Journal of Logical and Algebraic Methods in Programming 110 (2020) 100483

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Generalized rewrite theories, coherence completion, and symbolic methods

José Meseguer

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

ARTICLE INFO

Article history:
Received 19 December 2018
Received in revised form 16 August 2019
Accepted 18 August 2019
Available online 21 August 2019

Keywords:
Generalized rewrite theories
Coherence
Pattern predicates
Constrained narrowing
Symbolic invariant verification

ABSTRACT

A new notion of generalized rewrite theory suitable for symbolic reasoning and generalizing the standard notion in [19] is motivated and defined. Also, new requirements for *symbolic executability* of generalized rewrite theories that extend those in [33] for standard rewrite theories, including a generalized notion of *coherence*, are given. Symbolic executability, including coherence, is both ensured and made available for a wide class of such theories by automatable *theory transformations*. Using these foundations, several *symbolic reasoning methods* using generalized rewrite theories are studied, including: (i) symbolic description of sets of terms by *pattern predicates*; (ii) reasoning about universal reachability properties by *generalized rewriting*; (iii) reasoning about existential reachability properties by *constrained narrowing*; and (iv) symbolic verification of *safety properties* such as invariants and stability properties.

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., [35], (ii) logical LTL model checking, e.g., [38,9,10], (iii) rewriting modulo SMT and related approaches, e.g., [87,7], (iv) inductive theorem proving and program verification, e.g., [43,61], and (v) reachability logic theorem proving, e.g., [92,64,91]. One key issue is that the rewrite theories used in several of these approaches go *beyond* the standard notion of rewrite theory in, say [19], and also beyond the *executability requirements* in, say, [33]. 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 [87]. Furthermore, each of the approaches just mentioned makes *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 (e.g., in narrowing, extra variables in righthand sides of rules are unproblematic), in other ways symbolic execution imposes *strong restrictions* on the rewrite rules to be executed. For example, 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 assume and require that the rewrite theories we are going to symbolically execute are *coherent* [98,33]. But no theory of coherence, or methods for guaranteeing it, have yet been developed for these new kinds of theories.

E-mail address: meseguer@uiuc.edu.

https://doi.org/10.1016/j.jlamp.2019.100483
2352-2208/

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 begins by asking, and providing 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, an *as wide as possible* class of theories?

Questions (1) and (2) are answered as follows. Question (1) is answered in Section 3, which 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. Question (2) is then answered by using 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*? The answer to question (i) is new even for standard rewrite theories and can be quite useful to semi-automate equational abstractions [75]. The answer to question (ii) is very general: under mild conditions symbolic executability can be ensured for a very wide class of generalized rewrite theories by two theory transformations.

Once answers to the above foundational questions (1)–(2) have been given, one can ask, and provide answers to, the following high-level question: (3) What suitable *symbolic methods* can be developed to reason about generalized rewrite theories, including properties satisfied by the *initial models* of such theories? Since different methods are possible, the following answers are given:

1. Since the symbolic reasoning involved is about the behavior of the concurrent system specified by a generalized rewrite theory $\mathcal{R}$, the first order of business is to find a *language of state predicates* amenable to symbolic reasoning with the rewrite theory $\mathcal{R}$. The language proposed in Section 5 for this purpose further develops the language of *pattern predicates* proposed in [91], whose atomic formulas are constrained terms $u \mid \varphi$, where u is a constructor term[1] and φ is a quantifier-free (QF) formula, so that $u \mid \varphi$ describes the ground instances of u that satisfy the constraint φ.
2. Rules in a generalized rewrite theory $\mathcal{R}$ are conditional rules of the form $l \to r$ *if* φ, where φ is a QF formula. In Section 6.1 rewriting with such rules is shown to be sound and complete to answer universal reachability questions of the form $\mathcal{R} \models (\forall Y)\, t \to^* t'$, with Y the variables in t, t'. Due to the presence of extra variables in r and/or φ, execution of such rewrite rules may be hard to mechanize; therefore, sufficient conditions making this kind of generalized rewriting decidable are also given.
3. For symbolic model checking applications the crucial question is how to solve *existential reachability* problems of the form $\mathcal{R} \models \exists(A \to^* B)$, where A and B are pattern predicates in the sense explained above and $\exists$ abbreviates the existential closure of the reachability formula. That is, we want to know if there is a concrete state satisfying predicate A from which a concrete state satisfying B can be *reached*. For example, $\mathcal{R}$ may specify a cryptographic protocol, A may specify its initial states, and B may specify a class of *attack states*. In Section 6.2 I first show that for the wide class of rewrite theories characterized in Section 4.3, rewriting with $\mathcal{R}$ defines a *predicate transformer* $R!$ on the power set of the set of states, which can be *effectively computed* when such sets of states are definable by pattern predicates. I then show that such effective description of $R!$ in fact *coincides* with the notion of *constrained narrowing* also defined in Section 4.3, which therefore provides a sound and complete symbolic method for existential reachability analysis.
4. Since the most common symbolic model checking problems involve the verification of *invariants*, in Section 6.3 I explain in detail how *invariants* and their complements (which I call *coinvariants*), including the case of *inductive* invariants and coinvariants, can be analyzed and sometimes fully verified by the just-described constrained narrowing method.

One might wonder why a discussion of *rewriting modulo SMT* [86,87] has not been included in the above list of symbolic methods. The reasons are twofold. Firstly, the journal paper [87] already provides a detailed explanation of rewriting modulo SMT. Secondly, as I briefly explain in Section 7, rewriting modulo SMT can be naturally understood as a *special, restricted case* of the more general constrained narrowing method, which implicitly subsumes it. This does not decrease the usefulness of rewriting modulo SMT, since its implementation can be substantially more efficient than that of constrained narrowing.

Outline. Section 2 gathers preliminaries. Section 3 defines generalized rewrite theories, their categories of models, including initial ones, and studies in detail the coherence problem for such theories. Section 4 then defines several theory transformations that can automatically ensure coherence. The language of pattern predicates and its computability properties are presented in Section 5. The last three symbolic methods described above are then presented in Section 6. Related work and conclusions are discussed in Section 7. Proofs are relegated to Appendix A.

Comparison with the Conference Paper [72]. This paper substantially extends the conference paper [72] in the following ways:

[1] A constructor term u does not include equationally defined functions like $*$ for number multiplication, or *append* for lists, but only "data constructor" operators such as 0 and s for natural numbers in Peano notation, or *nil* and *cons* for lists. Constructors are explained in detail in Section 2.

1. Sections 5–6 on pattern predicates and on symbolic methods are entirely new and contain many new concepts and results.
2. Sections 1, 3 and 7 and the list of References have been substantially expanded.
3. Many fully developed examples are presented.
4. Proofs of all, previous and new, results are included.

2. Preliminaries on order-sorted algebra and variants

I present needed preliminaries on order-sorted algebra, logic, and variants. The material is adapted from [70,73]. The presentation is self-contained: only the notions of many-sorted signature and many-sorted algebra, e.g., [34], are assumed.

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

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

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

denotes the family of "subsort polymorphic" operators f. We can extend any $\Sigma = (S, \leq, \Sigma)$ to its *kind completion* $\widehat{\Sigma} = (S \uplus \widehat{S}, \widehat{\leqq}, \widehat{\Sigma})$ where: (i) $\widehat{\leqq}$ is the least partial order extending $\leq$ 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 $\Sigma = (S, \leq, \Sigma)$ an OS signature, an *order-sorted Σ-algebra* A is a many-sorted (S, Σ)-algebra A such that:

- whenever $s \leq s'$, 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 $A^{s_1\ldots s_n} = A_{s_1} \times \ldots \times A_{s_n}$.

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. *[70] 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 $s \leq s'$ then $t \in T_{\Sigma,s'}$,*
- *if $f : s_1 \ldots s_n \to s$ and $t_i \in T_{\Sigma,s_i}$ $1 \leq i \leq n$, 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 $s \in S$, $T_{\Sigma,s} \neq \emptyset$. An OS signature Σ is called *preregular* [51] 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 $s \neq s' \Rightarrow X_s \cap X_{s'} = \emptyset$, and the variables in X are always assumed disjoint from all constants in Σ. The *Σ-term algebra* on variables X, $T_{\Sigma}(X)$, is the *initial algebra* for the signature $\Sigma(X)$ obtained by adding to Σ the variables X *as extra constants*. Since a $\Sigma(X)$-algebra is just a pair (A, α), with A a Σ-algebra, and α an *interpretation of the constants* in X, i.e., an S-sorted function $\alpha \in [X \to A]$, the $\Sigma(X)$-initiality of $T_{\Sigma}(X)$ can be expressed as the following theorem:

Theorem 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. $QFForm(\Sigma)$ denotes the set of QF formulas. 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 $\Sigma \setminus \Omega$. 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 $\varphi \in Form(\Sigma)$ 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 $T' \subseteq T$. Given a theory $T' = (\Sigma', \Gamma')$ and a subsignature $\Sigma \subseteq \Sigma'$, the theory $T'|_\Sigma$ is, by definition, the pair $(\Sigma, \{\varphi \in Form(\Sigma) \mid \Gamma' \models \varphi\})$. The following two facts follow easily from this definition: (i) T' is a conservative extension of $T'|_\Sigma$, and (ii) if T' is a conservative extension of $T = (\Sigma, \Gamma)$, then $T \equiv T'|_\Sigma$.

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* $T_{\Sigma/E}(X)$ [70]. The inference system in [70] 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 an OS equational theory (Σ, E), an E-*unifier* of a system of Σ-equations, i.e., a conjunction $\phi = u_1 = v_1 \wedge \ldots \wedge u_n = v_n$ of Σ-equations is a substitution σ such that $u_i\sigma =_E v_i\sigma$, $1 \leq i \leq n$. An E-*unification algorithm* for (Σ, E) is an algorithm generating a *complete set* of E-unifiers $Unif_E(\phi)$ for any system of Σ equations ϕ, where "complete" means that for any E-unifier σ of ϕ there is a $\tau \in Unif_E(\phi)$ and a substitution ρ such that $\sigma =_E (\tau\rho)|_{dom(\sigma)\cup dom(\tau)}$, where $=_E$ here means that for any variable x we have $x\sigma =_E x(\tau\rho)|_{dom(\sigma)\cup dom(\tau)}$. The algorithm is *finitary* if it always terminates with a *finite set* $Unif_E(\phi)$ for any ϕ. Likewise, given a sort k that is the top of one of the connected components of the poset of sorts $(S, \leq)$ of the signature Σ, and given a finite set of terms $\{t_1, \ldots, t_n\} \subseteq T_\Sigma(X)_k$ with $n \geq 2$, a E-*unifier* of $\{t_1, \ldots, t_n\}$ is a substitution σ such that for each $1 \leq i < j \leq n$, $t_i\sigma =_E t_j\sigma$. The notion of a *complete set* of E-unifiers $Unif_E(\{t_1, \ldots, t_n\})$ for such a set of terms $\{t_1, \ldots, t_n\}$ is entirely analogous to that for a system of equations. The same E-unification algorithm can solve both systems of equations and sets of terms of the same kind.

