Narrowing-based Optimization of Rewrite Theories[⋆]

Technical Report DSIC-UPV, 2020

M. Alpuente[1], D. Ballis[2], S. Escobar[1], J. Meseguer[3], and J. Sapiña[1]

[1] VRAIN, Universitat Politècnica de València, Valencia, Spain
{alpuente,sescobar,sapina}@upv.es
[2] DMIF, Università degli Studi di Udine, Udine, Italy
demis.ballis@uniud.it
[3] University of Illinois at Urbana-Champaign, Urbana, IL, USA
meseguer@illinois.edu

Abstract. Partial evaluation has been never investigated in the context of rewrite theories that allow concurrent systems to be specified by means of rules, with an underlying equational theory being used to model system states as terms of an algebraic data type. In this paper, we develop a symbolic, narrowing-driven partial evaluation framework for rewrite theories that supports sorts, subsort overloading, rules, equations, and algebraic axioms. Our partial evaluation scheme allows a rewrite theory to be optimized by specializing the plugged equational theory with respect to the rewrite rules that define the system dynamics. This can be particularly useful for automatically optimizing rewrite theories that contain overly general equational theories which perform unnecessary computations involving matching modulo axioms, because some of the axioms may be blown away after the transformation. The specialization is done by using appropriate unfolding and abstraction algorithms that achieve significant specialization while ensuring the correctness and termination of the specialization. Our preliminary results demonstrate that our transformation can speed up a number of benchmarks that are difficult to optimize otherwise.

1 Introduction

Rewriting Logic (RWL) is a logic of change that extends order-sorted equational logic by adding rewrite rules that are used to describe non-deterministic transitions of concurrent systems. Rewriting Logic is efficiently implemented in the high-performance system Maude [9]. Roughly speaking, a rewrite theory seamlessly combines a *term rewriting system* (TRS), which specifies the system dynamics, with an *equational theory* that defines the static structure of the system states. The equational theory may contain equations and axioms (i.e., distinguished equations that specify algebraic laws such as commutativity, associativity, and unity for some theory operators) so that rewrite steps are performed *modulo* the equations and axioms.

[⋆] This work has been partially supported by the EU (FEDER) and the Spanish MCIU under grant RTI2018-094403-B-C32, and by Generalitat Valenciana under grant PROMETEO/2019/098. Julia Sapiña has been supported by the Generalitat Valenciana APOSTD/2019/127 grant.

Partial evaluation (PE) is a program optimization technique (also known as program specialization) that, given a program and some of its input data, produces a residual or specialized program. Running the residual program on the remaining data is generally faster and yields the same result as running the original program on all of its input data [17]. PE has been widely applied to a variety of programs, including functional programs (FP) [17] and logic programs (LP) [19], where it is usually called *partial deduction* (PD). The *Equational Narrowing-driven Partial Evaluation* (EQNPE) scheme of [1] extends PD to the specialization of order-sorted equational theories with respect to a set of input terms. The input equational theory $(\Sigma, E \uplus B)$ consists of a set E of convergent equations (that in the order-sorted setting means they are confluent and terminating, among other requirements) that are implicitly oriented from left to right as rewrite rules (and operationally used as simplification rules), and a set B of commonly occurring axioms such associativity, commutativity, and identity that are essentially used for pattern matching modulo axioms. Thanks to the use of (a form of) *narrowing*, the symbolic mechanism that extends term rewriting by replacing pattern matching with unification [29], the achieved transformation is strictly more powerful than the PE of both logic programs and functional programs [4]

In the following we consider a rewrite theory $\mathscr{R} = (\Sigma, E \uplus B, R)$ that extends the order-sorted equational theory $\mathscr{E} = (\Sigma, E \uplus B)$ with a set R of rewrite rules that specify concurrent system transitions. Rewrite theories can not only be executed by equational rewriting in Maude but also *symbolically* executed by narrowing at *two levels*: (i) narrowing with the (typically non-confluent and non-terminating) rules of R modulo $(E \uplus B)$, and (ii) narrowing with oriented equations $\vec{E}$ (the explicitly oriented version of the equations in E) modulo the axioms B. They both have practical applications: (i) narrowing with R modulo $(E \uplus B)$ is useful for solving *reachability goals* and *logical model checking* [26], and (ii) narrowing with $\vec{E}$ modulo B (or $(\vec{E},B)$-narrowing) is useful for equational unification and variant computation [14]. Both levels of narrowing should meet some conditions: (i) narrowing with R modulo $(E \uplus B)$ is performed in a "topmost" way (i.e., the rules in R rewrite the global system state) and there must be a finitary unification algorithm for $(E \uplus B)$, and (ii) narrowing with $\vec{E}$ modulo B requires that B is a theory with a finitary unification algorithm[4] and that $\vec{E}$ is convergent. When $(\Sigma, E \uplus B)$ additionally has the property that a finite complete set of most general $(\vec{E},B)$-*variants*[5] exists for each term, known as the *finite variant property* (FVP), $\mathscr{E}$-unification is finitary and *topmost* narrowing with R modulo the equations and axioms can be effectively performed.

For $(\vec{E},B)$-variant computation and (variant-based) $\mathscr{E}$-unification, the *folding variant narrowing (FVN) strategy* of [14] is used. The main idea of folding variant narrowing is to "fold"[6] , by subsumption modulo B, the $(\vec{E},B)$-narrowing tree that can in prac-

[4] This happens for B consisting of modular combinations of associativity, commutativity, and/or identity axioms for different theory operators.

[5] A variant of a term t consists of a pair (t', σ), where t' is the $(\vec{E},B)$-irreducible form of $t\sigma$ for a substitution σ.

[6] The notion of folding in folding variant narrowing is essentially a *subsumption* notion applied to some leaves of the narrowing tree so that less general leaves are subsumed (folded into) most general ones. Therefore, this notion is quite different from the classical folding operation

tice result in a finite, directed acyclic narrowing graph that symbolically and concisely summarizes the (generally infinite) narrowing search space of the theory. Nevertheless, finiteness of folding variant narrowing trees is only guaranteed for equational theories that satisfy the finite variant property.

Because both reachability goals and logical model checking generally require the whole search space of rewrite theories to be analyzed (i.e. all system states and transitions), the opportunities for optimizing rewrite theories by partial evaluation may appear to be scarce. Actually, partial evaluation typically removes some computation states by performing as much program computation as possible hence contracting the search space because some transitions are removed. The key idea in this paper for the specialization of a rewrite theory $\mathscr{R}$ is to partially evaluate the underlying equational theory $\mathscr{E}$ with respect to the function calls in the rules of $\mathscr{R}$ in such a way that $\mathscr{E}$ gets rid of any unneeded overgenerality. By this means, only the functional computations in $\mathscr{E}$ are compressed by partial evaluation, while keeping the concurrent computations with the rules of R. Depending on the properties of both, $\mathscr{E}$ and $\mathscr{R}$, the right unfolding and abstraction operators must be chosen to achieve the biggest optimization possible while ensuring termination and total correctness of the transformation. Moreover, in many cases we transform a rewrite theory whose operators obey structural, algebraic axioms such as associativity, commutativity, and unity, into a much simpler rewrite theory whose operators obey no axioms. This makes it possible to run such theories into an independent rewriting infrastructure that does not support rewriting modulo axioms. Furthermore, some costly analyses that may require significant (or even unaffordable) resources, both in time and space, can be now effectively performed after the transformation. This includes the analysis of cryptographic communication protocols that are currently handled by some ad-hoc combination of separate techniques and that can be recast as distinct instances of our generic partial evaluation scheme. See [1] for extra references on narrowing-driven partial evaluation.

After some brief preliminaries in Section 2, we sketch the specialization algorithm for rewrite theories in Section 3, which works in two phases: partial evaluation and compression refactoring. In Section 4.1, we provide a suitable unfolding operator dealing with theories that do not meet the finite variant property. Then, we discuss some preliminary experiments that show the optimization capability of our technique and we conclude.

2 Preliminaries

Let us recall some key concepts of rewriting logic theories [20] and equational unification [5]. We consider an order-sorted signature[7] $\Sigma{=}(\Sigma, \mathsf{S}, \leq)$ that consists of a poset of sorts $(\mathsf{S}, \leq)$ and an $\mathsf{S}^* \times \mathsf{S}$-indexed family of sets $\Sigma = \{\Sigma_{\mathsf{s}_1 \ldots \mathsf{s}_n, \mathsf{s}}\}_{(\mathsf{s}_1 \ldots \mathsf{s}_n, \mathsf{s}) \in \mathsf{S}^* \times \mathsf{S}}$

of Burstall and Darlington's fold/unfold transformation scheme [7], where unfolding is essentially the replacement of a call by its body, with appropriate substitutions, and folding is the inverse transformation, i.e., the replacement of some piece of code by an equivalent function call.

[7] This abuse of language of using Σ for both the triplet and the ranked set of function symbols is useful, and is in fact the notation we use later.

of function symbols. The poset $(\mathsf{S},\leq)$ of sorts for Σ is partitioned into equivalence classes $C_1,\ldots,C_n$ (called *connected components*) by the equivalence relation $(\leq\cup\geq)^+$. Throughout this paper, Σ is assumed to be *preregular*, so each term t has a least sort under $\leq$, denoted $ls(t)$ (see [16]). Σ is also assumed to be *kind-complete* [23], that is, for each sort $\mathsf{s}\in\mathsf{S}$, its connected component in the poset $(\mathsf{S},\leq)$ has a top sort under $\leq$, denoted $[\mathsf{s}]$ and called the connected component's *kind*, and for each function symbol $f\in\Sigma_{\mathsf{s}_1\ldots\mathsf{s}_n,\mathsf{s}}$, there is also an $f\in\Sigma_{[\mathsf{s}_1]\ldots[\mathsf{s}_n],[\mathsf{s}]}$. An order-sorted signature can always be extended to be kind-complete [23]. Maude automatically checks preregularity and adds a new "kind" sort $[\mathsf{s}]$ at the top of the connected component of each sort $\mathsf{s}\in\mathsf{S}$ specified by the user and automatically lifts each operator to the kind level. For technical reasons, it is useful to assume that Σ has no ad-hoc overloading[8]. However, this assumption entails no real loss of generality: any Σ can be transformed into a semantically equivalent signature with no ad-hoc overloading (by symbol renaming). Note that avoiding ad-hoc overloading ensures that Σ is *sensible*, in the sense that for any two typings $f:\mathsf{s}_1\ldots\mathsf{s}_n\longrightarrow\mathsf{s}$ and $f:\mathsf{s}'_1\ldots\mathsf{s}'_n\longrightarrow\mathsf{s}'$ of a n-ary function symbol f, if s_i and s'_i are in the same connected component of $(\mathsf{S},\leq)$ for $1\leq i\leq n$, then s and s' are also in the same connected component; this provides the right notion of *unambiguous* signature at the order-sorted level.

We assume an S-sorted family $\mathscr{X}=\{\mathscr{X}_{\mathsf{s}}\}_{\mathsf{s}\in\mathsf{S}}$ of disjoint variable sets. $\mathscr{T}_{\Sigma,\mathsf{s}}(\mathscr{X})$ and $\mathscr{T}_{\Sigma,\mathsf{s}}$ denote the sets of terms and ground terms of sort s, respectively. Note that $\mathsf{s}<\mathsf{s}'$ (s is a subsort of s') implies the set of terms of sort s are a subset of the set of terms of sort s', i.e., $\mathscr{T}_{\Sigma,\mathsf{s}}(\mathscr{X})\subseteq\mathscr{T}_{\Sigma,\mathsf{s}'}(\mathscr{X})$. We (ambiguously) write $\mathscr{T}_\Sigma(\mathscr{X})$ and $\mathscr{T}_\Sigma$ for *both* the corresponding term algebras and for the underlying sets of terms, i.e., $\mathscr{T}_\Sigma(\mathscr{X})=\cup_{\mathsf{s}\in\mathsf{S}}\mathscr{T}_{\Sigma,\mathsf{s}}(\mathscr{X})$ and $\mathscr{T}_\Sigma=\cup_{\mathsf{s}\in\mathsf{S}}\mathscr{T}_{\Sigma,\mathsf{s}}$. Throughout this paper we assume that $\mathscr{T}_{\Sigma,\mathsf{s}}\neq\emptyset$ for every sort s because this affords a simpler deduction system. The set of variables occurring in a term t is denoted by $Var(t)$. A ground term is a term without variables. In order to simplify the presentation, we often disregard sorts when no confusion can arise.

