Equational Unification and Matching, and Symbolic Reachability Analysis in Maude 3.2 (System Description)

Francisco Durán[1], Steven Eker[2], Santiago Escobar[3(✉)], Narciso Martí-Oliet[4], José Meseguer[5], Rubén Rubio[4], and Carolyn Talcott[2]

[1] Universidad de Málaga, Málaga, Spain
duran@lcc.uma.es
[2] SRI International, Menlo Park, CA, USA
eker@csl.sri.com, clt@cs.stanford.edu
[3] VRAIN, Universitat Politècnica de València, Valencia, Spain
sescobar@upv.es
[4] Universidad Complutense de Madrid, Madrid, Spain
{narciso,rubenrub}@ucm.es
[5] University of Illinois at Urbana-Champaign, Urbana, IL, USA
meseguer@illinois.edu

Abstract. Equational unification and matching are fundamental mechanisms in many automated deduction applications. Supporting them efficiently for as wide as possible a class of equational theories, and in a typed manner supporting type hierarchies, benefits many applications; but this is both challenging and nontrivial. We present Maude 3.2's efficient support of these features as well as of symbolic reachability analysis of infinite-state concurrent systems based on them.

1 Introduction

Unification is a key mechanism in resolution [41] and paramodulation-based [36] theorem proving. Since Plotkin's work [40] on *equational unification*, i.e.,

Durán was supported by the grant UMA18-FEDERJA-180 funded by J. Andalucía/FEDER and the grant PGC2018-094905-B-I00 funded by MCIN/AEI/10.13039/501100011033 and ERDF A way of making Europe. Escobar was supported by the EC H2020-EU grant 952215, by the grant RTI2018-094403-B-C32 funded by MCIN/AEI/10.13039/501100011033 and ERDF A way of making Europe, by the grant PROMETEO/2019/098 funded by Generalitat Valenciana, and by the grant PCI2020-120708-2 funded by MICIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR. Martí-Oliet and Rubio were supported by the grant PID2019-108528RB-C22 funded by MCIN/AEI/10.13039/501100011033 and ERDF A way of making Europe. Talcott was partially supported by the U. S. Office of Naval Research under award numbers N00014-15-1-2202 and N00014-20-1-2644, and NRL grant N0017317-1-G002.

© The Author(s) 2022
J. Blanchette et al. (Eds.): IJCAR 2022, LNAI 13385, pp. 529–540, 2022.
https://doi.org/10.1007/978-3-031-10769-6_31

E-unification modulo an equational theory E, it is widely used for increased effectiveness. Since Walther's work [47] it has been well understood that *typed* E-unification, exploiting types and subtype hierarchies, can drastically reduce a prover's search space. Many other automated deduction applications use typed E-unification as a key mechanism, including, inter alia: (i) constraint logic programming, e.g., [12,23]; (ii) narrowing-based infinite-state reachability analysis and model checking, e.g., [6,35]; (iii) cryptographic protocol analysis modulo algebraic properties, e.g., [8,19,28]; (iv) partial evaluation, e.g., [4,5]; and (v) SMT solving, e.g., [32,48]. The special case of typed *E-matching* is also a key component in all the above areas as well as in: (vi) E-generalization (also called anti-unification), e.g., [1,2]; and (vii) E-homeomorphic embedding, e.g., [3].

Maximizing the scope and effectiveness of typed E-unification and E-matching means efficiently supporting as wide a class of theories E as possible. Such efficiency crucially depends on both efficient algorithms (and their combinations) and —since the number of E-unifiers may be large— on computing complete *minimal* sets of solutions to reduce the search space. The recent Maude 3.2 release[1] provides this kind of efficient support for typed E-unification and E-matching in three, increasingly more general classes of theories E:

1. Typed B-unification and B-matching, where B is any combination of associativity (A) and/or commutativity (C) and/or unit element (U) axioms.
2. Typed $E \cup B$-unification and matching in the *user-definable* infinite class of theories $E \cup B$ with B as in (1), and $E \cup B$ having the *finite variant property* (FVP) [13,21].
3. Typed $E \cup B$-unification for the infinite class of *user-definable* theories $E \cup B$ with B as in (1), and E confluent, terminating, and coherent modulo B.