Given a set of equations B used for deduction modulo B, a preregular OS signature Σ is called B-*preregular*[2] iff for each $u = v \in B$ and substitution ρ, $ls(u\rho) = ls(v\rho)$.

Recall the notation for term positions, subterms, and term replacement from [31]: (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 [73,65] 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 $\vec{E} = \{t \to t' \mid (t = t') \in E\}$, have $vars(t') \subseteq vars(t)$, and are *convergent* modulo B, that is, sort-decreasing, strictly B-coherent, confluent, and terminating as rewrite rules modulo B [65],

then we call the rewrite theory $\mathcal{R} = (\Sigma, B, \vec{E})$ (in the sense of [19]) 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[3] 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) [55] Let $\mathcal{R} = (\Sigma, B, \vec{E})$ be a decomposition of $(\Sigma, E \uplus B)$. Then we have an equivalence:*

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

[2] 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 $u = v \in U \Rightarrow ls(u\rho) \geq ls(v\rho)$ for each ρ. Maude automatically checks B-preregularity of an OS signature Σ in this broader sense [24].

[3] See [90] 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 2.

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

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 $f : s_1 \dots s_n \rightarrow s$ 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.

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 $C_{\Omega/\vec{E}_\Omega,B_\Omega} = T_{\Omega/B_\Omega}$.

Example 1. The following Maude specification of lists of natural numbers illustrates the notions of: (i) a decomposition protection another, and (ii) a constructor decomposition.

```
fmod NAT-LIST is protecting BOOL-OPS .
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
```

The Maude notation for functional modules (introduced with keywords `fmod ... endfm`) is just a typewriter version of an order-sorted equational specification. An equational theory is always named (in this case it is called `NAT-LIST`). It may import other theories as subtheories (in this case the subtheory `BOOL-OPS` of Boolean operations in Maude's standard prelude). The keywords are self-explanatory: sorts are declared with the `sorts` keyword; the sort order $\leq$ is the transitive-reflexive closure of the binary relation declared by the `subsorts` keyword. Variables to be used in equations can be declared with their appropriate sorts using the `vars` keyword. Equations are declared with the `eq` keyword. The function symbols defining the module's signature Σ are declared with the `op` keyword. Each such declaration is started by its keyword and finished with a blank space and a period. All Maude functional modules *are* in fact decompositions of equational theories in the following sense: the theory $(\Sigma, E \cup B)$ has the decomposition $(\Sigma, B, \vec{E})$, where B are *associativity* and/or *commutativity* and/or *identity* element axioms that are respectively declared with the `assoc`, `comm`, and `id:` keywords for their corresponding operators. Instead, the equations E are declared with the `eq` keyword. Specifically: (i) natural number addition `_+_` is associative and commutative and has `0` as its identity element; the equality predicate on natural numbers `_~_` is commutative; and list concatenation `_;_` is associative. The `protecting BOOL-OPS` declaration states that the decomposition `NAT-LIST` protects that of `BOOL-OPS` in the sense of Definition 3. This is intuitively obvious for two reasons: (1) no new equations for the Boolean operations in `BOOL-OPS` are added in `NAT-LIST` (no confusion is caused between terms in the submodule), and (2) because of commutativity, the two equations for the equality predicate `_~_` fully define it for all pairs of natural numbers (no "junk" is added to the `Bool` sort by this new predicate), so that `NAT-LIST` adds *no junk and no confusion* to `BOOL-OPS`, which is the key idea about protecting inclusions.

Thanks to the `ctor` operator attribute, the subsignature $\Omega \subseteq \Sigma$ of constructors for `NAT-LIST` can be read off from the module's text. Therefore, Ω consists of: (i) the constructors for `BOOL-OPS`, namely, `true` and `false`, (ii) the `0` and `1`

numbers, (iii) the _+_ operator; (iv) the nil empty list, and (v) the list concatenation operator _;_. The axioms B_Ω are those of B minus the commutativity of the equationally defined equality predicate symbol _~_. Finally, the equations E_Ω are the last three equations in the module (the last one is added as a so-called strict B-coherence extension of the previous two, in the sense explained later in Section 3.1). Then, the decomposition inclusion $(\Omega, B_\Omega, \vec{E_\Omega}) \subseteq (\Sigma, B, \vec{E})$ is a protecting inclusion and a *constructor decomposition* of NAT-LIST in the sense of Definition 3. Symbols in $\Sigma \setminus \Omega$ are called *defined* function symbols. In NAT-LIST, besides the Boolean operations in BOOL-OPS, _~_ is the only such symbol.

The only still unclear issue is the meaning of the variant attribute for all the equations in NAT-LIST. This is just declared to allow Maude to compute *variants* of terms in NAT-LIST, in the sense explained right below.

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* [28,39] 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* [28] (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*[4] [39].

Example 2. Disregarding the Boolean operations in BOOL-OPS, the decomposition of NAT-LIST is FVP (but see below). Since the Boolean operations in BOOL-OPS are never used in the communication channel modules FT-CHANNEL and FT-CHANNEL-ABS of Example 5 that import NAT-LIST and, furthermore, both the decomposition of NAT-LIST and that of the equational part of FT-CHANNEL-ABS extending it are FVP, the computation of the variants $[\![l]\!]_{\vec{E},B}$ for l each lefthand side of a rule in FT-CHANNEL-ABS will allow us in Example 5 to compute the so-called coherence completion of FT-CHANNEL-ABS. This, in turn, will make it possible to automatically verify LTL properties of FT-CHANNEL-ABS by explicit-state model checking. For concrete examples of finite sets of variants of the form $[\![l]\!]_{\vec{E},B}$ for l a rule's lefthand side, please see Example 5.

A perceptive reader might have a lingering doubt: list concatenation _;_ is associative. But associative unification is in general infinitary. So, how is $[\![l]\!]_{\vec{E},B}$ going to yield a finite number of variants? The answer is that: (i) many practical associative unification problems do have a *finite*, complete set of solutions, and (ii) all the unification problems involved in computing the variant sets $[\![l]\!]_{\vec{E},B}$ for lefthand sides of FT-CHANNEL-ABS rules fall within this finitary set of solutions case. Therefore, a more careful statement is that NAT-LIST "is" FVP for all terms t such that all the associative unification problems arising when computing $[\![t]\!]_{\vec{E},B}$ have finite, complete sets of unifiers. We could say that NAT-LIST is FVP *in practice* for many commonly occurring variant computation problems.

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). [73] 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 [90], which has been implemented in Maude. Constructor variant sets of the form $[\![\langle l, r\rangle]\!]^\Omega_{\vec{E},B}$, where l and r are, respectively, the lefthand and righthand sides of a rewrite rule in a rewrite theory $\mathcal{R}$ play a crucial role in the theory transformation

[4] Maude 2.7.1 supports the computation of $[\![t]\!]_{\vec{E},B}$ by folding variant narrowing for B a combination of associative and/or commutative and/or identity axioms.

$\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma,l,r}$ of Section 4.3. Please, see the BANK-ACCOUNT example in Section 4.3 for examples of constructor variant sets of the form $[\![\langle l, r\rangle]\!]^{\Omega}_{E,B}$.

3. Generalized rewrite theories and coherence

There are two main reasons for further generalizing the notion of rewrite theory in [19], and for relaxing its *executability conditions* as specified in, e.g., [33]. 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 not always executable in the standard sense, can still be executed *symbolically* under fairly reasonable assumptions. For example, the notion of *rewriting modulo SMT* [87] (see also the related work [7]) 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 (discussed in Section 7) are also given in [43,61].

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* [98,33] 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 Definition 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[5]; (iv) $T = (\Delta, \Gamma)$, called the *background theory*, satisfies: (a) $\Sigma \subseteq \Delta$, (b) $(\Sigma, G) \subseteq T|_{\Sigma} \subseteq th(T_{\Sigma/G})$, (c) for each ground $\varphi \in QFForm(\Sigma)$, i.e., such that $vars(\varphi) = \emptyset$, $T_{\Sigma/G} \models \varphi \Leftrightarrow T \models \varphi$; and (v) ϕ is a so-called *frozenness* function,[6] 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 $\to_{\mathcal{R}}$ holds between them, denoted $u \to_{\mathcal{R}} v$, iff there exist a term u', a ϕ-*unfrozen*[7] 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) $u =_G u'[l\theta]_p$; 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 $l \to r$ *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 $u \to_{\mathcal{R}} v \Rightarrow u, v \in T_{\Sigma,[State]}$.

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) $\to_{\mathcal{R}} = \to_{\mathcal{R}'}$ (resp. (1) $T_{\Sigma/G} = T_{\Sigma/G'}$, (2) $T \equiv T'$, and (3) $\to_{\mathcal{R}}|_{T^2_{\Sigma}} = \to_{\mathcal{R}'}|_{T^2_{\Sigma}}$).

The case of a *standard rewrite theory* is the special case of theories $\mathcal{R} = (\Sigma, G, R, T, \phi)$ where: (1) condition (iv)-(c) in Definition 6 applies only to *positive* (i.e., negation-free) ground QF Σ-formulas φ, (2) $T = (\Sigma, G)$, and (3) for each $l \to r$ *if* φ in R, φ is a conjunction of equalities[8] $\varphi = \bigwedge_{i=1\ldots n} u_i = v_i$. 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 $T_0 = (\Sigma_0, \Gamma_0)$ 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 such that $T|_{\Sigma_0} \equiv T_0$. The possibility that the background theory $T = (\Delta, \Gamma)$ may have extra operators, so that $\Delta \supset \Sigma$, can be nicely illustrated by the case when (Σ, G) has a decomposition that *protects* a constructor decomposition $(\Omega, B, \emptyset)$, where B is a combination of associativity and/or

[5] This is of course a pragmatic decision: satisfiability procedures for QF formulas do not have to pay the typically quite expensive price of quantifier elimination. All I say in this work about generalized rewrite theories can be easily extended to allow rules whose conditions can be arbitrary first-order Σ-formulas.

[6] 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.

[7] 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'.