A *position* p in a term t is represented by a sequence of natural numbers (Λ denotes the empty sequence, i.e., the root position). The top symbol of t is denoted by $root(t)$. Positions are ordered by the *prefix* ordering: $p\leq q$, if $\exists w$ such that $p.w=q$. Given a term t, we let $Pos(t)$ and $Pos_\Sigma(t)$ respectively denote the set of positions and the set of non-variable positions of t (i.e., positions where a variable does not occur). The expression $t_{|p}$ denotes the *subterm* of t at position p, and $t[u]_p$ denotes the result of *replacing the subterm* $t_{|p}$ by the term u at position p.

A *substitution* σ is a sorted mapping from a finite subset of $\mathscr{X}$ to $\mathscr{T}_\Sigma(\mathscr{X})$. Substitutions are written as $\sigma=\{X_1\mapsto t_1,\ldots,X_n\mapsto t_n\}$ where the domain of σ is $Dom(\sigma)=\{X_1,\ldots,X_n\}$ and the set of variables introduced by terms $t_1,\ldots,t_n$ is written $Ran(\sigma)$. The identity substitution is denoted id. Substitutions are homomorphically extended to $\mathscr{T}_\Sigma(\mathscr{X})$. The application of a substitution σ to a term t is denoted by $t\sigma$. The restriction of σ to a set of variables $V\subset\mathscr{X}$ is denoted $\sigma_{|V}$; sometimes we write $\sigma_{|t_1,\ldots,t_n}$ to denote

[8] Given the overloaded operator $f:\mathsf{s}_1\ldots\mathsf{s}_m\longrightarrow\mathsf{s}_0$ and $f:\mathsf{s}'_1\ldots\mathsf{s}'_n\longrightarrow\mathsf{s}'_0$, subsort overloading means that $m=n$ and, for all i, $0\leq i\leq n$, s_i and s'_i belong to the same connected component. Otherwise, the overloading of f is called ad-hoc.

$\sigma_{|V}$ where $V = Var(t_1) \cup \cdots \cup Var(t_n)$. Composition of two substitutions is denoted by $\sigma\sigma'$ so that $t(\sigma\sigma') = (t\sigma)\sigma'$.

A Σ-*equation* is an unoriented pair $t = t'$, where $t,t' \in \mathscr{T}_{\Sigma,\mathsf{s}}(\mathscr{X})$ for some sort $\mathsf{s} \in \mathsf{S}$. Given Σ and a set $\mathscr{E}$ of Σ-equations, order-sorted equational logic induces a congruence relation $=_{\mathscr{E}}$ on $\mathscr{T}_{\Sigma}(\mathscr{X})$ (see [21]). An *equational theory* is a pair $(\Sigma,\mathscr{E})$, with Σ an order-sorted signature and $\mathscr{E}$ a set of Σ-equations. We often omit Σ when no confusion can arise.

A term t is more (or equally) general than t' modulo $\mathscr{E}$, denoted by $t \leq_{\mathscr{E}} t'$, if there is a substitution γ such that $t' =_{\mathscr{E}} t\gamma$. A substitution θ is more (or equally) general than σ modulo $\mathscr{E}$, denoted by $\theta \leq_{\mathscr{E}} \sigma$, if there is a substitution γ such that $\sigma =_{\mathscr{E}} \theta\gamma$, i.e., for all $x \in \mathscr{X}, x\sigma =_{\mathscr{E}} x\theta\gamma$. Also, $\theta \leq_{\mathscr{E}} \sigma\ [V]$ iff there is a substitution γ such that, for all $x \in V, x\sigma =_{\mathscr{E}} x\theta\gamma$. We also define $t \simeq_{\mathscr{E}} t'$ iff $t \leq_{\mathscr{E}} t'$ and $t' \leq_{\mathscr{E}} t$; and similarly $\theta \simeq_{\mathscr{E}} \sigma$.

An *$\mathscr{E}$-unifier* for a Σ-equation $t = t'$ is a substitution σ such that $t\sigma =_{\mathscr{E}} t'\sigma$. $CSU_{\mathscr{E}}(t = t')$ denotes a *complete* set of unifiers for the equation $t = t'$ modulo $\mathscr{E}$ so that any $\mathscr{E}$-unifier is an $\mathscr{E}$-instance of one of them. An $\mathscr{E}$-unification algorithm is *complete* if for any equation $t = t'$ it generates a complete set of $\mathscr{E}$-unifiers. Note that this set needs not be finite. A unification algorithm is said to be *finitary* if it always terminates. Note that a complete and finitary $\mathscr{E}$-unification algorithm may not exist even if a complete and finite set of $\mathscr{E}$-unifiers exists [5].

A *rewrite theory* is a triple $\mathscr{R} = (\Sigma,\mathscr{E},R)$, where $(\Sigma,\mathscr{E})$ is the equational theory modulo which we rewrite and R is a set of rewrite rules. Rules are of the form $l \Rightarrow r$ where terms $l,r \in \mathscr{T}_{\Sigma,\mathsf{s}}(\mathscr{X})$ for some sort s and $Var(r) \subseteq Var(l)$. The terms l and r are respectively called the *left-hand side* (or *lhs*) and the *right-hand side* (or *rhs*) of the rule. The set R of rules is required to be *sort-decreasing*, i.e., for each $l \Rightarrow r$ in R, and for each well-sorted substitution σ, $ls(l\sigma) \geq ls(r\sigma)$.

Let $\rightarrow \subseteq A \times A$ be a binary relation on a set A. We denote its transitive closure by $\rightarrow^+$, and its reflexive and transitive closure by $\rightarrow^*$. We define the *one-step rewrite relation* on $\mathscr{T}_{\Sigma}(\mathscr{X})$ for the set of rules R as follows: $t \rightarrow_R t'$ if there is a position $p \in Pos(t)$, a rule $l \Rightarrow r$ in R, and a substitution σ such that $t_{|p} = l\sigma$ and $t' = t[r\sigma]_p$. The relation $\rightarrow_{R/\mathscr{E}}$ for rewriting modulo $\mathscr{E}$ is defined as $=_{\mathscr{E}} \circ \rightarrow_R \circ =_{\mathscr{E}}$. A term t is called *$R/\mathscr{E}$-irreducible* iff there is no term u such that $t \rightarrow_{R/\mathscr{E}} u$. A substitution σ is $R/\mathscr{E}$-irreducible if, for every $x \in \mathscr{X}$, $x\sigma$ is $R/\mathscr{E}$-irreducible. We say that the relation $\rightarrow_{R/\mathscr{E}}$ is *terminating* if there is no infinite sequence $t_1 \rightarrow_{R/\mathscr{E}} t_2 \rightarrow_{R/\mathscr{E}} \cdots t_n \rightarrow_{R/\mathscr{E}} t_{n+1} \cdots$. We say that the relation $\rightarrow_{R/\mathscr{E}}$ is *confluent* if whenever $t \rightarrow^*_{R/\mathscr{E}} t'$ and $t \rightarrow^*_{R/\mathscr{E}} t''$, there exists a term t''' such that $t' \rightarrow^*_{R/\mathscr{E}} t'''$ and $t'' \rightarrow^*_{R/\mathscr{E}} t'''$. A rewrite theory $(\Sigma,\mathscr{E},R)$ is *convergent* if R is sort-decreasing and the relation $\rightarrow_{R/\mathscr{E}}$ is confluent and terminating. In a convergent order-sorted rewrite theory, for each term $t \in \mathscr{T}_{\Sigma}(\mathscr{X})$, there is a unique (up to $\mathscr{E}$-equivalence) $R/\mathscr{E}$-irreducible term t' that can be obtained by rewriting t to $R/\mathscr{E}$-irreducible or *normal* form, which is denoted by $t \rightarrow^!_{R/\mathscr{E}} t'$, or simply $t{\downarrow_{R/\mathscr{E}}}$ when t' is not relevant. For each $x \in Dom(\sigma)$, $\sigma{\downarrow_{R/\mathscr{E}}}$ is defined as $(\sigma{\downarrow_{R/\mathscr{E}}})(x) = \sigma(x){\downarrow_{R/\mathscr{E}}}$. A substitution σ is $R/\mathscr{E}$-irreducible (normalized) iff $x\sigma$ is so for each $x \in Dom(\sigma)$. For a set Q of terms, we denote by $Q{\downarrow_{R/\mathscr{E}}}$ the set of normal forms of the terms in Q.

Since $\mathscr{E}$-congruence classes can be infinite, $\rightarrow_{R/\mathscr{E}}$-reducibility is undecidable in general. Therefore, $R/\mathscr{E}$-rewriting is usually implemented [18] by R,$\mathscr{E}$-rewriting. We define the relation $\rightarrow_{R,\mathscr{E}}$ on $\mathscr{T}_{\Sigma}(\mathscr{X})$ by $t \rightarrow_{p,R,\mathscr{E}} t'$ (or simply $t \rightarrow_{R,\mathscr{E}} t'$) iff there is a non-variable position $p \in Pos_{\Sigma}(t)$, a rule $l \Rightarrow r$ in R, and a substitution σ such that $t_{|p} =_{\mathscr{E}} l\sigma$ and $t' = t[r\sigma]_p$. To ensure completeness of R,$\mathscr{E}$-rewriting w.r.t. $R/\mathscr{E}$-rewriting, we require *strict coherence*, ensuring that $=_{\mathscr{E}}$ is a bisimulation for $R,\mathscr{E}$-rewriting [24]: for any Σ-terms u, u', v if $u =_{\mathscr{E}} u'$ and $u \to_{R,\mathscr{E}} v$, then there exists a term v' such that $u' \to_{R,\mathscr{E}} v'$ and $v =_{\mathscr{E}} v'$. Note that, assuming $\mathscr{E}$-matching is decidable, $\to_{R,\mathscr{E}}$ is decidable and notions such as confluence, termination, irreducible term, and normalized substitution are defined for $\to_{R,\mathscr{E}}$ straightforwardly [24]. It is worth noting that Maude automatically provides strict $\mathscr{E}$-coherence completion for rules and equations for any combination of associativity and/or commutativity and/or identity axioms [27]. That is, the specified rules and equations are automatically completed to be coherent with no need for user intervention.

Besides the standard assumptions on $\mathscr{R}$ mentioned before, we consider the classical restriction that the set R of rules is coherent w.r.t. $\mathscr{E}$ (intuitively, this ensures that a rewrite step with $\mathscr{R}$ can always be postponed in favor of deterministically rewriting with $\mathscr{E}$).

2.1 Modeling Concurrent Systems as Rewrite Theories

Rewrite theories provide a natural computation model for concurrent systems as shown in the following example.

Example 1. Let us consider a rewrite theory $\mathscr{R} = (\Sigma, E \uplus B, R)$ that encodes a close variant of the handshake network protocol of [22]. The theory models an environment where several clients and servers coexist. The signature Σ includes several operators and sorts that model protocol entities. Names of the sorts are self-explanatory: for example, servers are typed with sort `Serv`, clients with sort `Cli`, and messages with sort `Message`.