For classes (1) and (2) the set of B- (resp. $E \cup B$-) unifiers is always *complete, minimal and finite*, except for the *AwoC* case when B contains an A but not C axiom for some binary symbol f.[2] The typing is order-sorted [22,29] and thus contains many-sorted and unsorted B- (resp. $E \cup B$-) unification as special cases. For class (3), Maude enumerates a possibly infinite complete set of $E \cup B$-unifiers, with the same AwoC exception on B. We discuss new features for classes (1)–(2), and a new narrowing modulo $E \cup B$-based *symbolic reachability analysis* feature for infinite-state systems specified in Maude as rewrite theories $(\Sigma, E \cup B, R)$ with equations $E \cup B$ in class (2) and concurrent transition rules R. In Sect. 5 we discuss various applications that can benefit from these new features.

In comparison with previous Maude tool papers reporting on new features —the last one was [16]— the new features reported here include: (i) computing minimal complete sets of most general B- (resp. $E \cup B$-) unifiers for classes (1) and (2) except for the AwoC case; (ii) a new $E \cup B$-matching algorithm for class (2); and (iii) a new symbolic reachability analysis for concurrent systems

[1] Publicly available at http://maude.cs.illinois.edu.

[2] In the AwoC case, Maude's algorithms are optimized to favor many commonly occurring cases where typed A-unification is finitary, and provides a finite set of solutions and an incompleteness warning outside such cases (see [18]).

based on narrowing with transition rules modulo equations $E \cup B$ in class (2) enjoying powerful state-space reduction capabilities based on the minimality and completeness feature (i) and on "folding" less general symbolic states into more general ones through subsumption. Section 3.1 shows the importance of the new $E \cup B$-matching algorithm for efficient computation of minimal $E \cup B$-unifiers.

Notation, Strict-B-Coherence, and FVP. For notation involving either term positions, $p \in pos(t)$, $t|_p$, $t[t']_p$, or substitutions, $t\theta$, $\theta\mu$, see [14]. Equations $(u = v) \in E$ oriented as rules $(u \to v) \in \overrightarrow{E}$ are *strictly coherent* modulo axioms B iff $(t =_B t' \wedge t \to_{\overrightarrow{E},B} w) \Rightarrow \exists w'(t \to_{\overrightarrow{E},B} w' \wedge w =_B w')$, where $t \to_{\overrightarrow{E},B} w$ iff $\exists (u \to v) \in \overrightarrow{E}, \exists \theta, \exists p \in pos(t)(u\theta =_B t|_p \wedge w = t[v\theta]_p)$. For $(\Sigma, E \cup B)$ an equational theory with $\overrightarrow{E}$ confluent, terminating and strictly coherent modulo B, (1) an $\overrightarrow{E}, B$-*t-variant* is a pair (v, θ) s.t. $v = (t\theta)!_{\overrightarrow{E},B} \wedge \theta = \theta!_{\overrightarrow{E},B}$, where $u!_{\overrightarrow{E},B}$ (resp. $\theta!_{\overrightarrow{E},B}$) denotes the $\overrightarrow{E}, B$-normal form of u, resp. θ; (2) for $\overrightarrow{E}, B$-t-variants (v, θ), (u, μ), the *more general relation* $(v, \theta) \sqsupseteq_B (u, \mu)$ holds iff $\exists \gamma(u =_B v\gamma \wedge \theta\gamma =_B \mu)$; (3) $(\Sigma, E \cup B)$ is *FVP* [13,21] iff any Σ-term t has a *finite* set of most general $\overrightarrow{E}, B$-t-variants. Footnote 5 explains how FVP can be checked.

2 Complete and Minimal Order-Sorted B-Unifiers

Throughout the paper we use the following equational theory $E \cup B$ of the Booleans as a running example (with self-explanatory, user-definable syntax[3]):

```
fmod BOOL-FVP is protecting TRUTH-VALUE .
   op _and_ : Bool Bool -> Bool [assoc comm] .
   op _xor_ : Bool Bool -> Bool [assoc comm] .
   op not_ : Bool -> Bool .
   op _or_ : Bool Bool -> Bool .
   op _<=>_ : Bool Bool -> Bool .
   vars X Y Z W : Bool .

   eq X and true = X [variant] .
   eq X and false = false [variant] .
   eq X and X = X  [variant] .
   eq X and X and Y = X and Y [variant] .    *** AC extension
   eq X xor false = X [variant] .
   eq X xor X = false [variant] .
   eq X xor X xor Y = Y [variant] .          *** AC extension
   eq not X = X xor true [variant] .
   eq X or Y = (X and Y) xor X xor Y [variant] .
   eq X <=> Y = true xor X xor Y [variant] .
endfm
```

[3] This module imports Maude's `TRUTH-VALUE` module and the command "`set include BOOL off .`" must be typed before the module to avoid default importation of `BOOL`.

