v_3 \rightarrow^{k_2}_{R_{G,C}} t$ with $k_1 + k_2 = m_2$ which is equivalent to $s \rightarrow^{m_1+k_1+1}_{R_{G,\star}} v_3 \rightarrow^{k_2}_{R_{G,C}} t$ and $m_1 + k_1 + k_2 + 1 = m_1 + m_2 + 1 = n$, as required. □

Thus, according to Lemmas 7 and 8 and Theorem 9, $(\mathcal{R}_F, s)$ will generate a rewrite path containing a cycle iff $|T_{\Sigma,s}| = \aleph_0$. To complete the proof, for any $s \in S$, we just to characterize when $\exists s' \in S_{\supset\varnothing}[s \rightarrow^*_{R_F} s' \rightarrow^+_{R_F} s']$ holds. Thus, define the set of *cycle sorts* by $cy(S_{\supset\varnothing}) = \{s \in S_{\supset\varnothing} \mid s \rightarrow^+_{R_F} s\}$. This set can be computed by search, since the sort set and rules are both finite. Then, we immediately obtain the following theorem.

Theorem 10. $\forall s \in S_{\supset\varnothing}\ |T_{\Sigma,s}| = \aleph_0$ *iff* $\bigvee_{s' \in cy(S_{\supset\varnothing})} R_F \vdash s \rightarrow^* s'$

Proof. By Lemmas 7 and 8 and Theorem 9, obtain $|T_{\Sigma,s}| = \aleph_0$ iff the formula $\exists s' \in S_{\supset\varnothing}[s \rightarrow^*_{R_F} s' \rightarrow^+_{R_F} s']$ holds. But by definition, any s' which satisfies the formula satisfies $s' \in cy(S_{\supset\varnothing})$, so reduce to $\exists s' \in cy(S_{\supset\varnothing})[s \rightarrow^*_{R_F} s']$. Since S is finite by assumption, $cy(S_{\supset\varnothing})$ is finite. So, reduce to $\bigvee_{s' \in cy(S_{\supset\varnothing})} s \rightarrow^*_{R_F} s'$, which holds iff $\bigvee_{s' \in cy(S_{\supset\varnothing})} R_F \vdash s \rightarrow^* s'$ holds, as required. □

A final consideration is how to check, for a theory (Σ, B), whether equivalence classes of terms $T_{\Sigma/B,s}$ are finite, given that $T_{\Sigma,s}$ is finite. Since $T_{\Sigma/B,s}$ is a set of B-equivalence classes $[t]$, each containing at least one $t' \in [t]$ with $t' \in T_{\Sigma,s}$, if $|T_{\Sigma,s}| < \aleph_0$, then $T_{\Sigma/B,s} < \aleph_0$. Nevertheless, in general, it may be the case that $|T_{\Sigma/B,s}| < \aleph_0$ but $|T_{\Sigma,s}| = \aleph_0$.

Example 5. $\Sigma = ((\{a, b\}, \{(a, b)\}), 0 :\to a, 1 :\to b, \_+\_ : a\, a \to a, \_+\_ : b\, b \to b)$. Let B contain a unit axiom for 0 over $(+)$. Then $|T_{\Sigma,a}| = |T_{\Sigma,b}| = \aleph_0$ but $|T_{\Sigma/B,a}| = 1$ and $|T_{\Sigma/B,b}| = \aleph_0$.

However, under some conditions on B, finiteness of $T_{\Sigma/B,s}$ can still be checked.

Lemma 10. *Suppose B is a set of associativity and/or commutativity axioms, $|\Sigma| < \aleph_0$, and that Σ is B-preregular. Then $|T_{\Sigma/B,s}| < \aleph_0$ iff $|T_{\Sigma,s}| < \aleph_0$.*

Proof. Since Σ is B-preregular, all axioms in B are sort preserving. Then obtain $[u]_B \in T_{\Sigma/AC,s}$ iff $[u]_B \subseteq T_{\Sigma,s}$, proving $(\Leftarrow)$. To show $(\Rightarrow)$, note that for any combination of associativity and/or commutativity axioms, $[u]_B$ is a *finite* set. Since $T_{\Sigma/B,s}$ is finite, then $T_{\Sigma,s}$ is a finite union of finite sets and thus finite. □

Let U be a set of unit axioms for unit elements $e_1 :\to s_1, \cdots e_n :\to s_n$ in Σ. Then define $\Sigma - U = \Sigma - \{e_1 :\to s_1, \cdots e_n :\to s_n\}$.