Messages are encoded as non-empty, associative, sequences $\mathtt{s}_1 \ldots \mathtt{s}_n$ where, for the sake of simplicity, each $\mathtt{s}_i$ is a term of sort `Symbol` in the alphabet `{a,b,c}`. We assume that `Symbol < Message`, hence any symbol is also a (one-symbol) message. Clients are represented as 5-tuples of the form `[C,S,Q,K,V]` of sort `Cli`, where `C` is the client's name, `S` is the name of the server `C` wants to communicate with, `Q` is a message encoding a client handshake request, `K` is a natural number (specified in Peano's notation) that determines an encryption/decryption key for messages, and `V` is a constant value that models the server handshake status. Initially, the status is set to the empty value `mt`, and it changes to `success` whenever the handshaking process succeeded. Servers are simply modeled by means of pairs of the form `[S,K]` of sort `Serv`, where `S` is a server name, and `K` is an encryption/decryption key. All network packets are represented as pairs of the form `Host <- CNT` of sort `Packet`, where `Host` is a client or server recipient, and `CNT` specifies the packet content. Specifically, `CNT` is a term `{H,M}`, with `H` being the sender's name and `M` being a message that represents either a client handshake request or a server response. System states are formalized as multisets $\mathtt{t}_1$ `&...&` $\mathtt{t}_m$ of clients, servers, and network packets via the ACU operator `_&_` whose unity element is the constant `null`. The protocol dynamics is specified by the following three rules that implements a handshake protocol where clients and servers agree on a shared key `K`.

```
rl [req] : [C,S,Q,K,mt] => (S <- {C,enc(Q,K)}) & [C,S,Q,K,mt] .
rl [reply] : (S <- {C,M}) & [S,K] => (C <- {S,dec(M,K)}) & [S,K] .
rl [rec] : (C <- {S,Q}) & [C,S,Q,K,mt] => [C,S,Q,K,success] .
```

More specifically, the rule `req` allows a client `C` to start a handshake request with a sever `S` by sending an encrypted message `enc(Q,K)` to `S` so that the message `Q` is encrypted by using the client's key `K`. The rule `reply` lets the server `S` consume a client handshake request packet `S <- {C,M}` by first decrypting the incoming message `M` with the server key and then sending a response packet back to `C` that includes the decrypted request message. The rule `rec` successfully completes the handshake between `C` and `S` whenever the server response packet `C <- {S,Q}` includes a message `Q` which is equal to the initial client request message. In this case, the status of the client is changed from `mt` to `success`. Note that the handshake succeeds when the client and server use the same key `K`.

Encryption and decryption capabilities are implemented by two functions (namely, `enc(M,K)` and `dec(M,K)`) that are specified by the equational theory $\mathscr{E}$ in $\mathscr{R}$. The equational theory $\mathscr{E}$ implements a *Caesar* cipher with key `K`, which is a simple substitution cipher where each symbol in the plaintext message is replaced by the symbol `K` positions forward in the alphabet. The cipher is circular, i.e., it works modulo the cardinality of the alphabet. For instance, `enc(a b,s(0))` would deliver `(b c)`, and `dec(a b,s(0))` would be the message `(c a)`. The equational theory $\mathscr{E}$ includes the equations[9] in Figure 1. In the specification, the equational attribute `variant` is used to identify the equations to be considered by the folding variant narrowing strategy.

2.2 Equational Theories as Rewrite Theories

Algebraic structures often involve axioms like associativity and/or commutativity of function symbols, which cannot be handled by ordinary term rewriting but are instead handled implicitly by working with congruence classes of terms [12]. This is why an equational theory is often decomposed into a disjoint union $\mathscr{E} = E \uplus B$, where B is a set of algebraic axioms (which are implicitly expressed in Maude as attributes of their corresponding operator using the `assoc`, `comm`, and `id:` keywords) that are used for B-matching, and E consists of (possibly conditional) equations that are implicitly oriented from left to right as a set $\vec{E}$ of rewrite rules (and operationally used as simplification rules modulo B). By doing this, a (well-behaved) rewrite theory $(\Sigma, B, \vec{E})$ is defined, with $\vec{E} = \{t \to t' \mid t = t' \in E\}$, which satisfies all of the conditions that we need. This is formalized by the notion of *decomposition* of the equational theory $(\Sigma, \mathscr{E})$ as follows.

Definition 1 (Decomposition [13]**).** *Let $(\Sigma, \mathscr{E})$ be a order-sorted equational theory. We call $(\Sigma, B, \vec{E})$ a* decomposition *of $(\Sigma, \mathscr{E})$ if $\mathscr{E} = E \uplus B$ and $(\Sigma, B, \vec{E})$ is an order-sorted rewrite theory satisfying the following properties:*

1. $\vec{E}$ is convergent

[9] Due to the lack of space, we omitted the definition of the operators `[_,_,_]`, `_<_`, `_+_` that respectively implements the usual *if-then-else* construct, the *less-than* relation and the (associative and commutative) addition over natural numbers.

```
var M : Message .
var X K : Nat .
var S : Symbol .
eq toNat(a) = 0 [variant] .
eq toNat(b) = toNat(a) + s(0) [variant] .
eq toNat(c) = toNat(b) + s(0) [variant] .
eq toSym(0)= a [variant] .
eq toSym(s(0)) = b [variant] .
eq toSym(s(s(0))) = c [variant] .
eq len = s(s(s(0))) --- Alphabet cardinality
eq shift(X) = [ s(X) < len,s(X), 0 ] [variant] .
eq unshift(0) = s(s(0)) [variant] .
eq unshift(s(X)) = X [variant] .
eq e(X,0) = X [variant] .
eq e(X,s(Y)) = e(shift(X),Y) [variant] .
eq d(X,0) = X [variant] .
eq d(X,s(Y)) = d(unshift(X),Y) [variant] .
eq enc(S,K) = toSym(e(toNat(S),K)) [variant] .
eq enc(S M,K) = toSym(e(toNat(S),K)) enc(M,K) [variant] .
eq dec(S,K) = toSym(d(toNat(S),K))[variant] .
eq dec(S M,K) = toSym(d(toNat(S),K)) dec(M,K) [variant] .
```

Fig. 1. Equational theory $\mathscr{E}$ encoding the *Caesar* cipher.

2. *B is* regular, *i.e., for each $t = t'$ in B, we have $Var(t) = Var(t')$, and* linear, *i.e., for each $t = t'$ in B, each variable occurs only once in t and in t'.*
3. *B is* sort-preserving, *i.e., for each $t = t'$ in B and substitution σ, we have $t\sigma \in \mathscr{T}_{\Sigma,\mathsf{s}}(\mathscr{X})$ iff $t'\sigma \in \mathscr{T}_{\Sigma,\mathsf{s}}(\mathscr{X})$. Furthermore, for each equation $t = t'$ in B, all variables in $Var(t)$ and $Var(t')$ have a common top sort.*
4. *B has a finitary and complete unification algorithm, which implies that B-matching is decidable.*
5. *The rewrite rules in $\vec{E}$ are* convergent, *i.e., confluent, terminating, and strictly coherent modulo B, and* sort-decreasing.

We often abuse notation and say that $(\Sigma, B, \vec{E})$ is a decomposition of an order-sorted equational theory $(\Sigma, \mathscr{E})$ even if $\mathscr{E} \neq E \uplus B$ but E is instead the explicitly extended B-coherent completion of a set E' such that $\mathscr{E} = E' \uplus B$.

In the following, we often consider rewrite theories (Σ, B, R) that are a decomposition of an order-sorted equational theory so that $R = \vec{E}$.

2.3 Narrowing in Rewriting Logic

Narrowing generalizes term rewriting by allowing free variables in terms (as in logic programming) and by performing unification (at non-variable positions) instead of matching in order to (non-deterministically) reduce the term. Function definition and evaluation are thus embedded within a symbolic logical framework and features such as existentially quantified variables, unification, and function inversion become available.

Definition 2 **((R,E)-narrowing** [26]**).** *Let $\mathcal{R} = (\Sigma, E, R)$ be an order-sorted rewrite theory. The* (R,E)-narrowing *relation on $\mathcal{T}_\Sigma(\mathcal{X})$ is defined as $t \leadsto_{\sigma,R,E} t'$ (or just $t \leadsto_\sigma t'$) if there exist $p \in Pos_\Sigma(t)$, a (renamed apart*[10]*) rule $l \Rightarrow r$ in R, and a E-unifier σ of $t_{|p}$ and l such that $t' = (t[r]_p)\sigma$. The narrowing step $t \leadsto_{\sigma,R,E} t'$ is also called a* (R,E)-narrowing step. *A term t is* (R,E)-narrowable *if there exist σ and t' such that $t \leadsto_{\sigma,R,E} t'$. Given the narrowing sequence $\alpha : (t_0 \leadsto_{\theta_1} t_1 \cdots \leadsto_{\theta_n} t_n)$, the computed substitution of α is $\theta = (\theta_1 \ldots \theta_n)_{|Var(t_0)}$ and we may write $t_0 \leadsto^n_\theta t_n$.*

Since (R,E)-narrowing has quite a large search space, suitable strategies are needed to improve the efficiency of narrowing by getting rid of useless computations.

First, we define the notion of a narrowing strategy. Given a (R,E)-narrowing sequence $\alpha : (t_0 \leadsto_{\theta_1} t_1 \cdots \leadsto_{\theta_n} t_n)$, we denote by α_i the narrowing sequence $\alpha_i : (t_0 \leadsto_{\theta_1} t_1 \cdots \leadsto_{\theta_i} t_i)$, which is a prefix of α. Given an order-sorted rewrite theory $\mathcal{R}$, we denote by $\mathcal{N}_\mathcal{R}(t)$ the (possibly infinite) set of all (R,E)-narrowing sequences stemming from t.

Definition 3 **(Narrowing Strategy).** *A* narrowing strategy *in RWL is a function of two arguments: a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ and a term $t \in \mathcal{T}_\Sigma(\mathcal{X})$, which we denote by $\mathcal{S}_\mathcal{R}(t)$, such that $\mathcal{S}_\mathcal{R}(t) \subseteq \mathcal{N}_\mathcal{R}(t)$. We require $\mathcal{S}_\mathcal{R}(t)$ to be prefix closed, i.e., for each narrowing sequence $\alpha \in \mathcal{S}_\mathcal{R}(t)$ of length n, and each $i \in \{1, \ldots, n\}$, we also have $\alpha_i \in \mathcal{S}_\mathcal{R}(t)$.*

Narrowing strategies have been investigated in [26, 30] that are complete under suitable conditions (i.e., for every solution, a most general answer is computed).

Example 2. The equational theory for exclusive-or has a decomposition into $\vec{E}$ consisting of the (implicitly oriented) equations (1)-(3) below, and B the associativity and commutativity (AC) axioms for symbol $\oplus$:

$$X \oplus 0 = X \quad (1) \qquad\qquad X \oplus X = 0 \quad (2) \qquad\qquad X \oplus X \oplus Y = Y \, (3)$$

Note that equations (1)-(2) are not strictly AC-coherent, but adding equation (3) is sufficient to recover that property (see [31, 11]).