The axioms B are the associativity-commutativity (AC) axioms for `xor` and `and` (specified with the `assoc comm` attributes). The equations E are terminating and confluent modulo B [42]. To achieve strict B-*coherence* [30], the needed AC-extensions [39] are added —for example, the AC-extension of `X xor X = false` is `X xor X xor Y = Y`. The equations E for `xor` and `and` define the theory of *Boolean rings*, *except for the missing*[4] *distributivity equation* `X and (Y xor Z) = (X and Y) xor (X and Z)`. The remaining equations in E define `or`, `not` and `<=>` as definitional extensions. The `variant` attribute declares that the equation will be used for folding variant narrowing [21]. The theory is FVP,[5] in class (2). In this section we will consider B-unification (for $B = AC$) using this example. $E \cup B$-unification for the same example will be discussed in Sect. 3.

For B any combination of associativity and/or commutativity and/or identity axioms, Maude's `unify` command computes a complete finite set of most general B-unifiers, except for the AwoC case. The new `irredundant unify` command always returns[6] a *finite, complete and minimal* set of B-unifiers, except for the AwoC case. The output of `unify` for the equation below can be found in [10, §13].

```
Maude> irredundant unify X and not Y and not Z =? W and Y and not X .
Decision time: 0ms cpu (0ms real)

Unifier 1                                  Unifier 2
X --> #1:Bool and #2:Bool                  X --> #2:Bool
Z --> #1:Bool and #2:Bool                  Z --> #1:Bool
Y --> #1:Bool                              Y --> #2:Bool
W --> #2:Bool and not #1:Bool              W --> not #1:Bool
```

3 $E \cup B$-Unification and Matching for FVP Theories

It is a general result from [21] that if $E \cup B$ is FVP and B-unification is finitary, then $E \cup B$-unification is *finitary* and a complete finite set of $E \cup B$-unifiers can be computed by *folding variant narrowing* [21]. Furthermore, assuming that $T_{\Sigma/E,s}$ is non-empty for each sort s, a finitary $E \cup B$-unification algorithm automatically provides a decision procedure for *satisfiability* of any *positive* (the $\wedge, \vee$-fragment) quantifier-free formula φ in the initial algebra $T_{\Sigma/E}$, since φ can be put in DNF, and a conjunction of equalities Γ is satisfiable in $T_{\Sigma/E}$ iff Γ is $E \cup B$-unifiable.

Since for our running example `BOOL-FVP` the equations $E \cup B$ are FVP and B-unification (in this case $B = AC$) is finitary, all this has useful consequences for

[4] By missing distributivity, this theory is *weaker* than the theory of Boolean rings. Nevertheless, its *initial algebra* $T_{\Sigma/E \cup B}$ is exactly the Booleans on {`true,false`} with the standard truth tables for all connectives. Thus, all equations provable in Boolean algebra hold in $T_{\Sigma/E \cup B}$, including the missing distributivity equation.

[5] This can be easily checked in Maude by checking the finiteness of the variants for each $f(X)$, resp. $f(X, Y)$, for each unary, resp. binary, symbol f in `BOOL-FVP` using the `get variants` command; see [9] for a theoretical justification of this check.

[6] Fresh variables follow the form `#1:Bool`.

`BOOL-FVP`. Indeed, $T_{\Sigma/E \cup B}$ is exactly the Booleans[7] on {`true`,`false`} with the well-known truth tables for `and`, `xor`, `not`, `or` and `<=>`. This means that $E \cup B$-unification provides a Boolean *satisfiability decision procedure* for a Boolean expression u on such symbols, namely, u is Boolean satisfiable iff the equation $u =$ `true` is $E \cup B$-unifiable. Furthermore, a ground assignment ρ to the variables of u is a satisfying assignment for u iff there exists an $E \cup B$-unifier α of $u =$ `true` and a ground substitution δ such that $\rho = \alpha\delta$. For the same reasons, u is a Boolean *tautology* iff the equation $u =$ `false` has no $E \cup B$-unifiers.

