Variants, Unification, Narrowing, and Symbolic Reachability in Maude 2.6[*]

Francisco Durán[1], Steven Eker[2], Santiago Escobar[3], José Meseguer[4], and Carolyn Talcott[2]

1 **Universidad de Málaga, Spain.** `duran@lcc.uma.es`
2 **SRI International, CA, USA.** `eker@csl.sri.com`, `clt@cs.stanford.edu`
3 **Universidad Politécnica de Valencia, Spain.** `sescobar@dsic.upv.es`
4 **University of Illinois at Urbana-Champaign, IL, USA.** `meseguer@illinois.edu`

Abstract

This paper introduces some novel features of Maude 2.6 focusing on the *variants* of a term. Given an equational theory $(\Sigma, Ax \cup E)$, the E,Ax-variants of a term t are understood as the set of all pairs consisting of a substitution σ and the E,Ax-canonical form of $t\sigma$. The equational theory $Ax \cup E$ has the *finite variant property* iff, for each term, there is a finite and complete set of most general variants. We have added support in Maude 2.6 for: (i) order-sorted unification modulo associativity, commutativity, and identity, (ii) variant generation, (iii) order-sorted unification modulo finite variant theories, and (iv) narrowing-based symbolic reachability modulo finite variant theories. We also explain how these features have a number of interesting applications in areas such as unification theory, cryptographic protocol verification, business processes, and proofs of termination, confluence, and coherence.

1998 ACM Subject Classification D.2.4 [Software Engineering]: Software/Program Verification, D.3.2 [Programming Languages]: Language Classifications, F.3.1 [Logics and Meanings of Programs]: Specifying and Verifying and Reasoning about Programs

Keywords and phrases Rewriting logic, narrowing, unification, variants

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

In [4] the Maude 2.4 features for order-sorted unification modulo axioms Ax, including commutativity (C) and associativity commutativity (AC), and for narrowing-based reachability analysis of rewrite theories modulo such axioms Ax were described. In this paper we present the new features of variant generation, variant-based unification, and symbolic reachability analysis modulo a theory with the finite variant property supported by Maude 2.6. The key distinction, now supported for the first time in Maude, is one between *dedicated* unification algorithms for a limited set of axioms Ax, and *generic* unification algorithms such as variant-based unification which can be applied to a much wider range of user-definable theories. As explained in Section 6, this opens up many applications, including: (i) unification-related applications; (ii) cryptographic protocol analysis; (iii) symbolic reachability analysis of concurrent systems; and (iv) formal reasoning capabilities such as termination proofs, and proofs of local confluence and coherence that can now be performed *modulo* a much wider set of equational theories thanks to the use of variants.

[*] S. Escobar has been supported by MICINN grant TIN 2010-21062-C02-02. J. Meseguer has been supported by NSF grant CCF 09-05584. C. Talcott and S. Eker have been supported by NSF grant 09-05607. F. Durán was supported by MICINN grant TIN2008-03107 and JA grant P07-TIC-03184.

Comon-Lundh and Delaune's notion of *variant* [7] characterizes the instances of a term w.r.t. an equational theory $E \cup Ax$ such that the equations E are convergent and coherent modulo axioms Ax. The E,Ax-*variants* of a term t are *pairs* (t',θ), with θ a substitution and t' the E,Ax-canonical form of $t\theta$. A preorder relation of generalization that holds between such pairs provides a notion of most general variants and also of completeness of a set of variants. An equational theory $E \cup Ax$ has the *finite variant property* iff there is a finite complete set of most general variants for each term. This property also ensures the existence of a generic *finitary* $E \cup Ax$-unification algorithm based on computing variants. Such generic unification algorithm involves performing Ax-unification using a dedicated algorithm and computing the E,Ax-variants.

As we explain in Section 2, the base of axioms now supported by Maude with a dedicated unification algorithm has been extended to include associative-commutative with identity (ACU) function symbols, in combination with the previously supported C, AC, and free ($\emptyset$) function symbols. On this extended axiom base, *Full Maude 2.6*, an extension of Maude written in Maude itself by taking advantage of its reflective capabilities, now offers the following new features for user-definable order-sorted theories $E \cup Ax$ with the finite variant property and satisfying some simple requirements: (i) *variant generation*, that is, computing the most general E,Ax-variants of a term (Section 3); (ii) *variant-based order-sorted unification modulo* $E \cup Ax$ (Section 4); and (iii) *narrowing-based symbolic reachability analysis* of a concurrent system whose equational subtheory satisfies the finite variant property (Section 5). There are several programming languages based on narrowing but none supporting narrowing modulo finite variant theories.