Given the term $t = X \oplus Y$, the following $(\vec{E},B)$-narrowing steps can be proved (variables X and Y of the equations (1)-(3) are respectively renamed as X' and Y' by applying the renaming apart technique; also, only the bindings for the variables X and Y of the input term are shown)

$X \oplus Y \leadsto_{\phi_1} X'$ using $\phi_1 = \{X \mapsto 0, Y \mapsto X'\}$ and Equation (1)
$X \oplus Y \leadsto_{\phi_2} X'$ using $\phi_2 = \{X \mapsto X', Y \mapsto 0\}$ and Equation (1)
$X \oplus Y \leadsto_{\phi_3} 0$ using $\phi_3 = \{X \mapsto X', Y \mapsto X'\}$ and Equation (2)
$X \oplus Y \leadsto_{\phi_4} Y'$ using $\phi_4 = \{X \mapsto Y' \oplus X', Y \mapsto X'\}$ and Equation (3)
$X \oplus Y \leadsto_{\phi_5} Y'$ using $\phi_5 = \{X \mapsto X', Y \mapsto Y' \oplus X'\}$ and Equation (3)
$X \oplus Y \leadsto_{\phi_6} U \oplus Y'$ using $\phi_6 = \{X \mapsto X' \oplus U, Y \mapsto X' \oplus Y'\}$ and Equation (3)

[10] The renaming is chosen to ensure that every substitution is idempotent, i.e., σ satisfies $Dom(\sigma) \cap Ran(\sigma) = \emptyset$ so that $(t\sigma)\sigma = t\sigma$.

In order to provide a finitary and complete unification algorithm for a decomposition $(\Sigma, B, \vec{E})$, two narrowing strategies are defined in [14] (*variant narrowing* and *folding variant narrowing*) that we summarize in the following sections after recalling the notion of *term variant* [10].

2.4 Term Variants

Intuitively, the *variants* of a term t are the normal forms of every possible *instance* of t. More precisely, the set of $(\vec{E}, B)$-variants of t contains all pairs (t', θ) where θ is a substitution and t' is equal to the $(\vec{E}, B)$-irreducible form of $t\theta$ modulo B.

Definition 4 (Term Variant [10]**).** *Given a term t and a decomposition $(\Sigma, B, \vec{E})$, we say that (t', θ) is an $(\vec{E}, B)$-*variant *of t if $t' =_B (t\theta)\downarrow_{\vec{E},B}$, where $Dom(\theta) \subseteq Var(t)$ and $Ran(\theta) \cap Var(t) = \emptyset$.*

Example 3. Consider the following basic specification for the addition of natural numbers without axioms:

$$0 + Y = Y$$
$$s(X) + Y = s(X + Y)$$

The set of variants for the term $N + 0$ is infinite, since we have $(0, \{N \mapsto 0\})$, $(s(0), \{N \mapsto s(0)\})$, …, $(s^k(0), \{N \mapsto s^k(0)\})$. Analogously, the variants of the term $0 + M$ are $(0, \{M \mapsto 0\})$, $(s(0), \{M \mapsto s(0)\})$, …, $(s^k(0), \{M \mapsto s^k(0)\})$.

In order to capture when a newly generated variant is subsumed by a previously generated one, we introduce the notion of *variant preordering with normalization*.

Definition 5 (More General Variant [14]**).** *Given a decomposition $(\Sigma, B, \vec{E})$ and two term variants $(t_1, \theta_1), (t_2, \theta_2)$ of a term t, we write $(t_1, \theta_1) \leq_{\vec{E},B} (t_2, \theta_2)$, meaning (t_1, θ_1) is a most general variant of t than (t_2, θ_2), iff there is a substitution ρ such that $t_1\rho =_B t_2$ and $(\theta_1\rho)_{|Var(t)} =_B (\theta_2\downarrow_{\vec{E},B})_{|Var(t)}$.*

Example 4. The term $N + M$ has an infinite set of most general variants in the theory of Example 3, since we have $(M, \{N \mapsto 0\})$, $(s(M), \{N \mapsto s(0)\})$, …, $(s^k(M)$, $\{N \mapsto s^k(0)\})$, but also $(M', \{N \mapsto 0, M \mapsto (0 + M')\})$. However, note that the variant $(M', \{N \mapsto 0, M \mapsto (0 + M')\})$ is subsumed by $(M, \{N \mapsto 0\})$ and is therefore discarded from the set of most general variants. The set of most general variants of the term $0 + M$ is finite and is $\{(M, id)\}$.

Definition 6 (Complete set of Variants [14]**).** *Given a decomposition $(\Sigma, B, \vec{E})$ and a term t, we write $[\![t]\!]_{\vec{E},B}$ for a* complete *set of variants of t, i.e., for any variant (t_2, θ_2) of t, there is a variant $(t_1, \theta_1) \in [\![t]\!]_{\vec{E},B}$ such that $(t_1, \theta_1) \leq_{\vec{E},B} (t_2, \theta_2)$. An equational theory has the* finite variant property *(FVP) (or it is called a* finite variant theory*) iff for all $t \in \mathcal{T}_\Sigma(\mathcal{X})$, $[\![t]\!]_{\vec{E},B}$ is a finite set.*

The specification of natural numbers of Example 3 is not a finite variant theory, since the term $N+M$ has an infinite number of most general variants, as shown in Example 4. The equational theory for exclusive-or of Example 2 is a finite variant theory as it is the following theory for Boolean expressions.

Example 5. Consider the following theory (written in Maude[11] syntax) that declares the two Boolean constants `true` and `false`, with the special attributes `assoc` and `comm` meaning that the infix operators "`and`" and "`or`" obey associativity and commutativity axioms:

```
fmod BOOL is
  sort Bool .
  ops true false : -> Bool .
  op not : Bool -> Bool .
  ops _and_ _or_ : Bool Bool -> Bool [assoc comm] .
  vars X Y : Bool .
  eq not(true) = false [variant] .
  eq not(false) = true [variant] .
  eq X and true = X [variant] .
  eq X and false = false [variant] .
  eq X or true = true [variant] .
  eq X or false = X [variant] .
endfm
```

There are five most general variants modulo AC for the term "`X and Y`", which are: $(\mathsf{X}\ \mathsf{and}\ \mathsf{Y}, id), (\mathsf{Y}, \{\mathsf{X} \mapsto \mathtt{true}\}), (\mathsf{X}, \{\mathsf{Y} \mapsto \mathtt{true}\}), (\mathtt{false}, \{\mathsf{X} \mapsto \mathtt{false}\}), (\mathtt{false}, \{\mathsf{Y} \mapsto \mathtt{false}\})$. Similarly, only five most general variants exist for "`X or Y`".

It is generally undecidable whether an equational theory has the FVP [6]; a simple semi-decision procedure is given in [8, 25] that works well in practice, and a t more sophisticated technique based on the dependency pair framework is given in [14]. The procedure in [8] is implemented in [3] and works by computing the variants of all flat terms $f(X_1,\dots,X_n)$ for any n-ary operator f in the theory and pairwise-distinct variables $X_1,\dots,X_n$ (of the corresponding sort); the theory does have the FVP iff there is a finite number of most general variants for every such term [8].

Unlike Peano's definition of natural numbers, Presburger's arithmetic is FVP.

Example 6. Consider the Maude functional modules of Figure 2 that define natural numbers with addition and other operations in Presburger's and Peano's style, respectively. Presburger's theory is FVP while Peano's theory is not FVP.

2.5 The variant narrowing strategy

Given a decomposition $(\Sigma, B, \vec{E})$, applying narrowing without any restriction can be very wasteful due to two main sources: (i) for axioms B such as associativity-commutativity,

[11] In Maude 3.0, only equations with the attribute `variant` are used for narrowing under the folding variant narrowing strategy.

```
fmod NAT-PRES is
 pr TRUTH-VALUE .
 sort Nat .

 op 0 : -> Nat [ ctor ] .
 op 1 : -> Nat [ ctor ] .
 op _+_ : Nat Nat -> Nat [ ctor assoc comm id: 0 ] .
 op _-_ : Nat Nat -> Nat .
 op _>=_ : Nat Nat -> Bool .
 op _>_ : Nat Nat -> Bool .

 eq n:Nat - n:Nat + m:Nat = 0 [ variant ] .
 eq (n:Nat + m:Nat) - n:Nat = m:Nat [ variant ] .
 eq n:Nat >= 1 + n:Nat + m:Nat = false [ variant ] .
 eq (n:Nat + m:Nat) >= n:Nat = true [ variant ] .
 eq n:Nat > n:Nat + m:Nat = false [ variant ] .
 eq (1 + n:Nat + m:Nat) > n:Nat = true [ variant ] .
endfm
```

Fig. 2. Presburger's style Equational specification for natural numbers.

```
fmod NAT-PEANO is
 pr TRUTH-VALUE .
 sort Nat .

 op 0 : -> Nat [ctor] .
 op s : Nat -> NzNat  [ctor] .
 op _<_ : Nat Nat -> Bool .
 op _+_: Nat Nat -> Nat [assoc comm id: 0].

 eq 0 < s(X:Nat) = true [variant] .
 eq X:Nat < 0 = false [variant] .
 eq s(X:Nat) < s(Y:Nat) = X < Y [variant] .
 eq X:Nat + 0 = X:Nat [variant].
 eq s(X:Nat) + Y:Nat = s(X:Nat + Y:Nat) .
endfm
```

Fig. 3. Peano's style equational specification for natural numbers.

the number of B-unifiers of an equation can be quite large; and (ii) if we narrow a term in all possible positions, the narrowing tree may grow in an explosive way. Let us first motivate the variant narrowing strategy with two ideas. First, for computing variants in a decomposition we are only interested in narrowing derivations that deliver normalized substitutions and lead to normalized terms.

Example 7. Continuing with Example 2, due to the prolific *AC*-unification algorithm there are some redundant narrowing steps with non-normalized substitutions, such as

$$X \oplus Y \rightsquigarrow_{\phi_7} Y' \quad \text{using } \phi_7 = \{X \mapsto X' \oplus X', Y \mapsto Y'\} \text{ and Equation (3)}$$

Note that this narrowing step with substitution ϕ_7 is not needed because the same effect is achieved with the normalized substitution ϕ_1. Also note that there are many infinite narrowing sequences, such as the one repeating substitution ϕ_6 again and again: $X \oplus Y \rightsquigarrow_{\phi_6} Z_1 \oplus Z_2 \rightsquigarrow_{\phi_6'} Z_1' \oplus Z_2' \rightsquigarrow_{\phi_6''} Z_1'' \oplus Z_2'' \rightsquigarrow \cdots$ where $\phi_6' = \{Z_1 \mapsto U' \oplus Z_1', Z_2 \mapsto U' \oplus Z_2'\}$ and $\phi_6'' = \{Z_1' \mapsto U'' \oplus Z_1'', Z_2' \mapsto U'' \oplus Z_2''\}$.

Our second idea is to give priority to *most general* narrowing steps, instead of dealing with more instantiated ones, and to select one and only one narrowing step among those having the same generality, following a *don't care* approach.

These optimizations are formalized as follows. First, a relative generality preorder is defined on narrowing steps.

Definition 7 (Preorder and Equivalence of Narrowing Steps [14]**).** *Given a decomposition $(\Sigma, B, \vec{E})$, consider two narrowing steps $\alpha_1 : t \rightsquigarrow_{\sigma_1,\vec{E},B} s_1$ and $\alpha_2 : t \rightsquigarrow_{\sigma_2,\vec{E},B} s_2$. Let $V = Var(t)$. We write $\alpha_1 \preceq_B \alpha_2$ if $\sigma_1 \leq_B \sigma_2[V]$ and $\alpha_1 \prec_B \alpha_2$ if $\sigma_1 <_B \sigma_2[V]$ (i.e., σ_1 is strictly most general than σ_2 on V). We write $\alpha_1 \simeq_B \alpha_2$ if $\sigma_1 \simeq_B \sigma_2[V]$, i.e. $\alpha_1 \preceq_B \alpha_2$ and $\alpha_2 \preceq_B \alpha_1$.*

The relation $\alpha_1 \simeq_B \alpha_2$ between narrowing steps defines a set of equivalence classes of narrowing steps. In what follows, we will be interested in choosing a unique representative $\underline{\alpha} \in [\alpha]_{\simeq_B}$ in each equivalence class of narrowing steps from t. Therefore, $\underline{\alpha}$ will always denote the chosen unique representative $\underline{\alpha} \in [\alpha]_{\simeq_B}$ that is minimal w.r.t. the order $\preceq_B$.