A complete, finite set of $E \cup B$-unifiers can be computed with Maude's `variant unify` command whenever $E \cup B$ is FVP, except for the AwoC case. Instead, the new[8] `filtered variant unify` command computes a *finite, complete and minimal* set of $E \cup B$-unifiers, which can be considerably smaller than that computed by `variant unify`. For our `BOOL-FVP` example, `filtered variant unify` gives us a Boolean satisfiability decision procedure plus a symbolic specification of satisfying assignments. Such a procedure is not practical: it cannot compete with standard SAT-solvers; but that was never our purpose: our purpose here is to illustrate with simple examples how $E \cup B$-unification works for the *infinite* class of *user-definable* FVP theories $E \cup B$, of which `BOOL-FVP` is just a simple example; dozens of other examples can be found in [32].

The difference between the `variant unify` and the new `filtered variant unify` command is illustrated with the following example; its unfiltered output can be found in [10, §14]. Note that the single $E \cup B$-unifier gives us a compact symbolic description of this Boolean expression's satisfying assignments.

```
Maude> filtered variant unify (X or Y) <=> Z =? true .
rewrites: 3224 in 12765ms cpu (14776ms real) (252 rewrites/second)

Unifier 1
X --> #1:Bool xor #2:Bool
Y --> #1:Bool
Z --> #2:Bool xor (#1:Bool and (#1:Bool xor #2:Bool))

No more unifiers.
Advisory: Filtering was complete.
```

The computation of a minimal set of $E \cup B$-unifiers relies on filtering by $E \cup B$-matching between two $E \cup B$-unifiers, as explained in the following section.

3.1 FVP $E \cup B$-Matching and Minimality of $E \cup B$-Unifiers

By definition, a term u $E \cup B$-*matches* another term v iff there is a substitution γ such that $u =_{E \cup B} v\gamma$. Besides the existing `match` command modulo axioms

[7] Each connective's truth table can be checked with Maude's `reduce` command. Actually, need only check `and` and `xor` (other connectives are definitional extensions).

[8] In Maude, different command names are used to emphasize different algorithms. The word 'filtered' is used instead of 'irredundant' because irredundancy is not guaranteed in the AwoC case.

B, Maude's new `variant match` command computes a complete, minimal set of $E \cup B$-*matching substitutions* for any FVP theory $E \cup B$ in class (2), except for the AwoC case. Such an algorithm could always be derived from an $E \cup B$-unification algorithm by replacing u by $\overline{u}$, where all variables in u are replaced by fresh constants in $\overline{u}$, and computing the $E \cup B$-unifiers of $\overline{u} = v$. But a more efficient special-purpose algorithm has been designed and implemented for this purpose. $E \cup B$-matching algorithms are automatically provided by Maude for any *user-definable* theory in class (2) with the `variant match` command.

```
Maude> variant match in BOOL-FVP : Z and W <=? X .
rewrites: 12 in 21ms cpu (27ms real) (545 rewrites/second)

Matcher 1          Matcher 2          Matcher 3
Z --> true         Z --> X            Z --> X
W --> X            W --> true         W --> X
```

This is a good moment to ask and answer a relevant question: Why is computing a complete *minimal* set of $E \cup B$-unifiers for a unification problem Γ, where $E \cup B$ is an FVP theory in class (2) except for the AwoC case, *nontrivial*? We first need to explain how minimality is achieved. Suppose that α and β are two $E \cup B$-unifiers of a system of equations Γ with, say, typed variables $x_1, \ldots, x_n$. We then say that α *is more general than* β modulo $E \cup B$, denoted $\alpha \sqsupseteq_{E \cup B} \beta$, iff there is a substitution γ such that for each x_i, $1 \leq i \leq n$, $\gamma(\alpha(x_i)) =_{E \cup B} \beta(x_i)$. But this exactly means that the vector $[\beta(x_1), \ldots, \beta(x_n)]$ $E \cup B$-matches the vector $[\alpha(x_1), \ldots, \alpha(x_n)]$ with $E \cup B$-matching substitution γ. A complete set of $E \cup B$-unifiers of Γ is by definition *minimal* iff for any two different unifiers α and β in it we have $\alpha \not\sqsupseteq_{E \cup B} \beta$ *and* $\beta \not\sqsupseteq_{E \cup B} \alpha$, i.e., the two associated $E \cup B$-matching problems fail.