[8] Admittedly, one can define generalized rewrite theories with even more general rules having additional "rewrite conditions," i.e., rules of the form $l \to r$ *if* $\varphi \wedge \bigwedge_{i=1\ldots n} u_i \to v_i$. Then, generalized rewrite theories would specialize to standard rewrite theories whose rules also allow rewrite conditions. I leave this further generalization as future work.

commutativity axioms. Then, we can choose T to be the theory obtained by adding to the *equality enrichment* $(\Sigma^{\equiv}, G^{\equiv})$ [53] of (Σ, G), for each top sort $s_\top \in [s]$ of each connected component $[s]$ of Σ, the axioms $x = y \Leftrightarrow x \equiv y = true$ and $x \neq y \Leftrightarrow x \equiv y = false$, with x, y of sort $s_\top$. T then satisfies conditions (iv)–(a)–(c) in Definition 6.

Example 3. This QLOCK protocol example is borrowed from [91]. It is a dynamic, open system generalization of the original QLOCK in [44]. 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 Definition 6, where ϕ maps each $f \in \widehat{\Sigma}$ to $\emptyset$ (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 [44], where (Σ, B) defines the protocol's 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 $F = \{0 : \to Nat,\ s\_ : Nat \to Nat,\ \emptyset : \to MSet,\ nil : \to List,\ \_\ \_ : MSet\ MSet \to MSet,\ \_;\_ : List\ List \to List, dupl : MSet \to Pred, tt : \to Pred, \_|\_|\_|\_ : MSet\ MSet\ MSet\ List \to Conf, < \_ > : Conf \to State\}$, where underscores denote operator argument placement. The axioms B are the associativity-commutativity of the multiset union $\_\ \_$ with identity $\emptyset$, 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 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.

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

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 $h : (A, \to_A) \to (B, \to_B)$ is an isomorphism in this category iff: (i) h is a Σ-isomorphism, and (ii) $b \to_{B_{[s]}} b'$ implies $h^{-1}(b) \to_{A_{[s]}} h^{-1}(b')$. Intuitively, such an isomorphism could be called a "bijective algebraic bisimulation," and a homomorphism an "algebraic simulation map."

Given a generalized rewrite theory $\mathcal{R} = (\Sigma, G, R, T, \phi)$ we say that a Σ-transition system $(A, \to_A)$ *satisfies* the theory $\mathcal{R}$, denoted $(A, \to_A) \models \mathcal{R}$ iff: (i) $A \in \mathbf{OSAlg}_{(\Sigma,G)}$, and (ii) for each $\alpha \in [X \to A]$ the unique Σ-homomorphism $\_\alpha : T_{\Sigma/G}(X) \to A$ is a Σ-transition system homomorphism $\_\alpha : \mathcal{T}_\mathcal{R}(X) \to (A, \to_A)$. This defines a full subcategory $\mathbf{Trans}_\mathcal{R} \subseteq \mathbf{Trans}_\Sigma$.

Lemma 1. *$\mathcal{T}_\mathcal{R}$ is the initial object of* $\mathbf{Trans}_\mathcal{R}$.

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 $\mathcal{R}$ [19], whose associated Σ-transition system is the transitive closure $(T_{\Sigma/G}, \to^*_\mathcal{R})$ of $\mathcal{T}_\mathcal{R}$. Roughly speaking, $\mathcal{T}_\mathcal{R}$ is the "one step rewrite" fragment of the initial reachability model in [19].

Definition 6 is very general; in a sense too much so: 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., [33,65], 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 [65]. Under these extra assumptions on $\mathcal{R}$, a much simpler rewrite relation $\rightarrow_{R/B}$ with the rules R modulo B as well as a much simpler Σ-transition systems can be defined:

Definition 8. ($\rightarrow_{R/B}$ Relation and 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.

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) $v = u'[r\theta]_p$.

The Σ-transition system $\mathcal{C}_{\mathcal{R}}(X)$ (resp. $\mathcal{C}_{\mathcal{R}}$) is 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) $[v] = [w!_{\vec{E},B}]$.

3.1. The coherence problem

Note that it follows from the above definition of canonical transition systems 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, $E = \{a = b\}$, $B = \emptyset$, and $R = \{a \rightarrow c\}$, where $\rightarrow_{\mathcal{C}_{\mathcal{R}}} = \emptyset$, but $\rightarrow_{\mathcal{R}} = \{(\{a, b\}, \{c\})\}$. 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 $h : \mathcal{C}_{\mathcal{R}}(X) \rightarrow \mathcal{T}_{\mathcal{R}}(X)$ (resp. $h : \mathcal{C}_{\mathcal{R}} \rightarrow \mathcal{T}_{\mathcal{R}}$) is an isomorphism. Coherence can be characterized by an easier to check condition that generalizes ideas in [98,33]:

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

In both Definition 8 and Theorem 4, a perceptive reader may have noticed a notational discrepancy between two relations: the relation $\rightarrow_{R/B}$ and the relation $\rightarrow_{\vec{E},B}$ already defined in Section 2. One would instead have expected to see the relations $\rightarrow_{R,B}$ and $\rightarrow_{\vec{E},B}$. Let me clarify and resolve this matter by first defining, for any generalized rewrite theory $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$ satisfying the assumptions in Definition 8, the rewrite relation $\rightarrow_{R,B}$. 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 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) $u|_p =_B l\theta$; and (iii) $v = u[r\theta]_p$. Two things to note are that: (1) $\rightarrow_{R,B} \subseteq \rightarrow_{R/B}$, and (2) if $\mathcal{R}$ is topmost, the relations $\rightarrow_{R,B}$ and $\rightarrow_{R/B}$ *coincide*. It is for this second reason, and for the sake of simplicity, that, for the moment, I have stated Definition 8 and Theorem 4 in terms of the relation $\rightarrow_{R/B}$. But, as I show below, both of them can be restated in terms of $\rightarrow_{R,B}$.

A third thing to note is that, for non-topmost rewrite theories, the relation $\rightarrow_{R,B}$ is typically much easier to implement than the relation $\rightarrow_{R/B}$, since we can use a B-matching algorithm to determine whether $u|_p =_B l\theta$ for some position p in u. The fourth and most crucial thing to say is that, by making the rules in R *strictly B-coherent* [71], we can safely *replace* $\rightarrow_{R/B}$ by $\rightarrow_{R,B}$ throughout.

Strict B-Coherence. The word "coherence" has several related, yet distinct, technical meanings. Coherence in the sense shown in Theorem 4 (which more precisely should be called *strong coherence* [98,33] between the rules R and the equations E modulo B), is a certain "commutativity-like" property between application of rules R and application of oriented equations $\vec{E}$ modulo B. Instead, the simpler notion of *strict B-coherence* [71] is the property that the equality relation $=_B$ defines a *bisimulation* (and therefore, for all practical purposes, a semantic equivalence) between the relations $\rightarrow_{R,B}$ and $\rightarrow_{R/B}$. Since we have $\rightarrow_{R,B} \subseteq \rightarrow_{R/B}$ and $=_B; \rightarrow_{R/B} = \rightarrow_{R/B}$, strict B-coherence boils down to the property that if $u \rightarrow_{R/B} v$ and $u =_B u'$ then there exists v' such that: (i) $u' \rightarrow_{R,B} v'$, and (ii) $v =_B v'$. Strict B-coherence is related to a more general, yet less well-behaved, notion of coherence modulo equations in [55].

The usefulness of strict B-coherence can best be illustrated by its absence.

Example 4. (Sorting). Consider the following generalized rewrite theory $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$ where: (i) Σ has sorts *Nat*, *Bool*, *NeList* (non-empty lists), and *List*, subsorts $Nat < NeList < List$, constants $0, 1$ of sort *Nat*, *nil* of sort *List*, *true* and *false* of sort *Bool*, a binary $+$ operator of sort *Nat*, a subsort-overloaded list concatenation operator $\_;\_ : NeList\ NeList \to NeList$, $\_;\_ : List\ List \to List$, and a binary operator $\_ \geq \_ : Nat\ Nat \to Bool$. B are the axioms of associativity, commutativity and unit element 0 for $+$, and of associativity for $\_;\_$, and E are the equations $n + m \geq n = true$, $n \geq n + m + 1 = false$, $l; nil = l$, and $nil; l = l$, where n, m have sort *Nat* and l has sort *List*, R has the single rule[9]:

$$n; m \to m; n \ \textit{if}\ n \geq m = true \wedge n \neq m,$$

T is the theory $th(T_{\Sigma/E \uplus B})$, and $\phi(f) = \emptyset$ for each operator in Σ.

In this theory, the relation $\to_{R,B}$ *fails to be* strictly B-coherent. For example, since $T_{\Sigma/E \uplus B} \models n + 1 \geq n = true \wedge n + 1 \neq n$, and $(0; n + 1); n =_B 0; (n + 1; n)$, we have $(0; n + 1); n \to_{R/B} 0; (n; n + 1)$. However, the term $(0; n + 1); n$ *cannot be rewritten* with the relation $\to_{R,B}$, neither at the top, nor at position 1.

What can be done to make $\to_{R,B}$ strictly B-coherent? To *complete it* by adding to R all its B-*extensions*. The general definition for an arbitrary B having linear and regular equations is given in [71]; but all we need here is the Peterson and Stickel definition [81] for the case where B consists of associativity and/or commutativity axioms, which adds to R the rules:

1. $l; (n; m) \to l; (m; n)$ *if* $n \geq m = true \wedge n \neq m$
2. $(n; m); l' \to (m; n); l'$ *if* $n \geq m = true \wedge n \neq m$
3. $l; ((n; m); l') \to l; ((m; n); l')$ *if* $n \geq m = true \wedge n \neq m$

where l, l' have sort *List*. Call $\overline{R}$ the set obtained by adding to R the above B-extensions. Note that, using rule (1), we can now perform the rewrite step $(0; n + 1); n \to_{R,B} 0; (n; n + 1)$.

I refer the reader to [71] for a detailed treatment of strict B-coherence of conditional rewrite theories. Since we are now considering generalized rewrite theories of the form $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$, a slight generalization of the framework in [71] is needed: (i) we need to generalize the rewrite conditions used in [71] to QF Σ-formulas φ, (ii) we should replace the rewriting satisfaction of conditions by the more abstract condition satisfaction based on the validity check $T \models \varphi\theta$, and (iii) we need to impose the additional requirement of rewriting only at positions p not frozen by the frozenness map ϕ.

