Generalized Rewrite Theories and Coherence Completion

José Meseguer

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

Abstract. A new notion of generalized rewrite theory suitable for symbolic reasoning and generalizing the standard notion in [3] is motivated and defined. Also, new requirements for *symbolic executability* of generalized rewrite theories that extend those in [8] for standard rewrite theories, including a generalized notion of *coherence*, are given. Finally, symbolic executability, including coherence, is both ensured and made available for a wide class of such theories by automatable *theory transformations*.
Keywords: rewriting, coherence, variants, symbolic execution.

1 Introduction

Symbolic methods are used to reason about concurrent systems specified by rewrite theories in many ways, including: (i) cryptographic protocol verification, e.g., [10], (ii) logical LTL model checking, e.g., [2], (iii) rewriting modulo SMT and related approaches, e.g., [22,1], (iv) inductive theorem proving and program verification, e.g., [12,16], and (v) reachability logic theorem proving, e.g., [25,17,24]. One key issue is that the rewrite theories used in several of these approaches go *beyond* the standard notion of rewrite theory in, say [3], and also beyond the *executability requirements* in, say, [8]. For example: (1) conditions in rules are not just conjunctions of equations, but quantifier-free (QF) formulas in an, often decidable, *background theory* T (e.g., Presburger arithmetic); and (2) the rewrite rules may model *open systems* interacting with an environment, so that they may have extra variables in their righthand sides [22]. Furthermore, each of the approaches just mentioned uses *different* assumptions about the rewrite theories they handle: no general notion has yet been proposed.

There are also unsolved issues about *symbolic executability*: even though symbolic execution methods in some ways *relax* executability requirements, in other ways they impose *strong restrictions* on the rewrite rules to be executed. For example, in narrowing-based reachability analysis the presence of extra variables in righthand sides of rules is unproblematic. Nevertheless, unless *both* the lefthand and righthand sides of a rewrite rule are terms in an equational theory having a *finitary* unification algorithm, symbolic reachability analysis becomes extremely difficult and is usually outside the scope of current methods. There is also plenty of *terra incognita*. For example, we all optimistically assume and require that the rewrite theories we are going to symbolically execute are of course *coherent*

[27,8]. But no theory of coherence, or methods for guaranteeing it, have yet been developed for these new kinds of theories.

The upshot of all this is that, as usual, the new wine of symbolic reasoning requires new wineskins. This work is all about such new wineskins. It asks, and provides answers for, two main questions: (1) How can the notion of rewrite theory be *generalized* to support symbolic reasoning? and (2) What are the appropriate *symbolic executability requirements* needed for such rewrite theories; and how can they be *ensured* for, and made available to, a widest possible class of theories?

Outline and Main Contributions. Section 2 gathers preliminaries. Section 3 motivates and presents a notion of *generalized rewrite theory* suitable for symbolic reasoning and subsuming the standard notion as a special case. It also defines an *initial model semantics* for such theories in an associated category of algebraic transition systems. Finally, it uses such a semantics to identify *symbolic executability requirements*, including a generalized notion of *coherence* and an easier to check characterization of it. Section 4 then addresses and provides solutions for two related problems: (i) how can (ground) coherence be ensured *automatically* under reasonable requirements? and (ii) how can the class of generalized rewrite theories that can be symbolically executed be made *as wide as possible* by means of adequate *theory transformations*? Note that the answer to question (i) is new even for standard rewrite theories and can be quite useful to semi-automate equational abstractions [21]. This automation method is an interesting instance of what might be called *theoretical dogfooding*, where the new symbolic methods of variant computation [11,20,23] are applied to *complete* a rewrite theory into a ground coherent one. The answer to question (ii) is very general: under mild conditions symbolic executability can be ensured for a wide class of generalized theories by two theory transformations. Related work and conclusions are discussed in Section 5. Proofs are relegated to Appendix A.

2 Preliminaries on Order-Sorted Algebra and Variants

I present needed preliminaries on order-sorted algebra, logic, and variants. The material is adapted from [19,20]. The presentation is self-contained: only the notions of many-sorted signature and many-sorted algebra, e.g., [9], are assumed.

Definition 1. *An* order-sorted (OS) signature *is a triple* $\Sigma = (S, \leqslant, \Sigma)$ *with* $(S, \leqslant)$ *a poset and* (S, Σ) *a many-sorted signature.* $\widehat{S} = S/\equiv_{\leqslant}$*, the quotient of* S *under the equivalence relation* $\equiv_{\leqslant} = (\leqslant \cup \geqslant)^+$*, is called the set of* connected components, or kinds *of* $(S, \leqslant)$*. The order* $\leqslant$ *and equivalence* $\equiv_{\leqslant}$ *are extended to sequences of same length in the usual way, e.g.,* $s'_1 \ldots s'_n \leqslant s_1 \ldots s_n$ *iff* $s'_i \leqslant s_i$*,* $1 \leqslant i \leqslant n$*.* Σ *is called* sensible *if for any two* $f : w \to s, f : w' \to s' \in \Sigma$*, with* w *and* w' *of same length, we have* $w \equiv_{\leqslant} w' \;\Rightarrow\; s \equiv_{\leqslant} s'$*. A* many-sorted signature Σ *is the special case where the poset* $(S, \leqslant)$ *is discrete, i.e.,* $s \leqslant s'$ *iff* $s = s'$*.*

For connected components $[s_1], \ldots, [s_n], [s] \in \widehat{S}$

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

denotes the family of "subsort polymorphic" operators f. We can extend any $\Sigma = (S, \leqslant, \Sigma)$ to its kind completion $\widehat{\Sigma} = (S \uplus \widehat{S}, \widehat{\leqslant}, \widehat{\Sigma})$ *where: (i) $\widehat{\leqslant}$ is the least partial order extending $\leqslant$ such that $s < [s]$ for each $s \in S$, and (ii) we add to each family of subsort polymorphic operators $f^{[s_1]\ldots[s_n]}_{[s]}$ in Σ the operator $f : [s_1] \ldots [s_n] \to [s]$.* □

Definition 2. *For $\Sigma = (S, \leqslant, \Sigma)$ an OS signature, an* order-sorted Σ-algebra *A is a many-sorted (S, Σ)-algebra A such that:*

- *whenever $s \leqslant s'$, then we have $A_s \subseteq A_{s'}$, and*
- *whenever $f : w \to s, f : w' \to s' \in f^{[s_1]\ldots[s_n]}_{[s]}$ and $\overline{a} \in A^w \cap A^{w'}$, then we have $f^{w,s}_A(\overline{a}) = f^{w',s'}_A(\overline{a})$, where $A^{s_1 \cdots s_n} = A_{s_1} \times \ldots \times A_{s_n}$.*

*A Σ-*homomorphism *$h : A \to B$ is a many-sorted (S, Σ)-homomorphism such that $([s] = [s'] \wedge a \in A_s \cap A_{s'}) \Rightarrow h_s(a) = h_{s'}(a)$. This defines a category* $\mathbf{OSAlg}_\Sigma$. **Notation***: $h : A \cong B$ denotes an isomorphism $h : A \to B$.* □

Theorem 1. *[19] The category $\mathbf{OSAlg}_\Sigma$ has an initial algebra. Furthermore, if Σ is sensible, then the term algebra T_Σ with:*

- *if $a : \epsilon \to s$ then $a \in T_{\Sigma,s}$ (ϵ denotes the empty string),*
- *if $t \in T_{\Sigma,s}$ and $s \leqslant s'$ then $t \in T_{\Sigma,s'}$,*
- *if $f : s_1 \ldots s_n \to s$ and $t_i \in T_{\Sigma,s_i}$ $1 \leqslant i \leqslant n$, then $f(t_1, \ldots, t_n) \in T_{\Sigma,s}$,*

is initial, i.e., there is a unique Σ-homomorphism to each Σ-algebra.

For $[s] \in \widehat{S}$, $T_{\Sigma,[s]}$ denotes the set $T_{\Sigma,[s]} = \bigcup_{s' \in [s]} T_{\Sigma,s'}$. T_Σ will (ambiguously) denote: (i) the term algebra; (ii) its underlying S-sorted set; and (iii) the set $T_\Sigma = \bigcup_{s \in S} T_{\Sigma,s}$. An OS signature Σ is said to *have non-empty sorts* iff for each $s \in S$, $T_{\Sigma,s} \neq \varnothing$. An OS signature Σ is called *preregular* [14] iff for each $t \in T_\Sigma$ the set $\{s \in S \mid t \in T_{\Sigma,s}\}$ has a least element, denoted $ls(t)$. *We will assume throughout that Σ has non-empty sorts and is preregular.*

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

Theorem 2. *(Freeness Theorem). If Σ is sensible, for each $A \in \mathbf{OSAlg}_\Sigma$ and $\alpha \in [X \to A]$, there exists a unique Σ-homomorphism, $\_\alpha : T_\Sigma(X) \to A$ extending α, i.e., such that for each $s \in S$ and $x \in X_s$ we have $x\alpha_s = \alpha_s(x)$.*

In particular, when $A = T_\Sigma(Y)$, an interpretation of the constants in X, i.e., an S-sorted function $\sigma \in [X \to T_\Sigma(Y)]$ is called a *substitution*, and its unique homomorphic extension $\_\sigma : T_\Sigma(X) \to T_\Sigma(Y)$ is also called a substitution. Define

$dom(\sigma) = \{x \in X \mid x \neq x\sigma\}$, and $ran(\sigma) = \bigcup_{x \in dom(\sigma)} vars(x\sigma)$. Given variables Z, the substitution $\sigma|_Z$ agrees with σ on Z and is the identity elsewhere.

The first-order language of *equational Σ-formulas* is defined in the usual way: its atoms are *Σ-equations* $t = t'$, where $t, t' \in T_\Sigma(X)_{[s]}$ for some $[s] \in \widehat{S}$ and each X_s is assumed countably infinite. The set $Form(\Sigma)$ of *equational Σ-formulas* is then inductively built from atoms by: conjunction ($\wedge$), disjunction ($\vee$), negation ($\neg$), and universal ($\forall x{:}s$) and existential ($\exists x{:}s$) quantification with sorted variables $x{:}s \in X_s$ for some $s \in S$. $\varphi \in Form(\Sigma)$ is called *quantifier-free* (QF) iff it does not contain any quantifiers. The literal $\neg(t = t')$ is denoted $t \neq t'$. Given a Σ-algebra A, a formula $\varphi \in Form(\Sigma)$, and an assignment $\alpha \in [Y \to A]$, with $Y = fvars(\varphi)$ the free variables of φ, the *satisfaction relation* $A, \alpha \models \varphi$ is defined inductively in the usual way. By definition, $A \models \varphi$ holds iff for each $\alpha \in [Y \to A]$ $A, \alpha \models \varphi$ holds, where $Y = fvars(\varphi)$ are the free variables of φ. We say that φ is *valid* (or *true*) in A iff $A \models \varphi$. For a subsignature $\Omega \subseteq \Sigma$ and $A \in \mathbf{OSAlg}_\Sigma$, the *reduct* $A|_\Omega \in \mathbf{OSAlg}_\Omega$ agrees with A in the interpretation of all sorts and operations in Ω and discards everything in $\Sigma \backslash \Omega$. If $\varphi \in Form(\Omega)$ we have the equivalence $A \models \varphi \;\Leftrightarrow\; A|_\Omega \models \varphi$. Given a set of formulas $\Gamma \subseteq Form(\Sigma)$ we say that $A \in \mathbf{OSAlg}_\Sigma$ *satisfies* Γ, written $A \models \Gamma$ iff $\forall \varphi \in \Gamma \; A \models \varphi$. An OS *theory* T is a pair $T = (\Sigma, \Gamma)$ with Σ an OS signature and $\Gamma \subseteq Form(\Sigma)$. For $T = (\Sigma, \Gamma)$, $\mathbf{OSAlg}_{(\Sigma,\Gamma)}$ denotes the full subcategory of $\mathbf{OSAlg}_\Sigma$ with objects those $A \in \mathbf{OSAlg}_\Sigma$ such that $A \models \Gamma$, called the *(Σ, Γ)-algebras*. Given $T = (\Sigma, \Gamma)$ we call $\varphi \in Form(\Omega)$ a *logical consequence* of T, or *true* in T, denoted $T \models \varphi$ or $\Gamma \models \varphi$, iff $\forall A \in \mathbf{OSAlg}_{(\Sigma,\Gamma)} \; A \models \varphi$. Note that the notion of satisfaction and the Freeness theorem yield the implication $T \models \varphi \Rightarrow T \models \varphi\theta$ for any substitution θ. Note also that any Σ-algebra A has an associated theory $th(A) = (\Sigma, \{\varphi \in Form(\Sigma) \mid A \models \varphi\})$. A *theory inclusion* $T = (\Sigma, \Gamma) \subseteq (\Sigma', \Gamma') = T'$ holds iff $\Sigma \subseteq \Sigma'$ and $\Gamma' \models \Gamma$, and is called a *conservative extension* iff $\forall \varphi \in Form(\Sigma) \;\; T \models \varphi \;\Leftrightarrow\; T' \models \varphi$. Call $T = (\Sigma, \Gamma)$ and $T' = (\Sigma, \Gamma')$ *semantically equivalent* (denoted $T \equiv T'$) iff $T \subseteq T'$ and $T' \subseteq T$.