What is *nontrivial* is computing a minimal complete set of $E \cup B$-unifiers *efficiently*. One could do so inefficiently by simulating $E \cup B$-matching with $E \cup B$-unification, and more efficiently by using an $E \cup B$-matching algorithm. Maude achieves still greater efficiency by directly computing the $\alpha \sqsupseteq_{E \cup B} \beta$ relation. The key difference between the `variant unify` command and the new `filtered variant unify` command is that the second computes a $E \cup B$-minimal set of $E \cup B$-unifiers of Γ using the $\alpha \sqsupseteq_{E \cup B} \beta$ relation, whereas the first only computes a set of B-minimal $E \cup B$-unifiers of Γ using the cheaper $\alpha \sqsupseteq_B \beta$ relation. There are three ideas we use to make it fast in practice: (i) variant matching is faster than variant unification because one side is variable-free; (ii) enumerating the variant matchers between two variant unifiers is far more expensive than checking existence of a matcher; and (iii) variant unifiers are discarded on-the-fly avoiding further narrowing steps and computation.

4 Narrowing-Based Symbolic Reachability Analysis

In Maude, concurrent systems are specified in so-called *system modules* as *rewrite theories* of the form: $\mathcal{R} = (\Sigma, G, R)$, where G is an equational theory either of the

form B in class (1), or $E \cup B$ in classes (2) or (3), and R are the *system transition rules*, specified as rewrite rules. When the theory $\mathcal{R}$ is *topmost*, meaning that the rules R rewrite the entire state, narrowing with rules R modulo the equations G is a *complete* symbolic reachability analysis method for *infinite-state systems* [35]. That is, given a term u with variables $\overrightarrow{x}$, representing a typically infinite set of initial states, and another term v with variables $\overrightarrow{y}$, representing a possibly infinite set of target states, narrowing can answer the question: *can an instance of u reach an instance of v?* That is, does the formula $\exists \overrightarrow{x}, \overrightarrow{y} \;\; u \rightarrow^* v$ hold in $\mathcal{R}$? Note that, if the *complement* of a system invariant I can be symbolically described as the set of ground instances of terms in a set $\{v_1, \ldots, v_n\}$ of pattern terms, then narrowing provides a semi-decision procedure for verifying whether the system specified by $\mathcal{R}$ fails to satisfy I starting from an initial set of states specified by u. Namely, I holds iff no instance of any v_i can be reached from some instance of u.

Assuming G is in class (1) or (2), Maude's `vu-narrow` command implements narrowing with R modulo G by performing G-unification at each narrowing step. However, the number of symbolic states that need to be explored can be *infinite*. This means that if no solution exists for the narrowing search, Maude will search forever, so that only *depth-bounded searches* will terminate. The great advantage of the new `{fold} vu-narrow {filter,delay}` command is that it performs a powerful *symbolic state space reduction* by: (i) removing a newly explored symbolic state v' if it $E \cup B$-matches a previously explored state v and replacing transition with target v' by transitions with target v; and (ii) using minimal sets of $E \cup B$-unifiers for each narrowing step and for checking common instances between a newly explored state and the target term (ensured by words `filter` and `delay`). This can make the entire search space finite and allow full verification of invariants for some infinite-state systems. Consider the following Maude specification of Lamport's bakery protocol.

```
mod BAKERY is
  sorts Nat LNat Nat? State WProcs Procs .
  subsorts Nat LNat < Nat? .  subsort WProcs < Procs .
  op 0 : -> Nat .
  op s : Nat -> Nat .
  op [_] : Nat -> LNat .            *** number-locking operator
  op < wait,_> : Nat -> WProcs .
  op < crit,_> : Nat -> Procs .
  op mt : -> WProcs .               *** empty multiset
  op __ : Procs Procs -> Procs [assoc comm id: mt] .    *** union
  op __ : WProcs WProcs -> WProcs [assoc comm id: mt] . *** union
  op _|_|_ : Nat Nat? Procs -> State .
  vars n m i j k : Nat . var x? : Nat? . var PS : Procs . var WPS : WProcs .

  rl [new]:  m | n | PS => s(m) | n | < wait,m > PS [narrowing] .
  rl [enter]: m | n | < wait,n > PS => m | [n] | < crit,n > PS [narrowing] .
  rl [leave]: m | [n] | < crit,n > PS => m | s(n) | PS [narrowing] .
endm
```