2 Implementation of Order-Sorted ACU Unification

The addition of ACU to the theories handled by the dedicated unification algorithm in Maude required substantial changes to the unification infrastructure implemented in previous versions of Maude for C and AC theories because of the problems associated with collapse theories. In this section we give an overview of the techniques used and highlight a novel algorithm for selecting sets of Diophantine basis elements during the computation of ACU unifiers.

Combining Unification Algorithms. The basic approach to solving unification problems where function symbols are drawn from more than one theory is variable abstraction where *alien subterms*, i.e., subterms headed by a symbol from a theory different from that of the top symbol of the parent term, are replaced by fresh variables to form *pure* unification subproblems which only involve variables and function symbols from a single theory and which can be passed to a unification algorithm for such a theory. Proving termination of combinations of algorithms is nontrivial, as variables are necessarily shared between theories and the unification of variables in one theory can create new unification subproblems in another theory, potentially ad infinitum. Stickel's algorithm [22], which combined the AC and free theories, required an elaborate termination proof by Fages [15]. Boudet et al. [2] proposed a much simpler approach where all unification subproblems and variable bindings in a given theory are solved (and re-solved if another subproblem in that theory is created) simultaneously. This method requires a simultaneous E-unification algorithm for each theory E and was the method implemented in Maude for C, AC, and $\emptyset$ prior to the addition of ACU.

Collapse theories add two major complications to the combination of unification algorithms. Firstly, theory clashes where two terms with top symbols from different theories are required to unify can no longer be treated as a failure, since if one or other top symbol belongs to a collapse theory, a collapse may occur, yielding solutions. Secondly, *compound cycles*, that is, problems of the form $x_1 =^? t_1(\dots,x_2,\dots), x_2 =^? t_2(\dots,x_3,\dots),\dots,x_n =^? t_n(\dots,x_1,\dots)$ where the terms t_i are pure in different theories, can no longer be treated as failure, since solutions may be possible via collapse.

Several authors have proposed combination schemes that can handle collapse theories. We use a simplified version of an algorithm due to Boudet [1]. The original algorithm also handles nonregular theories but we omit that capability to simplify the implementation. The key idea is that each theory E needs a restricted simultaneous E-unification algorithm which solves the simultaneous unification problem for pure equations that are pure in E but where certain variables may be marked as only being allowed to unify with other variables. A theory clash subproblem $f(\dots) =^? g(\dots)$ is split into a disjunction of two subproblems each of which is a conjunction $x =^? f(\dots) \wedge x =^? g(\dots)$ where x is a fresh variable. In one subproblem x is marked in the f equation and in the other subproblem x is marked in the g equation; either or both branches of the search may return solutions. Restricted unification is also used to break compound cycles. Because we do not handle nonregular theories, Boudet-style variable-elimination algorithms are unnecessary.

Boudet's algorithm assumes that theories are disjoint; i.e., that they do not share function symbols. Because in Maude this is not quite true — identities can contain symbols from other theories — we need to handle a special kind of variable elimination. We illustrate the issue with the following example:

```
fmod CYCLE is sort S . vars X Y : S . ops a b c d : -> S .
  op f : S S -> S [assoc comm id: g(c, d)] .
  op g : S S -> S [assoc comm id: f(a, b)] .
endfm
Maude> unify X =? f(Y, a, b) /\ Y =? g(X, c, d) .
```

Here the unification problem would already be in solved form but for the compound cycle formed by the X and Y variables. Restricted unification cannot break this cycle, since neither of the right-hand sides can collapse out of their theory. However, putting $Y = g(c,d)$ eliminates Y from the first equation yielding $X = f(a,b)$ which eliminates X from the second equation, yielding a solution. This situation is somewhat pathological in Maude programs, and we do not really care about performance in its handling. Maude handles it by looking for this kind of cyclic dependency between theories when the signature is preprocessed and setting a flag so that a brute force variable elimination algorithm will be used to try and break compound cycles at unification time.

Diophantine Basis Element Selection. We solve restricted simultaneous ACU unification using an extension of the simultaneous AC unification algorithm in [2]. For an ACU function symbol f we are presented with a set of flattened pure equations that take the form $f(x_1^{p_1}, \dots, x_n^{p_n}) =^? f(y_1^{q_1}, \dots, y_m^{q_m})$ or $x_1 =^? f(y_1^{q_1}, \dots, y_m^{q_m})$. Each f-equation yields a Diophantine equation $p_1X_1 + \cdots + p_nX_n = q_1Y_1 + \cdots + q_mY_m$ or, respectively, $X_1 = q_1Y_1 + \cdots + q_mY_m$ where the X_i's and Y_i's are non-negative Diophantine variables. If an original variable is marked in some equation, the corresponding Diophantine variable receives an upper-bound of 1. Also, we may be able to obtain an upper-bound from order-sorting information, using the signature analysis technique in [12].