How should Definition 8 and Theorem 4 be reformulated in terms of the $\to_{R,B}$ relation? We should just require in both cases that the rules R are strictly B-coherent, that is, that the relation $=_B$ makes $\to_{R,B}$ and $\to_{R/B}$ bisimilar. Then we can replace $\to_{R/B}$ by $\to_{R,B}$ everywhere in Definition 8 and Theorem 4.

This, however, only solves the issue of strict B-coherence. We should not lose track of the different and actually thornier issue of *strong coherence* between rules R and oriented equations $\vec{E}$ modulo B characterized in Theorem 4. How can we ensure that a generalized rewrite theory $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$, whose rules R are strictly B-coherent and where $(\Sigma, B, \vec{E})$ is a convergent rewrite theory is actually strongly coherent (resp. strongly ground coherent)?

There are essentially two approaches. The first is to slightly generalize the methods developed in [33] to *check* the strong coherence (resp. strong ground coherence) of a given generalized rewrite theory $\mathcal{R}$. Such methods are based on suitable *critical pairs* modulo B between conditional rules in R and (oriented) conditional equations in $\vec{E}$ modulo axioms B. The generalization in question is relatively straightforward. The equational conditions (conjunctions of equalities) in rewrite rules considered in [33] need to be generalized to rule conditions that are QF formulas; and the executability conditions in [33] need to be likewise generalized to allow, for example, rewrite rules specifying open systems, such as the *join* rule in Example 3. I refer the reader to [33] for a detailed description of the conditions ensuring strong coherence (resp. ground strong coherence) that need to be generalized. The key theorem to be generalized is Theorem 5 in [33].

Rather than delving into the details of this theorem and its generalization, let me sketch how its natural generalization can be applied to the strictly B-coherent version of the generalized rewrite theory of Example 4 obtained by adding to it the B-extension rules (1)–(3) to get a bigger set $\overline{R}$ of rules. First of all, condition (ii) in Theorem 5 of [33] is easily checked to hold, since the only equations that can have rewrites with rules in $\overline{R}$ below them are the equations $l; nil = l$, and $nil; l = l$, which are both linear. Let me illustrate how condition (i) in Theorem 5 of [33] would be checked by illustrating it with a specific critical pair modulo B between a rule in $\overline{R}$ and an oriented equation in E. For example, the rule $l; (n; m) \to$

[9] It is certainly the case that this rule's condition could have been made simpler by: (i) defining a new predicate $\_ > \_ : Nat\ Nat \to Bool$, and (ii) replacing the QF condition $n \geq m = true \wedge n \neq m$ by the single equality $n > m = true$. My interest, however, is to give examples of generalized rewrite theories whose rules have QF conditions that are not conjunctions of equalities. In this simple example, this extra generality could have been avoided, but not at all easily in the QLOCK example (Example 3), and certainly not without paying a heavy price in many other useful examples. Furthermore, properties crucial for symbolic executability such as the *decidability* of the background theory T (or that of an appropriate reduct of it) can be easily lost in the, now unnecessary, effort to force conditions to be conjunctions of equalities.

$l; (m; n)$ *if* $n \geq m = true \wedge n \neq m$ and the renamed rule $nil; l' = l'$ have a B-unifier of the form $\{l \mapsto nil, l' \mapsto (n; m)\}$ yielding a conditional critical pair of the form:

$$(n \geq m = true \wedge n \neq m) \Rightarrow [n; m \rightarrow m; n]$$

But this conditional critical pair can be discharged with rule $n; m \rightarrow m; n$ *if* $n \geq m = true \wedge n \neq m$ by using the (suitable generalization of) the *context joinability* method explain in Section 4.1 of [33]. Specifically, using the generalized rewrite theory $(\Sigma(\{\overline{n}, \overline{m}\}), E \uplus B, \overline{R}, T \cup \{\overline{n} \geq \overline{m} = true \wedge \overline{n} \neq \overline{m}\}, \phi)$, where we have added $\overline{n}, \overline{m}$ as fresh *constants* and the ground axiom $\overline{n} \geq \overline{m} = true \wedge \overline{n} \neq \overline{m}$ to T, we can now rewrite with $\rightarrow_{\overline{R},B}$ the *ground* term $\overline{n}; \overline{m}$ to $\overline{m}; \overline{n}$ in this extended theory, and therefore discharge the given critical pair. In a similar way, showing that a conditional critical pair $\psi \Rightarrow [u \rightarrow v]$ is *unfeasible* (see Section 4.1 of [33]) for $(\Sigma, E \uplus B, \overline{R}, T, \phi)$ now amounts to proving that ψ is T*-unsatisfiable*.

There is, however, a second, completely different and new, method that can be used to solve the strong ground coherence problem of a *topmost* generalized rewrite theory $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$ under the assumptions in Definition 8, and is applicable also to the special case of a topmost rewrite theory under the assumptions in [33]. The issue is no longer to *check* whether a (generalized) topmost rewrite theory $\mathcal{R}$ is strongly coherent. Instead, the question asked and answered for the first time is: Can we, under suitable conditions, *transform* a generalized topmost rewrite theory $\mathcal{R}$ into a semantically equivalent theory $\overline{\mathcal{R}}_l$, called its *ground coherence completion*, so that $\overline{\mathcal{R}}_l$ 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 [75], 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 $(\Sigma, B, \vec{E})$ a decomposition of $(\Sigma, E \uplus B)$. Furthermore: (i) $\mathcal{R}$ is *topmost*; (ii) there are protecting inclusions of decompositions[10]

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

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 $l \rightarrow r$ *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., [68]. Regarding assumptions (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 most specifications of rewrite theories the lefthand side l of a rule $l \rightarrow r$ *if* φ is almost always a constructor term. In practice, l may fail to be a constructor in two special cases: (1) the case of an *equational abstraction* [75], 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; or (2) the somewhat subtle case, illustrated by the sorting rewrite theory in Example 4, where the signature Ω of constructors has a supersignature $\Sigma_1 \supseteq \Omega$, where for every operator in $\Sigma_1 \setminus \Omega$ there is a subsort-overloaded typing of it in Ω itself. Specifically, in Example 4 the constructor decomposition has the form $(\Omega, B, \emptyset)$, with Ω consisting of the constants 0, 1 of sort *Nat*, *nil* of sort *List*, *true* and *false* of sort *Bool*, a binary $+$ operator of sort *Nat*, and the list concatenation operator $\_;\_ : NeList\ NeList \rightarrow NeList$, and B are the axioms of associativity, commutativity and unit element 0 for $+$, and of associativity for $\_;\_$. Instead, the decomposition $(\Sigma_1, B, \vec{E}_1)$ keeps these same axioms and adds: (i) the (associative) *defined* operator $\_;\_ : List\ List \rightarrow List$, and (ii) its defining equations E_1, namely, $l; nil = l$, and $nil; l = l$. The somewhat subtle points is that the lefthand sides of the B-extension rules (1)–(3) added to the original sorting rule of Example 4 are *not* Ω-terms, but only Σ_1-terms. In case (2), the decomposition $(\Sigma_1, B_1, \vec{E}_1)$ is typically FVP in practice; in fact, for the just-mentioned Example 4 the above $(\Sigma_1, B, \vec{E}_1)$ is FVP. In case (1) the equational abstractions used for model checking purposes tend to be simple enough that either they are FVP or can be easily made so by a simple theory transformation. In summary, therefore, conditions (i)–(iii) cover a very general class of practical applications. There is, furthermore, a very useful special case of condition (ii), namely, the case when $(\Sigma_1, B_1, \vec{E}_1) = (\Omega, B_\Omega, \vec{E}_\Omega)$.

[10] 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.

4.1. 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 $(\varphi\alpha)!_{\vec{E},B} =_B \varphi'$. That is, $l \to r \text{ if } \varphi$ is *more general* than $l' \to r' \text{ if } \varphi'$ up to B-equality, making $l' \to r' \text{ if } \varphi'$ 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 $\mathcal{R}$, and $\overline{\mathcal{R}}_l$ is ground coherent.*

Example 5. 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 fault-tolerant communication channel, which thus becomes finite-state and therefore analyzable by standard LTL model checking. The Maude specifications of such a fault-tolerant out-of-order communication channel `FT-CHANNEL` and its equational abstraction `FT-CHANNEL-ABS` import the functional module `NAT-LIST` of Example 1 and are as follows:

```
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] .

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

mod FT-CHANNEL-ABS is including FT-CHANNEL .
vars S S' : MsgSet .
eq S S = S [variant] .           *** set idempotency
eq S S S' = S S' [variant] .     *** B-coherence extension
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 acknowledgments, 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 explicit-state LTL model checking is impossible. For this reason, `FT-CHANNEL-ABS` specifies an *equational abstraction* [75], 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, FT-CHANNEL-ABS is *not* ground coherent, and therefore its explicit-state LTL model checking would be incorrect, because lack of ground coherence means that the canonical transition system $\mathcal{C}_{\mathcal{R}}$ is such that $\mathcal{C}_{\mathcal{R}} \notin \mathbf{Trans}_{\mathcal{R}}$, and in fact $\mathcal{C}_{\mathcal{R}}$ has *fewer transitions* than the initial model $\mathcal{T}_{\mathcal{R}}$. Since in Maude explicit-state LTL model checking (or even just reachability analysis using the search command) would use $\mathcal{C}_{\mathcal{R}}$, but this is only permitted when, by ground coherence, $\mathcal{C}_{\mathcal{R}}$ and $\mathcal{T}_{\mathcal{R}}$ are *isomorphic*, all bets are off regarding the correctness of any such reachability analysis or LTL model checking. This lack of ground 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 are 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 Definition 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-OPS module; (ii) the frozenness function ϕ just freezes the if-then-else operator with the frozen keyword (therefore the rewrite theories defined by FT-CHANNEL and FT-CHANNEL-ABS are *topmost*); and (iii) at the equational level of FT-CHANNEL-ABS 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 set 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 for Boolean operations in the imported BOOL-OPS module from Maude's standard prelude. 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.

Let us focus on the transformation of 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 the multiset `{J,K} S` where `{J,K}` is a message sending item `J` marked as message 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 the if-then-else operator 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.

The "variants" of the above `[recv]` rule which are added by the theory transformation $\mathcal{R} \mapsto \overline{\mathcal{R}}_l$ are:

```
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]] .
```

4.2. 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 $(\Sigma_1, B_1, \vec{E}_1)$ laying between $(\Omega, B_\Omega, \vec{E}_\Omega)$ and a possibly conditional $(\Sigma, B, \vec{E})$ the special case, described in [91], of a transformation $\mathcal{R} \mapsto \mathcal{R}_\Omega$ making all righthand sides constructor terms. The extra generality of the $\mathcal{R} \mapsto \mathcal{R}_{\Sigma_1}$ transformation is useful because it has a better chance of becoming the identity transformation[11] for many rules in $\mathcal{R}$.

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 $l \to r$ *if* φ with $l \notin T_{\Sigma_1}(X)$. 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 $l \to r$ *if* φ in R into the rule $l \to r'$ *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 6. 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 $\geq$ 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 $th(T_{\Sigma_1/E_1 \uplus B_1})$ [73], 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:

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

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 in the standard sense, the $\mathcal{R} \mapsto \mathcal{R}_{\Sigma_1}$ transformation *preserves standard rule executability*. To explain this, I need to explain the general sense in which a rewrite rule $l \to r$ *if* φ 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 [33], 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) $vars(r) \subseteq vars(l) \cup \bigcup_{j \leq n} vars(v_j)$, 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 $l \to r$ *if* φ in R, then they also hold for its transformed rule $l \to r'$ *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 $t|_p \to x_p$ 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 standard sense, then those is $\mathcal{R}_{\Sigma_1}$ are also executable.*

4.3. 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 where $E_\Omega = \emptyset$ reduce to just B_Ω-unification steps; and (2) ensure that $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1, l, r}$ is *ground coherent*.

[11] For example, since in Example 5 the underlying equational theories of both `FT-CHANNEL` and `FT-CHANNEL-ABS` are FVP, for them we can chose $(\Sigma_1, B_1, \vec{E}_1) = (\Sigma, B, \vec{E})$, so that the $\mathcal{R} \mapsto \mathcal{R}_{\Sigma_1}$ transformation becomes the identity transformation.

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 $l \to r$ *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

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

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 $l' \to r'$ *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 $\alpha : l' \to r'$ *if* $(\varphi\gamma)!_{\vec{E},B}$ can be executed *backwards* as the rule $\alpha^{-1} : r' \to l'$ *if* $(\varphi\gamma)!_{\vec{E},B}$, which can be very useful for backwards symbolic reachability analysis (more on this in Section 6.3). 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.*

As an immediate corollary of Theorem 7, we have the isomorphisms of Σ-transition systems: $\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}} \cong \mathcal{T}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}} \cong \mathcal{T}_{\mathcal{R}}$. Furthermore, $\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}$ enjoys a very remarkable property, very useful for symbolic execution (see Section 6), which in general is not satisfied by other canonical models $\mathcal{C}_{\mathcal{R}}$, even assuming $\mathcal{R}$ coherent:

Proposition 1. *Let $[u], [v] \in C_{\Sigma/\vec{E},B}$ be such that $[u] \to_{\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}} [v]$. Then, there is a rewrite rule $l' \to r'$ if $(\varphi\gamma)!_{\vec{E},B}$ in $R^{\Omega}_{\Sigma_1,l,r}$ and a $\vec{E}_\Omega, B_\Omega$-normalized ground substitution δ such that $u =_{B_\Omega} l'\delta$, $v =_{B_\Omega} r'\delta$, and $T \models (\varphi\gamma)!_{\vec{E},B}\delta$, and therefore a rewrite step $u \to_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} r'\delta$, with $v =_{B_\Omega} r'\delta$.*

In plain English, what Proposition 1 tells us about the $[u] \to_{\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}} [v]$ rewrite relation is that we can always choose the rewrite rule $l' \to r'$ *if* $(\varphi\gamma)!_{\vec{E},B}$ in $R^{\Omega}_{\Sigma_1,l,r}$ and the $\vec{E}_\Omega, B_\Omega$-normalized ground substitution δ in such a way that in the rewrite step $u \to_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} r'\delta$ the term $r'\delta$ is already $\vec{E}_\Omega, B_\Omega$-*normalized*, so that $v =_{B_\Omega} r'\delta$ and there is no need for the usual additional step of $\vec{E}_\Omega, B_\Omega$-normalizing $r'\delta$.

Example 7. The $\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ transformation can be illustrated by the following Maude specification of a bank account system which is an open system and uses various auxiliary functions to update an account's state after each transaction:

```
fmod NAT-PRES-MONUS is protecting TRUTH-VALUE .
    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 be deposited in the account if 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 intended 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` constants 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, $\vec{E}_\Omega = \emptyset$, 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] .

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


    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' .