The states of BAKERY have the form "m | x? | PS" with m the ticket-dispensing counter, x? the (possibly locked) counter to access the critical section, and PS a multiset of processes either waiting or in the critical section. BAKERY is infinite-state: [new] creates new processes, and the counters can grow unboundedly. When a waiting process enters the critical section with [enter], the second counter n is locked as [n]; and it is unlocked and incremented when it leaves it with [leave]. The key invariant is *mutual exclusion*. Note that the term "i | x? | < crit, j > < crit, k > PS" describes all states in the *complement* of mutual exclusion states. Without the fold option, narrowing does not terminate, but with the following command we can verify that BAKERY satisfies mutual exclusion, not just for the initial state "0 | 0 | mt", but for the much more general infinite set of initial states with waiting processes only "m | n | WPS".

```
Maude> {fold} vu-narrow {filter,delay}
        m | n | WPS =>* i | x? | < crit, j > < crit, k > PS .

No solution.
rewrites: 4 in 1ms cpu (1ms real) (2677 rewrites/second)
```

The new vu-narrow {filter,delay} command can achieve dramatic state space reductions over the previous vu-narrow command by filtering $E \cup B$-unifiers. This is illustrated by a simple cryptographic protocol example in [10, §15] exploiting the unitary nature of unification in the exclusive-or theory [24].

5 Applications and Conclusion

Maude can be used as a meta-tool to develop new formal tools because: (i) its underlying equational and rewriting logics are logical —and reflective meta-logical— frameworks [7,27,46]; (ii) Maude's efficient support of logical reflection through its META-LEVEL module; (iii) Maude's rewriting, search, model checking, and strategy language features [11,15]; and (iv) Maude's symbolic reasoning features [15,33], the latest reported here. We refer to [11,15,31,33] for references on various Maude-based tools. Many of them can benefit from these new features.

By way of example we mention some areas ready to reap such benefits: (1) *Formal Analysis of Cryptographic Protocols*. The new features can yield substantial improvements to tools such as Maude-NPA [19], Tamarin [28] and AKISS [8]. (2) *Model Checking of Infinite-State Systems*. The narrowing-based LTL symbolic model checker reported in [6,20], and the addition of new symbolic capabilities to Real-Time Maude [37,38] can both benefit from the new features. (3) *SMT Solving*. In Sect. 3 we noted that FVP $E \cup B$-unification makes satisfiability of positive QF formulas in $T_{\Sigma/E \cup B}$ decidable. Under mild conditions, this has been extended in [32,44] to a procedure for satisfiability in $T_{\Sigma/E \cup B}$ of all QF formulas which will also benefit from the new features. (4) *Theorem Proving*. The new Maude Inductive Theorem Prover under construction [34], as well as

Maude's Invariant Analyzer [43] and Reachability Logic Theorem Prover [45] all use equational unification and narrowing modulo equations; so all will benefit from the new features. (5) *Theory Transformations* based on equational unification, e.g., partial evaluation [4], ground confluence methods [17] or program termination methods [25,26] could likewise become more efficient.

In conclusion, we have presented and illustrated with examples new equational unification and matching, and symbolic reachability analysis features in Maude 3.2. Thanks to the above-mentioned properties (i)–(iv) of Maude as a meta-tool, we hope that this work will encourage other researchers to use Maude and its symbolic features to develop new tools in many different logics.