An OS *equational theory* (resp. *conditional equational theory*) is an OS theory $T = (\Sigma, E)$ with E a set of Σ-equations (resp. conditional Σ-equations of the form $\bigwedge_{i=1\ldots n} u_i = v_i \Rightarrow t = t'$). $\mathbf{OSAlg}_{(\Sigma,E)}$ always has an *initial algebra* $T_{\Sigma/E}$, and *free algebras* $T_{\Sigma/E}(X)$ [19]. The inference system in [19] is *sound and complete* for OS equational deduction, i.e., for any OS equational theory (Σ, E), and Σ-equation $u = v$ we have an equivalence $E \vdash u = v \;\; \Leftrightarrow \;\; E \models u = v$. Deducibility $E \vdash u = v$ is abbreviated as $u =_E v$, called *E-equality*.

Given a set of equations B used for deduction modulo B, a preregular OS signature Σ is called *B-preregular*[1] iff for each $u = v \in B$ and substitution

[1] If $B = B_0 \uplus U$, with B_0 associativity and/or commutativity axioms, and U identity axioms, the B-preregularity notion can be *broadened* by requiring only that: (i) Σ is B_0-preregular in the standard sense, so that $ls(u\rho) = ls(v\rho)$ for all $u = v \in B_0$ and substitutions ρ; and (ii) the axioms U oriented as rules $\vec{U}$ are *sort-decreasing* in the sense that $u = v \in U \Rightarrow ls(u\rho) \geqslant ls(v\rho)$ for each ρ. Maude automatically checks B-preregularity of an OS signature Σ in this broader sense [4].

ρ, $ls(u\rho) = ls(v\rho)$. Recall the notation for term positions, subterms, and term replacement from [6]: (i) positions in a term viewed as a tree are marked by strings $p \in \mathbb{N}^*$ specifying a path from the root, (ii) $t|_p$ denotes the subterm of term t at position p, and (iii) $t[u]_p$ denotes the result of *replacing* subterm $t|_p$ at position p by u. Recall also from [20,18] that given an equational theory $(\Sigma, E \uplus B)$ with Σ is B-*preregular*, $=_B$ *decidable*, and such that:

1. each equation $u = v \in B$ is *regular*, i.e., $vars(u) = vars(v)$, and *linear*, i.e., there are no repeated variables in u, and no repeated variables in v;
2. the equations E, when oriented as rewrite rules $\vec{E} = \{t \to t' \mid (t = t') \in E\}$, are *convergent* modulo B, that is, sort-decreasing, strictly B-coherent, confluent, and terminating as rewrite rules modulo B [18],

then we call the rewrite theory $\mathcal{R} = (\Sigma, B, \vec{E})$ (in the sense of [3]) a *decomposition* of the given equational theory $(\Sigma, E \uplus B)$. Given such a decomposition $\mathcal{R} = (\Sigma, B, \vec{E})$, the equality relation $=_{E \uplus B}$ becomes then *decidable* thanks to the rewrite relation $\to_{\vec{E},B}$, where $u \to_{\vec{E},B} v$ holds[2] between two Σ-terms u and v iff there is a position p, a rule $(t \to t') \in \vec{E}$ and a substitution θ such that $u|_p =_B t\theta$ and $v = u[t'\theta]_p$. Such decidability follows from the following theorem:

Theorem 3. *(Church-Rosser Theorem) [15] Let $\mathcal{R} = (\Sigma, B, \vec{E})$ be a decomposition of $(\Sigma, E \uplus B)$. Then we have an equivalence:*

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

where $t!_{\vec{E},B}$ denotes the canonical form of term t by rewriting with $\to_{\vec{E},B}$, which exists and is unique up to B-equality thanks to the convergence of $\to_{\vec{E},B}$.

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

Quite often, the signature Σ on which $T_{\Sigma/E \uplus B}$ is defined has a natural decomposition as a disjoint union $\Sigma = \Omega \uplus \Delta$, where the elements of $C_{\Sigma/\vec{E},B}$ are Ω-terms, whereas the function symbols $f \in \Delta$ are viewed as *defined functions* which are *evaluated away* by $\vec{E}, B$-simplification. Ω (with same poset of sorts as Σ) is then called a *constructor subsignature* of Σ. Call a decomposition $\mathcal{R} = (\Sigma, B, \vec{E})$ of $(\Sigma, E \uplus B)$ *sufficiently complete* with respect to the *constructor subsignature* Ω iff for each $t \in T_\Sigma$ we have $t!_{\vec{E},B} \in T_\Omega$. Sufficient completeness is closely related to *protecting* inclusions of decompositions.

[2] See [23] for the more general definition of both convergence and the relation $\to_{\vec{E},B}$ when Σ is B-preregular in the broader sense of Footnote 1.

Definition 3. *(Protecting, Constructor Decomposition). A decomposition* $\mathcal{R} = (\Sigma, B, \vec{E})$ protects *decomposition* $\mathcal{R}_0 = (\Sigma_0, B_0, \vec{E}_0)$ *iff* $\Sigma_0 \subseteq \Sigma$, $B_0 \subseteq B$, *and* $\vec{E}_0 \subseteq \vec{E}$, *and for all* $t, t' \in T_{\Sigma_0}(X)$ *we have: (i)* $t =_{B_0} t' \Leftrightarrow t =_B t'$, *(ii)* $t = t!_{\vec{E}_0, B_0} \Leftrightarrow t = t!_{\vec{E}, B}$, *and (iii)* $C_{\Sigma_0/\vec{E}_0, B_0} \cong C_{\Sigma/\vec{E}, B}|_{\Sigma_0}$.

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

The notion of *variant* answers, in a sense, two questions: (i) how can we best describe symbolically the elements of $C_{\Sigma/\vec{E},B}(X)$ that are *reduced substitution instances* of a *pattern term* t? and (ii) given an original pattern t, how many other patterns do we need to "cover" all reduced instances of t in $C_{\mathcal{R}}(X)$?

Definition 4. *Given a decomposition* $\mathcal{R} = (\Sigma, B, \vec{E})$ *and a* Σ*-term* t, *a* variant *[5,11] of* t *is a pair* (u, θ) *such that: (i)* $u =_B (t\theta)!_{\vec{E},B}$, *(ii)* $dom(\theta) = vars(t)$, *and (iii)* $\theta = \theta!_{\vec{E},B}$, *that is,* $x\theta = (x\theta)!_{\vec{E},B}$ *for all variables* x. (u, θ) *is called a* ground variant *iff, furthermore,* $u \in T_\Sigma$. *Note that if* (u, θ) *is a ground variant of some* t, *then* $[u]_B \in C_{\Sigma/\vec{E},B}$. *Given variants* (u, θ) *and* (v, γ) *of* t, (u, θ) *is called* more general *than* (v, γ), *denoted* $(u, \theta) \sqsupseteq_B (v, \gamma)$, *iff there is a substitution* ρ *such that: (i)* $(\theta\rho)|_{vars(t)} =_B \gamma$, *and (ii)* $u\rho =_B v$. *Let* $[\![t]\!]_{\vec{E},B} = \{(u_i, \theta_i) \mid i \in I\}$ *denote a* complete set of variants *of* t, *that is, a set of variants such that for any variant* (v, γ) *of* t *there is an* $i \in I$, *such that* $(u_i, \theta_i) \sqsupseteq_B (v, \gamma)$. *A decomposition* $\mathcal{R} = (\Sigma, B, \vec{E})$ *of* $(\Sigma, E \uplus B)$ *has the* finite variant property *[5] (FVP) iff for each* Σ*-term* t *there is a* finite *complete set of variants* $[\![t]\!]_{\vec{E},B} = \{(u_1, \theta_1), \ldots, (u_n, \theta_n)\}$.

If B has a finitary unification algorithm and $\mathcal{R} = (\Sigma, B, \vec{E})$ is FVP, then for any term t the finite set $[\![t]\!]_{\vec{E},B}$ of its variants can be computed by *folding variant narrowing* [11]. Maude 2.7.1 supports the computation of $[\![t]\!]_{\vec{E},B}$ for B a combination of associative and/or commutative and/or identity axioms.

If a decomposition $\mathcal{R} = (\Sigma, B, \vec{E})$ is FVP and protects a constructor decomposition $\mathcal{R}_\Omega = (\Omega, B_\Omega, \vec{E}_\Omega)$, the notion of *constructor variant* answers the following related question: given a pattern t what are the reduced instances of t which "cover" all reduced *ground* instances of t?

Definition 5. *(Constructor Variant). [20] Let* $\mathcal{R} = (\Sigma, B, \vec{E})$ *be a decomposition of* $(\Sigma, E \uplus B)$, *and let* $\mathcal{R}_\Omega = (\Omega, B_\Omega, \vec{E}_\Omega)$ *be a constructor decomposition of* $\mathcal{R}$. *Then an* $\vec{E}, B$*-variant* (u, θ) *of a* Σ*-term* t *is called a* constructor $\vec{E}, B$-variant *of* t *iff* $u \in T_\Omega(X)$. *Let* $[\![t]\!]^\Omega_{\vec{E},B}$ *denote a complete set of constructor variants of a term* t, *i.e., for each constructor variant* (v, β) *of* t *there is a* $(w, \alpha) \in [\![t]\!]^\Omega_{\vec{E},B}$ *such that* $(w, \alpha) \sqsupseteq_B (v, \beta)$.

Under mild conditions on a constructor decomposition $\mathcal{R}_\Omega = (\Omega, B_\Omega, \vec{E}_\Omega)$ protected by an FVP $\mathcal{R} = (\Sigma, B, \vec{E})$, if B has a finitary unification algorithm the

set $[\![t]\!]^{\Omega}_{\vec{E},B}$ is finite and can be effectively computed according to the algorithm in [23], which has been implemented in Maude. Both the sets $[\![t]\!]_{\vec{E},B}$ and $[\![t]\!]^{\Omega}_{\vec{E},B}$ will play a key role in the various notions of ground coherence completion of a generalized rewrite theory presented in Section 4.

3 Generalized Rewrite Theories and Coherence

There are two main reasons for further generalizing the notion of rewrite theory in [3], and for relaxing its *executability conditions* as specified in, e.g., [8]. The first is that it has proved very useful to model *open systems* that interact with a typically non-deterministic external environment by rewrite rules that have extra variables in their righthand sides, so that a term t may be rewritten to a possibly *infinite* number of righthand side instances by different instantiations of such extra variables. The second reason is that for symbolic reasoning it is very useful to allow conditional rewrite rules $l \to r$ *if* φ where φ is not just a conjunction of equalities but a QF equational formula, which is viewed as a *constraint* imposed by the rule and interpreted in a suitable *background theory* T. The key point is that the notion of generalized rewrite theory thus obtained, although in general not executable in the standard sense, can still be executed *symbolically* under fairly reasonable assumptions. For example, the notion of *rewriting modulo SMT* [22] (see also the related work [1]) shows how such generalized theories can be symbolically executed under some typing restrictions and the requirement that satisfiability of a rule's condition φ is always decidable. Related, yet different, notions of symbolic execution are also given in [12,16].

The purpose of this section is fourfold: (1) to give a general definition of such generalized rewrite theories with no executability or decidability assumptions at all; (2) to define a category of *transition system* models for generalized rewrite theories; (3) to first add executability assumptions to the equations in such theories; and (4) to then extend the notion of *coherence* [27,8] to generalized rewrite theories. This will have two important consequences: (i) it will provide essential conditions for *symbolic execution* of such generalized rewrite theories; and (ii) it will make the notion of *ground coherence completion* of a generalized rewrite theory presented in Section 4 as widely applicable as possible.

Definition 6. *(Generalized Rewrite Theory). A* generalized rewrite theory *is a 5-tuple* $\mathcal{R} = (\Sigma, G, R, T, \phi)$*, where: (i)* Σ *is* kind-complete*, so that its set of sorts is* $S \uplus \widehat{S}$*, (see Def. 1); (ii)* (Σ, G) *is a (possibly conditional) equational theory; (iii)* R *is a set of (possibly conditional)* Σ-rewrite rules*, i.e., sequents* $l \to r$ *if* φ*, with* $l, r \in T_{\Sigma}(X)_{[s]}$ *for some* $[s] \in \widehat{S}$*, and* φ *a QF* Σ*-formula; (iv)* T*, called the* background theory*, satisfies* $(\Sigma, G) \subseteq T \subseteq th(T_{\Sigma/G})$*; and (v)*

ϕ is a so-called frozenness *function,*[3] *mapping each subsort-polymorphic family $f^{[s_1]\ldots[s_n]}_{[s]}$ in Σ to the subset $\phi(f^{[s_1]\ldots[s_n]}_{[s]}) \subseteq \{1,\ldots,n\}$ of its* frozen arguments.