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

To further illustrate the $\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ transformation, let us focus on 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 system's state, which is a #-separated *pair*. Its left pattern `< bal: n pend: x overdraft: false >` describes the current state of the account, which is not in the red. Its right pattern `withdraw(m), msgs` describes a multiset of messages with an actual request `withdraw(m)` to withdraw the amount of money `m` and the remaining messages described by the variable `msgs`. The rule's righthand 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 5) 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. Otherwise, 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, as already explained, the equations defining all these auxiliary functions are FVP. Thanks to the $\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ transformation, all the lefthand- and righthand-sides of the transformed rules become *constructor* terms. Specifically, the transformed rules for the above `[w]` rule are:

```
    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* $E \uplus B$. Now instead, 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. Constrained constructor pattern predicates

This section addresses, and gives an answer to, the following crucial question:

What is a good language for state predicates?

The question is crucial because the degree of *automation* possible when reasoning symbolically about concurrent systems will vary greatly depending on the answer one gives to it. The naive answer would of course be: "first-order logic." But such an answer, while seemingly sufficient,[12] is not at all adequate in practice. It misses five key points: (i) the *specific domain* on which we are reasoning, in this case the initial model $\mathcal{T}_\mathcal{R}$, which is in fact the *intended model* specified by a generalized rewrite theory $\mathcal{R}$; (ii) the concrete needs of *symbolic reasoning* based on various symbolic procedures — for starters, think just about equational unification — to increase automation; (iii) the *additional advantages* that derive not just from reasoning about the initial model $\mathcal{T}_\mathcal{R}$ but, assuming $\mathcal{R}$ is or has been made ground coherent, about its isomorphic *canonical model* $\mathcal{C}_\mathcal{R}$; (iv) the even greater advantages that accrue from using the theory transformation $\mathcal{R} \mapsto \overline{\mathcal{R}}^\Omega_{\Sigma_1,l,r}$, because the canonical models $\mathcal{C}_\mathcal{R}$ and $\mathcal{C}_{\overline{\mathcal{R}}^\Omega_{\Sigma_1,l,r}}$ *coincide*, but $\overline{\mathcal{R}}^\Omega_{\Sigma_1,l,r}$ is a *much simpler* rewrite theory than $\mathcal{R}$ in a way that substantially facilitates automation; and (v) the fact that a well-chosen language can drastically reduce, or even eliminate, the explicit use of *quantifiers*, since explicit handling of quantifiers can substantially increase reasoning complexity.

The first thing to do in the quest for an answer to the question about a suitable language for state predicates aware of the above points (i)–(v) is to *look at the model for states* provided by $\mathcal{C}_{\overline{\mathcal{R}}^\Omega_{\Sigma_1,l,r}}$. It is of course the canonical term algebra $C_{\Sigma/\vec{E},B}$. But this is only half of the story, because we also have the reduct identity: $C_{\Sigma/\vec{E},B}|_\Omega = C_{\Omega/\vec{E}_\Omega,B_\Omega}$. This often provides an enormous simplification, because $C_{\Omega/\vec{E}_\Omega,B_\Omega}$ is typically a much simpler algebra than $C_{\Sigma/\vec{E},B}$. For example, in practical applications the theory $(\Omega, E_\Omega \uplus B_\Omega)$ is almost always FVP; and often satisfiability of QF formulas in $C_{\Omega/\vec{E}_\Omega,B_\Omega}$ is decidable [73]. In a very large number of practical applications we actually have $E_\Omega = \emptyset$; that is, the constructors Ω are *free modulo* B_Ω, so that $C_{\Sigma/\vec{E},B}|_\Omega = T_{\Omega/B_\Omega}$. Since this is the most common case in practice, and offers unique opportunities for efficient symbolic reasoning, in the rest of this section I will focus on a language of state predicates for equational theories whose constructors are free modulo B_Ω. I refer to [91] for a generalization of the ideas presented here to the case of constructors subtheories of the form $(\Omega, E_\Omega \uplus B_\Omega)$. However, even though I restrict myself to the case of free constructors modulo B_Ω, the treatment I give here contains various new results and ideas beyond those in [91]. Although the emphasis in this paper is on such new theoretical results and ideas, please note that all the logical operations on constrained constructor pattern predicates proved to be computable in Section 5.1 as well as the subsumption check defined in Section 5.2 have already been implemented in Maude and their implementation is used in an essential manner in the reachability logic theorem prover described in [91].

Looking at the model of states T_{Ω/B_Ω}, and viewing a language of state predicates as a precise way of specifying *sets* of states suggests a preliminary answer to the following, more specialized question: *what is a good, simple language for atomic state predicates?* I propose a simple, preliminary answer: *constructor patterns!* By a constructor pattern I just mean an Ω-term $u \in T_\Omega(X)$. The "pattern" description hints at the way it is interpreted as a predicate, that is, at its intended semantics, which is the set:

$$[\![u]\!] = \{[u\rho] \in T_{\Omega/B_\Omega} \mid \rho \in [X \to T_\Omega]\} \subseteq T_{\Omega/B_\Omega}.$$

[12] This answer is actually *insufficient*. To begin with, it misses the need for *inductive reasoning principles* that need to be added to first-order logic in order to have any fighting chance to reason effectively about the properties of a concurrent system.

In plain English, u is used as a pattern describing all (B_Ω-equivalence classes of) *ground* instances $u\rho$ of u. Note that the variables of a constructor pattern u *range over* T_{Ω/B_Ω}, and are *existentially quantified* in an *implicit* manner (see point (v) above). This existential quantification can be made explicit by the following, equivalent definition:

$$[\![u]\!] = \{[w] \in T_{\Omega/B_\Omega} \mid \exists \rho \in [X \to T_\Omega] \ s.t. \ w =_{B_\Omega} u\rho\}.$$

The second thing to ask ourselves when looking for a suitable language for state predicates is: *are state predicates closed under state transitions?* To clarify this issue we should look at the rewrite rules in the transformed theory $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$. The rules in this theory have the form $l \to r$ *if* φ with $l, r \in T_\Omega(X)_{State}$. To make sense when talking about "closed under state transitions," we should think of each such rewrite rule as a "predicate transformer." Therefore, what we are really asking is: is the set of states $[w] \in T_{\Omega/B_\Omega}$ obtained by rewriting the states $[u\rho] \in [\![u]\!]$ with the rule $l \to r$ *if* φ *definable* as a state predicate in the language we are seeking? A partial answer can be given as follows. If the rule is *unconditional*, i.e., has the form $l \to r$, with $l, r \in T_\Omega(X)_{State}$, the answer is *yes!* (more on this in Section 6.2). The problem, however, comes from the fact that the condition φ in a rule $l \to r$ *if* φ in $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ can be an arbitrary QF Σ-formula. The full, correct answer is also *yes!* (more on this in Section 6.2); but one has to generalize atomic state predicates from constructor patterns to *constrained constructor patterns* of the form: $u \mid \varphi$, where $u \in T_\Omega(X)$ is in $\vec{E}_\Omega, B_\Omega$-normal form, and φ is a QF Σ-formula. Their semantics is defined as follows:

$$[\![u \mid \varphi]\!] = \{[u\rho] \in T_{\Omega/B_\Omega} \mid \rho \in [X \to T_\Omega] \wedge T_{\Sigma/E \uplus B} \models \varphi\rho\}.$$

Note that $[\![u \mid \varphi]\!] \subseteq [\![u]\!] \subseteq T_{\Omega/B_\Omega}$. In fact, we can identify each constructor pattern u with the corresponding constrained constructor pattern $u \mid \top$.

Note, also, that it follows easily from the above semantic definition that for any constrained constructor pattern $u \mid \varphi$ and any *variable renaming*, i.e., any bijective and sort-preserving substitution σ such that for each $x \in dom(\sigma)$, $\sigma(x) \in X$, we have the set-theoretic equality $[\![u \mid \varphi]\!] = [\![u' \mid \varphi']\!]$ for any $u' \mid \varphi'$ such that $(u\sigma \mid \varphi\sigma) =_B (u' \mid \varphi')$, where we view $u\sigma \mid \varphi\sigma$ and $u' \mid \varphi'$ as *terms* in an extended signature consisting of $\mid$-separated pairs of terms, whose first components are Σ-terms, and whose second components are Σ-formulas.[13] Since: (i) by B regular, $=_B$ is a variable-preserving equivalence relation, (ii) variable renamings are closed under composition, inverses, and include the identity substitution, and (iii) if $(u \mid \varphi) =_B (v \mid \psi)$ then $(u \mid \varphi)\sigma =_B (v \mid \psi)\sigma$ for any variable renaming σ, we can think more abstractly of a constrained constructor pattern *up to variable renaming and B-equality*, i.e., up to the *equivalence relation*:

$$u \mid \varphi \approx_B u' \mid \varphi' \ \Leftrightarrow \ (\exists \sigma) \ s.t. \ \sigma \ \textit{variable renaming} \wedge u' \mid \varphi' =_B u\sigma \mid \varphi\sigma \, .$$

We denote by $[u \mid \varphi]_{\approx_B}$, or just $[u \mid \varphi]$, the $\approx_B$-equivalence class of $u \mid \varphi$. Therefore, since the semantics of constructor pattern predicates is preserved by $\approx_B$-equivalences, unless indicated otherwise, given two constructor patterns, say, $u \mid \varphi$ and $v \mid \psi$, we can always assume *without loss of generality*, that, up to variable renaming, we have $vars(u \mid \varphi) \cap vars(v \mid \psi) = \emptyset$.

We can now define constrained constructor pattern predicates and their semantics. Recall that X denotes the countably infinite S-sorted set of variables used in the language of Σ-formulas.

Definition 9. Let $(\Omega, B_\Omega, \emptyset)$ be a *constructor decomposition* of $(\Sigma, B, \vec{E})$, and let *State* be a chosen top sort in a chosen connected component of Σ. The set $AtPatPred(\Omega, \Sigma)$ of *atomic predicates* is the set of constrained constructor patterns of the form $u \mid \varphi$ with $u \in T_\Omega(X)_{State}$ and φ a QF Σ-formula. The algebra $PatPred(\Omega, \Sigma)$ of *constrained constructor pattern predicates* is then the free $\{\vee, \wedge, \bot\}$-algebra on the set of generators $AtPatPred(\Omega, \Sigma)$. The *semantics* $[\![A]\!]$ of a constrained constructor pattern predicate A is defined as the unique $\{\vee, \wedge, \bot\}$-homomorphism:

$$[\![\_]\!] : PatPred(\Omega, \Sigma) \to \mathcal{P}(T_{\Omega/B_\Omega, State})$$

extending the already defined semantic function $u \mid \varphi \mapsto [\![u \mid \varphi]\!]$ for atomic predicates. That is, using capital letters $A, B, \ldots, P, Q, \ldots$, and so on, as variables ranging over $PatPred(\Omega, \Sigma)$, $[\![\_]\!]$ homomorphically maps any pattern predicate A to a subset $[\![A]\!] \subseteq T_{\Omega/B_\Omega, State}$ as follows:

1. $[\![\bot]\!] = \emptyset$.
2. $[\![A \vee B]\!] = [\![A]\!] \cup [\![B]\!]$
3. $[\![A \wedge B]\!] = [\![A]\!] \cap [\![B]\!]$.

There is no need to add $\top$ and $\bot$ as pattern predicates, since $[\![x{:}State \mid x = x]\!] = T_{\Omega/B_\Omega}$, and $[\![x{:}State \mid x \neq x]\!] = \emptyset$; but it is useful to add $\bot$.

[13] The equivalence $=_B$ can be easily extended to the equivalence $=_{B \cup AC_\vee \cup AC_\wedge}$, where $AC_\vee \cup AC_\wedge$ are the associativity and commutativity axioms for $\vee$ and $\wedge$. In this way, formulas that are equal up to order and parentheses for logical connectives and B-equivalence of terms can be identified. Everything I will say later about constrained constructor patterns modulo $=_B$ (resp. $\approx_B$, see below), extends easily to $=_{B \cup AC_\vee \cup AC_\wedge}$ and $\approx_{B \cup AC_\vee \cup AC_\wedge}$.

Note that the $\{\vee, \wedge, \bot\}$-algebra $\mathcal{P}(T_{\Omega/B_\Omega,State})$ is a *distributive lattice*. Note also that if $T_{\Omega/B_\Omega,State}$ is countable, then $\mathcal{P}(T_{\Omega/B_\Omega,State})$ has the power of the continuum, so there is no hope whatsoever for the $\{\vee, \wedge, \bot\}$-algebra $\mathcal{P}(T_{\Omega/B_\Omega,State})$ to be *computable*.

5.1. Making pattern predicate operations computable

However, for Σ a finite signature, (i) the *set* $PatPred(\Omega, \Sigma)$ is countable; and (ii) we can *symbolically lift* in an *effective, computable* way all $\{\vee, \wedge, \bot\}$ operations on the image *subalgebra* $[\![PatPred(\Omega, \Sigma)]\!] \subseteq \mathcal{P}(T_{\Omega/B_\Omega,State})$ of $PatPred(\Omega, \Sigma)$ under the $\{\vee, \wedge, \bot\}$-homomorphism $[\![\_]\!]$ by performing them on the *computable* $\{\vee, \wedge, \bot\}$-*algebra* $\vee.PatPred(\Omega, \Sigma)/\approx_B \cup Ax_\vee$, where $\vee.PatPred(\Omega, \Sigma)/\approx_B \cup Ax_\vee$ denotes the free $\{\vee, \bot\}$-algebra modulo $Ax_\vee$ on the set $AtPatPred(\Omega, \Sigma)/\approx_B$, which is the quotient of the set $AtPatPred(\Omega, \Sigma)$ of atomic pattern predicates under the equivalence relation $\approx_B$, and where $\vee.Ax$ consists of the following axioms: (i) the associativity and commutativity ($AC_\vee$) axioms for $\vee$, and (ii) the $\vee$-identity axiom for $\bot$, $\bot \vee A = A$. Since: (a) the equivalence relation $\approx_B$ on atomic predicates is trivially computable, (b) equality modulo $\vee.AC$ is also computable, and (c) the (oriented) equation $\bot \vee A = A$ is confluent and terminating modulo $AC_\vee$, we can identify $\vee.PatPred(\Omega, \Sigma)/\approx \cup Ax_\vee$ with the set of its normal forms modulo $AC_\vee$ by the (oriented) equation $\bot \vee A = A$. This means that the $\{\vee, \bot\}$-algebra $\vee.PatPred(\Omega, \Sigma)/\approx \cup Ax_\vee$ is *computable*.[14]

To make $\vee.PatPred(\Omega, \Sigma)/\approx_B \cup Ax_\vee$ into a *computable* $\{\vee, \wedge, \bot\}$-algebra "all we need to do" is to define the function $\wedge$ and show it is computable. Rather than doing this directly, it is easier to do it by stages.

Stage 1. Define the $\{\vee, \wedge, \bot\}$-algebra $PatPred(\Omega, \Sigma)/\approx_B \cup Ax$, which, by definition, is the free $\{\vee, \wedge, \bot\}$-algebra modulo Ax on the set $AtPatPred(\Omega, \Sigma)/\approx_B$, and where Ax adds to $Ax_\vee$ the following additional axioms: (i) the associativity and commutativity ($AC_\wedge$) axioms for $\wedge$, (ii) the axiom $\bot \wedge A = \bot$, and (iii) the distributivity axiom $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$. The algebra $PatPred(\Omega, \Sigma)/\approx_B \cup Ax$ is *computable* for the following reasons: (a) $\approx_B$ is trivially computable, (b) equality modulo $AC_\vee \cup AC_\wedge$ is also computable, and (c) the equations $\bot \vee A = A$, $\bot \wedge A = \bot$, and $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ are confluent and terminating modulo $AC_\vee \cup AC_\wedge$. Therefore, $PatPred(\Omega, \Sigma)/\approx_B \cup Ax$ is like a canonical term algebra, except that instead of using variables to generate it we use $\approx_B$-equivalence classes of atomic predicates as generators. Note, furthermore, that the *normal form* modulo $\vee.AC \cup \wedge.AC$ under these equations of any pattern predicate is a *disjunctive normal form*, i.e., it is either $\bot$, or a disjunction of (one or more) conjunctions of atomic predicates.

Stage 2. All we have left to do is to define a *computable function*

$$\_ \wedge \_ : (\vee.PatPred(\Omega, \Sigma)/\approx_B \cup Ax_\vee)^2 \to \vee.PatPred(\Omega, \Sigma)/\approx_B \cup Ax_\vee.$$

This computable function is defined as follows:

(Stage 2.a). We map each $([A], [B]) \in (\vee.PatPred(\Omega, \Sigma)/\approx_B \cup Ax_\vee)^2$ to the disjunctive normal form $[A \wedge B]_{dnf}$ of $[A \wedge B]$ modulo $AC_\vee \cup AC_\wedge$ in $PatPred(\Omega, \Sigma)/\approx_B \cup Ax$. This defines a computable function:

$$\_ \wedge_1 \_ : (\vee.PatPred(\Omega, \Sigma)/\approx_B \cup Ax_\vee)^2 \to PatPred(\Omega, \Sigma)/\approx_B \cup Ax$$

where $[A] \wedge_1 [B] = [A \wedge B]_{dnf}$.

(Stage 2.b). We then define a second function:

$$unif : PatPred(\Omega, \Sigma)/\approx_B \cup Ax \to \vee.PatPred(\Omega, \Sigma)/\approx_B \cup Ax_\vee$$

as follows: $unif(\bot) = \bot$. In all other cases we need to define the image by $unif$ of a disjunctive normal form

$$\bigvee_{i \in I} \bigwedge_{j \in J_i} [u_{i_j} \mid \varphi_{i_j}]$$

modulo $AC_\vee \cup AC_\wedge$ with I and each J_i non-empty index sets. But, since we define $unif$ to be disjunction-preserving, all we have left to do is to define for each $k \in I$ the image by $unif$ of a conjunction of ($\approx_B$-equivalence classes of) atomic predicates $\bigwedge_{j \in J_k} [u_{k_j} \mid \varphi_{k_j}]$ modulo $AC_\wedge$. First of all, if $\{[u_{k_j} \mid \varphi_{k_j}] \mid j \in J_k\}$ is a singleton set of the form $\{[u_k \mid \varphi_k]\}$, then $unif(\bigwedge_{j \in J_k} [u_{k_j} \mid \varphi_{k_j}]) = [u_k \mid \varphi_k]$. Otherwise, $\{[u_{k_j} \mid \varphi_{k_j}] \mid j \in J_k\} = \{[u_{k_1} \mid \varphi_{k_1}], \ldots, [u_{k_n} \mid \varphi_{k_n}]\}$, with $n > 1$, and where we can assume without loss of generality that the representatives of the $\approx_B$-equivalence classes are such that for $1 \leq l < r \leq n$ we have: $vars(u_{k_l} \mid \varphi_{k_l}) \cap vars(u_{k_r} \mid \varphi_{k_r}) = \emptyset$. Then we define:

[14] By definition, a Σ-algebra A is called *computable* iff: (i) its elements can be effectively specified in a finitary way; (ii) equality between elements is decidable; and (iii) for each $f \in \Sigma$ the function f_A is recursive. For a precise definition see, e.g., [12,74]. Alternatively, one can formalize the informal description (i)–(iii) just given using Shoenfield's notion of a *space* and of a *recursive* (also called *computable*) function between two spaces (see [89], and for a brief introduction Section 3.2 of [69]). In Shoenfield's terms we just say that A is a space, and for each $f \in \Sigma$ the function f_A is recursive. The ten thousand feet high description is even simpler: A is computable if its elements can be implemented in a computer as a data type, and its operations can be programmed as terminating functions operating on such a data type.

$$unif(\bigwedge_{j\in J_k}[u_{k_j} \mid \varphi_{k_j}]) = \bigvee_{\alpha\in Unif_{B_\Omega}(\{u_{k_1},\ldots,u_{k_n}\})} [(u_{k_1} \mid \bigwedge_{1\leq l\leq n}\varphi_{k_l})\alpha].$$

Note that such a definition does not depend on the choice of variable-disjoint representatives $\{u_{k_1},\ldots,u_{k_n}\}$, and therefore defines a function *unif*. This is because, for any other such choice, say, $\{u'_{k_1},\ldots,u'_{k_n}\}$, there will be a variable renaming σ such that $Unif_{B_\Omega}(\{u_{k_1},\ldots,u_{k_n}\}) = \{\sigma\alpha \mid \alpha \in Unif_{B_\Omega}(\{u_{k_1},\ldots,u_{k_n}\})\}$. After these two steps we can define our desired computable function $\_\wedge\_$ as the function composition $\_\wedge\_ = unif \circ \_\wedge_1\_$. That is, for each $([A],[B]) \in (\vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee)^2$ we define $[A]\wedge[B] = unif([A\wedge B]_{dnf})$. The key semantic property satisfied by this definition of conjunction is:

Proposition 2. *Under the variable disjointness assumption that, for $1 \leq l < r \leq n$, $vars(u_{k_l} \mid \varphi_{k_l}) \cap vars(u_{k_r} \mid \varphi_{k_r}) = \emptyset$, the following set equality holds:*

$$\bigcap_{1\leq l\leq n}[\![u_{k_l} \mid \varphi_{k_l}]\!] = \bigcup_{\alpha\in Unif_{B_\Omega}(\{u_{k_1},\ldots,u_{k_n}\})}[\![(u_{k_1} \mid \bigwedge_{1\leq l\leq n}\varphi_{k_l})\alpha]\!].$$

This, plus the fact that the $\{\vee,\wedge,\bot\}$-homomorphism $[\![\_]\!] : PatPred(\Omega,\Sigma) \to \mathcal{P}(T_{\Omega/B_\Omega,State})$ preserves all $\approx_B \cup Ax$-equalities, i.e., that $A =_{\approx_B\cup Ax} B$ implies $[\![A]\!] = [\![B]\!]$, means that the $\{\vee,\wedge,\bot\}$-homomorphism $[\![\_]\!] : PatPred(\Omega,\Sigma) \to \mathcal{P}(T_{\Omega/B_\Omega,State})$ factors as the composition of $\{\vee,\wedge,\bot\}$-homomorphisms:

$$PatPred(\Omega,\Sigma) \xrightarrow{[\_]} \vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee \xrightarrow{[\![\_]\!]} \mathcal{P}(T_{\Omega/B_\Omega,State})$$

where, by definition, for each $[A] \in \vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee$, $[\![[A]]\!] = [\![A]\!]$, and $[\_]$ is the unique $\{\vee,\wedge,\bot\}$-homomorphism induced by the composition:

$$AtPatPred(\Omega,\Sigma) \xrightarrow{[\_]} AtPatPred(\Omega,\Sigma)/\approx_B \hookrightarrow \vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee$$

where $AtPatPred(\Omega,\Sigma) \xrightarrow{[\_]} AtPatPred(\Omega,\Sigma)/\approx_B$ is just the quotient map to $\approx_B$-equivalence classes, and $\hookrightarrow$ is set-theoretic inclusion.

It is this second $\{\vee,\wedge,\bot\}$-homomorphism $\vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee \xrightarrow{[\![\_]\!]} \mathcal{P}(T_{\Omega/B_\Omega,State})$ that I have called the *lifting* of the set-theoretic operations in the image $\{\vee,\wedge,\bot\}$-subalgebra $[\![PatPred(\Omega,\Sigma)]\!] \subseteq \mathcal{P}(T_{\Omega/B_\Omega,State})$ to corresponding *computable symbolic operations* in the algebra $\vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee$.

5.2. Some basic properties and pattern predicate subsumption

The proof of the following basic properties about atomic pattern predicates and pattern predicates follows easily from their basic definitions and is left to the reader.

Lemma 2. *The following results hold for atomic pattern predicates and pattern predicates:*

1. $[\![u \mid \varphi\vee\psi]\!] = [\![u \mid \varphi]\!] \cup [\![u \mid \psi]\!]$.
2. *Given QF Σ-formulas φ,ψ, if $T_{\Sigma/E\cup B} \models \varphi \Rightarrow \psi$, then $[\![u \mid \varphi]\!] \subseteq [\![u \mid \psi]\!]$.*
3. *For each Ω-substitution α and pattern predicate A, $[\![A\alpha]\!] \subseteq [\![A]\!]$.*