The general solution to a set of non-negative Diophantine equations is a set of basis elements from which all solutions can be obtained by linear combination. Upper-bound information may trivially eliminate some basis elements from consideration and can be used by the Diophantine solver to terminate the search for basis elements early.

A fresh variable z_k is allocated for each basis element α_k and unifiers are formed by finding sets of basis elements that satisfy certain properties and constructing assignments $x_i \leftarrow f(\dots, z_k^{\alpha_{k,i}}, \dots)$ where k ranges over the indices of the selected basis elements and $\alpha_{k,i}$ is the value of X_i in the basis element α_k.

The criteria for choosing the sets of basis elements is the key difference between AC unification, ACU unification, and restricted ACU unification. With AC unification, every selection of basis elements whose sum yields a nonzero value for each X_i and Y_i must be considered. With ACU unification that requirement is lifted because of the availability of an identity element. The identity element also means that any assignment including basis element α_k generalizes the same assignment with α_k removed by assigning the identity element to z_k and thus there is a single most general solution, formed by selecting all the basis elements.

In the case of restricted ACU unification, we may have upper-bounds on variables because they are marked. In Maude, order-sorted considerations may place upper-bounds on variables, and may also place a lower-bound of 1 on variables where the corresponding original variable has a sort that cannot take the identity element. In order to find a complete set of unifiers we need to find all maximal sets of basis elements whose sum satisfies the upper and lower-bounds on the variables.

Several explicit schemes for searching the subsets of basis elements were tried but the search was typically the dominant cost for ACU unification, often rendering the solution of quite modest unification problems impractical. In the current implementation this search is performed symbolically using a Binary Decision Diagram (BDD) [3] based algorithm. A BDD variable is allocated for each basis element, whose value, true or false, denotes whether the basis element is included in the subset. A BDD, called *legal*, is constructed, which evaluates to true on exactly those valuations that correspond to selections of basis elements that satisfy the upper- and lower-bound constraints on each Diophantine variable. Enforcement of the upper-bounds on the sum is done using dynamic programming and the BDD *ite* operation. Using the BDD *legal*, a second BDD, called *maximal*, is constructed which is true on exactly those valuations where *legal* is true, and changing a false into a true makes *legal* false. These valuations of the BDD variables and thus the subsets of basis elements they encode are then recovered by tracing the paths from the root to the true terminal in *maximal*. This method yielded a dramatic speed up (from hours to milliseconds) on problems of useful size.

Admissible Equational Theories. Maude 2.6 currently provides a built-in order-sorted *Ax*-unification algorithm for all order-sorted theories (Σ, Ax) such that:

- the order-sorted signature Σ is preregular modulo *Ax* (see [5, Section 3.8]);
- the axioms *Ax* associated to function symbols are as follows:
  - there can be arbitrary function symbols and constants with no equational attributes;
  - the `iter` equational attribute[1] can be declared for some unary symbols;
  - the `comm` or `assoc comm` or `assoc comm id:` attributes[2] can be declared for some binary function symbols, but then no other equational attributes must be given for such symbols.

Explicitly excluded are theories with binary function symbols having any combination of: (i) the `idem` attribute[3]; (ii) the `id:`, `left id:`, or `right id:` attributes without `assoc comm`; or (iii) the `assoc` attribute without `comm`.

3 Variants and Variant Generation

Variant generation for an equational theory $(\Sigma, E \cup Ax)$ is defined modulo *Ax* using the order-sorted *Ax*-unification procedure described in Section 2.

The equational theories that are admissible for variant generation are as follows. Let `fmod` $(\Sigma, Ax \cup E)$ `endfm` be an order-sorted functional module where *E* is a set of equations specified with the `eq` keyword, and *Ax* is a set of axioms such that (Σ, Ax) satisfies the restrictions of Section 2. Furthermore, the equations *E* must satisfy the following extra conditions:

- The equations *E* are unconditional and convergent, sort-decreasing and coherent modulo *Ax*.
- An equation's left-hand side cannot be a variable, and the `owise` feature is not allowed.

[1] Maude provides a built-in mechanism called the `iter` (short for iterated operator) theory whose goal is to permit the efficient input, output, and manipulation of very large stacks of a unary operator. See [6] for additional details.
[2] The operator attribute `assoc` stands for associativity, `comm` for commutativity and `id:` for identity.
[3] The operator attribute `idem` stands for idempotency.