*Given a generalized rewrite theory $\mathcal{R} = (\Sigma, G, R, T, \phi)$ and terms $u, v \in T_{\Sigma,[s]}(X)$ for some $[s] \in \widehat{S}$, the rewrite relation $\rightarrow_{\mathcal{R}}$ holds between them, denoted $u \rightarrow_{\mathcal{R}} v$, iff there exist a term u', a ϕ-*unfrozen[4] *position p in u', a rule $l \rightarrow r$ if φ in R and a substitution θ such that: (i) $T \models \varphi\theta$; (ii) $u =_G u' = u'[l\theta]_p$; and (iii) $u'[r\theta]_p =_G v$.*

A generalized rewrite theory $\mathcal{R} = (\Sigma, G, R, T, \phi)$ is called topmost *iff there is a kind $[State] \in \widehat{S}$ such that: (i) for each $l \rightarrow r$ if φ in R, $l, r \in T_\Sigma(X)_{[State]}$; and (ii) for each subsort-polymorphic family $f^{[s_1]\ldots[s_n]}_{[s]}$ in Σ and $i \in \{1,\ldots,n\}$, if $[s_i] = [State]$, then $i \in \phi(f^{[s_1]\ldots[s_n]}_{[s]})$. For $\mathcal{R}$ topmost $u \rightarrow_{\mathcal{R}} v \Rightarrow u, v \in T_{\Sigma,[State]}$.*

Call $\mathcal{R} = (\Sigma, G, R, T, \phi)$ and $\mathcal{R}' = (\Sigma, G', R', T', \phi)$ semantically equivalent*, denoted $\mathcal{R} \equiv \mathcal{R}'$ (resp.* ground semantically equivalent*, denoted $\mathcal{R} \equiv_{gr} \mathcal{R}'$) iff: (1) $(\Sigma, G) \equiv (\Sigma, G')$, (2) $T \equiv T'$, and (3) $\rightarrow_{\mathcal{R}} = \rightarrow_{\mathcal{R}'}$ (resp. (1) $T_{\Sigma/G} = T_{\Sigma/G'}$, (2) $T \equiv T'$, and (3) $\rightarrow_{\mathcal{R}}|_{T^2_\Sigma} = \rightarrow_{\mathcal{R}'}|_{T^2_\Sigma}$).*

Note that the case of a *standard rewrite theory* is the special case where $\mathcal{R} = (\Sigma, G, R, T, \phi)$ is such that $T = (\Sigma, G)$ and for each $l \rightarrow r$ *if* φ in R, φ is a conjunction of equalities[5] $\varphi = \bigwedge_{i=1\ldots n} u_i = v_i$. In such a special case we omit the background theory and write $\mathcal{R} = (\Sigma, G, R, \phi)$ as usual. Note also that the QF formulas φ in the conditions of rules in R may not be arbitrary Σ-formulas, but formulas in a theory $T_0 = (\Sigma_0, I_0)$ such that $\Sigma_0 \subseteq \Sigma$. For example, T_0 may be the theory of Presburger arithmetic. In such a case, the background theory T in $\mathcal{R} = (\Sigma, G, R, T, \phi)$ is assumed to be a *conservative extension* of T_0.

Example 1. This QLOCK protocol example is borrowed from [24], where it is used to verify some of its properties in Reachability Logic by symbolic methods. It illustrates the new features of generalized rewrite theories, including a background theory, negative constraints in conditions, and "open system" rules modeling interaction with an outside environment. QLOCK can be formalized as a generalized rewrite theory $\mathcal{R} = (\widehat{\Sigma}, E \uplus B, R, th(T_{\widehat{\Sigma}/E \uplus B}), \phi)$, in the sense of Def. 6, where ϕ maps each $f \in \widehat{\Sigma}$ to $\varnothing$ (no frozen positions), and $\widehat{\Sigma}$ is the kind completion of signature Σ below. $\mathcal{R}$ models a dynamic version of the QLOCK mutual exclusion protocol [13], where (Σ, B) defines the protocol's

[3] This is supported in Maude by the `frozen` operator attribute, which forbids rewrites below the specified argument positions. For example, when giving a rewriting semantics to a CCS-like process calculus, the process concatenation operator $\_\cdot\_$, appearing in process expressions like $a \cdot P$, will typically be frozen in its second argument.

[4] By definition this means that there is no function symbol f and position q such that: (i) $p = q \cdot i \cdot q'$, (ii) $u'|_q = f(u_1, \ldots, u_n)$, and (iii) $i \in \phi(f^{[s_1]\ldots[s_n]}_{[s]})$. Intuitively this means that the frozenness restrictions ϕ do not block rewriting at position p in u'.

[5] Admittedly, it is possible to allow more general rules with additional "rewrite conditions" of the form $l \rightarrow r$ *if* $\varphi \wedge \bigwedge_{i=1\ldots n} u_i \rightarrow v_i$ in a generalized rewrite theory. Then, generalized rewrite theories would specialize to standard rewrite theories whose rules also allow rewrite conditions. I leave this further generalization as future work.

states, involving natural numbers, lists, and multisets over natural numbers. Σ has sorts $S = \{Nat, List, MSet, Conf, State, Pred\}$ with subsorts $Nat < List$ and $Nat < MSet$ and operators $F = \{0 : \to Nat,\ s\_ : Nat \to Nat,\ \varnothing : \to MSet,\ nil : \to List,\ \_\,\_ : MSet\ MSet \to MSet,\ \_;\_ : List\ List \to List, dupl : MSet \to Pred, tt : \to Pred, \_|\_|\_|\_ : MSet\ MSet\ MSet\ List \to Conf, < \_ > : Conf \to State\}$, where underscores denote operator argument placement. The axioms B are the associativity-commutativity of the multiset union $\_\,\_$ with identity $\varnothing$, and the associativity of list concatenation $\_;\_$ with identity nil. The only equation in E is $dupl(s\,i\,i) = tt$. It defines the $dupl$ predicate by detecting a duplicated element i in the multiset $s\,i\,i$ (where s could be empty). The *states* of QLOCK are B-equivalence classes of ground terms of sort $State$.

QLOCK [13] is a mutual exclusion protocol where the number of processes is unbounded. Furthermore, in the *dynamic* version of QLOCK presented below, such a number can grow or shrink. Each process is identified by a number. The system configuration has three sets of processes (normal, waiting, and critical) plus a waiting queue. To ensure mutual exclusion, a normal process must first register its name at the end of the waiting queue. When its name appears at the front of the queue, it is allowed to enter the critical section. The first three rewrite rules in R below specify how a *normal* process i first transitions to a *waiting* process, then to a *critical* process, and back to normal. The last two rules in R specify how a process can dynamically join or exit the system.

$$
\begin{array}{lllllllllllll}
n2w : & < n\ i & | & w & | & c & | & q & > & \to & < n & | & w\ i & | & c & | & q\ ;\ i > \\
w2c : & < n & | & w\ i & | & c & | & i\ ;\ q > & \to & < n & | & w & | & c\ i & | & i\ ;\ q > \\
c2n : & < n & | & w & | & c\ i & | & i\ ;\ q > & \to & < n\ i & | & w & | & c & | & q\ > \\
join : & < n & | & w & | & c & | & q\ > & \to & < n\ i & | & w & | & c & | & q\ > \ \mathit{if}\ \varphi \\
exit : & < n\ i & | & w & | & c & | & q\ > & \to & < n & | & w & | & c & | & q\ >
\end{array}
$$

where $\varphi \equiv dupl(n\,i\,w\,c) \neq tt$, i is a number, n, w, and c are, respectively, normal, waiting, and critical process identifier sets, and q is a queue of process identifiers. Note that *join* makes QLOCK an *open* system in the sense explained earlier in this section. In the intended use of QLOCK, any state $< n \mid w \mid c \mid q >$ will be such that the multiset $n\,w\,c$ is actually a *set*, so that $dupl(n\,w\ c) \neq tt$ holds. Note that this is an invariant preserved by all the above rules.

Transition System Semantics of Generalized Rewrite Theories. Given a generalized rewrite theory $\mathcal{R} = (\Sigma, G, R, T, \phi)$ we can associate to it the transition system $\mathcal{T}_{\mathcal{R}} = (T_{\Sigma/G}, \to_{\mathcal{R}})$, resp. $\mathcal{T}_{\mathcal{R}}(X) = (T_{\Sigma/G}(X), \to_{\mathcal{R}})$, where, by definition, given $[u], [v] \in T_{\Sigma/G,[s]}$ (resp. $[u], [v] \in T_{\Sigma/G,[s]}(X)$) for some $[s] \in \widehat{S}$, $[u] \to_{\mathcal{R}} [v]$ holds iff $u \to_{\mathcal{R}} v$ holds in the sense of Definition 6. Both $\mathcal{T}_{\mathcal{R}}$ and $\mathcal{T}_{\mathcal{R}}(X)$ are Σ-transition system in the following sense:

Definition 7. *(Σ-Transition System and Homomorphism). Given a kind-complete OS signature Σ, a* Σ-transition system *is a pair $(A, \to_A)$ where: (i) A is a Σ-algebra; and (ii) $\to_A$ is a $\widehat{S}$-indexed family of relations $\to_A = \{\to_{A_{[s]}} \subseteq A^2_{[s]}\}_{[s]\in\widehat{S}}$.*

A homomorphism *of Σ-transition systems $h : (A, \to_A) \to (B, \to_B)$ is a Σ-homomorphism $h : A \to B$ such that for each $[s] \in \widehat{S}$ and $a, a' \in A_{[s]}$, $a \to_{A_{[s]}} a'$ implies $h(a) \to_{B_{[s]}} h(a')$. This defines a category* $\mathbf{Trans}_{\Sigma}$.

Note that $h : (A, \rightarrow_A) \rightarrow (B, \rightarrow_B)$ is an isomorphism in this category iff: (i) h is a Σ-isomorphism, and (ii) $b \rightarrow_{B_{[s]}} b'$ implies $h^{-1}(b) \rightarrow_{A_{[s]}} h^{-1}(b')$. Intuitively, such an isomorphism could be called an "algebraic bisimulation," and a homomorphism an "algebraic simulation."

Given a generalized rewrite theory $\mathcal{R} = (\Sigma, G, R, T, \phi)$ we say that a Σ-transition system $(A, \rightarrow_A)$ *satisfies* the theory $\mathcal{R}$, denoted $(A, \rightarrow_A) \models \mathcal{R}$ iff: (i) $A \in \mathbf{OSAlg}_{(\Sigma,G)}$, and (ii) for each $\alpha \in [Y \rightarrow A]$ the unique Σ-homomorphism $\_\alpha : T_{\Sigma/G}(X) \rightarrow A$ is a Σ-transition system homomorphism $\_\alpha : \mathcal{T}_{\mathcal{R}}(X) \rightarrow (A, \rightarrow_A)$. This defines a full subcategory $\mathbf{Trans}_{\mathcal{R}} \subseteq \mathbf{Trans}_{\Sigma}$ whose initial object is $\mathcal{T}_{\mathcal{R}}$. When $\mathcal{R} = (\Sigma, G, R, \phi)$ is a standard rewrite theory, the Σ-transition system $\mathcal{T}_{\mathcal{R}}$ is closely related to the *initial reachability model* of $\mathcal{R}$ [3], whose associated Σ-transition system is the transitive closure $(T_{\Sigma/G}, \rightarrow_{\mathcal{R}}^*)$ of $\mathcal{T}_{\mathcal{R}}$. Roughly speaking, $\mathcal{T}_{\mathcal{R}}$ is the "one step rewrite" fragment of the initial reachability model in [3].