The relation $\preceq_B$ provides an improvement on narrowing executions in two ways. First, narrowing steps with most general computed substitutions will be selected instead of narrowing steps with more specific computed substitutions. As a particular case, when both a rewriting step and a narrowing step are available for (even different positions of) the same term, the rewriting step will always be chosen. Second, the relation $\simeq_B$ provides a further optimization, since just one narrowing (or rewriting) step is chosen for each equivalence class, which further reduces the width of the narrowing tree.

The described strategy is formalized by the notion of *variant narrowing*.

Definition 8 (Variant Narrowing [14]**).** *Given a decomposition $(\Sigma, B, \vec{E})$ and a narrowing step $\alpha : t \rightsquigarrow_{\sigma,\vec{E},B} t'$, α is a* variant narrowing step *if it satisfies: (i) $\sigma|_{Var(t)}$ is $(\vec{E},B)$-irreducible and (ii) $\underline{\alpha}$ is the chosen unique representative of its $\simeq_B$-equivalence class.*

Following the notation of [14], a variant narrowing step from t to t' in $(\Sigma, B, \vec{E})$ with substitution σ is denoted as $t \rightsquigarrow_{\sigma,\underline{\vec{E}},B} t'$.

Note that we easily extend the variant narrowing strategy to variants, i.e., $(t,\theta) \rightsquigarrow_{\sigma,\underline{\vec{E}},B} (t',\theta')$ iff $t \rightsquigarrow_{\sigma,\underline{\vec{E}},B} t'$ and $\theta' = \theta\sigma$.

2.6 The folding variant narrowing strategy

The variant narrowing strategy defined above has no memory of previous steps. The *folding narrowing* strategy of [14] does consider previous steps to avoid the repeated generation of useless or unnecessary computation steps. This is done by considering the narrowing tree as a graph, where some leaves are connected to other nodes by implicit "fold" arrows. When combined with the variant narrowing strategy, the *folding variant narrowing strategy* is achieved that is complete for variant generation and terminates when the input term has a finite set of *most general variants.*

Let us define the *folding variant narrowing strategy* by introducing a *folding narrowing* relation on term variants. The following definition normalizes each computed variant, which is not performed in the original definition of [14].

Definition 9 (Folding Variant Narrowing Strategy). *Let $\mathscr{R} = (\Sigma, B, \vec{E})$ be a decomposition. Given a Σ-term t, the frontier from term variant $I = (t, id)$ is defined as*

$$\begin{aligned}
&\mathit{Frontier}(I)_0 = \{(t{\downarrow_{\vec{E},B}}, id)\},\\
&\mathit{Frontier}(I)_{n+1} = \{(y{\downarrow_{\vec{E},B}}, (\rho\sigma){\downarrow_{\vec{E},B}}) \mid (\exists (z,\rho) \in \mathit{Frontier}(I)_n : (z,\rho) \leadsto_{\sigma,\underline{\vec{E}},B} (y,\rho\sigma)) \wedge \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\nexists k \le n, (w,\tau) \in \mathit{Frontier}(I)_k : (w,\tau) \le_{\vec{E},B} (y,\rho\sigma))\},\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad n \ge 0
\end{aligned}$$

The folding variant narrowing strategy, denoted by $VN^{\circlearrowleft}_{\mathscr{R}}$, is defined as

$$VN^{\circlearrowleft}_{\mathscr{R}}(t) = \{\alpha \mid \alpha : t \leadsto^{k}_{\sigma,\vec{E},B} t' \wedge \exists k \ge 0 : (t', \sigma) \in \mathit{Frontier}((t, id))_k\}$$

Example 8. For the input term $X \oplus Y$, non-normalized narrowing steps such as

$$(X \oplus Y, id) \leadsto_{\phi_7} (Y', \phi_7), \text{ using } \phi_7 = \{X \mapsto X' \oplus X', Y \mapsto Y'\} \text{ and Equation (3)}$$

are not in $VN^{\circlearrowleft}_{\mathscr{R}}$ becaus a variant narrowing step in $VN^{\circlearrowleft}_{\mathscr{R}}$ computes the normalized version of the same substitution; e.g., the variant narrowing step $(X \oplus Y, id) \leadsto_{\phi_1} (X', \phi_1)$ is computed using $\phi_1 = \{X \mapsto 0, Y \mapsto X'\}$ and Equation (1), and $(X', \phi_1) \le_{\vec{E},B} (Y', \phi_7)$. Furthermore, the sequence $(X \oplus Y, id) \leadsto_{\phi_6} (Z_1 \oplus Z_2, \phi_6) \leadsto_{\phi'_6} (Z'_1 \oplus Z'_2, \phi_6\phi'_6)$ corresponding to the two-step prefix of the infinite variant narrowing derivation of Example 7 is not in $VN^{\circlearrowleft}_{\mathscr{R}}$ because $(Z_1 \oplus Z_2, \phi_6) \le_{\vec{E},B} (Z'_1 \oplus Z'_2, \phi_6\phi'_6)$.

For a decomposition $(\Sigma, B, \vec{E})$, completeness of folding variant narrowing w.r.t. $(\vec{E},B)$-normalized substitutions is proved in [14, Theorem 4].

3 Partial Evaluation of Rewrite Theories

In this section, we briefly present the specialization procedure NPER$^{\mathscr{U}}$ that allows a rewrite theory $\mathscr{R} = (\Sigma, E \uplus B, R)$ to be optimized by specializing the underlying equational theory $\mathscr{E} = (\Sigma, E \uplus B)$ with respect to the (calls in the) rewrite rules of R. The procedure NPER$^{\mathscr{U}}$ is parametric w.r.t. an unfolding operator $\mathscr{U}$ that is used to construct finite narrowing derivations for a given expression. NPER$^{\mathscr{U}}$ is based on a suitable extension of the equational, narrowing-driven partial evaluation algorithm for equational theories EQNPE$^{\mathscr{U}}$ of [1] shown below.

3.1 Partial Evaluation of Equational Theories

Given $\mathscr{E} = (\Sigma, E \uplus B)$ and a set Q of calls (henceforth called *specialized calls*), the main goal of EQNPE$^{\mathscr{U}}$ is to derive a new equational theory $\mathscr{E}'$ that computes the same answers (and values) for any input term that is a recursive instance (modulo axioms) of a term in Q. The procedure follows the style of Gallagher's partial deduction method [15], with two distinct control levels: the local level, which is controlled by an unfolding operator, and the global level, which is managed by an abstraction operator.

Unfolding. To partially evaluate $\mathscr{E}$ with respect to Q[12], the EQNPE$^{\mathscr{U}}$ algorithm starts by constructing in $\mathscr{E}$ a finite, possibly partial (folding variant) narrowing tree for each input term t of Q. This is done by using the unfolding operator $\mathscr{U}$ that determines when and how to stop the narrowing computations.

Abstraction. In order to guarantee that all possible executions for t in the original theory $\mathscr{E}$ are *covered* by the specialization, every (sub-)term in any leaf of the tree is required to be *equationally closed* w.r.t. Q. The equational closedness extends the classical PD closedness by recursing over the term structure (in order to handle nested function calls) and by considering B-equivalence of terms.

Roughly speaking, consider a natural partition[13] of the signature as $\Sigma = \mathscr{D} \uplus \mathscr{C}$, where the values computed by simplification (i.e., reduction to canonical form) with $\vec{E}$ modulo B are constructor terms, whereas the function symbols $f \in \mathscr{D}$ are viewed as defined functions that are evaluated away by simplification with $\vec{E}$ modulo B. A term u is closed modulo B w.r.t. Q (we also say that u is Q-closed modulo B) iff either: (i) it does not contain defined function symbols of $\mathscr{D}$, or (ii) there exists a substitution θ and a (possibly renamed) $q \in Q$ such that $u =_B q\theta$, and the terms in θ are recursively Q-closed. For instance, given a defined binary symbol $\bullet$ that does not obey any structural axioms (in particular the commutativity), the term $t = a \bullet (Z \bullet a)$ is closed w.r.t. $Q = \{a \bullet X, Y \bullet a\}$ or $\{X \bullet Y\}$, but it is not with Q being $\{a \bullet X\}$; however, it would be closed if $\bullet$ were commutative.

A set T of terms is closed modulo B (w.r.t. Q and Σ) if $closed_B(Q,t)$ holds for each t in T. A set R of rules is closed modulo B (w.r.t. Q and Σ) if the set that can be formed by taking the right-hand sides of all of the rules in R also is closed modulo B. We often omit Σ when no confusion can arise.

Note that several iterations of i) and ii) may be needed because some of the leaves in deployed narrowing trees might include calls, i.e., (sub-)terms, that are not Q-closed modulo B. At each iteration, an abstraction operator is applied to properly add the uncovered (sub-)terms to the set of already partially evaluated calls, yielding a new set of

[12] For simplicity, we assume that Q is normalized w.r.t. the equational theory $\mathscr{E}$. If this were not the case, for each $t \in Q$ that is not in canonical form such that $t\downarrow_{\vec{E},B} = C(\overline{t_i})$, where $C(\ )$ is the (possibly empty) constructor context of $t\downarrow_{\vec{E},B}$ and $\overline{t_i}$ are the maximal calls in $t\downarrow_{\vec{E},B}$, we would replace t in Q with the normalized terms $\overline{t_i}$, and add a suitable "bridge" equation $t = C(\overline{t_i})$ to the resulting specialization.

[13] This distinction between constructor and defined symbols is more sophisticated than the standard division in the TRSs literature since Rewriting Logic supports overloaded symbols that can play both roles. Consider, e.g., the sort poset `Zero One < Nat` and the equation `s(s(X:Nat))=X:Nat`; in this setting, `s:Zero-> One` is a constructor symbol, whereas `s:Nat -> Nat` is a defined symbol.

terms which may need further evaluation. The process is iteratively repeated as far as new terms are introduced yet ensuring that the set cannot grow infinitely. The abstraction operator guarantees that only finitely many expressions are evaluated, thus ensuring global termination of the specialization.

Theory generation. The EQNPE$^{\mathscr{U}}$ algorithm does not explicitly compute a partially evaluated equational theory. It does so implicitly, by computing a (generally augmented) set Q' of partially evaluated terms that unambiguously determine the desired partially evaluated equations E as the set of *resultants* $t\sigma = t'$ associated with the derivations in the narrowing tree from the root $t \in Q'$ to the leaf t with computed answer substitution σ, such that the closedness condition modulo B w.r.t. Q' is satisfied for all function calls that appear in the right-hand sides of the equations in E'.

In the following, we assume the existence of the function GENTHEORY$(Q',(\Sigma,E \uplus B))$ that delivers the partially evaluated equational theory $\mathscr{E}' = (\Sigma',E' \uplus B')$ univocally determined by Q' and the original equational theory $\mathscr{E} = (\Sigma,E \uplus B)$.

3.2 The NPER$^{\mathscr{U}}$ scheme for the Specialization of Rewrite Theories

We first provide some auxiliary notions that are useful to describe the generic NPER$^{\mathscr{U}}$ scheme. Roughly speaking, the specialization of the rewrite theory $\mathscr{R} = (\Sigma,E \uplus B,R)$ is achieved by partially evaluating the hosted equational theory $\mathscr{E} = (\Sigma,E \uplus B)$ w.r.t. the rules of R, which is done by using the partial evaluation procedure EQNPE$^{\mathscr{U}}$ of Section 3.1. By providing suitable unfolding and abstraction operators, different instances of the specialization scheme can be defined. An unfolding operator that is able to deal with theories that do not meet the finite variant property is introduced in Section 4.1.