- All equations must be *variant-preserving* [14], i.e., if two left-hand sides of E (possibly renamed) overlap — i.e., there is a substitution θ s.t. $(l_1\theta)|_p =_{Ax} l_2\theta$, where p can be a variable or non-variable position of l_1 — then either:
  1. $l_1\theta$ does not have a pattern modulo Ax, i.e., for every term u s.t. $u =_{Ax} l_1\theta$, u is reducible in E modulo Ax below the root position, or
  2. $l_1\theta$ has a pattern modulo Ax, i.e., there is a term u s.t. $u =_{Ax} l_1\theta$ and u is reducible in E modulo Ax only at the root position, but then the matching substitution is E,Ax-irreducible.

  Variant-preservingness is necessary for an eager generation of variants; see [5] for details.
- An equation's right-hand side must be a *strongly irreducible term*, i.e., for any E,Ax-normalized substitution σ, the term $t\sigma$ is E,Ax-irreducible. A term containing only variables and non-defined (constructor) symbols is strongly irreducible.

The above conditions ensure that $(\Sigma, E\cup Ax)$ has the finite variant property. We refer the reader to [14] for a detailed explanation of variants and variant generation as well as for automated methods for ensuring the finite variant property. Any rewrite theory `mod` $(\Sigma, Ax \cup E \cup G, R)$ `endm` where G is an additional set of equations is also considered admissible for variant generation if the equational part $(\Sigma, Ax \cup E)$ satisfies the conditions described above. Note that when an equational theory $(\Sigma, Ax \cup E \cup G)$ is provided to Full Maude, each equation in E (used for variant computation) must include the `variant` attribute.

Given a module *ModId*, Full Maude provides a variant generation command of the form:

`(get variants` [`in` *ModId* `:`] *t* ` .)`

ACU-Coherence Completion. The convergence and sort-decreasingness of equational Maude specifications can be checked using Maude's Church-Rosser Checker (CRC) [10] and Termination Checker (MTT) [8]. For theories Ax that are combinations of associativity, commutativity, and identity axioms, we can make any specification Ax-coherent by using a procedure which adds Ax-extensions and always terminates (see [20], and [6, Section 4.8] for a more informal explanation).

The user modules are automatically completed for Ax-coherence when used for variant generation and variant-based unification (Section 4) and narrowing (Section 5). The user can access these automatically completed user modules by invoking the command

`(acu coherence completion` [*<module-expr.>*] ` .)`

where *<module-expr.>* is any module expression. If no module expression is given the default current module is completed.

A corresponding `acuCohComplete` function is available at the metalevel of Maude.

```
op acuCohComplete : Module -> Module .
```

A Motivating Example. Consider, for example, the following Petri-net-like specification of a vending machine to buy apples (`a`) or cakes (`c`) with dollars (`$`) and/or quaters (`q`):

```
(mod VENDING-MACHINE is
   sorts Coin Item Marking Money State . subsort Money Item < Marking .
   op empty : -> Money . op <_> : Marking -> State . subsort Coin < Money .
   op __ : Money Money -> Money [assoc comm id: empty] .
   op __ : Marking Marking -> Marking [assoc comm id: empty] .
   ops $ q : -> Coin . ops a c : -> Item . var M : Marking .
   rl [buy-c] : < M $ > => < M c > .
   rl [buy-a] : < M $ > => < M a q > .
   eq [change]: q q q q = $ [variant] .
 endm)
```

The equational theory underlying this rewrite theory contains two subsort-overloaded ACU symbols and an equation `q q q q = $` (for variant computation). Note that the module is not ACU-coherent. It is automatically completed for coherence modulo ACU by replacing the `change` equation by the equation "`eq [change-Ext]: M q q q q = M $ [variant] .`". Note also that this equation satisfies all the conditions above for admissible theories, especially strongly right irreducibility and variant preservingness. We can get variants of a term as follows.

```
Maude> (get variants in VENDING-MACHINE : < $ q q X:Marking > .)
Variant 1
< $ q q X:Marking >, empty substitution
Variant 2
< $ $ #5:Marking >, X:Marking --> q q #5:Marking
```

These two variants represent a finite, complete, and maximal set of variants for the given term. For instance, the variant `{< $ $ q q Y:Marking >, X:Marking --> q q q q Y:Marking}` is an instance of the first variant above, i.e., the canonical form `< $ $ q q Y:Marking >` is an instance of the normal form `< $ q q X:Marking >` of the first variant, and the (normalized version) of the instantiating substitution (`X:Marking --> $ Y:Marking`) is an instance of the empty substitution of the first variant. Note that this variant is not an instance of the second variant above because the substitution `X:Marking --> q q q q Y:Marking` is normalized before comparing it with the substitution `X:Marking --> q q #5:Marking` of the second variant above.