Definition 6 is very general: in $\mathcal{R} = (\Sigma, G, R, T, \phi)$, besides the generality of the rules R, no assumptions are made about the (possibly conditional) equations G which we are rewriting *modulo* in each transition $u \rightarrow_{\mathcal{R}} v$. In such generality, even *symbolic* execution of $\mathcal{R}$ may be hard to attain. We can substantially improve the situation if we assume that $G = E \uplus B$, with B regular and linear unconditional axioms for which Σ is B-preregular and $=_B$ is decidable, and such that (Σ, G) has a *decomposition* $(\Sigma, B, \vec{E})$. Strictly speaking, such decompositions have only been defined in Section 2 for G a set of *unconditional* equations. However, as shown in, e.g., [8,18], the notion of decomposition of $(\Sigma, E \uplus B)$ generalizes to conditional equations E by means of the notion of a *convergent, strongly deterministic* rewrite theory $(\Sigma, B, \vec{E})$. Likewise, the Church-Rosser Theorem, the notion of canonical term algebra $C_{\Sigma/\vec{E},B}$, and the isomorphism $C_{\Sigma/E,B} \cong T_{\Sigma/E \uplus B}$ naturally extend to the conditional case for such decompositions [18]. Under such conditions, we can achieve a much simpler rewrite relation $\rightarrow_{R/B}$ with the rules R modulo B. Given two terms $u, v \in T_{\Sigma,[s]}(X)$ for some $[s] \in \widehat{S}$, the rewrite relation $u \rightarrow_{R/B} v$ holds iff there exists a $u' \in T_{\Sigma}(X)$ with $u =_B u'$, a ϕ-*unfrozen* position p in u', a rule $l \rightarrow r$ *if* φ in R and a substitution θ such that: (i) $T \models \varphi\theta$; (ii) $u'|_p = l\theta$; and (iii) $v = u'[r\theta]_p$. Under these extra assumptions on $\mathcal{R}$, much simpler Σ-transition systems can be defined:

Definition 8. *(Canonical Σ-Transition System). Let $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$ be such that $(\Sigma, E \uplus B)$ has a decomposition $(\Sigma, B, \vec{E})$ in the above-mentioned sense. Then the Σ-transition system $\mathcal{C}_{\mathcal{R}}(X)$ (resp. $\mathcal{C}_{\mathcal{R}}$) is defined as the pair $(C_{\Sigma/\vec{E},B}(X), \rightarrow_{\mathcal{C}_{\mathcal{R}}})$ (resp. $(C_{\Sigma/\vec{E},B}, \rightarrow_{\mathcal{C}_{\mathcal{R}}}|_{C^2_{\Sigma/\vec{E},B}})$) where for $[u], [v] \in C_{\Sigma/\vec{E},B}(X)$ (resp. $[u], [v] \in C_{\Sigma/\vec{E},B}$), $[u] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [v]$ holds iff there exists $w \in T_{\Sigma}(X)$ such that: (i) $u \rightarrow_{R/B} w$, and (ii) $[v] = [w!_{\vec{E},B}]$.*

The Coherence Problem. Note that it follows from the above definition and from Definition 6 that if $[u]_B \rightarrow_{\mathcal{C}_{\mathcal{R}}} [v]_B$, then $[u]_{E \uplus B} \rightarrow_{\mathcal{R}} [v]_{E \uplus B}$. And since the isomorphism $h : C_{\Sigma/\vec{E},B} \cong T_{\Sigma/E \uplus B}$ (resp. $h : C_{\Sigma/\vec{E},B}(X) \cong T_{\Sigma/E \uplus B}(X)$) is precisely the mapping $h : [u]_B \mapsto [u]_{E \uplus B}$, this means that we have a homomorphism of Σ-transition systems $h : \mathcal{C}_{\mathcal{R}} \rightarrow \mathcal{T}_{\mathcal{R}}$ (resp. $h : \mathcal{C}_{\mathcal{R}}(X) \rightarrow \mathcal{T}_{\mathcal{R}}(X)$).

However, although h is a Σ-isomorphism, it fails in general to be an isomorphism of Σ-transition systems. This is well-known for even trivially simple rewrite theories $\mathcal{R} = (\Sigma, E \uplus B, R, \phi)$ such as $\mathcal{R}$ with Σ unsorted and consisting of constants a, b, c, $E = \{a = b\}$, $B = \varnothing$, and $R = \{a \to c\}$, where $\to_{\mathcal{C}_\mathcal{R}} = \varnothing$, but $\to_\mathcal{R} = \{(\{a, b\}, \{c\})\}$. Since $\mathcal{T}_\mathcal{R}$ is initial in $\mathbf{Trans}_\mathcal{R}$, this of course means that in general $\mathcal{C}_\mathcal{R} \notin \mathbf{Trans}_\mathcal{R}$, and likewise $\mathcal{C}_\mathcal{R}(X) \notin \mathbf{Trans}_\mathcal{R}$. Therefore, canonical transition systems, although simpler than $\mathcal{T}_\mathcal{R}$ or $\mathcal{T}_\mathcal{R}(X)$, *cannot* be used to reason correctly about $\mathcal{R}$-computations. This is the so-called *coherence problem*.

Call $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$ with decomposition $(\Sigma, B, \vec{E})$ *coherent* (resp. *ground coherent*) iff the Σ-transition system homomorphism $h : \mathcal{C}_\mathcal{R}(X) \to \mathcal{T}_\mathcal{R}(X)$ (resp. $h : \mathcal{C}_\mathcal{R} \to \mathcal{T}_\mathcal{R}$) is an isomorphism. Coherence can be characterized by an easier to check condition that generalizes ideas in [27,8]:

Theorem 4. *Let $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$ with $(\Sigma, E \uplus B)$ a decomposition of $(\Sigma, B, \vec{E})$. Then $\mathcal{R}$ is coherent (resp. ground coherent) iff for each $u, v \in T_\Sigma(X)$ (resp. $u \in T_\Sigma$, $v \in T_\Sigma(X)$) such that $u \to_{R/B} v$ (resp. $u \to_{R/B} v$ and $v!_{\vec{E},B} \in T_\Sigma$) there is a term $v' \in T_\Sigma(X)$ such that $u!_{\vec{E},B} \to_{R/B} v'$ and $v!_{\vec{E},B} =_B v'!_{\vec{E},B}$.*

The methods developed in [8] to *check* the coherence of a given $\mathcal{R}$ are based on adequate *critical pairs* modulo B between conditional rules in R and (oriented) conditional equations in $\vec{E}$. By generalizing the conditions in [8] from conjunctions of equalities to QF equational formulas and dropping the executability conditions in [8], general methods for *coherence checking* entirely similar to those in [8] can be developed for generalized rewrite theories. This, however, is *not* the focus of this paper. Instead, both for the special case of the rewrite theories in [8] and for the generalized rewrite theories in Def. 6 above, a *different* question is asked and answered for the first time: Can we, under suitable conditions, *transform* a generalized rewrite theory $\mathcal{R}$ into a semantically equivalent theory $\overline{\mathcal{R}}$, called its *ground coherence completion*, so that $\overline{\mathcal{R}}$ is itself ground coherent? This question is answered in Section 4 below.

4 Coherence Completion of Generalized Rewrite Theories

I present below several theory transformations making a given generalized rewrite theory ground coherent. I also explain how these methods can be automated and how they can be applied to: (i) make rewrite theories symbolically executable; (ii) reason about *equational abstractions* of rewrite theories [21], and (iii) achieve symbolic execution of a widest possible class of such rewrite theories. But first some assumptions on $\mathcal{R}$ need to be made.

Assumptions on $\mathcal{R}$. The generalized rewrite theory $\mathcal{R}$ has the form $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$, with $(\Sigma, E \uplus B)$ a decomposition of $(\Sigma, B, \vec{E})$. Furthermore:

(i) $\mathcal{R}$ is *topmost*; (ii) there are protecting inclusions of decompositions[6]

$$(\Omega, B_\Omega, \vec{E}_\Omega) \subseteq (\Sigma_1, B_1, \vec{E}_1) \subseteq (\Sigma, B, \vec{E})$$

where: (a) Ω, Σ_1 and Σ share the same poset of sorts; (b) E_Ω and E_1 are *unconditional* equations; (c) $(\Omega, B_\Omega, \vec{E}_\Omega)$ is a *constructor* decomposition of $(\Sigma, B, \vec{E})$ and, a fortiori, of $(\Sigma_1, B_1, \vec{E}_1)$; and (d) $(\Sigma_1, B_1, \vec{E}_1)$ is an FVP decomposition; and (iii) each rewrite rule $l \to r$ *if* φ in R is such that l is a Σ_1-term.

Are these assumptions "reasonable"? Regarding assumption (i), many rewrite theories of interest, including theories specifying distributed object-oriented systems and rewriting logic specifications of concurrent programming languages, can be easily specified as topmost rewrite theories by simple theory transformations, e.g., [26]. Regarding assumption (ii)–(iii), some remarks are in order. First, the specification of a constructor subsignature Ω is either explicit in most applications or typically easy to carry out. Second, in virtually all practical specifications of rewrite theories the lefthand side l of a rule $l \to r$ *if* φ is almost always a constructor term. The only case in which this may happen to fail in practice is the case of an *equational abstraction* [21], where l typically *was* a constructor term *before* the abstraction was defined, but after such abstraction definition a *smaller* signature Ω of constructors can be defined. This means that for some applications the decomposition $(\Sigma_1, B_1, \vec{E}_1)$ may specify an equational abstraction. However, $\mathcal{R}$ need *not* be an equational abstraction of another rewrite theory. The FVP decomposition $(\Sigma_1, B_1, \vec{E}_1)$ may have other meanings, including $(\Sigma_1, B_1, \vec{E}_1) = (\Omega, B_\Omega, \vec{E}_\Omega)$, so that the general assumptions are not at all restricted to equational abstractions. This will become clear in what follows.

The $\mathcal{R} \mapsto \overline{\mathcal{R}}_l$ Transformation. For $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$ satisfying the above assumptions, the theory $\overline{\mathcal{R}}_l$ has the form $\overline{\mathcal{R}}_l = (\Sigma, E \uplus B, \overline{R}_l, T, \phi)$, where

$$\overline{R}_l = \{l' \to (r\gamma)!_{\vec{E},B} \text{ if } (\varphi\gamma)!_{\vec{E},B} \mid (l', \gamma) \in [\![l]\!]_{\vec{E}_1, B_1} \ \wedge \ l \to r \text{ if } \varphi \in R\}.$$

As an optimization, we can remove from $\overline{\mathcal{R}}_l$ those rules *B-subsumed* by other rules in $\overline{\mathcal{R}}_l$, where the B subsumption relation $(l \to r \text{ if } \varphi) \sqsupseteq_B (l' \to r' \text{ if } \varphi')$ holds between rules iff there is a substitution α such that $l\alpha =_B l'$, $r\alpha =_B r'$ and $\varphi\alpha =_B \varphi'$. That is, $l \to r$ *if* φ is *more general* than $l' \to r'$ *if* φ' up to B-equality, making $l' \to r'$ *if* φ' redundant. The transformation $\mathcal{R} \mapsto \overline{\mathcal{R}}_l$ can be easily automated as a meta-level function in Maude 2.7.1 using the `metaGetIrredundantVariant` function.

Theorem 5. *Under the above assumptions on $\mathcal{R}$, $\overline{\mathcal{R}}_l$ is semantically equivalent to $\mathcal{R}$. Furthermore, $\overline{\mathcal{R}}_l$ is ground coherent.*

Example 2. The $\mathcal{R} \mapsto \overline{\mathcal{R}}_l$ transformation can be used to obtain a ground coherent theory for an equational abstraction of an infinite-state, out-of-order and

[6] Recall that the strongly deterministic and convergent rules $\vec{E}$ may be *conditional*. We are therefore using Definition 3 in its straightforward generalization to the conditional case.

fault-tolerant communication channel, which thus becomes finite-state and therefore analyzable by standard LTL model checking. Full details are given in Appendix B. Here I illustrate the transformation by focusing on one of the rules, namely, the message reception rule:

```
rl [recv] : [L,N] {J,K} S [P,M] =>
              [K ~ M, [L,N] S ack(K) [P ; J, M + 1],
                      [L,N] S ack(K) [P,M]] .
```

The rule's lefthand side describes a state in which the sender's state `[L,N]` consists of a list `L` of items still to be sent, and a counter `N`, and the receiver's state `[P,M]` consists of a list `P` of items already received and a counter `M`. The channel's contents is a multiset of messages with multiset union denoted by juxtaposion. In this case the contents of the channel is the multiset `{J,K} S` where `{J,K}` is a message sending item `J` marked as message number number `K` sent by the sender to ensure in-order communication. The rest of the messages in the channel are described by the variable `S` of sort `MsgSet`. The rule's righthand side describes two alternative behaviors of the receiver by means of an if-then-else operator

```
op [_,_,_] : Bool Channel Channel -> Channel [frozen] .
```

which is declared `frozen` so that no further rewrites below it are possible until after the if-then-else has been evaluated away. Depending on the equality test `K ~ M` between the message number `K` in the message and the receiver's counter `M`, the sender either appends the item at the end of its list and increases its counter, or discards the message without changing its counter. But in either case an `ack(K)` message signaling the receipt of message number `K` is sent to the sender.