Given $\Sigma = \mathscr{D} \uplus \mathscr{C}$, let $\mathscr{D}_E$ be the set of the defined symbols of $\mathscr{D}$ that appear in the set of equations E. Given a term t, a *maximal function call* in t is a subterm $t_{|w}$ of t, with $w \in Pos(t)$, such that (i) $root(t_{|w}) \in \mathscr{D}_E$, and (ii) there does not exist $w' \in Pos(t)$, such that $w' \leq w$ and $t_{|w'} \in \mathscr{D}_E$. Given a rewrite rule $s \Rightarrow t$ of R, by $mcalls(s \Rightarrow t)$ we denote the set of all the maximal function calls that occur in s and t. Also, $mcalls(R)$ is the set of all maximal calls in the rules of R. The NPER$^{\mathscr{U}}$ procedure is outlined in Algorithm 1.

Algorithm 1 Symbolic Specialization of Rewrite Theories NPER$^{\mathscr{U}}(\mathscr{R})$

```
Require:
    A rewrite theory $\mathscr{R} = (\Sigma,E \uplus B,R)$, an unfolding operator $\mathscr{U}$
 1: function NPER$^{\mathscr{U}}(\mathscr{R})$
    Phase 1. Partial Evaluation
 2:     $R' \leftarrow \{(l\downarrow_{\vec{E},B}) \Rightarrow (r\downarrow_{\vec{E},B}) \mid l \Rightarrow r \in R\}$
 3:     $Q \leftarrow mcalls(R')$
 4:     $Q' \leftarrow$ EQNPE$^{\mathscr{U}}((\Sigma,E \uplus B),Q)$
 5:     $\mathscr{E}' \leftarrow$ GENTHEORY$(Q',(\Sigma,E \uplus B))$
    Phase 2. Compression
 6:     $\mathscr{R}'' \leftarrow$ COMPRESS$((\Sigma,E \uplus B,R'),\mathscr{E}',Q')$
 7: return $\mathscr{R}''$
```

Roughly speaking, given the rewrite theory $\mathscr{R} = (\Sigma, E \uplus B, R)$, the procedure consists of two phases. Phase 1 applies the EQNPE$^{\mathscr{U}}$ algorithm to specialize the equational theory $\mathscr{E} = (\Sigma, E \uplus B)$ w.r.t. a set Q of specialized calls that consists of all of the maximal functions calls that appear in the $(\vec{E}, B)$-normalized version R' of the rewrite rules of R. We must normalize the rules in R before initializing Q with the maximal calls in the rules because, for each t in Q, the FV-narrowing tree for t is not rooted with t but with $t_{\downarrow \vec{E},B}$, hence we would loose the connection between the partially evaluated equational theory and the rules of R if the rules were not correspondingly normalized.

This phase produces the new set of specialized calls Q' from which the partial evaluation $\mathscr{E}' = (\Sigma', E' \uplus B')$ of $\mathscr{E}$ w.r.t. Q is univocally derived by executing GENTHEORY$(Q', (\Sigma, E \uplus B))$.

Phase 2 is performed by the COMPRESS post-processing, shown in Algorithm 2, that takes as input the $(\vec{E}, B)$-normalized R' rewrite theory $\mathscr{R}' = (\Sigma, E \uplus B, R')$, the computed partial evaluation $\mathscr{E}' = (\Sigma', E' \uplus B')$, and the final set of specialized calls Q' from which $\mathscr{E}'$ derives. The algorithm computes a new, much more compact equational theory $\mathscr{E}'' = (\Sigma'', E'' \uplus B'')$ where unused symbols and unnecessary repetition of variables are removed, and equations of E' are simplified by renaming similar expressions w.r.t. an independent renaming function ρ that is derived from set of specialized calls Q'.

Formally, for each t of sort s in Q' with $root(t) = f$, $\rho(t) = f_t(\overline{x_n : s_n})$, where $\overline{x_n}$ are the distinct variables in t in the order of their first occurrence and $f_t \colon \overline{s_n} \to s$ is a new function symbol that does not occur in Σ or Q' and is different from the root symbol of any other $\rho(t')$, with $t' \in Q'$ and $t' \neq t$. By abuse, we let $\rho(T)$ denote the set $T' = \{\rho(t) \mid t \in T\}$ for a given set of terms T.

Algorithm 2 Compression algorithm

```
Require:
    A rewrite theory $\mathscr{R}' = (\Sigma, E \uplus B, R')$, a partial evaluation $\mathscr{E}' = (\Sigma', E' \uplus B')$ of $(\Sigma, E \uplus B)$
    w.r.t. a set of specialized calls $Q$.
1: function COMPRESS($\mathscr{R}, \mathscr{E}', Q$)
2:     Let $\rho$ be an independent renaming for $Q$ in
3:             $E'' \leftarrow \bigcup_{t \in Q} \{\rho(t)\theta = RN_\rho(t') \mid t\theta = t' \in E'\}$
4:             $R'' \leftarrow \{RN_\rho(l) \Rightarrow RN_\rho(r) \mid l \Rightarrow r \in R'\}$
5:             $\Sigma'' \leftarrow (\Sigma' \setminus \{f \mid f \text{ occurs in } ((E \uplus B) \setminus (E' \uplus B'))\}) \cup \{root(\rho(t)) \mid t \in Q\}$
6:     $B'' = \{ax(f) \in B' \mid f \in \Sigma' \cap \Sigma''\}$
7: return $(\Sigma'', E'' \uplus B'', R'')$
    where
```

$$RN_\rho(t) = \begin{cases} c(\overline{RN_\rho(t_n)}) & \text{if } t = c(\overline{t_n}) \text{ with } c : \overline{s_n} \to s \in \Sigma \ s.t.\ c \in \mathscr{C},\ ls(t) = s,\ n \geq 0 \\ \rho(u)\theta' & \text{if } \exists\theta, \exists u \in Q \text{ s.t. } t =_B u\theta \text{ and } \theta' = \{x \mapsto RN_\rho(x\theta) \mid x \in Dom(\theta)\} \\ t & \text{otherwise} \end{cases}$$

Essentially, the COMPRESS algorithm of Figure 2 can be seen as a refactoring transformation that recursively computes, by means of the function RN_ρ, a new equation set E'' by replacing each call in E' by a call to the corresponding renamed function according to ρ. Furthermore, a new rewrite rule set R'' is also produced by consistently applying RN_ρ to the rewrite rules of R'. Formally, each rewrite rule $l \Rightarrow r$ in R' is transformed into the rewrite rule $RN_\rho(l) \Rightarrow RN_\rho(r)$, in which every maximal function call t in the rewrite rule is recursively renamed according to the independent renaming ρ. The algorithm also computes the specialized signature Σ'' and restricts the set B' to those axioms obeyed by the function symbols in $\Sigma' \cap \Sigma''$. Finally, the rewrite theory $\mathscr{R}'' = (\Sigma'', E'' \uplus B'', R'')$ is delivered as the final outcome.

Note that, while the independent renaming suffices to rename the left-hand sides of the equations in E' (since they are mere instances of the specialized calls), the right-hand sides are renamed by means of the auxiliary function RN_ρ, which recursively replaces each call in the given expression by a call to the corresponding renamed function (according to ρ).

Given the rewrite theory $\mathscr{R} = (\Sigma, E \uplus B, R)$ that satisfies all of the executability conditions we required before, we can prove that they are also satisfied by the specialization.

Lemma 1 (preservation of executability conditions). *Let $\mathscr{R} = (\Sigma, E \uplus B, R)$ be a rewrite theory where: R is (ground) coherent with E modulo B; and 2) $\vec{\mathscr{E}} = (\Sigma, B, \vec{E})$ is a decomposition. Let $\mathscr{R}' = \mathrm{NPER}^{\mathscr{U}}_{\mathscr{A}}(\mathscr{R})$ be a specialization of $\mathscr{R}$ under the renaming ρ such that $\mathscr{R}' = (\Sigma', E' \uplus B', R')$. Then,*

1. *R' is (ground) coherent with E' modulo B'; and*
2. *$\vec{\mathscr{E}}' = (\Sigma', B', \vec{E}')$ is a decomposition.*

The following result establishes the strong correctness of the $\mathrm{NPER}^{\mathscr{U}}_{\mathscr{A}}$ specialization Algorithm 1, and it states that the specialized rewrite theory $\mathscr{R}'$ and the original theory $\mathscr{R}$ are equivalent in the very strong sense that all computations in $\mathscr{R}$ are preserved in $\mathscr{R}'$.

Theorem 1 (strong correctness). *Let $\mathscr{R} = (\Sigma, E \uplus B, R)$ be a topmost rewrite theory where: 1) $\mathscr{E} = (\Sigma, E \uplus B)$ is a convergent theory; and 2) R is coherent with E modulo B. Let $Q' = \mathrm{EQNPE}^{\mathscr{U}}_{\mathscr{A}}((\Sigma, E \uplus B), Q)$, with $Q = mcalls(R)$. Let $\mathscr{R}' = \mathrm{NPER}^{\mathscr{U}}_{\mathscr{A}}(\mathscr{R})$ be a specialization of $\mathscr{R}$ under the renaming ρ so that $\mathscr{E}' = (\Sigma', E' \uplus B')$ is Q'-closed. Let $u \in \mathscr{T}_\Sigma(\mathscr{X})$ be Q'-closed and $u' = RN_\rho(u)$. Then,*

1. *$(u \rightarrow^*_{R/E,B} v)$ if and only if $(u' \rightarrow^*_{R'/E',B'} v')$, with $v' =_{B'} RN_\rho(v)$.*
2. *If $\mathscr{R}$ satisfies the FVP, then for any $(\vec{E}, B)$-normalized computed substitution σ, $(u \leadsto^*_{\sigma,R,E\uplus B} v)$ if and only if $(u' \leadsto^*_{\sigma',R',E'\uplus B'} v')$, with $v' =_{B'} RN_\rho(v)$ and $\sigma'(x) =_{B'} RN_\rho(\sigma(x))$ for $x \in Dom(\sigma)$.*

Proof. The proof is immediate by the strong correctness of $\mathrm{EQNPE}^{\mathscr{U}}((\Sigma, E \uplus B), Q)$ (Theorem 6 in [1].)

4 Instantiating the Specialization Scheme for Rewrite Theories

In this section, we formulate an instance of the generic specialization scheme of Section 3 by providing a concrete implementation $\mathscr{U}_{\overline{fvp}}$ of the generic unfolding operator $\mathscr{U}$ that implements the local control and is based on folding variant narrowing. Since termination of folding variant narrowing is not generally guaranteed, the unfolding operator $\mathcal{U}_{\overline{fvp}}$ must incorporate some mechanism to stop the construction of the narrowing trees. For this purpose, a number of standard techniques can be applied, including depth-bounds, loop-checks, well-founded orderings, well-quasi orderings, etc.

4.1 Instantiating the Specialization Scheme for Rewrite Theories: The non-FVP case

In this section, we assume a rewrite theory $\mathscr{R} = (\Sigma, E \uplus B, R)$ whose embedded equational theory $\mathcal{E} = (\Sigma, E \uplus B)$ does not satisfy the folding variant property.