The procedure for variant generation is also available at the metalevel of Maude thanks to the `getVariants` function.

```
op getVariants : Module Term -> VariantFourSet .
```

Handling of Other Axioms. Variant generation and variant-based unification (Section 4) and narrowing (Section 5) have also been extended to deal with *any* combination of associativity and/or commutativity and/or identity axioms except associativity without commutativity. The general idea, borrowed from [9], is to replace a specification $(\Sigma,(Ax \cup Id) \cup E)$ where Ax contains C, AC, or ACU axioms and Id contains all other identity axioms, by a *semantically equivalent* specification $(\Sigma, Ax \cup (\vec{Id} \cup \widehat{E}))$, where the Id axioms have been oriented as rules, and the equations $\widehat{E}$ are the $\vec{Id}, Ax$*-variants* of the original equations E.

A command is available in Full Maude of the form:

(`remove id attributes` [<*module-expr.*>] `.`)

It shows the specified module with the identity attributes (`id`, `right id`, and `left id`) transformed into rules and the equations $\widehat{E}$ obtained using $\vec{Id}, Ax$-variants. If no module expression is given, the default current module is used.

A corresponding function `removeIds` is available at the metalevel of Maude.

```
op removeIds : Module -> Module .
```

4 Variant-based Equational Order-Sorted Unification

The intimate connection between E, Ax-variants and $E \cup Ax$-unification is as follows. Suppose that we extend the equational theory $(\Sigma, E \cup Ax)$ to $(\widehat{\Sigma}, \widehat{E} \cup Ax)$ by adding to Σ a new sort Truth, not related to any sort in Σ, with a constant `tt`, and for each top sort [s] of each connected component s, an operator `eq` : [s] $\times$ [s] $\rightarrow$ Truth; and where $\widehat{E}$ extends E by adding for each top sort [s] and x of sort [s] an extra rule $eq(x,x) \rightarrow tt$. Then, given any two terms t, t', if θ is an (E, Ax)-unifier of t and t', then the E, Ax-canonical forms of $t\theta$ and $t'\theta$ must be Ax-equal and therefore the pair $(\mathtt{tt}, \theta)$ must be a variant of the term $eq(t,t')$, i.e., $eq(t,t')\theta \rightarrow^! \mathtt{tt}$. Furthermore, if the term $eq(t,t')$ has a finite set of most general variants, then we are *guaranteed* that the set of most general (E, Ax)-unifiers of t and t' is *finite* and subsumes $(\mathtt{tt}, \theta)$.

Given a module *ModId* of the general form `mod` $(\Sigma, Ax \cup E \cup G, R)$ `endm` where $(\Sigma, Ax \cup E)$ satisfies the requirements of Section 3, Full Maude provides a command for $E \cup Ax$-equational unification based on variant generation of the form:

(`variant unify` [`in` *ModId* `:`] t `=?` t' `.`)

Consider again the vending machine. We can ask whether there is an $E \cup Ax$-equational unifier of two configurations, one containing a dollar and two quarters and another containing two quarters:

```
Maude> (variant unify in VENDING-MACHINE : < q q X:Marking > =? < $ Y:Marking > .)
Solution 1
X:Marking --> q q Y:Marking
Solution 2
X:Marking --> $ #12:Marking ; Y:Marking --> q q #12:Marking
```

There are no more general unifiers. For instance, `X:Marking --> q q`, `Y:Marking --> empty` is an instance of the first solution by using the identity property of the operator for markings.

The procedure for variant-based equational unification is also available at the metalevel thanks to the `metaVariantUnify` function.

```
op metaVariantUnify : Module Term Term -> SubstitutionSet .
```

A useful special case of the variant-based equational unification feature is that of Ax'-unification for theories (Σ, Ax') where: (i) $Ax' = Ax \cup Ids$, (ii) (Σ, Ax) satisfies the requirements in Section 3, and (iii) Ids is a collection of `id:`, `left id:`, `right id:` axioms. This case is handled by invoking the `variant unify` command directly on (Σ, Ax'), since Full Maude first invokes the `remove id attributes` transformation command described in Section 3.