Besides the *associativity* axiom for the list concatenation operator `_;_` and the *associativity-commutativity* axioms for the multiset union operator `_ _` plus the usual equations for if-then-else and the number equality predicate, the key equations in this module are:

```
eq L ; nil = L [variant] .
eq nil ; L = L [variant] .
eq L ; nil ; Q = L ; Q [variant] .  *** B-coherence extension

eq S null = S [variant] .
eq S S = S [variant] .
eq S S S' = S S' [variant] .     *** B-coherence extension
```

The first three equations make `nil` an indentity element for list concatenation. The fourth equation likewise makes `null` an identity element for multiset union. With these equations alone the system is *infinite-state* due to the possibility of message loss modeled by the conditional rule

```
crl [loss] : [L,N] S S' [P,M] => [L,N] S' [P,M] if S =/= null .
```

which makes the specification into a *generalized* rewrite theory due to its QF negative condition. Message loss forces the sender to keep resending each item by means of a `[send]` rule not presented here. The system is made *finite-state*, and

therefore verifiable by standard LTL model checking, by means of the equational abstraction [21] provided by the last two idempotency equations, because the unbounded multiset of messages in the channel thus becomes a set of bounded size. All equations involved are FVP so that the requirements in Theorem 5 are met. For $\mathcal{R}$ the generalized rewrite theory specifying this equationally-abstracted channel, its gound coherence completion $\mathcal{R} \mapsto \overline{\mathcal{R}}_l$ is described in full detail in Appendix B. Here we can just get a flavor for this theory transformation by focusing on the "variants" of the above `[recv]` rule which are added, namely, the following rules:

```
rl [recv] : [L,N] {J,K} [P,M] =>
              [(K ~ M), [L,N] ack(K) [P ; J, M + 1],
                      [L,N] ack(K) [P,M]] .
rl [recv] : [L,N] {J,K} [P,M] =>
              [K ~ M, [L,N] {J,K} ack(K) [P ; J, M + 1],
                      [L,N] {J,K} ack(K) [P,M]] .
rl [recv] : [L,N] {J,K} S [P,M] =>
              [K ~ M, [L,N] {J,K} S ack(K) [P ; J, M + 1],
                      [L,N] {J,K} S ack(K) [P,M]] .
```

The $\mathcal{R} \mapsto \mathcal{R}_{\Sigma_1}$ Transformation. The transformation $\mathcal{R} \mapsto \mathcal{R}_{\Sigma_1}$ is not a coherence completion, but a stepping stone towards a more powerful such completion discussed later. The problem solved by the transformation $\mathcal{R} \mapsto \mathcal{R}_{\Sigma_1}$ has everything to do with symbolic execution and is the following. As already mentioned, a generalized rewrite theory $\mathcal{R}$ of practical interest will typically have rules $l \to r$ *if* φ where the lefthand side l is either a constructor term, or at least a Σ_1-term with $(\Sigma_1, B_1, \vec{E}_1)$ FVP. But what about the rule's righthand side r? Nothing can be assumed in general about r. It can be an arbitrary Σ-term because *auxiliary functions* in Σ may be needed to *update* the state. This poses a serious challenge for *symbolic reasoning* about $\mathcal{R}$, which typically will use symbolic methods such as equational unification and reachability analysis by narrowing modulo an equational theory. As long as r is an Ω-term or at least a Σ_1-term with $(\Sigma_1, B_1, \vec{E}_1)$ FVP, this can easily be done *after each symbolic transition step*, because we can use variant-based unification to compute unifiers in the FVP theories $(\Omega, B_\Omega, \vec{E}_\Omega)$ or $(\Sigma_1, B_1, \vec{E}_1)$, and likewise narrowing modulo such theories to perform symbolic reachability analysis. Instead, if, as usual, r is an arbitrary Σ-term, symbolic reasoning, while not impossible, becomes much harder: if the decomposition $(\Sigma, B, \vec{E})$ is unconditional, we can still perform variant $E \uplus B$-unification by variant narrowing as supported in Maude 2.7.1 for convergent unconditional theories, and likewise narrowing-based reachability analysis based on such $E \uplus B$-unification; but the number of unifiers is in general *infinite*, leading to impractical search spaces with potentially infinite branching at each symbolic state. In Lenin's words: *what is to be done*? Perform the $\mathcal{R} \mapsto \mathcal{R}_{\Sigma_1}$ transformation! This transformation generalizes to a general FVP decomposition $(\Sigma_1, B_1, \vec{E}_1)$ between $(\Omega, B_\Omega, \vec{E}_\Omega)$ and a possibly conditional $(\Sigma, B, \vec{E})$ the special case, described in [24], of a transformation $\mathcal{R} \mapsto \mathcal{R}_\Omega$ making all righthand sides constructor terms. The extra generality of $\mathcal{R} \mapsto \mathcal{R}_{\Sigma_1}$ is useful, because it has a better chance of becoming the identity transformation

for many rules in $\mathcal{R}$. Note that, since righthand sides in $\mathcal{R}_{\Sigma_1}$ are Σ_1-terms, a rule $\alpha : l \to r \ \textit{if} \ \varphi$ can be applied *backwards*, as the rule $\alpha^{-1} : r \to l \ \textit{if} \ \varphi$, to perform *backwards* symbolic reachability analysis, as done in Maude-NPA [10].

The transformation $\mathcal{R} \mapsto \mathcal{R}_{\Sigma_1}$ is defined as follows. By our assumptions on $\mathcal{R}$ each rewrite rule has the form $l \to r \ \textit{if} \ \varphi$ with $l \in T_{\Sigma_1}(X)$. For symbolic reasoning purposes it will be very useful to also achieve that $r \in T_{\Sigma_1}(X)$. If $\mathcal{R} = (\Sigma, E \cup B, R, T, \phi)$, $\mathcal{R}_{\Sigma_1}$ has the form $\mathcal{R}_{\Sigma_1} = (\Sigma, E \cup B, R_{\Sigma_1}, T, \phi)$, where the rules in R_{Σ_1} are obtained from those in R by transforming each $l \to r \ \textit{if} \ \varphi$ in R into the rule $l \to r' \ \textit{if} \ \varphi \wedge \hat{\theta}$, where: (i) $r' \in T_{\Sigma_1}(X)$ is the Σ_1-*abstraction* of r obtained by replacing each length-minimal position p of r where the *top symbol* $top(t|_p)$ of $t|_p$ does not belong Σ_1 by a fresh variable x_p whose sort is the least sort of $t|_p$, and (ii) $\hat{\theta} = \bigwedge_{p \in P} t|_p = x_p$, where P is the set of all length-minimal positions in r with $top(t|_p) \notin \Sigma_1$. As an optimization, whenever $p, p' \in P$ are such that $t_p =_B t'_p$, we can use the same fresh variable for x_p and $x_{p'}$.

Example 3. Since, by specifying order in the natural numbers with constructors an ACU addition $+$, constants $0, 1$ of sort Nat, and $\top, \bot$ of sort $Bool$, Presburger arithmetic with $>$ and $\geqslant$ predicates and extended also with an if-then-else operator $[\_, \_, \_]$ added to any desired sort has an FVP decomposition with signature Σ_1 with decidable $th(T_{\Sigma_1/E_1 \uplus B_1})$ [20], if we have a topmost system whose states are pairs $\langle n, m \rangle$ of natural numbers, and where one of its rules has the form:

$$\langle n, m \rangle \to [n > m, \langle n * m, m \rangle, \langle n, n * m \rangle]$$

then, since the multiplication operator $\_ * \_$ is in Σ but outside Σ_1, the set P of length-minimal positions of the righthand side is $P = \{2.1, 3.2\}$. And since the terms at such positions are both $n * m$, we obtain the transformed rule:

$$\langle n, m \rangle \to [n > m, \langle y, m \rangle, \langle n, y \rangle] \ \textit{if} \ y := n * m.$$

where y has sort Nat and I have used Maude's "matching condition" notation $y := n * m$ for the equation $n * m = y$ to emphasize its executability by matching, which, operationally, corresponds to viewing it as an *equational* rewrite condition of the form $n * m \to^*_{\vec{E}, B} y$.

Although a generalized rewrite theory $\mathcal{R}$ need not be executable, the $\mathcal{R} \mapsto \mathcal{R}_{\Sigma_1}$ transformation *preserves rule executability*. To explain this, I need to explain the general sense in which a rewrite rule $l \to r \ \textit{if} \ \varphi$ in R with $\varphi = \bigwedge_{i=1..n} u_i = v_i$ a conjunction of equalities becomes *executable* by evaluating its condition φ by $\vec{E}, B$ rewriting and B-matching. The sense, as explained in [8], is that we view φ as a $\vec{E}, B$-rewrite condition $\bigwedge_{i=1..n} u_i \to v_i$ and require the following *strong determinism* conditions: (i) $\forall j \in [1..n], vars(u_j) \subseteq vars(l) \cup \bigcup_{k<j} vars(v_k)$, (ii) $vars(r) \subseteq vars(l) \cup \bigcup_{j \leqslant n} vars(v_j)$, and (iii) each v_j is *strongly* $\vec{E}, B$*-irreducible* in the precise sense that $v_j\sigma$ is in $\vec{E}, B$-normal form for each $\vec{E}, B$-normalized substitution σ. The point is that if properties (i)–(ii) hold for the original rule $l \to r \ \textit{if} \ \varphi$ in R, then they also hold for its transformed rule $l \to r' \ \textit{if} \ \varphi \wedge \hat{\theta}$

in R_{Σ_1}. This is clear for (i) and (ii) by construction, and follows also for (iii) because in each rewrite condition $t|_p \rightarrow x_p$ obtained from $\hat{\theta}$ the variable x_p is trivially strongly $\vec{E}, B$-irreducible. In summary we have:

Theorem 6. *Under the above assumptions on $\mathcal{R}$ (dropping the topmost assumption), $\mathcal{R}_{\Sigma_1}$ is semantically equivalent to $\mathcal{R}$. Furthermore, if the rules in $\mathcal{R}$ are executable in the above sense, then those is $\mathcal{R}_{\Sigma_1}$ are also executable.*

The $\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ Transformation. We can now use the previous $\mathcal{R} \mapsto \mathcal{R}_{\Sigma_1}$ transformation to achieve *simultaneously* two important goals: (1) obtain a generalized rewrite theory $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ ground semantically equivalent to $\mathcal{R}$ and such that the lefthand and righthand sides of each of its rules are *constructor* terms; this can be very useful for symbolic executability purposes, since we only need to perform $E_\Omega \uplus B_\Omega$-unification steps, which in many examples may reduce to just B_Ω-unification steps; and (2) ensure that $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ is *ground coherent*.

As already mentioned, the transformation $\mathcal{Q} \mapsto \mathcal{Q}_{\Sigma_1}$ will be used here as a stepping stone. Therefore, we may assume without loss of generality that *it has already been applied*, so that the input theory in this, second transformation $\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ is of the form $\mathcal{R} = \mathcal{Q}_{\Sigma_1}$. Therefore, $\mathcal{R} = (\Sigma, E \cup B, R, T, \phi)$ is such that in each rule $l \rightarrow r$ *if* φ in R both l and r are Σ_1-terms, where $(\Sigma_1, B_1, \vec{E_1})$ is an FVP decomposition protecting a constructor decomposition $(\Omega, B_\Omega, \vec{E}_\Omega)$ and itself protected by $(\Sigma, B, \vec{E})$. The transformed theory $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ has then the form $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r} = (\Sigma, E \cup B, R^{\Omega}_{\Sigma_1,l,r}, T, \phi)$, where

$$R^{\Omega}_{\Sigma_1,l,r} = \{l' \rightarrow r' \textit{ if } (\varphi\gamma)!_{\vec{E},B} \mid (l \rightarrow r \textit{ if } \varphi) \in R \wedge (\langle l', r' \rangle, \gamma) \in [\![\langle l, r \rangle]\!]^{\Omega}_{\vec{E}_1,B_1}\}$$

where we assume without loss of generality that a pairing operator $\langle \_, \_ \rangle$ has been added as a free constructor to each kind in Σ_1 and therefore also to Ω. The key point, of course, is that now the lefthand and righthand sides of a rule $l' \rightarrow r'$ *if* $(\varphi\gamma)!_{\vec{E},B}$ in $R^{\Omega}_{\Sigma_1,l,r}$ are *constructor terms*. This has two important advantages: (1) such rules can be symbolically executed, for example for reachability analysis, by performing $E_\Omega \uplus B_\Omega$-unification, which it typically much simpler and efficient that $E_1 \uplus B_1$-unification; and (2) a rule $\alpha : l' \rightarrow r'$ *if* $(\varphi\gamma)!_{\vec{E},B}$ can be executed *backwards* as the rule $\alpha^{-1} : r' \rightarrow l'$ *if* $(\varphi\gamma)!_{\vec{E},B}$, which can be very useful for backwards symbolic reachability analysis. Here are the key properties:

Theorem 7. *Under the above assumptions on $\mathcal{R}$, $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ is ground semantically equivalent to $\mathcal{R}$. Furthermore, $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ is ground coherent.*

Example 4. The $\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ transformation can be illustrated by a bank account system which is an open system and uses various auxiliary functions to update an account's state after each transaction. Full details are given in Appendix B. Here I illustrate the transformation by focusing on one of the rewrite rules, namely, the rule `[w]` specifying how money can be withdrawn from an account:

```
rl [w] :  < bal: n pend: x overdraft: false > # withdraw(m),msgs =>
      [ m > n , < bal: n pend: x overdraft: true > # msgs ,
      < bal: (n - m) pend: (x - m) overdraft: false > # msgs ]  .
```