References

1. Aït-Kaci, H., Sasaki, Y.: An axiomatic approach to feature term generalization. In: De Raedt, L., Flach, P. (eds.) ECML 2001. LNCS (LNAI), vol. 2167, pp. 1–12. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44795-4_1
2. Alpuente, M., Ballis, D., Cuenca-Ortega, A., Escobar, S., Meseguer, J.: ACUOS^2: a high-performance system for modular ACU generalization with subtyping and inheritance. In: Calimeri, F., Leone, N., Manna, M. (eds.) JELIA 2019. LNCS (LNAI), vol. 11468, pp. 171–181. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19570-0_11
3. Alpuente, M., Cuenca-Ortega, A., Escobar, S., Meseguer, J.: Order-sorted homeomorphic embedding modulo combinations of associativity and/or commutativity axioms. Fundam. Inform. **177**(3–4), 297–329 (2020)
4. Alpuente, M., Cuenca-Ortega, A., Escobar, S., Meseguer, J.: A partial evaluation framework for order-sorted equational programs modulo axioms. J. Log. Algebraic Methods Program. **110**, 100501 (2020)
5. Alpuente, M., Falaschi, M., Vidal, G.: Partial evaluation of functional logic programs. ACM Trans. Program. Lang. Syst. **20**(4), 768–844 (1998)
6. Bae, K., Escobar, S., Meseguer, J.: Abstract logical model checking of infinite-state systems using narrowing. In: RTA 2013. LIPIcs, vol. 21, pp. 81–96. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2013)
7. Basin, D., Clavel, M., Meseguer, J.: Rewriting logic as a metalogical framework. ACM Trans. Comput. Log. **5**, 528–576 (2004)
8. Chadha, R., Cheval, V., Ciobâcă, Ş, Kremer, S.: Automated verification of equivalence properties of cryptographic protocols. ACM Trans. Comput. Log. **17**(4), 23:1–23:32 (2016)
9. Cholewa, A., Meseguer, J., Escobar, S.: Variants of variants and the finite variant property. Technical report, CS Dept. University of Illinois at Urbana-Champaign, February 2014. http://hdl.handle.net/2142/47117
10. Clavel, M., et al.: Maude manual (version 3.2.1). SRI International, February 2022. http://maude.cs.illinois.edu
11. Clavel, M., et al.: All About Maude, A High-Performance Logical Framework. Lecture Notes in Computer Science, vol. 4350. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71999-1
12. Colmerauer, A.: An introduction to Prolog III. Commun. ACM **33**(7), 69–90 (1990)
13. Comon-Lundh, H., Delaune, S.: The finite variant property: how to get rid of some algebraic properties. In: Giesl, J. (ed.) RTA 2005. LNCS, vol. 3467, pp. 294–307. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-32033-3_22

14. Dershowitz, N., Jouannaud, J.-P.: Rewrite systems. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, pp. 243–320. North-Holland (1990)
15. Durán, F., et al.: Programming and symbolic computation in Maude. J. Log. Algebraic Methods Program. **110**, 100497 (2020)
16. Durán, F., Eker, S., Escobar, S., Martí-Oliet, N., Meseguer, J., Talcott, C.: Associative unification and symbolic reasoning modulo associativity in Maude. In: Rusu, V. (ed.) WRLA 2018. LNCS, vol. 11152, pp. 98–114. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99840-4_6
17. Durán, F., Meseguer, J., Rocha, C.: Ground confluence of order-sorted conditional specifications modulo axioms. J. Log. Algebraic Methods Program. **111**, 100513 (2020)
18. Eker, S.: Associative unification in Maude. J. Log. Algebraic Methods Program. **126**, 100747 (2022)
19. Escobar, S., Meadows, C., Meseguer, J.: Maude-NPA: cryptographic protocol analysis modulo equational properties. In: Aldini, A., Barthe, G., Gorrieri, R. (eds.) FOSAD 2007-2009. LNCS, vol. 5705, pp. 1–50. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03829-7_1
20. Escobar, S., Meseguer, J.: Symbolic model checking of infinite-state systems using narrowing. In: Baader, F. (ed.) RTA 2007. LNCS, vol. 4533, pp. 153–168. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73449-9_13
21. Escobar, S., Sasse, R., Meseguer, J.: Folding variant narrowing and optimal variant termination. J. Algebraic Log. Program. **81**, 898–928 (2012)
22. Goguen, J., Meseguer, J.: Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations. Theoret. Comput. Sci. **105**, 217–273 (1992)
23. Jaffar, J., Maher, M.J.: Constraint logic programming: a survey. J. Log. Program. **19**(20), 503–581 (1994)
24. Kapur, D., Narendran, P.: Matching, unification and complexity. SIGSAM Bull. **21**(4), 6–9 (1987)
25. Lucas, S., Meseguer, J., Gutiérrez, R.: The 2D dependency pair framework for conditional rewrite systems. Part I: definition and basic processors. J. Comput. Syst. Sci. **96**, 74–106 (2018)
26. Lucas, S., Meseguer, J., Gutiérrez, R.: The 2D dependency pair framework for conditional rewrite systems - Part II: advanced processors and implementation techniques. J. Autom. Reason. **64**(8), 1611–1662 (2020)
27. Martí-Oliet, N., Meseguer, J.: Rewriting logic as a logical and semantic framework. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, 2nd. Edition, pages 1–87. Kluwer Academic Publishers (2002). First published as SRI Technical report SRI-CSL-93-05, August 1993
28. Meier, S., Schmidt, B., Cremers, C., Basin, D.: The TAMARIN prover for the symbolic analysis of security protocols. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 696–701. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_48
29. Meseguer, J.: Membership algebra as a logical framework for equational specification. In: Presicce, F.P. (ed.) WADT 1997. LNCS, vol. 1376, pp. 18–61. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-64299-4_26
30. Meseguer, J.: Strict coherence of conditional rewriting modulo axioms. Theor. Comput. Sci. **672**, 1–35 (2017)