5 Narrowing-based Symbolic Reachability Analysis

Narrowing [16] generalizes term rewriting by allowing free variables in terms and by performing unification instead of matching. Likewise, narrowing *modulo* $Ax \cup E$ [18] generalizes rewriting with rules R modulo $Ax \cup E$. Given an order-sorted rewrite theory $(\Sigma, Ax \cup E, R)$, where R is a set of unconditional rewrite rules such that the left-hand sides are non-variable terms and the rules are explicitly $Ax \cup E$-coherent [19], and $(\Sigma, Ax \cup E)$ is an equational theory such that a finitary $Ax \cup E$-unification procedure is available, the $(R, Ax \cup E)$*-narrowing* relation is defined as $t \rightsquigarrow_{\sigma,p,R,Ax\cup E} t'$ iff there is a non-variable position p of t, a (possibly renamed) rule $l \to r$ in R, and a unifier $\sigma \in Unif_{Ax\cup E}(t|_p, l)$ such that $t' = \sigma(t[r]_p)$.

The classical application of $(R, Ax \cup E)$-narrowing is to perform $R \cup Ax \cup E$*-unification* when the rules R are understood as *equations*. Indeed the variant-based equational order-sorted unification algorithm of Section 4 is based on an E, Ax-narrowing strategy, called *folding variant narrowing* [14], that terminates when $E \cup Ax$ has the finite variant property [7], even though full E, Ax-narrowing typically does not terminate when Ax contains AC axioms (see [7, 14]).

Instead, when the rules R are understood as *transition rules*, a completely different application of $R, Ax \cup E$-narrowing is that of *symbolic reachability analysis* [19]. Specifically, we consider transition systems specified by order-sorted rewrite theories of the form `mod` $(\Sigma, Ax \cup E, R)$ `endm` where: (i) $E \cup Ax$ satisfies the requirements of Section 3, and (ii) the transition rules R are $E \cup Ax$-coherent and *topmost* (so that rewriting is always done at the top of the term). Then, narrowing modulo $E \cup Ax$ is a *complete* deductive method [19] for symbolic reachability analysis, that is, for solving existential queries of the form $\exists \bar{x} : t(\bar{x}) \to^* t'(\bar{x})$ in the sense that the formula holds for $(\Sigma, Ax \cup E, R)$ iff there is a narrowing sequence $t \rightsquigarrow^*_{R,E\cup Ax} u$ such that u and t' have a $E \cup Ax$-unifier.

This symbolic reachability analysis is supported by Full Maude's `search` command, which has the form:

`(search` [`[`n`,`m`]`] [`in` *ModId* `:`] t_1 *SearchArrow* t_2 `.)`

where: n and m are optional arguments providing, respectively, a bound on the number of desired solutions and the maximum depth of the search; *ModId* is the module where the search takes place; t_1 is the starting *non-variable term*, which may contain variables; t_2 is the term specifying the pattern that has to be reached, with variables, some of then possibly shared with t_1; and *SearchArrow* is an arrow indicating the form of the narrowing proof from t_1 until t_2, where `~>1` indicates a narrowing proof consisting of exactly one step; `~>+` indicates a proof of one or more steps; `~>*` indicates a proof of none, one, or more steps; and `~>!` indicates that the reached term cannot be further narrowed. This narrowing-based search command was already introduced in [4] but now can be performed modulo theories with the finite variant property.

Consider again the vending machine of Section 3. We can use the narrowing search command to answer the question: *Is there any combination of one or more coins that returns exactly an apple and a cake?* This can be done by searching for states that are reachable from a term `< M:Money >` and match the desired pattern at the end.

```
Maude> (search [1] in VENDING-MACHINE : < M:Money > ~>* < a c > .)
Solution 1
M:Money --> $ q q q
```

Note that we must restrict the search to just one solution, because narrowing does not terminate for this reachability problem even though the above solution is indeed the *only* solution.

Narrowing-based reachability analysis is also available at the metalevel by using the following `metaNarrowSearch` function.

```
op metaNarrowSearch :
  Module Term Term Substitution Qid Bound Bound Bound -> ResultTripleSet .
```

If a non-identity substitution is provided in the fourth argument, then any computed substitution must be an instance of the provided one, i.e., we can restrict the computed narrowing sequences to some concrete shape. The `Qid` metarepresents the appropriate search arrow, similar to the `metaSearch` command (see [5, Section 11.4.6]). For the bounds, the first one is the number of computed solutions, the second one is the maximum length of the narrowing sequences, i.e., the depth of the narrowing tree, and the third one is the chosen solution (in order to provide all solutions in a sequential way, as many meta-level commands in Maude do).