The rule's lefthand side describes the state of the account, which consists of `#`-separated pair. The record `< bal: n pend: x overdraft: false >` is the first component. The `bal`ance `n` is the amount of money currently in the account, `x` is the amount of money `pend`ing to be withdrawn in the future, which can be thought of as the amount corresponding to previously written but not yet cashed checks and other withdrawals, and `overdraft` is a Boolean flag whose `false` value indicates that the account is not in the red. Its second component is a multiset of messages built up with an *associative-commutative* multiset union operator `_,_` with *identity* element the empty multiset `mt`. It models the checks and other withdrawals pending to be cashed. Here such a multiset has the form `withdraw(m),msgs` so that there is an actual request `withdraw(m)` to withdraw the amount of money `m` and the remaining messages described by the variable `msgs`. The rule's righthad side describes the account's behavior in response to such a withdrawal request by means of an if-then-else operator (exactly as in Example 2) and the predicate `m > n` testing whether or not the requested money exceeds the account's current balance. If this is the case, the request is rejected and the account goes into an overdraft state. Othewise, the request is honored, the balance is updated, and the pending debt is decreased accordingly. What this rewrite rule clearly illustrates is that, although its lefthand side only involves constructors, its righthand side involves several defined functions needed to update the state, namely, the if-then-else operator, the `m > n` predicate, and the "monus" operator on natural numbers `_-_` used to decrease both the balance and the pending debt. Fortunately, the equations defining all these auxiliary functions are FVP, so that this rule, as well as the other rules in the example only involve Σ_1-terms. This means that this example meets the requirements for the input theory in the $\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ transformation. To give a flavor for the transformation itself, in which all the lefthand- and righthand-sides of the transformed rules become *constructor* terms, I list below the transformed rules for the above `[w]` rule. One feature of the terms below that might seem puzzling is the presence of the natural number addition operator `+`. The point is that `+` is a *free constructor modulo associativity-commutativity* axioms and the *identity* axiom for `0` (ACU), because the additive natural numbers are the free commutative monoid generated by `1`. As shown in [20], this yields a variant-based decision procedure for QF-satisfiability, not just for Presburger arithmetic, but for *all* other auxiliary functions, like monus and if-then-else, involved in this example.

```
rl [w] : < bal: n + m + x pend: m overdraft: false >
         # msgs,withdraw(m + x)
  =>
 < bal: n pend: 0 overdraft: false > # msgs .


rl [w] : < bal: n + m pend: m + x overdraft: false >
```

```
          # msgs,withdraw(m)
 =>
 < bal: n pend: x overdraft: false > # msgs .

rl [w] : < bal: n pend: y overdraft: false >
          # msgs',withdraw(1 + n + x)
 =>
 < bal: n pend: y overdraft: true > # msgs' .
```

The relevant question about this example is: *what is gained in translation?* And the relevant answer is: very much, particularly for narrowing-based reachability analysis. The reason is that, before the transformation, each narrowing step would take place by unifying a symbolic state with a rule's lefthand side *modulo* $E \uplus B$. Instead, now, the unification of symbolic states with lefthand sides of rules takes place modulo $B = B_\Omega$, that is, just modulo ACU, which is much more efficient that $E \uplus B$-unification by folding variant narrowing. In some sense, what has been achieved could be called a process of *total evaluation*, where the defined functions appearing in righthand sides of rules have been completely *evaluated away* by means of their constructor variants. Such total evaluation is what makes possible the reduction from $E \uplus B$-unification to just ACU-unification.

5 Related Work and Conclusions

Closely related work falls into three categories: (i) the already-mentioned symbolic reasoning techniques for rewrite theories, e.g., [10,2,22,1,12,16,25,17,24]; (ii) executability techniques for standard rewrite theories, including [27,8]; and (iii) variant-based symbolic computation, including [5,11,20,23], and also [7], where a limited form of "equational coherence completion" was introduced. In relation to all the work in (i)–(iii), the main contributions of this paper are: (1) a new notion of *generalized rewrite theory*, of rewriting in a generalized rewrite theory, and an initial model semantics for such theories; (2) new *symbolic executability requirements*, including a new notion of *coherence* that is a substantial generalization of the standard notions in [27,8]; and (3) new automatable *theory transformations* both to ensure ground coherence of generalized rewrite theories by *coherence completion*, and to make symbolic executability applicable to a widest possible class of such theories. It is worth noting that the new coherence completion transformations apply, in particular, to standard rewrite theories.

The most obvious next step is to implement all the theory transformations presented in Section 4. This can easily be done by computing variants in Maude, and constructor variants in the Maude implementation of [23]. This will enable new applications, both in symbolic reasoning and in equational abstraction. It could also be used to substantially extend the features of the current Maude Coherence Checker [8].

Acknowledgments. I thank the referees for their constructive criticism and valuable suggestions to improve the paper. This work has been partially supported by NRL under contract number N00173-17-1-G002.

References

1. Arusoaie, A., Lucanu, D., Rusu, V.: Symbolic execution based on language transformation. Computer Languages, Systems & Structures 44, 48–71 (2015)
2. Bae, K., Escobar, S., Meseguer, J.: Abstract Logical Model Checking of Infinite-State Systems Using Narrowing. In: Rewriting Techniques and Applications (RTA'13). LIPIcs, vol. 21, pp. 81–96. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2013)
3. Bruni, R., Meseguer, J.: Semantic foundations for generalized rewrite theories. Theor. Comput. Sci. 360(1-3), 386–414 (2006)
4. Clavel, M., Durán, F., Eker, S., Meseguer, J., Lincoln, P., Martí-Oliet, N., Talcott, C.: All About Maude – A High-Performance Logical Framework. Springer LNCS Vol. 4350 (2007)
5. Comon-Lundth, H., Delaune, S.: The finite variant property: how to get rid of some algebraic properties, in Proc *RTA'05*, Springer LNCS 3467, 294–307, 2005
6. Dershowitz, N., Jouannaud, J.P.: Rewrite systems. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Vol. B, pp. 243–320. North-Holland (1990)
7. Durán, F., Lucas, S., Meseguer, J.: Termination modulo combinations of equational theories. In: Frontiers of Combining Systems, 7th International Symposium, FroCoS 2009, Trento, Italy, September 16-18, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5749, pp. 246–262. Springer (2009)
8. Durán, F., Meseguer, J.: On the Church-Rosser and coherence properties of conditional order-sorted rewrite theories. J. Algebraic and Logic Programming 81, 816–850 (2012)
9. Ehrig, H., Mahr, B.: Fundamentals of Algebraic Specification 1. Springer (1985)
10. Escobar, S., Meadows, C., Meseguer, J.: Maude-NPA: cryptographic protocol analysis modulo equational properties. In: Foundations of Security Analysis and Design V, FOSAD 2007/2008/2009 Tutorial Lectures, LNCS, vol. 5705, pp. 1–50. Springer (2009)
11. Escobar, S., Sasse, R., Meseguer, J.: Folding variant narrowing and optimal variant termination. J. Algebraic and Logic Programming 81, 898–928 (2012)
12. Falke, S., Kapur, D.: Rewriting induction + linear arithmetic = decision procedure. In: Proc. IJCAR 2012. vol. 7364, pp. 241–255. Springer LNCS (2012)
13. Futatsugi, K.: Fostering proof scores in CafeOBJ. In: Proc. ICFEM 2010. vol. 6447, pp. 1–20. Springer LNCS (2010)
14. Goguen, J., Meseguer, J.: Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. Theoretical Computer Science 105, 217–273 (1992)
15. Jouannaud, J.P., Kirchner, H.: Completion of a set of rules modulo a set of equations. SIAM Journal of Computing 15, 1155–1194 (November 1986)
16. Kop, C., Nishida, N.: Automatic constrained rewriting induction towards verifying procedural programs. In: Garrigue, J. (ed.) Programming Languages and Systems - 12th Asian Symposium, APLAS 2014, Singapore, November 17-19, 2014, Proceedings. Lecture Notes in Computer Science, vol. 8858, pp. 334–353. Springer (2014)

17. Lucanu, D., Rusu, V., Arusoaie, A., Nowak, D.: Verifying reachability-logic properties on rewriting-logic specifications. In: Logic, Rewriting, and Concurrency - Essays dedicated to José Meseguer on the Occasion of His 65th Birthday. vol. 9200, pp. 451–474. Springer LNCS (2015)
18. Lucas, S., Meseguer, J.: Normal forms and normal theories in conditional rewriting. J. Log. Algebr. Meth. Program. 85(1), 67–97 (2016)
19. Meseguer, J.: Membership algebra as a logical framework for equational specification. In: Proc. WADT'97. pp. 18–61. Springer LNCS 1376 (1998)
20. Meseguer, J.: Variant-based satisfiability in initial algebras. Sci. Comput. Program. 154, 3–41 (2018)
21. Meseguer, J., Palomino, M., Martí-Oliet, N.: Equational abstractions. Theoretical Computer Science 403(2-3), 239–264 (2008)
22. Rocha, C., Meseguer, J., Muñoz, C.A.: Rewriting modulo SMT and open system analysis. Journal of Logic and Algebraic Methods in Programming 86, 269–297 (2017)
23. Skeirik, S., Meseguer, J.: Metalevel algorithms for variant satisfiability. J. Log. Algebr. Meth. Program. 96, 81–110 (2018)
24. Skeirik, S., Stefanescu, A., Meseguer, J.: A constructor-based reachability logic for rewrite theories. Tech. rep., University of Illinois Computer Science Department (March 2017), available at : `http://hdl.handle.net/2142/95770`. To appear (shorter version) in *Proc. LOPSTR 2107*, Springer LNCS 2018.
25. Stefanescu, A., Ştefan Ciobâcă, Mereuta, R., Moore, B.M., Serbanuta, T., Rosu, G.: All-path reachability logic. In: Proc. RTA-TLCA 2014. vol. 8560, pp. 425–440. Springer LNCS (2014)
26. Thati, P., Meseguer, J.: Symbolic reachability analysis using narrowing and its application to the verification of cryptographic protocols. J. Higher-Order and Symbolic Computation 20(1–2), 123–160 (2007)
27. Viry, P.: Equational rules for rewriting logic. Theoretical Computer Science 285, 487–517 (2002)

A Proofs of Theorems

Proof of Theorem 4.

Proof. Call (†) the above property, claimed equivalent to coherence. Since $h : \mathcal{C}_{\mathcal{R}}(X) \to \mathcal{T}_{\mathcal{R}}(X)$ is already a Σ-algebra isomorphism, it will be a Σ-transition system isomorphism iff the following property holds:

$$(\ddagger)\ \ \forall u, v \in T_\Sigma(X)\ u \to_{\mathcal{R}} v\ \Rightarrow\ \exists v' \in T_\Sigma(X)\ u!_{\vec{E},B} \to_{R/B} v'\ \wedge\ v!_{\vec{E},B} =_B v'!_{\vec{E},B}.$$

But since we always have $u \to_{R/B} v \Rightarrow u \to_{\mathcal{R}} v$, we get $(\ddagger) \Rightarrow (\dagger)$. So we only need to prove $(\dagger) \Rightarrow (\ddagger)$. But, by definition, $u \to_{\mathcal{R}} v$ iff there exists a term $u' \in T_\Sigma(X)$, an ϕ-*unfrozen* position p in u', a rule $l \to r$ *if* φ in R and a substitution θ such that: (i) $T \models \varphi\theta$; (ii) $u =_{E \uplus B} u' = u'[l\theta]_p$; and (iii) $u'[r\theta]_p =_{E \uplus B} v$. Letting $w = u'[r\theta]_p$ this means that $u =_{E \uplus B} u'$, $u' \to_{R/B} w$, and $w =_{E \uplus B} v$, which by (†) implies that there is a $w' \in T_\Sigma(X)$ such that $u'!_{\vec{E},B} \to_{R/B} w'$ and $w!_{\vec{E},B} =_B w'!_{\vec{E},B}$. But by the Church-Rosser Theorem, $u!_{\vec{E},B} =_B u'!_{\vec{E},B}$ and $v!_{\vec{E},B} =_B w'!_{\vec{E},B}$. Therefore, we get $u!_{\vec{E},B} \to_{R/B} w'$ and $v!_{\vec{E},B} =_B w'!_{\vec{E},B}$, yielding $(\ddagger)$, as desired. The proof for the ground coherence case is just a restriction of the above proof requiring $u, v \in T_\Sigma$ in $(\ddagger)$. □

Proof of Theorem 5.