Lemma 11. *Let B_0 be a set of associative and/or commutative axioms and U a set of unit axioms in Σ, $B = B_0 \uplus U$, $|\Sigma| < \aleph_0$, and $\Sigma = ((S, <), F)$ be B-preregular according to Footnote 1. If $|T_{\Sigma-U,s}| = \aleph_0$, then $|T_{\Sigma/B,s}| = \aleph_0$.*

Proof. We can orient a unit axiom $f(x, e) = x$ as a rewrite rule $f(x, e) \to x$, so that the set U becomes a set of rewrite rules $R(U)$. In this way the theory $(\Sigma, B_0 \uplus U)$ can be decomposed as a convergent rewrite theory $(\Sigma, B_0, R(U))$. Observe $T_{\Sigma-U/B_0} \subseteq C_{\mathcal{R}_U}$ and $C_{\mathcal{R}_U} \cong T_{\Sigma/B}$. By Lemma 10, $|T_{\Sigma-U,s}| = \aleph_0$ iff $|T_{\Sigma-U/B_0,s}| = \aleph_0$. Thus, $\aleph_0 = |T_{\Sigma-U,s}| = |T_{\Sigma-U/B_0,s}| \leqslant |C_{\mathcal{R}_B,s}| = |T_{\Sigma/B,s}|$. Since $|T_{\Sigma/B,s}| \leqslant \aleph_0$, obtain $|T_{\Sigma/B,s}| = \aleph_0$, as required. □

The following lemma gives sufficient conditions such that $|T_{\Sigma,s}| = \aleph_0$ but $|T_{\Sigma/B,s}| < \aleph_0$ when B is a combination of associativity and/or commutativity and/or unit axioms.

Lemma 12. *Let B_0 be a set of associative and/or commutative axioms and U a set of unit axioms in Σ, $B = B_0 \uplus U$, $|\Sigma| < \aleph_0$, and $\Sigma = ((S, <), F)$ be B-preregular according to Footnote 1. Let $f : s_1 s_2 \to s'$ with $ls(e) \leqslant s_1, s_2 \leqslant s' \leqslant s$ and let e be a unit element satisfying either a left-unit, right-unit, or left- and right-unit axiom(s) for f with $s \in S$. If $\nexists g : w \to s'' \in F/\{f, e\}[s'' \leqslant s]$ then $|T_{\Sigma,s}| = \aleph_0$ and $T_{\Sigma/B,s} = \{\{e\}\}$.*

Proof. By an easy structural induction, $\forall u \in T_{\Sigma,s}[u!_{R(U),B_0} = e]$. □

C.4 Decidable Sort Classifications

Here, we present a summary of the results of the previous sections by illustrating how our methods can be used to compute a partitioning of S that respects sort classifications.

Corollary 7. *Let B be a set of associative and/or commutative axioms, $|\Sigma| < \aleph_0$, and Σ be B-preregular. Then S has the following computable partitioning:*

$$S = S_{\supset\varnothing} \uplus S_{\varnothing} = S_{\infty} \uplus S_F \uplus S_{\varnothing}$$

where $S_{\infty} = \{s \in S_{\supset\varnothing} \mid |T_{\Sigma/B,s}| = \aleph_0\}$ and $S_F = S_{\supset\varnothing}/S_{\infty}$.

Proof. First apply Lemma 10 to reduce to the case with no axioms. By Lemma 6, $s \not\to_{R_M,ACI} *$ iff $s \in S_{\varnothing}$, and $S_{\supset\varnothing} = S/S_{\varnothing}$. Thus, obtain $\Sigma_{\supset\varnothing}$. By Theorem 10, if $s \in S_{\supset\varnothing}$ then $s \in S_F$ iff $\neg(\bigvee_{s' \in cy(S_{\supset\varnothing})} R_F \vdash s \to^* s')$. Otherwise, by definition, $s \in S_{\infty}$. Since each step—performing search via $(=_{ACI};\to_{R_M};=_{ACI})$, filtering $F_{\supset\varnothing}$, computing $cy(S_{\supset\varnothing})$, and search over R_F—is decidable, the entire sort classification algorithm is decidable, as required. □

In the more general ACU case, this partitioning can no longer be computed by the methods we have presented. However, in many cases we can still compute such a partition, for example if all sorts s for which $|T_{\Sigma,s}| = \aleph_0$ fall into one of the cases laid out in Lemmas 11 and 12. Otherwise, the partitioning algorithm will fail to classify some sorts, leaving some proof obligations for the user.

D Implementation Details and Example

In this appendix, we present further details about our Maude implementation of the variant satisfiability algorithm and show some examples. Since Maude directly implements rewriting logic, the code is just a rewrite theory where rewrites correspond to query evaluation. Since rewriting logic is reflective [10], we can directly represent metalevel entities in Maude using the `META-LEVEL` module. Essentially, the algorithm follows the outline sketched in Section 4; it takes a reflected theory M and formula $\phi = \bigwedge G \wedge \bigwedge D$ as input. Thanks to mixfix parsing, we can use a more natural notation to write ϕ:

$$\overline{u}_1 \texttt{ ==? } \overline{v}_1 \texttt{ /\textbackslash } \cdots \texttt{ /\textbackslash } \overline{u}_k \texttt{ ==? } \overline{v}_k \texttt{ /\textbackslash } \overline{u}'_1 \texttt{ =!? } \overline{v}'_1 \texttt{ /\textbackslash } \cdots \texttt{ /\textbackslash } \overline{u}'_l \texttt{ =!? } \overline{v}'_l$$

where each $\overline{u}_i, \overline{v}_i$ and $\overline{u}'_j, \overline{v}'_j$ for $1 \leqslant i \leqslant k$ and $1 \leqslant j \leqslant l$ is a meta-term. We have developed functions corresponding to the different subalgorithms presented in Section 4 and shown in the diagram in the Introduction (except that currently, finite sort checks in the presence of axioms are not implemented yet). Let $\overline{t}$ denote a metaterm. Then some of the primary functions include:

(a) `ctor-variants`$(M, \overline{t})$ which computes constructor variants of $\overline{t}$
(b) `ctor-unifiers`$(M, \bigwedge G)$ which computes $mgu^{\Omega}_{M}(\bigwedge G)$
(c) `mgci`$(M, \overline{t})$ which computes the most general constructor instances of $\overline{t}$

(d) `sort-finite?`(M, s) which checks whether $|T_{M,s}| < \aleph_0$
(e) `ctor-refine`(M) which computes the constructor sort refinement of M
(f) `consistent?`$(M, \bigwedge D)$ which checks if $\bigwedge D$ is consistent in M

Before we proceed, note that the complete codebase, including an appropriate Maude binary, and examples, can be downloaded from:

http://maude.cs.illinois.edu/tools/var-sat/

We show an example of how the tool may be run below (this example is included with the tool distribution, so the interested reader may check it). Note that Maude, in general, is not a whitespace sensitive language, so we can generally arrange syntactic items as we wish. A signature is specified by the `sort`, `subsort`, and `op` declarations which define the sorts, subsort relation, and operators respectively. The constructor subsignature is the signature which has the same sorts and subsorts, but only the operators marked with the `[ctor]` tag are included. Finally, the `var` and `eq` declarations declare variables and equations. The example theory `ZERO?` above is from [27] (Example 1).

Example Commandline Output

```
fmod ZERO? is
  sorts Nat Bool .
  op 0     :     -> Nat  [ctor] .
  op top   :     -> Bool [ctor] .
  op bot   :     -> Bool [ctor] .
  op s     : Nat -> Nat  [ctor] .
  op zero? : Nat -> Bool .
  var N:Nat .
  eq zero?(s(N)) = bot .
  eq zero?(0)    = top .
endfm

red var-sat(upModule(ZERO?,true),
            'zero?['N:Nat] ==? 'X:Bool /\
            'X:Bool =!? 'top.Bool      /\
            'X:Bool =!? 'bot.Bool) .
reduce in TEST-ZERO? :
    var-sat(upModule(ZERO?,true),
            'zero?['N:Nat] ==? 'X:Bool /\
            'X:Bool =!? 'bot.Bool      /\
            'X:Bool =!? 'top.Bool) .
rewrites: 428 in 4ms cpu (0ms real) (107000 rewrites/second)
result Bool: false
```

Note that the term syntax in the formula which is an input to `var-sat` and in the module varies; this is due to the fact that our algorithm takes meta-terms

Example Variants and Constructor Variants

```
red variants(upModule('ZERO?,true), 'zero?['N:Nat]) .
reduce in TEST-ZERO? :
    variants(upModule('ZERO?,true), 'zero?['N:Nat]) .
rewrites: 17 in 0ms cpu (0ms real) (~ rewrites/second)
result VariantTripleSet:
    {'bot.Bool,'N:Nat <- 's['#4:Nat],4} |
    {'top.Bool,'N:Nat <- '0.Nat,2} |
    {'zero?['#1:Nat],'N:Nat <- '#1:Nat,1}

red ctor-variants(upModule('ZERO?,true), 'zero?['N:Nat]) .
reduce in TEST-ZERO? :
    ctor-variants(upModule('ZERO?,true), 'zero?['N:Nat]) .
rewrites: 365 in 0ms cpu (0ms real) (~ rewrites/second)
result VariantTripleSet:
    {'bot.Bool,'N:Nat <- 's['#4:Nat],4} |
    {'top.Bool,'N:Nat <- '0.Nat,2}
```

as input. The function `upModule` gives us a meta-level representation of the module `ZERO?`. In this simple example, `var-sat` returns `false`, since a totally defined predicate cannot evaluate to both true and false. The example above was originally used in [27] to show that variants and variant unifiers are in general insufficient to reduce the satisfiability problem from one theory into its subtheory. To see why, we can compute the variants and constructor variants of `zero?(N)` as show above.

There are three most general variants: `top`, `bot`, and `zero?(N)` where `N` is a `Nat`, but just `top` and `bot` are the most general constructor variants. Why is the extra variant a problem? Because in the constructor subtheory, there are no equations. Then, we obtain the variant unifier—but not constructor unifier!—of unification problem `zero?(N:Nat) ==? X:Bool` where `N:Nat` $\mapsto$ `N:Nat` and `X:Bool` $\mapsto$ `zero?(N:Nat)`. When we apply this variant unifier to the disequations: `'X:Bool =!? bot` and `'X:Bool =!? top`, we obtain the two disequations: `zero?(N:Nat) =!? bot` and `zero?(N:Nat) =!? top` which are both *consistent* with the empty theory; this is clearly *not* what we want. However, by restricting ourselves to just the constructor unifiers, the disequations are trivially inconsistent, as we expected.