31. Meseguer, J.: Symbolic reasoning methods in rewriting logic and Maude. In: Moss, L.S., de Queiroz, R., Martinez, M. (eds.) WoLLIC 2018. LNCS, vol. 10944, pp. 25–60. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-57669-4_2
32. Meseguer, J.: Variant-based satisfiability in initial algebras. Sci. Comput. Program. **154**, 3–41 (2018)
33. Meseguer, J.: Symbolic computation in Maude: some tapas. In: LOPSTR 2020. LNCS, vol. 12561, pp. 3–36. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-68446-4_1
34. Meseguer, J., Skeirik, S.: Inductive reasoning with equality predicates, contextual rewriting and variant-based simplification. In: Escobar, S., Martí-Oliet, N. (eds.) WRLA 2020. LNCS, vol. 12328, pp. 114–135. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-63595-4_7
35. Meseguer, J., Thati, P.: Symbolic reachability analysis using narrowing and its application to verification of cryptographic protocols. High.-Order Symb. Comput. **20**(1–2), 123–160 (2007)
36. Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning (in 2 volumes), pp. 371–443. Elsevier and MIT Press (2001)
37. Ölveczky, P.C.: Real-time Maude and its applications. In: Escobar, S. (ed.) WRLA 2014. LNCS, vol. 8663, pp. 42–79. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12904-4_3
38. Ölveczky, P.C., Meseguer, J.: Semantics and pragmatics of real-time Maude. High.-Order Symb. Comput. **20**(1–2), 161–196 (2007)
39. Peterson, G.E., Stickel, M.E.: Complete sets of reductions for some equational theories. J. Assoc. Comput. Mach. **28**(2), 233–264 (1981)
40. Plotkin, G.: Building-in equational theories. In: Meltzer, B., Michie, D. (eds.) 1971 Proceedings of the Seventh Annual Machine Intelligence Workshop on Machine Intelligence 7, Edinburgh, pp. 73–90. Edinburgh University Press (1972)
41. Robinson, J.A.: A machine-oriented logic based on the resolution principle. J. Assoc. Comput. Mach. **12**(1), 23–41 (1965)
42. Rocha, C., Meseguer, J.: Five isomorphic Boolean theories and four equational decision procedures. Technical report UIUCDCS-R-2007-2818, CS Department, University of Illinois at Urbana-Champaign, February 2007. http://hdl.handle.net/2142/11295
43. Rocha, C., Meseguer, J.: Proving safety properties of rewrite theories. In: Corradini, A., Klin, B., Cîrstea, C. (eds.) CALCO 2011. LNCS, vol. 6859, pp. 314–328. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22944-2_22
44. Skeirik, S., Meseguer, J.: Metalevel algorithms for variant satisfiability. J. Log. Algebr. Meth. Program. **96**, 81–110 (2018)
45. Skeirik, S., Stefanescu, A., Meseguer, J.: A constructor-based reachability logic for rewrite theories. Fundam. Inform. **173**(4), 315–382 (2020)
46. Stehr, M.-O., Meseguer, J.: Pure type systems in rewriting logic: specifying typed higher-order languages in a first-order logical framework. In: Owe, O., Krogdahl, S., Lyche, T. (eds.) From Object-Orientation to Formal Methods. LNCS, vol. 2635, pp. 334–375. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-39993-3_16
47. Walther, C.: A mechanical solution of Schubert's steamroller by many-sorted resolution. Artif. Intell. **26**(2), 217–224 (1985)
48. Zheng, Y., et al.: Z3str2: an efficient solver for strings, regular expressions, and length constraints. Formal Methods Syst. Design **50**(2–3), 249–288 (2017)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.