Proof. Note that $\overline{\mathcal{R}}_l \equiv \mathcal{R}$ iff $\mathcal{T}_{\mathcal{R}}(X) = \mathcal{T}_{\overline{\mathcal{R}}_l}(X)$. To see that $\mathcal{T}_{\mathcal{R}}(X) = \mathcal{T}_{\overline{\mathcal{R}}_l}(X)$ just note that: (i) both Σ-transition systems share the same Σ-algebra, namely, $T_{\Sigma/E \uplus B}(X)$, (ii) since, up to variable renaming, $[\![l]\!]_{\vec{E}_1,B_1}$ contains the identity variant (l, id) we have $R \subseteq \overline{R}_l$ and therefore $(\rightarrow_{\mathcal{R}}) \subseteq (\rightarrow_{\overline{\mathcal{R}}_l})$; (iii) any rewrite $u \rightarrow_{\overline{\mathcal{R}}_l} v$ with a rule $l' \rightarrow (r\gamma)!_{\vec{E},B}$ *if* $(\varphi\gamma)!_{\vec{E},B}$ and substitution θ, where $(l', \gamma) \in [\![l]\!]_{\vec{E}_1,B_1}$ and $l \rightarrow r$ *if* $\varphi \in R$, can be performed with $l \rightarrow r$ *if* φ and substitution $\gamma\theta$, since $l\gamma\theta =_{E \uplus B} l'\theta$ and $r\gamma\theta =_{E \uplus B} r'\theta$, and $T \models \varphi\gamma\theta$ iff $T \models (\varphi\gamma)!_{\vec{E},B}\theta$.

To prove that $\overline{\mathcal{R}}_l$ is ground coherent we show that the characterization in Theorem 4 holds. Suppose that $u \in T_\Sigma$, $v \in T_\Sigma(X)$ $u \rightarrow_{\overline{R}_l/B} v$ and $v!_{\vec{E},B} \in T_\Sigma$. We then must show that there is a term $v' \in T_\Sigma(X)$ such that $u!_{\vec{E},B} \rightarrow_{\overline{R}_l/B} v'$ and $v!_{\vec{E},B} =_B v'!_{\vec{E},B}$. But $u \rightarrow_{\overline{R}_l/B} v$ just means that there is a rule $l' \rightarrow (r\gamma)!_{\vec{E},B}$ *if* $(\varphi\gamma)!_{\vec{E},B}$ in $\overline{R}_l$ and substitution θ, where $(l', \gamma) \in [\![l]\!]_{\vec{E}_1,B_1}$ and $l \rightarrow r$ *if* $\varphi \in R$, such that, since $\overline{\mathcal{R}}_l$ is topmost, we have $u =_B l'\theta$, $v = (r\gamma)!_{\vec{E},B}\theta$, and $T \models (\varphi\gamma)!_{\vec{E},B}\theta$ and, by assumption, $v!_{\vec{E},B} \in T_\Sigma$. In general, $\gamma\theta$ need not be a ground substitution. But we can choose a ground substitution η such that $\gamma\theta\eta$ is ground. Furthermore, since by confluence $u!_{\vec{E},B} =_B (l\gamma\theta)!_{\vec{E},B}$ and $u!_{\vec{E},B}$ is ground, by $\vec{E}, B$-rewriting being substitution-closed we must also have $u!_{\vec{E},B} =_B (l\gamma\theta\eta)!_{\vec{E},B}$. But this means that $(u!_{\vec{E},B}, ((\gamma\theta\eta)!_{\vec{E},B})|_{vars(l)})$ is a variant of l, since, by Ω a constructor signature and $\gamma\theta\eta$ is ground, up to B-equivalence we can choose $(\gamma\theta\eta)!_{\vec{E},B}$ to be an Ω-substitution. Therefore, we must have a variant $(l'', \mu) \in [\![l]\!]_{\vec{E}_1,B_1}$ and a substitution δ with $dom(\delta) \subseteq ran(\mu)$ such that $u!_{\vec{E},B} =_B l''\delta$, and $((\gamma\theta\eta)!_{\vec{E},B})|_{vars(l)} =_B \mu\delta$. But this means that we have a decomposition $\gamma\theta\eta =_{E \uplus B} \mu\delta \uplus (\gamma\theta\eta)|_{dom(\gamma\theta\eta)-vars(l)}$, with each component a *ground* substitution. Therefore, we have as well a composition $\gamma\theta\eta =_{E \uplus B} \mu\delta(\gamma\theta\eta)|_{dom(\gamma\theta\eta)-vars(l)}$ such that: (i) since $T \models (\varphi\gamma)!_{\vec{E},B}\theta$ we also have $T \models (\varphi\gamma)!_{\vec{E},B}\theta\eta$, and therefore $T \models \varphi\mu\delta(\gamma\theta\eta)|_{dom(\gamma\theta\eta)-vars(l)}$. But this means that we have a rewrite step $u!_{\vec{E},B} \rightarrow_{\overline{R}_l/B} v'$ with rule $l'' \rightarrow (r\mu)!_{\vec{E},B}$ *if* $(\varphi\mu)!_{\vec{E},B}$ and substitution $\delta(\gamma\theta\eta)|_{dom(\gamma\theta\eta)-vars(l)}$ such that $v' = (r\mu)!_{\vec{E},B}\delta(\gamma\theta\eta)|_{dom(\gamma\theta\eta)-vars(l)}$. Furthermore, since $v = (r\gamma)!_{\vec{E},B}\theta$, and $v!_{\vec{E},B} \in T_\Sigma$, by confluence and $\vec{E}, B$-rewriting being substitution-closed we also have $(r\gamma\theta\eta)!_{\vec{E},B} =_B v!_{\vec{E},B}$, and, by the Church-Rosser Theorem, $v'!_{\vec{E},B} = ((r\mu)!_{\vec{E},B}\delta(\gamma\theta\eta)|_{dom(\gamma\theta\eta)-vars(l)})!_{\vec{E},B} =_B v!_{\vec{E},B}$, as desired. □

Proof of Theorem 6.

Proof. Preservation of rule executability has already been shown. The semantic equivalence $\mathcal{R} \equiv \mathcal{R}_{\Sigma_1}$ follows form the following observations: (1) If $u \rightarrow_{\mathcal{R}} v$ with rule $l \rightarrow r$ *if* φ in R, so that $u =_{E \uplus B} u' = u'[l\mu]_p$ and $u'[r\mu]_p =_{E \uplus B} v$, then $u \rightarrow_{\mathcal{R}_{\Sigma_1}} v$ with rule $l \rightarrow r'$ *if* $\varphi \wedge \hat{\theta}$ in R_{Σ_1} and substitution $\theta\mu$, where $\theta = \{x_p \mapsto t|_p\}_{p \in P}$, so that $u =_{E \uplus B} u' = u'[l\mu]_p = u'[l\theta\mu]_p$, and, since $r = r'\theta$, $u'[r'\theta\mu]_p = u'[r\mu]_p =_{E \uplus B} v$. (2) Conversely, If $u \rightarrow_{\mathcal{R}_{\Sigma_1}} v$ with rule $l \rightarrow r'$ *if* $\varphi \wedge \hat{\theta}$ in R_{Σ_1} and substitution α such that $u =_{E \uplus B} u' = u'[l\alpha]_p$ and $u'[r'\alpha]_p =_{E \uplus B} v$,

since $T_{\Sigma/E \uplus B} \models (\varphi \wedge \hat{\theta})\alpha$, and $r = r'\theta$, we must have $r'\alpha =_{E \uplus B} r\alpha$, and therefore $u'[r'\alpha]_p =_{E \uplus B} u'[r\alpha]_p =_{E \uplus B} v$, so that $u \rightarrow_{\mathcal{R}} v$. □

Proof of Theorem 7.

Proof. To prove that $\mathcal{R} \equiv_{gr} \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ we just need to show $\rightarrow_{\mathcal{R}}|_{T^2_\Sigma} = \rightarrow_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}|_{T^2_\Sigma}$. But any rewrite step (ground or not) $u \rightarrow_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}|_{T^2_\Sigma} v$, say with rule $l' \rightarrow r'$ *if* $(\varphi\gamma)!_{\vec{E},B}$ where $(l \rightarrow r \text{ if } \varphi) \in R$ and $(\langle l', r'\rangle, \gamma) \in [\![\langle l, r\rangle]\!]^{\Omega}_{\vec{E}_1,B_1}$, say with substitution θ, is also a rewrite step $u \rightarrow_{\mathcal{R}} v$ with substitution $\gamma\theta$. Therefore, $\rightarrow_{\mathcal{R}}|_{T^2_\Sigma} \supseteq \rightarrow_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}|_{T^2_\Sigma}$. Two show $\rightarrow_{\mathcal{R}}|_{T^2_\Sigma} \subseteq \rightarrow_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}|_{T^2_\Sigma}$, assume $u \rightarrow_{\mathcal{R}} v$ with u, v ground terms. Since $\mathcal{R}$ is topmost, this means that we have a substitution θ and a rule $l \rightarrow r$ *if* φ in R such that $T \models \varphi\theta$, $u =_{E \uplus B} l\theta$ and $v_{E \uplus B} r\theta$. In general, θ need not be a ground substitution. However, we can choose a ground substitution η such that $\theta\eta$ is ground. And, since u and v ground terms, and equational deduction is closed under substitution, we also get $T \models \varphi\theta\eta$, $u =_{E \uplus B} l\theta\eta$ and $v_{E \uplus B} r\theta\eta$, and therefore the same rewrite step $u \rightarrow_{\mathcal{R}} v$ can also be achieved with ground substitution $\theta\eta$ mapping all variables in the rule to ground terms. But by sufficient completeness this means that $((\langle l\theta\eta, r\theta\eta\rangle)!_{\vec{E},B}, (\theta\eta)!_{\vec{E},B})$ is a constructor variant of $\langle l, r\rangle$ and therefore we have $(\langle l', r'\rangle, \gamma) \in [\![\langle l, r\rangle]\!]^{\Omega}_{\vec{E}_1,B_1}$ and substitution δ such that $\langle l', r'\rangle\delta =_{B_1} (\langle l\theta\eta, r\theta\eta\rangle)!_{\vec{E},B}$ and $(gamma\delta)|_{vars(\langle l,r\rangle)} =_{B_1} ((\theta\eta)!_{\vec{E},B})|_{vars(\langle l,r\rangle)}$, which gives us a rewrite step $u \rightarrow_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}|_{T^2_\Sigma} v$ with rule $l' \rightarrow r'$ *if* $(\varphi\gamma)!_{\vec{E},B}$ and substitution $\theta\eta$, as desired.

The proof that $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ is ground coherent reasons in a way similar to both the proof of the above containment $\rightarrow_{\mathcal{R}}|_{T^2_\Sigma} \subseteq \rightarrow_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}|_{T^2_\Sigma}$ and (even more closely) the proof of ground coherence of $\overline{\mathcal{R}}_l$ in Theorem 5. It is left to the reader. □

B Examples

Example 2. Consider the following Maude specification of a fault-tolerant out-of-order communication channel:

```
fmod NAT-LIST is protecting BOOL .
sorts Nat NatList . subsorts Nat < NatList .
ops 0 1 : -> Nat [ctor] .
op _+_ : Nat Nat -> Nat [ctor assoc comm id: 0] .
op _~_ : Nat Nat -> Bool [comm] .          *** equality predicate
op nil : -> NatList [ctor] .
op _;_ : NatList NatList -> NatList [ctor assoc] .

vars N M : Nat .   vars L Q : NatList .

eq N ~ N = true [variant] .
eq N ~ N + M + 1 = false [variant] .
eq L ; nil = L [variant] .
```

```
eq nil ; L = L [variant] .
eq L ; nil ; Q = L ; Q [variant] .  *** B-coherence extension
endfm

mod FT-CHANNEL is protecting NAT-LIST .
sorts Msg MsgSet Channel .
subsorts Msg < MsgSet .
op null : -> MsgSet [ctor] .
op __ : MsgSet MsgSet -> MsgSet [ctor assoc comm] .
op [_,_]_[_,_] : NatList Nat MsgSet NatList Nat -> Channel [ctor] .
op {_,_} : Nat Nat -> Msg [ctor] .
op ack : Nat -> Msg [ctor] .
op [_,_,_] : Bool Channel Channel -> Channel [frozen] . *** if-then-else

vars N M I J K : Nat .  vars L P Q R : NatList .
var MSG : Msg .  vars S S' : MsgSet . vars CH CH' : Channel .

eq [true,CH,CH'] = CH [variant] .
eq [false,CH,CH'] = CH' [variant] .

eq S null = S [variant] .
eq S S = S [variant] .
eq S S S' = S S' [variant] .     *** B-coherence extension

rl [send] : [J ; L,N] S [P,M] => [J ; L,N] {J,N} S [P,M] .
rl [recv] : [L,N] {J,K} S [P,M] =>
              [K ~ M, [L,N] S ack(K) [P ; J, M + 1],
                      [L,N] S ack(K) [P,M]] .
rl [ack-recv] : [J ; L,N] ack(K) S [P,M] =>
                  [K ~ N,[L,N + 1] S [P,M],
                         [J ; L,N] S [P,M]] .

crl [loss] : [L,N] S S' [P,M] => [L,N] S' [P,M] if S =/= null .
endm
```

The sender (resp. receiver) is located at the left (resp. right) side of the channel and has a buffer storing a list of numbers and a counter. The channel is a multiset of messages modeling out-of-order communication; and is lossy, as modeled by the `[loss]` rule. Fault-tolerant in-order communication is ensured by: (i) sending messages of the form `{J,N}` with `J` the number being sent and `N` the value of the sender's counter, (ii) the receiver sending acknowledgements, and (iii) the sender beginning to send the next item only after receipt of the previous one has been acknowledged. Because of the `[send]` rule, the number of messages in the channel is *unbounded*, so standard LTL model checking is impossible. For this reason, the above module specifies an *equational abstraction* [21], where the contents of the channel becomes a *set* thanks to the idempotency equation `S S = S`, so that LTL model checking becomes possible.