Full Maude's `search` command also supports a more general form of symbolic reachability analysis that uses narrowing *with simplification*. We can allow more general rewrite theories of the form `mod` $(\Sigma, Ax \cup E \cup G, R)$ `endm` where: (i) $E \cup Ax$ satisfies the requirements of Section 3, (ii) G is an additional set of equations, and (iii) the rules R are $E \cup Ax \cup G$-coherent and topmost. The remaining equations G are now used in the combined relation $\leadsto_{R,E\cup Ax};\to^{!}_{E\cup G,Ax}$. Note that this combined relation may be incomplete, i.e., given a reachability problem of the form $\exists \overline{x} : t(\overline{x}) \to^{*} t'(\overline{x})$ and a solution σ (i.e., $\sigma(t) \to^{*}_{R,E\cup Ax\cup G} \sigma(t')$), the relation $\leadsto_{R,E\cup Ax};\to^{!}_{E\cup G,Ax}$ may not be able to find a solution more general than σ.

6 Applications

The key usefulness of the new `variant unify` feature is to *greatly extend the range of theories* for which a unification algorithm can be provided by Maude. The key distinction is one between *dedicated* algorithms for a given theory, and *generic* algorithms such as folding variant narrowing which can be applied to a wide range of user-defined theories. As explained in this paper, Maude 2.6 has a dedicated algorithm supporting order-sorted unification modulo axioms Ax which may contain C, AC, and ACU axioms. The `variant unify` feature then allows us to *automatically* derive a finitary unification algorithm for any theory $E \cup Ax$ that satisfies the requirements in Section 3 and therefore enjoys the finite variant property. In particular, as shown in [7], a good number of cryptographic theories of practical interest satisfy the finite variant property modulo axioms such as AC or $\emptyset$.

Support for variant-based unification can therefore be exploited by cryptographic protocol analysis tools performing symbolic reachability analysis. Such protocols can be modeled as rewrite theories $(\Sigma, E \cup Ax, R)$, where the algebraic properties of the cryptographic functions are specified by equations $E \cup Ax$, and the protocol's transition rules are specified by the rewrite rules R. Thus the narrowing search feature modulo a theory $E \cup Ax$ satisfying the finite variant property is a feature which, by being available also at the metalevel, can be the basis of a protocol analysis tool performing reachability analysis for protocol verification. This is exactly the approach that has been followed for analyzing cryptographic protocols modulo algebraic properties in the Maude-NPA tool [13, 21], which has been able to analyze a substantial collection of cryptographic protocols modulo their algebraic properties. With the `metaNarrowSearch` operator, this same functionality becomes now available to other protocol analysis tools. As an example, business processes can be similarly analyzed to check for violations. The Document Logic Analysis tool [17] represents document processing protocols as theories in rewriting logic and uses symbolic reachability analysis in Maude to look for forgeries and invalid signatures.

The usefulness of variants and variant generation goes beyond the availability of finitary unification algorithms and symbolic reachability analysis for cryptographic protocols and for other concurrent systems. As demonstrated by its recent applications to termination algorithms modulo axioms in [9], and to algorithms for checking confluence and coherence of rewrite theories modulo axioms, such as those used in the most recent Maude CRC and ChC tools [10, 11], computing the E,Ax*-variants* of a term may be just as important as computing $E \cup Ax$-unifiers. The key idea is the following. Suppose that R is a collection of rewrite rules modulo axioms Ax for which we want to prove, say, termination, or confluence. We may not have any tools for checking such properties that can work modulo the given set of axioms Ax. However, we can *decompose* Ax as a disjoint union $E \cup Ax'$, where E is convergent, sort-decreasing and coherent modulo Ax', and where we have methods to prove, e.g., termination or confluence modulo Ax'. As shown in [9], we can transform R,Ax into a semantically equivalent theory $\widehat{R} \cup E,Ax'$, where $\widehat{R}$ specializes each rule in R to the family of E,Ax'-variants of their left-hand sides. If $E \cup Ax'$ has the finite variant property, we are sure that $\widehat{R}$ will be a finite set; but in practice $\widehat{R}$ can often be finite without such a property. For example, Ax can be the theory A of associativity, for which unification is not even finitary, yet in an order-sorted setting A can often be added as a rule so that $\widehat{R}$ is finite in practice. We refer to [9, 10, 11] for details.

Acknowledgements. We are very thankful to the other members of the Maude team, namely, Manuel Clavel, Patrick Lincoln, and Narciso Martí-Oliet, with whom we have designed and built the Maude language and system.

References

1 A. Boudet. Unification in a combination of equational theories: an efficient algorithm. In *Proceedings of the tenth international conference on Automated deduction*, CADE-10, pages 292–307. Springer-Verlag, 1990.

2 A. Boudet, E. Contejean, and H. Devie. A new AC-unification algorithm with a new algorithm for solving Diophantine equations. In *Proceedings of the 5th IEEE Symposium on Logic in Computer Science*, pages 289–299. IEEE Computer Society Press, 1990.