Since $\mathcal{E}$ does not have the FVP, the FVN strategy may lead to the creation of an infinite narrowing tree for some specialized calls in Q. In this case, an *equational order-sorted extension* $\breve{\trianglelefteq}_B$ [1] of the classical homeomorphic embedding relation $\trianglelefteq$ can be used to detect the risk of non-termination. Roughly speaking, a homeomorphic embedding relation is a structural preorder under which a term t is greater than (i.e., it embeds) another term t', written as $t \trianglerighteq t'$, if t' can be obtained from t by deleting some parts; e.g., $s(s(X+Y)*(s(X)+Y))$ embeds $s(Y*(X+Y)))$. When iteratively computing a sequence $t_1, t_2, \ldots, t_n$, finiteness of the sequence can be guaranteed by using the embedding as a whistle: whenever a new expression t_{n+1} is to be added to the sequence, we first check whether t_{n+1} embeds any of the expressions already in the sequence. If that is the case, we say that $\trianglelefteq$ whistles, i.e., it has detected (potential) non-termination and the computation has to be stopped. Otherwise, t_{n+1} can be safely added to the sequence and the computation can proceed.

In our context, we say that a narrowing derivation D is *admissible* w.r.t. $\breve{\trianglelefteq}_B$ if and only if it does not contain a pair of comparable narrowing redexes (i.e., rooted by the same operation symbol) s and t, where s precedes t in D, such that $s \breve{\trianglelefteq}_B t$.

The embedding-based unfolding operator $\mathcal{U}_{\overline{fvp}}$ for theories that do not satisfy the FVP can hence be defined as follows.

Definition 10 (Unfolding operator $\mathcal{U}_{\overline{fvp}}$). *Given the equational theory $\mathcal{E} = (\Sigma, E \uplus B)$ and a call t_0 to be specialized in $\mathcal{E}$, we define*

$$\boxed{\begin{array}{ll} \mathcal{U}_{\overline{fvp}}(t_0, \mathcal{E}) = \{t_n \mid & t_0 \leadsto^n t_n \in VN^{\circlearrowleft}_{\vec{\mathcal{E}}}(t_0), \\ & t_0 \leadsto^{n-1} t_{n-1} \textit{ is admissible w.r.t. } \breve{\trianglelefteq}_B \textit{ and} \\ & \textit{either } \nexists w : t_0 \leadsto^n t_n \leadsto w \in VN^{\circlearrowleft}_{\vec{\mathcal{E}}}(t_0) \\ & \textit{or } t_0 \leadsto^n t_n \textit{ is not admissible w.r.t. } \breve{\trianglelefteq}_B.\} \end{array}}$$

Given a set Q of specialized calls , we define $\mathcal{U}_{\overline{fvp}}(Q, \mathcal{E}) = \bigcup_{t \in Q} \mathcal{U}_{\overline{fvp}}(t, \mathcal{E})$.

As for the global level of control, for theories that do not have the FVP (and also for theories that have it), it is enforced by means of an abstraction operator that is based on an equational order sorted extension of the least general generalization algorithm of [2], and it guarantees that the number of unfolded narrowing trees is kept finite. Computing a *least general generalization* (lgg) for two expressions t_1 and t_2, also known as *least general anti-unifier*, means finding the least general expression t such that both t_1 and t_2 are instances of t under appropriate substitutions. Due to the algebraic axioms, in general there can be more than one least general generalizer of two expressions. As a simple example, we record the travel history of a person using a list (with associative list constructor symbol '.') that is ordered by the chronology in which the visits were made; e.g., `paris.paris.bonn.nyc` denotes that `paris` has been visited twice before visiting `bonn` and then `nyc`. The travel histories `paris.paris.bonn.nyc` and `bonn.bonn.rome` have two incomparable least general generalizers modulo axioms (a) `L1.bonn.L2` and (b) `C.C.L`, meaning that (a) the two travelers visited `bonn`, and (b) they consecutively repeated a visit to their own first city. Note that the two generalizers are least general and incomparable, since neither of them is an instance of the other modulo axioms.

Example 9. Consider a specific instance of the rewrite theory of Example 1 where servers and clients reach consensus on a pre-shared fixed key; for simplicity assume `K=s(s(0))`. Let $\mathscr{R} = (\Sigma, E \uplus B, R)$ be such a rewrite theory, where $\mathscr{E} = (\Sigma, E \uplus B)$ is the equational theory of $\mathscr{R}$. In $\mathscr{E}$, the FVN trees associated to encryption and decryption capabilities may be infinite. For instance, the FVN tree for the call `enc(M,s(s(0))` is infinite since the message `M` may have an arbitrary size. In fact, terms of the form $(\mathtt{t}_1 \ldots \mathtt{t}_n\ \mathtt{enc}(\mathtt{M}', \mathtt{s}(\mathtt{s}(0))))$ can be narrowed from $\mathtt{enc}(\mathtt{M}, \mathtt{s}(\mathtt{s}(0)))$, where $\mathtt{enc}(\mathtt{M}', \mathtt{s}(\mathtt{s}(0)))$ can be further narrowed to unravel an unlimited sequence of identical terms modulo renaming. Nonetheless homeomorphic embedding detects this non-terminating behaviour since $\mathtt{enc}(\mathtt{M}', \mathtt{s}(\mathtt{s}(0))$ embeds $\mathtt{enc}(\mathtt{M}, \mathtt{s}(\mathtt{s}(0)))$.

By using the unfolding operator $\mathscr{U}_{\overline{fvp}}$, the first phase of the NPER$^{\mathscr{U}}(\mathscr{R})$ Algorithm 1 computes the initial set $\mathtt{Q} = \{\mathtt{enc}(\mathtt{M}, \mathtt{s}(\mathtt{s}(0)), \mathtt{dec}(\mathtt{M}, \mathtt{s}(\mathtt{s}(0))\}$ consisting of the maximal functional calls in $\mathscr{R}$. Then, the equational theory $\mathscr{E}$ is partially evaluated by EQNPE$^{\mathscr{U}_{\overline{fvp}}}$ w.r.t. `Q`. During the partial evaluation process, $\mathscr{U}_{\overline{fvp}}$ only unravels finite fragments of the FVN narrowing trees that are rooted by the specialized calls, thereby yielding the partial evaluation $\mathscr{E}'$ of $\mathscr{E}$ in Figure 4.

The second phase of the algorithm produces the compressed equational theory $\mathscr{E}''$ of Figure 5 by computing the following renaming for the theory functions. This greatly simplifies $\mathscr{E}'$ since it gets rid of long sequences of nested calls and non-variable function arguments.

```
dec(M : Message, s(s(0))) ↦ f0(M : Message)
enc(M : Message, s(s(0))) ↦ f1(M : Message)
toSym(unshift(unshift(toNat(X : Symbol)))) ↦ f3(X : Symbol)
toSym([[toNat(X : Symbol) < s(s(0)), s(toNat(X : Symbol)), 0] < s(s(0)),
      s([toNat(X : Symbol) < s(s(0)), s(toNat(X : Symbol)), 0]), 0]) ↦ f2(X : Symbol)
```

Finally, it is worth noting that the resulting specialization $\mathscr{E}''$ provides a highly optimized version of $\mathscr{E}$ for an arbitrarily fixed key `K=s(s(0))`, where both functional and structural compression are achieved. Specifically, data structures in $\mathscr{E}$ for natural numbers and their associated operations for message encryption and decryption are totally removed from $\mathscr{E}''$. Note that the `_+_`, operator together with its associative and commutative axioms, disappears from $\mathscr{E}''$, thereby avoiding expensive matching operations modulo axioms. Encryption in $\mathscr{E}''$ (resp., decryption) is now a direct mapping `f0` (resp., `f1`) that associates messages to their corresponding crypted (resp. decrypted)

```
eq dec(a,s(s(0))) = b [variant] .
eq dec(b,s(s(0))) = c [variant] .
eq dec(c,s(s(0))) = a [variant] .
eq dec(S:Symbol M:Message, s(s(0))) =
   toSym(unshift(unshift(toNat(S:Symbol))))
   dec(M:Message, s(s(0))) [variant] .
eq enc(a,s(s(0))) = c [variant] .
eq enc(b,s(s(0))) = a [variant] .
eq enc(c,s(s(0))) = b [variant] .
eq enc(S:Symbol M:Message, s(s(0))) =
   toSym([[toNat(S:Symbol) < s(s(0)),s(toNat(S:Symbol)),0] < s(s(0)),
   s([toNat(S:Symbol) < s(s(0)),s(toNat(S:Symbol)),0]),0]),
   enc(M:Message, s(s(0))) [variant] .
eq toSym([[toNat(a) < s(s(0)),s(toNat(a)),0] <
   s(s(0)),s([toNat(a) < s(s(0)),s(toNat(a)),0]),0]) = c [variant] .
eq toSym([[toNat(b) < s(s(0)),s(toNat(b)),0] <
   s(s(0)),s([toNat(b) < s(s(0)),s(toNat(b)),0]),0]) = a [variant] .
eq toSym([[toNat(c) < s(s(0)),s(toNat(c)),0] <
   s(s(0)),s([toNat(c) < s(s(0)),s(toNat(c)),0]),0]) = b [variant] .
eq toSym(unshift(unshift(toNat(a)))) = b [variant] .
eq toSym(unshift(unshift(toNat(b)))) = c [variant] .
eq toSym(unshift(unshift(toNat(c)))) = a [variant] .
```

Fig. 4. NPER$^{\mathscr{U}_{fvp}}$ Phase 1: Partial evaluation of $\mathscr{E}$ w.r.t. `Q`

counterparts, avoiding a huge amount of computation in the profuse domain of natural numbers. Finally, the computed renaming is also applied to $\mathscr{R}$ by respectively replacing the maximal function calls `enc(M,s(s(0))` and `dec(M,s(s(0))` with `f0(M)` and `f1(M)` into the rewrite rules of $\mathscr{R}$. This allows the (renamed) rewrite rules to be able to access the new specialized encryption and decryption functionality provided by $\mathscr{E}''$.

Note that our methodology may, in some cases, transform a rewrite theory whose equational theory does not satisfy the FVP, into one that does. This allows narrowing-based reachability problems to be solved in the specialized program, whereas it is not possible into the original one. For instance, by restricting the handshake protocol to messages of a fixed size (e.g. 3 symbols), we could get a specialization that meets the FVP, in which the following reachability goal `[Cli-A,Srv-A,Q,K,mt]` & `[Srv-A,K]` & `(Srv-A <- {Cli-A,abc}) =>*  [Srv-A,K]` & `[Cli-A,Srv-A,Q,K,success]` can be solved. The solution allows to infer the client key `K=s(s(0))` and the non-encrypted message `Q=(bca)` in the initial state whenever the crypted message `abc` is sent to the server.

4.2 Instantiating the Specialization Scheme for Rewrite Theories: The FVP case

Let $\mathscr{R} = (\Sigma, E \uplus B, R)$ be a rewrite theory whose embedded equational theory $\mathscr{E} = (\Sigma, E \uplus B)$ meets the folding variant property. In this scenario, folding variant narrowing trees are always finite objects that can be effectively constructed in finite time.

```
eq f0(a) = b [variant] .
eq f0(b) = c [variant] .
eq f0(c) = a [variant] .
eq f2(a) = c [variant] .
eq f2(b) = a [variant] .
eq f2(c) = b [variant] .
eq f1(a) = c [variant] .
eq f1(b) = a [variant] .
eq f1(c) = b [variant] .
eq f3(a) = b [variant] .
eq f3(b) = c [variant] .
eq f3(c) = a [variant] .
eq f0(S:Symbol M:Message) = f3(S:Symbol) f0(M:Message) [variant] .
eq f1(S:Symbol M:Message) = f2(S:Symbol) f1(M:Message) [variant] .
```

Fig. 5. NPER$^{\mathcal{U}_{\overline{fvp}}}$ Phase 2: Compression of $\mathcal{E}'$

Therefore, it is possible to define an unfolding operator that constructs the complete narrowing tree for any possible specialized call in $\mathcal{E}$. The unfolding operator $\mathcal{U}_{fvp}$ is thus defined as follows.