However, the above specification is *not* ground coherent, and therefore any LTL model checking would be incorrect. This lack of coherence occurs for two

different reasons: (1) even without the set idempotency abstraction, the *identity* equations for list concatenation and for multiset union cause lack of coherence; and (2) to make things worse, the idempotency equation used in the abstraction causes additional coherence problems. All these coherence problems can be solved automatically by the $\mathcal{R} \mapsto \overline{\mathcal{R}}_l$ transformation. Specifically, in this case $\mathcal{R}$ is a generalized rewrite theory $\mathcal{R} = (\widehat{\Sigma}, E \uplus B, R, th(T_{\widehat{\Sigma}/E \uplus B}), \phi)$, in the sense of Def. 6, where: (i) its signature Ω of constructors is specified by the operators declared with the `ctor` keyword, plus the `true` and `false` constants in the imported `BOOL` module; (ii) the frozenness function ϕ just freezes the if-then-else operator with the `frozen` keyword; and (iii) at the equational level, there are protecting inclusions:

$$(\Omega, B_\Omega, \vec{E}_\Omega) \subseteq (\Sigma_1, B_1, \vec{E}_1) \subseteq (\Sigma, B, \vec{E})$$

where $B_\Omega = B_1$ are the equational axioms declared by the `assoc` and/or `comm` keywords, $\vec{E}_\Omega$ are the identity equations for concatenation and union, and the idempotency equation and its B_Ω-coherence extension, $\vec{E}_1$ adds to $\vec{E}_\Omega$ the equations for the if-then-else operator, and $(\Sigma, B, \vec{E})$ adds additional functions symbols, equations and axioms in the imported `BOOL` module (see [4]). Note that, due to the negative condition in the `[loss]` rule, this is indeed a *generalized* rewrite theory.

The key point is that $(\Sigma_1, B_1, \vec{E}_1)$ is FVP, a fact that can be easily checked in Maude. Therefore the $\mathcal{R} \mapsto \overline{\mathcal{R}}_l$ transformation is well defined. Specifically, by computing variants of the lefthand sides using Maude, the coherence completion adds to the rules R in the module the following rules:

```
rl [send] : [J,N] S [P,M] => [J,N] {J,N} S [P,M] .

rl [recv] : [L,N] {J,K} [P,M] =>
              [(K ~ M), [L,N] ack(K) [P ; J, M + 1],
                      [L,N] ack(K) [P,M]] .
rl [recv] : [L,N] {J,K} [P,M] =>
              [K ~ M, [L,N] {J,K} ack(K) [P ; J, M + 1],
                      [L,N] {J,K} ack(K) [P,M]] .
rl [recv] : [L,N] {J,K} S [P,M] =>
              [K ~ M, [L,N] {J,K} S ack(K) [P ; J, M + 1],
                      [L,N] {J,K} S ack(K) [P,M]] .

rl [ack-recv] : [J,N] ack(K) S [P,M] =>
                  [K ~ N,[nil,N + 1] S [P,M],
                         [J,N] S [P,M]] .
rl [ack-recv] : [J ; L,N] ack(K) [P,M] =>
                  [K ~ N,[L,N + 1] null [P,M],
                         [J ; L,N] null [P,M]] .
rl [ack-recv] : [J,N] ack(K) [P,M] =>
                  [K ~ N,[nil,N + 1] null [P,M],
```

```
                          [J,N] null [P,M]] .
rl [ack-recv] : [J ; L,N] ack(K) [P,M] =>
                  [K ~ N,[L,N + 1] ack(K) [P,M],
                          [J ; L,N] ack(K) [P,M]] .
rl [ack-recv] : [J ; L,N] ack(K) S [P,M] =>
                  [K ~ N,[L,N + 1] ack(K) S [P,M],
                          [J ; L,N] ack(K) S [P,M]] .
rl [ack-recv] : [J,N] ack(K) [P,M] =>
                  [K ~ N,[nil,N + 1] ack(K) [P,M],
                          [J,N] ack(K) [P,M]] .
rl [ack-recv] : [J,N] S ack(K) [P,M] =>
                  [K ~ N,[nil,N + 1] S ack(K) [P,M],
                          [J,N] S ack(K) [P,M]] .

crl [loss] : [L,N] S' [P,M] => [L,N] S' [P,M]  if null =/= null .
crl [loss] : [L,N] S [P,M] => [L,N] null [P,M] if S =/= null .
crl [loss] : [L,N] S [P,M] => [L,N] S [P,M] if S =/= null .
crl [loss] : [L,N] S S1 S' [P,M] => [L,N] S1 S' [P,M]
                                                 if S S1 =/= null .
crl [loss] : [L,N] S S' [P,M] => [L,N] S' [P,M] if S S' =/= null .
crl [loss] : [L,N] S S' [P,M] => [L,N] S S' [P,M] if S =/= null .
```

Some of them, namely, the new `[send]` rule, the first new `[recv]` rule, the first three new `[ack-recv]` rules, and the first two new `[loss]` rules would be needed for coherence *even without the idempotency abstraction*. The remaining rules are needed for coherence due to that abstraction. Of course, the rule

```
crl [loss] : [L,N] S' [P,M] => [L,N] S' [P,M]  if null =/= null .
```

has an unsatisfiable condition and can therefore be dropped.

Example 4. Consider the following Maude specification of a bank account:

```
fmod BOOL-FVP is protecting TRUTH-VALUE .
  op _/\_ : Bool Bool -> Bool .
  op _\/_ : Bool Bool -> Bool .
  op ~ : Bool -> Bool .
  vars B X Y Z : Bool .
  eq true /\ B = B  [variant] .
  eq false /\ B = false [variant] .
  eq false \/ B = B  [variant] .
  eq true \/ B = true [variant] .
  eq ~(true) = false [variant] .
  eq ~(false) = true [variant] .
endfm

fmod NAT-PRES-MONUS is protecting BOOL-FVP .
    sort Nat .
    ops 0 1 : -> Nat [ctor] .
```

```
    op _+_ : Nat Nat -> Nat [ctor assoc comm id: 0] .

    vars n n' m x y x' y' : Nat . vars b b' : Bool .

    op _>_ : Nat Nat -> Bool .
    op _>=_ : Nat Nat -> Bool .

    eq m + n + 1 > n = true [variant] .
    eq n > n + m = false [variant] .

    eq m + n >= n =  true [variant] .
    eq n >= m + n + 1 = false [variant] .

    op _-_ : Nat Nat -> Nat .  *** monus

    eq n - (n + m) = 0 [variant] .
    eq (n + m) - n = m [variant] .
endfm

mod BANK-ACCOUNT is protecting NAT-PRES-MONUS .
    sorts Account Msg MsgConf State StatePair .
    subsort  Msg < MsgConf .

    op < bal:_pend:_overdraft:_> : Nat Nat Bool -> Account [ctor] .
    op mt : -> MsgConf [ctor] .
    op withdraw : Nat -> Msg [ctor] .
    op _,_ : MsgConf MsgConf -> MsgConf [ctor assoc comm id: mt] .
    op _#_ : Account MsgConf -> State [ctor] .  *** state ctor
    op [_,_,_] : Bool State State -> State .  *** if-then-else

    vars n n' m x y x' y' : Nat . vars b b' : Bool .
    vars s s' : State . var msgs : MsgConf .

    eq [true,s,s'] = s [variant] .
    eq [false,s,s'] = s' [variant] .

     *** requesting to draw money having sufficient funds; the amount
     *** requested is added to the amount of pending withdraw requests

    rl [w-req] : < bal: n + m + x pend: x overdraft: false > # msgs =>
       < bal: n + m + x pend: x + m overdraft: false > # withdraw(m),msgs .

     *** actual withdrawing of money from account

    rl [w] :  < bal: n pend: x overdraft: false > # withdraw(m),msgs =>
         [ m > n , < bal: n pend: x overdraft: true > # msgs ,
         < bal: (n - m) pend: (x - m) overdraft: false > # msgs ]  .

    *** more money can at any time be deposited in the account if it is not
    *** in overdraft
```

```
    rl [dep] :  < bal: n pend: x overdraft: false > # msgs =>
                          < bal: n + m pend: x overdraft: false > # msgs .
endm
```

An account's state has the form `< bal: n pend: x overdraft: b > # msgs` where `n` is the current balance; `x` is the amount of money that is currently *pending to be withdrawn* due to previous `withdraw(m)` messages; we can think of such messages as writing of checks, requesting wire transfers, etc.; `b` is a Boolean flag indicating whether or not the account is in the red (if it is, it gets blocked in the sense that no rule can be applied); and `msgs` is a multiset of such withdrawal messages awaiting withdrawal. The meaning of the three rules is explained in the comments. Note that the deposit rule `[dep]` has an extra variable `m` on the righthand side and models a non-deterministic environment form which new money can arrive to the account. Therefore, `BANK-ACCOUNT` models an *open system* in the sense of Section 3. It is not executable in the standard Maude sense, but is *symbolically executable* in Maude by narrowing with the rules modulo the equations (more on this later).

Note that, at the equational level, we have protecting inclusions:

$$(\Omega, B_\Omega, \vec{E}_\Omega) \subseteq (\Sigma_1, B_1, \vec{E_1}) \subseteq (\Sigma, B, \vec{E})$$

where the signature Ω of constructors has the `true` and `false` constats in the imported module `TRUTH-VALUE`, plus the operators declared with the `ctor` keyword, and $B_\Omega = B_1 = B$ are ACU axioms for $+$ and for multiset union. Therefore, $\vec{E}_\Omega = \varnothing$, that is, these are *free* constructors modulo ACU. In this case, we furthermore have $(\Sigma_1, B_1, \vec{E_1}) = (\Sigma, B, \vec{E})$, so that $E_1 = E$ defines all the remaining non-constructor functions and can be oriented as convergent rules modulo ACU. The key point is that $(\Sigma_1, B_1, \vec{E_1}) = (\Sigma, B, \vec{E})$ is FVP, as can easily be checked in Maude. This means that the rewrite theory $\mathcal{R}$ specified by `BANK-ACCOUNT` satisfies the input requirements for the $\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ transformation. By computing the *constructor variants* of the pairs $\langle l, r \rangle$ for the left- and right-hand sides l, r of the above three rules, we get the transformed module:

```
mod BANK-ACCOUNT-CTOR is protecting NAT-PRES-MONUS .
    sorts Account Msg MsgConf State StatePair .
    subsort  Msg < MsgConf .

    op < bal:_pend:_overdraft:_> : Nat Nat Bool -> Account [ctor] .
    op mt : -> MsgConf [ctor] .
    op withdraw : Nat -> Msg [ctor] .
    op _,_ : MsgConf MsgConf -> MsgConf [ctor assoc comm id: mt] .
    op _#_ : Account MsgConf -> State [ctor] .  *** state constructor
    op [_,_,_] : Bool State State -> State [frozen] . *** if-then-else

    vars n n' m x y x' y' : Nat . vars b b' : Bool .
    vars s s' : State . vars msgs msgs' : MsgConf .
```

```
    eq [true,s,s'] = s [variant] .
    eq [false,s,s'] = s' [variant] .

     *** requesting to draw money having sufficient funds; the amount
     *** requested is added to the amount of pending withdraw requests

    rl [w-req] : < bal: n + m + x pend: x overdraft: false > # msgs =>
        < bal: n + m + x pend: x + m overdraft: false > # withdraw(m),msgs .

     *** actual withdrawing of money from account
     *** (done with ctor variants of original rule)

    rl [w] : < bal: n + m + x pend: m overdraft: false >
             # msgs,withdraw(m + x)
      =>
     < bal: n pend: 0 overdraft: false > # msgs .


    rl [w] : < bal: n + m pend: m + x overdraft: false >
             # msgs,withdraw(m)
     =>
     < bal: n pend: x overdraft: false > # msgs .

    rl [w] : < bal: n pend: y overdraft: false >
             # msgs',withdraw(1 + n + x)
     =>
     < bal: n pend: y overdraft: true > # msgs' .

    *** more money can at any time be deposited in the account if it
    *** is not in overdraft

    rl [dep] : < bal: n pend: x overdraft: false > # msgs =>
                     < bal: n + m pend: x overdraft: false > # msgs .
endm
```