3 R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35:677–691, 1986.

4 M. Clavel, F. Durán, S. Eker, S. Escobar, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott. Unification and narrowing in Maude 2.4. In R. Treinen, editor, *Rewriting Techniques and Applications, 20th International Conference, RTA 2009, Brasília, Brazil, June 29 - July 1, 2009, Proceedings*, volume 5595 of *Lecture Notes in Computer Science*, pages 380–390. Springer, 2009.

5 M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. Maude Manual (Version 2.6). March 2011, `http://maude.cs.uiuc.edu`.

6 M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude - A High-Performance Logical Framework: How to Specify, Program, and Verify Systems in Rewriting Logic*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.

7 H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In J. Giesl, editor, *Term Rewriting and Applications, 16th International Conference, RTA 2005, Nara, Japan, April 19-21, 2005, Proceedings*, volume 3467 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2005.

8 F. Durán, S. Lucas, and J. Meseguer. MTT: The Maude termination tool (system description). In A. Armando, P. Baumgartner, and G. Dowek, editors, *Automated Reasoning 4th International Joint Conference, IJCAR 2008 Sydney, Australia, August 12-15, 2008 Proceedings*, volume 5195 of *Lecture Notes in Computer Science*, pages 313–319. Springer, 2008.

9 F. Durán, S. Lucas, and J. Meseguer. Termination modulo combinations of equational theories. In S. Ghilardi and R. Sebastiani, editors, *Frontiers of Combining Systems, 7th International Symposium, FroCoS 2009, Trento, Italy, September 16-18, 2009. Proceedings*, volume 5749 of *Lecture Notes in Computer Science*, pages 246–262. Springer, 2009.

10 F. Durán and J. Meseguer. A Church-Rosser checker tool for conditional order-sorted equational Maude specifications. In P. C. Ölveczky, editor, *8th International Workshop on Rewriting Logic and its Applications*, volume 6381 of *Lecture Notes in Computer Science*, pages 69-85. Springer, 2010.

11 F. Durán and J. Meseguer. A Maude coherence checker tool for conditional order-sorted rewrite theories. In P. C. Ölveczky, editor, *8th International Workshop on Rewriting Logic and its Applications*, volume 6381 of *Lecture Notes in Computer Science*, pages 86-103. Springer, 2010.

12 S. Eker. Fast matching in combinations of regular equational theories. In J. Meseguer, editor, *Proceedings of the First International Workshop on Rewriting Logic and its Applications*, volume 4 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1996.

13 S. Escobar, C. Meadows, and J. Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design V, FOSAD 2007/2008/2009 Tutorial Lectures*, volume 5705 of *Lecture Notes in Computer Science*, pages 1–50. Springer, 2009.

14 S. Escobar, R. Sasse, and J. Meseguer. Folding variant narrowing and optimal variant termination. *The Journal of Logic and Algebraic Programming*, 2011. In Press.

15 F. Fages. Associative-commutative unification. *Journal of Symbolic Computation*, 3:257–275, June 1987.

16 J.-M. Hullot. Canonical forms and unification. In W. Bibel and R. A. Kowalski, editors, *CADE*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334. Springer, 1980.

17 S. Iida, G. Denker, and C. Talcott. Document logic: Risk analysis of business processes through document authenticity. In *Second International Workshop on Dynamic and Declarative Business Processes (DDBP)*. IEEE Digital Library, 2009. Extended version to appear in Journal of Research and Practice in Information Technology, 2011.

18 J.-P. Jouannaud, C. Kirchner, and H. Kirchner. Incremental construction of unification algorithms in equational theories. In J. Díaz, editor, *ICALP*, volume 154 of *Lecture Notes in Computer Science*, pages 361–373. Springer, 1983.

19 J. Meseguer and P. Thati. Symbolic reachability analysis using narrowing and its application to verification of cryptographic protocols. *Higher-Order and Symbolic Computation*, 20(1-2):123–160, 2007.

20 G. Peterson and M. Stickel. Complete sets of reductions for some equational theories. *Journal of ACM*, 28(2):233–264, 1981.

21 R. Sasse, S. Escobar, C. Meadows, and J. Meseguer. Protocol analysis modulo a combination of theories: A case study in Maude-NPA. In *6th International Workshop on Security and Trust Management (STM'10)*, Lecture Notes in Computer Science. Springer, 2010. To appear.

22 M. E. Stickel. A complete unification algorithm for associative-commutative functions. In *Proceedings of the 4th international joint conference on Artificial intelligence - Volume 1*, pages 71–76. Morgan Kaufmann Publishers Inc., 1975.