Definition 11 (Unfolding operator $\mathcal{U}_{fvp}$). *Given the equational theory $\mathcal{E} = (\Sigma, E \uplus B)$ and a set Q of calls to be specialized in $\mathcal{E}$, we define*

$$\boxed{\mathcal{U}_{fvp}(Q,\mathcal{E}) = \bigcup_{t \in Q}\{t' \mid t \leadsto^{!} t' \in VN^{\circlearrowleft}_{\mathcal{E}}(t).\}}$$

where $t \leadsto^{!} t'$ denotes a narrowing derivation from t to the unnarrowable term t'.

The advantage of using $\mathcal{U}_{fvp}$ instead of $\mathcal{U}_{\overline{fvp}}$ is twofold. First, $\mathcal{U}_{fvp}$ disregards the embedding check of Definition 10, which may be extremely time-consuming when $\mathcal{E}$ includes several operators that obey associative and commutative axioms[14]. Second, $\mathcal{U}_{fvp}$ exhaustively explores the whole narrowing tree of a term, while $\mathcal{U}_{\overline{fvp}}$ does not. This may lead to a greater degree of specialization when $\mathcal{U}_{fvp}$ is applied.

5 Preliminary experimental evaluation

The NPER$^{\mathcal{U}_{\overline{fvp}}}$ specialization algorithm has been implemented in a prototype system that we have evaluated on a set of preliminary benchmark programs. Table 1 contains the experiments that we have performed using an Intel Xeon E5-1660 3.3GHz CPU with 64 GB RAM running Maude v3.0 and considering the average of ten executions for each test. These experiments together with the source code of all examples are also publicly available at [28].

[14] This is due to the huge search space generated by AC operators when checking an embedding relation. Indeed, given an AC operator $\circ$ and a term $t = t_1 \circ t_2 \ldots \circ t_n$, all possible permutations of t must be checked.

We have considered two variants of the handshake protocol previously discussed in the paper for input messages of three different sizes: one hundred thousand symbols, five hundred thousand symbols, and one million symbols (Column M_{size}). The two variants differ in the introduction of an extra function (Fibonacci) in the underlying equational theory to make the key generation heavier, and in this case, we introduce a generous time bound to stop the execution after a substantial number of rewrites. We have benchmarked the original rewrite theory $\mathscr{R}$ and the specialized rewrite theory $\mathscr{R}'$ on these data. We do not explicitly show the specialization times since they are negligible for all problems ($<$ 100 ms). For each benchmark, the number of rewrites for a common initial state in each rewrite theory is shown in columns $\hat{~}Rews_{\mathscr{R}}$ and $\hat{~}Rews_{\mathscr{R}'}$, respectively. The percentage of reduction in terms of number of rewrites is shown in the *Reduction* column.

The relative speedups that we achieved thanks to specialization are given in the *Speedup* column and computed as the ratio $T_{\mathscr{R}}/T_{\mathscr{R}'}$. Our figures are very encouraging and show that the specialized theories achieve a significant improvement in execution time when compared to the original rewrite theory, with an average speedup for these benchmarks of 3.47.

	M_{size}	$\hat{~}Rews_{\mathscr{R}}$	$\hat{~}Rews_{\mathscr{R}'}$	*Reduction*	$T_{\mathscr{R}}$ (ms)	$T_{\mathscr{R}'}$ (ms)	*Speedup*
Handshake Protocol	100K	2,600,115	400,002	84.62%	221	96	2.30
w/o Fibonacci	500K	13,000,205	2,000,002	84.62%	1,950	731	2.67
(success)	1M	26,000,100	4,000,002	84.62%	5,137	2,191	2.34
Handshake Protocol	100K	92,003,651	10,000,051	89.13%	10,200	1,716	5.94
with Fibonacci	500K	442,003,651	50,000,051	88.69%	53,424	12,185	4.38
(time bound)	1M	879,503,651	100,000,051	88.63%	129,112	40,857	3.16

Table 1. Experimental results for the specialization of the Handshake protocol

Bibliography

[1] M. Alpuente, A. Cuenca-Ortega, S. Escobar & J. Meseguer (2020): *A Partial Evaluation Framework for Order-Sorted Equational Programs modulo Axioms*. *J. Log. Algebr. Meth. Program.* 110.

[2] M. Alpuente, S. Escobar, J. Espert & J. Meseguer (2014): *A Modular Order-Sorted Equational Generalization Algorithm*. *Information and Computation* 235, pp. 98–136.

[3] M. Alpuente, S. Escobar, J. Sapiña & A. Cuenca-Ortega (2017): *Inspecting Maude variants with GLINTS*. *TPLP* 17(5-6), pp. 689–707, doi:10.1017/S147106841700031X. Available at https://doi.org/10.1017/S147106841700031X.

[4] M. Alpuente, M. Falaschi & G. Vidal (1998): *Partial Evaluation of Functional Logic Programs*. *ACM TOPLAS* 20(4), pp. 768–844.

[5] Franz Baader & Wayne Snyder (2001): *Unification Theory*. In John Alan Robinson & Andrei Voronkov, editors: *Handbook of Automated Reasoning (in 2 volumes)*, Elsevier and MIT Press, pp. 445–532.

[6] C. Bouchard, K. A. Gero, C. Lynch & P. Narendran (2013): *On Forward Closure and the Finite Variant Property*. In: *Proc. of the 9th Int'l Symposium on Frontiers of Combining Systems (FroCos 2013)*, *Lecture Notes in Computer Science* 8152, Springer-Verlag, Berlin, pp. 327–342.

[7] R.M. Burstall & J. Darlington (1977): *A Transformation System for Developing Recursive Programs*. *Journal of the ACM* 24(1), pp. 44–67.

[8] A. Cholewa, J. Meseguer & S. Escobar (2014): *Variants of Variants and the Finite Variant Property*. Technical Report, CS Dept. University of Illinois at Urbana-Champaign. Available at http://hdl.handle.net/2142/47117.

[9] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer & C. Talcott (2007): *All About Maude: A High-Performance Logical Framework*. *LNCS* 4350, Springer-Verlag.

[10] H. Comon-Lundh & S. Delaune (2005): *The Finite Variant Property: How to Get Rid of Some Algebraic Properties*. In Jürgen Giesl, editor: *Proc. RTA 2005*, *Lecture Notes in Computer Science* 3467, Springer, pp. 294–307.

[11] F. Durán & J. Meseguer (2010): *A Maude Coherence Checker Tool for Conditional Order-Sorted Rewrite Theories*. In Peter Csaba Ölveczky, editor: *WRLA*, *Lecture Notes in Computer Science* 6381, Springer, pp. 86–103. Available at http://dx.doi.org/10.1007/978-3-642-16310-4_7.

[12] S. Eker (2003): *Associative-Commutative Rewriting on Large Terms*. In Robert Nieuwenhuis, editor: *Rewriting Techniques and Applications, 14th International Conference, RTA 2003, Proceedings*, *Lecture Notes in Computer Science* 2706, Springer, pp. 14–29.

[13] S. Escobar, J. Meseguer & R. Sasse (2009): *Variant Narrowing and Equational Unification*. *Electronic Notes Theoretical Computer Science* 238(3), pp. 103–119. Available at http://dx.doi.org/10.1016/j.entcs.2009.05.015.

[14] S. Escobar, R. Sasse & J. Meseguer (2012): *Folding variant narrowing and optimal variant termination*. *J. Log. Algebr. Program.* 81(7-8), pp. 898–928.

[15] J. P. Gallagher (1993): *Tutorial on Specialisation of Logic Programs*. In: *Proc. of the ACM SIGPLAN Symposium on Partial Evaluation and Program Manipulation (PEPM 1993)*, ACM, New York, pp. 88–98.

[16] J. Goguen & J. Meseguer (1992): *Order-Sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations*. *Theoretical Computer Science* 105, pp. 217–273.

[17] N.D. Jones, C.K. Gomard & P. Sestoft (1993): *Partial Evaluation and Automatic Program Generation*. Prentice-Hall, Englewood Cliffs, NJ.

[18] J.-P. Jouannaud, C. Kirchner & H. Kirchner (1983): *Incremental Construction of Unification Algorithms in Equational Theories*. In: *Proc. of 10th Colloquium on Automata, Languages and Programming (ICALP 1983)*, *LNCS* 154, Springer, pp. 361–373.

[19] J.W. Lloyd & J.C. Shepherdson (1991): *Partial Evaluation in Logic Programming*. *Journal of Logic Programming* 11, pp. 217–242.

[20] J. Meseguer (1992): *Conditional Rewriting Logic as a Unified Model of Concurrency*. *Theoretical Computer Science* 96(1), pp. 73–155.

[21] J. Meseguer (1997): *Membership Algebra As a Logical Framework for Equational Specification*. In Francesco Parisi-Presicce, editor: *Proc. of 12th International Workshop on Recent Trends in Algebraic Development Techniques, WADT'97*, *LNCS* 1376, Springer, pp. 18–61. Available at `http://dx.doi.org/10.1007/3-540-64299-4_26`.

[22] J. Meseguer (2008): *The Temporal Logic of Rewriting: A Gentle Introduction*. In: *Concurrency, Graphs and Models: Essays Dedicated to Ugo Montanari on the Occasion of his 65th Birthday*, *Lecture Notes in Computer Science* 5065, Springer-Verlag, Berlin, pp. 354–382.

[23] J. Meseguer (2016): *Order-Sorted Rewriting and Congruence Closure*. In Bart Jacobs & Christof Löding, editors: *Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 2016*, *Lecture Notes in Computer Science* 9634, Springer, pp. 493–509.

[24] J. Meseguer (2017): *Strict Coherence of Conditional Rewriting Modulo Axioms*. *Theor. Comput. Sci.* 672, pp. 1–35, doi:`10.1016/j.tcs.2016.12.026`. Available at `http://dx.doi.org/10.1016/j.tcs.2016.12.026`.

[25] J. Meseguer (2018): *Variant-based satisfiability in initial algebras*. *Sci. Comput. Program.* 154, pp. 3–41, doi:`10.1016/j.scico.2017.09.001`. Available at `https://doi.org/10.1016/j.scico.2017.09.001`.

[26] J. Meseguer & P. Thati (2007): *Symbolic Reachability Analysis using Narrowing and its Application to Verification of Cryptographic Protocols*. *Higher-Order and Symbolic Computation* 20(1-2), pp. 123–160.

[27] José Meseguer (2020): *Generalized rewrite theories, coherence completion, and symbolic methods*. *J. Log. Algebr. Meth. Program.* 110, doi:`10.1016/j.jlamp.2019.100483`. Available at `https://doi.org/10.1016/j.jlamp.2019.100483`.

[28] Presto (2020): *The* PRESTO *Website*. `http://safe-tools.dsic.upv.es/presto`.

[29] James R. Slagle (1974): *Automated Theorem-Proving for Theories with Simplifiers Commutativity, and Associativity*. *J. ACM* 21(4), pp. 622–642. Available at http://doi.acm.org/10.1145/321850.321859.

[30] Prasanna Thati & José Meseguer (2006): *Complete symbolic reachability analysis using back-and-forth narrowing*. *Theor. Comput. Sci.* 366(1-2), pp. 163–179, doi:10.1016/j.tcs.2006.07.008. Available at https://doi.org/10.1016/j.tcs.2006.07.008.

[31] Patrick Viry (2002): *Equational rules for rewriting logic*. *Theoretical Computer Science* 285(2), pp. 487–517. Available at http://dx.doi.org/10.1016/S0304-3975(01)00366-8.