4. *Call $\{u_1,\ldots,u_k\} \subseteq T_\Omega(X)_s$ a* pattern set *for sort s modulo B_Ω iff $T_{\Omega/B_\Omega,s} = \bigcup_{1\leq i\leq k}\{[u_i\rho] \mid \rho \in [X \to T_\Omega]\}$. Then, if in the atomic predicate $u \mid \varphi$ variable $x{:}s$ of sort s appears in u we have,*

$$[\![u \mid \varphi]\!] = \bigcup_{1\leq i\leq k}[\![(u \mid \varphi)\{x{:}s \mapsto u_i\}]\!]$$

where all variables in the pattern set are fresh *variables not appearing in $u \mid \varphi$.*

Note that (1) in the above lemma allows assuming without loss of generality that in an atomic pattern predicate $u \mid \varphi$, the formula φ is a conjunction of literals. This is so because, up to Boolean equivalence, we may assume φ in disjunctive normal form $\varphi = \varphi_1 \vee \ldots \vee \varphi_n$ and can then apply (1) to decompose $[\![u \mid \varphi]\!]$ into the union $\bigcup_{1\leq i\leq n}[\![u \mid \varphi_i]\!]$. This is quite useful because, as we shall see in Section 6, symbolic methods often transform a constructor pattern $u \mid \varphi$ into another constructor pattern $(v \mid \varphi\wedge\psi)\alpha$ for some Ω-substitution α. But if φ and ψ are conjunctions of literals, then so is $\varphi\wedge\psi$, and this can make it easier to perform other symbolic operations.

Pattern Predicate Subsumption. Given atomic pattern predicates $u \mid \varphi$ and $v \mid \psi$, the *subsumption relation* $u \mid \varphi \sqsubseteq_{B_\Omega} v \mid \psi$ (meaning that $u \mid \varphi$ is *less general* than $v \mid \psi$, which therefore *subsumes* $u \mid \varphi$) holds by definition iff there is an Ω-substitution β such that: (i) $u =_{B_\Omega} v\beta$, and (ii) $T_{\Sigma/E\uplus E} \models \varphi \Rightarrow \psi\beta$. Unless we are within a decidable fragment for validity

of QF formulas in $T_{\Sigma/E \uplus B}$, checking (ii) is in general an inductive theorem proving problem and therefore undecidable; but there are various syntactic and/or simplification-based methods that can verify (ii) in many practical cases.

More generally, given pattern predicates $\bigvee_{i\in I} u_i \mid \varphi_i$ and $\bigvee_{j\in J} v_j \mid \psi_j$ the subsumption relation $\bigvee_{i\in I} u_i \mid \varphi_i \sqsubseteq_{B_\Omega} \bigvee_{j\in J} v_j \mid \psi_j$ holds iff for each $i \in I$ there is a $j \in J$ such that $u_i \mid \varphi_i \sqsubseteq_{B_\Omega} v_j \mid \psi_j$. Note that if $A \sqsubseteq_{B_\Omega} B$, then it follows immediately from (2)–(3) in Lemma 2 that we have the set containment $[\![A]\!] \subseteq [\![B]\!]$; but the converse does not follow: we may have $[\![A]\!] \subseteq [\![B]\!]$ without having $A \sqsubseteq_{B_\Omega} B$. In fact, (4) in Lemma 2 provides an easy counterexample taking $A \equiv u \mid \varphi$ and $B \equiv \bigvee_{1\leq i\leq k}(u \mid \varphi)\{x{:}s \mapsto u_i\}$.

Often, the substitution β witnessing a subsumption $u \mid \varphi \sqsubseteq_{B_\Omega} v \mid \psi$ can easily be found using a B_Ω-matching algorithm; but this is not always the case: when a B_Ω-matching substitution α is such that $u =_{B_\Omega} v\alpha$ but the set of variables $Y = vars(\psi\alpha) \setminus vars(\varphi)$ is non-empty, then α will usually not be good enough to witness the subsumption $u \mid \varphi \sqsubseteq_{B_\Omega} v \mid \psi$. We may need to more carefully choose a substitution β of the form $\beta = \alpha \uplus \gamma$, where γ has domain Y and range contained in $vars(\varphi)$ to be able to show that $T_{\Sigma/E\uplus B} \models \varphi \Rightarrow \psi\beta$. This need for finding a witness β more subtle than just a simple B_Ω-matching substitution can be illustrated with an example.

Example 8. Consider the unsorted signature $\Sigma = \{0, 1, \_+\_, \_*\_\}$ with $\Omega = \{0, 1, \_+\_\}$, B_Ω the associativity-commutativity and identity 0 for $\_+\_$, B obtained by adding to B_Ω the associativity-commutativity for $\_*\_$, and E the equations $x*0 = 0$, $x*1 = x$, and $x*(y+z) = (x*y)+(x*z)$. Then, to prove the pattern subsumption:

$$y+y+y+y \mid y \neq 0 \sqsubseteq_{B_\Omega} x+x \mid x = m*n \wedge m \neq 0 \wedge n \neq 0$$

the obvious B_Ω-matching substitution $\alpha = \{x \mapsto y+y\}$ is not good enough due to the "dangling variables" $Y = \{n, m\}$. But we can extend α to the substitution $\beta = \{x \mapsto y+y, m \mapsto 1+1, n \mapsto y\}$, which allows us to prove $T_{\Sigma/E\uplus B} \models y \neq 0 \Rightarrow y+y = (1+1)*y \wedge 1+1 \neq 0 \wedge y \neq 0$, since the first conjunct in the conclusion follows by E, B-simplification, the last conjunct follows from the hypothesis, and $1+1 \neq 0$ holds in T_{Ω/B_Ω} because the equation $1+1=0$ has no B_Ω-unifiers.

5.3. Relaxing the freeness modulo B_Ω requirement

There are two ways in which the requirement that the constructor subspecification $(\Omega, E_\Omega \cup B_\Omega) \subseteq (\Sigma, E \cup B)$ has free constructors modulo axioms B_Ω, i.e., that $E_\Omega = \emptyset$, can be relaxed. The first, adopted in [91], is to assume that $(\Omega, E_\Omega \cup B_\Omega)$ is an FVP theory. This assumption is very natural and holds in practice for an overwhelming number of constructor specifications. For a simple example, consider sets (whose elements belong to a subsort), with a union operator $\_\cup\_$ and empty set $\emptyset$, where B_Ω are the associative-commutative axioms for $\_\cup\_$, and $E_\Omega = \{S \cup S = S, S \cup \emptyset = S\}$, which is FVP. All results presented so far extend naturally to the FVP case. The semantics of an atomic pattern predicate $u \mid \varphi$ also extends naturally as follows:

$$[\![u \mid \varphi]\!] = \{[(u\rho)!_{E_\Omega, B_\Omega}] \in C_{\Omega/E_\Omega, B_\Omega} \mid \rho \in [X \to T_\Omega] \wedge T_{\Sigma/E\uplus B} \models \varphi\rho\}.$$

All operations on pattern predicates remain computable. The reason for this is that, since $(\Omega, E_\Omega \cup B_\Omega)$ is FVP, we obtain finitary unification and matching algorithms modulo $E_\Omega \cup B_\Omega$ that can be used, respectively, to compute pattern predicate conjunctions $A \wedge B$ and pattern predicate subsumptions $A \sqsubseteq_{E_\Omega \cup B_\Omega} B$.

A second way in which the need for relaxing the freeness modulo B_Ω requirement arises is because we may have a subsignature $\Sigma_0 \subseteq \Sigma$ such that satisfiability in $T_{\Sigma/E\cup B|\Sigma_0}$ of Σ_0-formulas[15] in some class $\mathcal{C}$ is decidable. Consider, for example, the case of the set constructors mentioned above, where the ground constructor terms in the subsort of elements are the rational numbers, and $\mathcal{C}$ is the class of linear arithmetic formulas. In many such cases the constructor subsignature will typically have the form: $(\Omega_1 \uplus \Omega_0, E_{\Omega_1} \uplus E_{\Omega_0} \uplus B_{\Omega_1} \uplus B_{\Omega_0})$, where Ω_1 is the signature of "standard" constructors, so that it is very reasonable to assume that $(\Omega_1, E_{\Omega_1} \uplus B_{\Omega_1})$ is FVP, and $(\Omega_0, E_{\Omega_0} \uplus B_{\Omega_0})$ is the signature of constructors for $T_{\Sigma/E\cup B|\Sigma_0}$. By the way, there is no need to assume that $(\Omega_0, E_{\Omega_0} \uplus B_{\Omega_0})$ is a finite specification. In some cases Ω_0 may contain all canonical representations of elements of $T_{\Sigma/E\cup B|\Sigma_0}$ as constants, and $E_{\Omega_0} \uplus B_{\Omega_0}$ may be the so-called *diagram* (see, e.g., [62]) of $T_{\Sigma/E\cup B|\Omega_0}$. To get real traction out of the decidable satisfiability of $\mathcal{C}$-formulas in $T_{\Sigma/E\cup B|\Sigma_0}$, a very useful property is that for any Ω_1-substitution θ and formula $\varphi \in \mathcal{C}$ we have $\varphi\theta \in \mathcal{C}$. This can be achieved by making some reasonable assumptions on the sorts S_0 of Ω_0 and on Ω_1 and Ω_0, as done in, e.g., [87,22]. Specifically, it is enough to assume that: (i) Ω_0 (and therefore Σ_0) is many-sorted, (ii) the sorts S_0 are minimal elements in the poset $(S, \leq)$ of sorts of Σ, which is also that of Ω_1, and (iii) all $\Omega_1 \uplus \Omega_0$-terms having a sort on S_0 are Ω_0-terms. A great conceptual simplification can then be achieved by reasoning about constructor $\Omega_1 \uplus \Omega_0$-terms using only Ω_1-patterns. This can be done by Ω_0-*abstracting* a $\Omega_1 \uplus \Omega_0$-term u as a pair $(u^\circ, \alpha_u^{\Omega_0})$, where u° is a Ω_1-term and $\alpha_u^{\Omega_0}$ is a substitution such that $u = u^\circ\alpha_u^{\Omega_0}$. Let me briefly explain the $(u^\circ, \alpha_u^{\Omega_0})$ construction, spelled out also in [87,22]. Any $\Omega_1 \uplus \Omega_0$-term u is always of the form $u = u[v_1, \ldots, v_n]_{p_1,\ldots,p_n}$, where $p_1, \ldots, p_n$ are the maximal positions in u where terms having a sort in S_0

[15] As explained in, e.g., [73], there is no need to distinguish between function and predicate symbols: by adding an extra sort *Pred* of predicates, any model of a first-order logic theory can be naturally understood as a Σ_0-algebra for some Σ_0.

(and therefore a Ω_0-term) appear, and the $v_1, \ldots, v_n$ are the biggest-possible Ω_0-subterms in question. Then we define the Ω_0-abstraction pair $(u^\circ, \alpha_u^{\Omega_0})$ as follows. $u^\circ = u[x_1, \ldots, x_n]_{p_1,\ldots,p_n}$, where, if v_i is a variable of sort $s_0 \in S_0$, then $x_i \equiv v_i$, and otherwise x_i is a fresh new variable with same sort as v_i, and where $\alpha_u^{\Omega_0}$ is the substitution $\alpha_u^{\Omega_0} = \{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\}$ and we denote by $\widehat{\alpha}_u^{\Omega_0}$ the associated conjunction $\widehat{\alpha}_u^{\Omega_0} \equiv x_1 = v_1 \wedge \ldots \wedge x_n = v_n$.

The key question now is: when the constructor subspecification of $(\Sigma, E \cup B)$ has the form $(\Omega_1 \uplus \Omega_0, E_{\Omega_1} \uplus E_{\Omega_0} \uplus B_{\Omega_1} \uplus B_{\Omega_0})$ and satisfies the assumptions and sort restrictions mentioned above, what is a good language of state predicates for $(\Sigma, E \cup B)$ and chosen sort *State*? The obvious answer is: the pattern predicates that can be built out of atomic pattern predicates of the form $u \mid \varphi$ with u a Ω_1-term of sort *State* and φ a QF Σ-formula. The *semantics* $[\![u \mid \varphi]\!]$ of $u \mid \varphi$ is exactly as the one given above for the FVP case, in the understanding that now $(\Omega, E_\Omega \uplus B_\Omega) = (\Omega_1 \uplus \Omega_0, E_{\Omega_1} \uplus E_{\Omega_0} \uplus B_{\Omega_1} \uplus B_{\Omega_0})$. The remarkable fact is that the algebra of pattern predicates is still *computable*. This is because a conjunction $u \mid \varphi \wedge v \mid \psi$ can be computed by variant $E_{\Omega_1} \uplus B_{\Omega_1}$-unification, and the subsumption relation has the form $A \sqsubseteq_{E_{\Omega_1} \cup B_{\Omega_1}} B$. This is very nice but not entirely obvious, since, to agree with set intersection in its semantic interpretation, a conjunction $u \mid \varphi \wedge v \mid \psi$ should be computed by $E_\Omega \uplus B_\Omega$-unification, which need not be finitary at all. The recent work of S. Ciobâcă, A. Arusoaie, and D. Lucanu [22] saves the day. Under the above assumptions on the decomposition of the constructor subspecification they provide a $E_\Omega \uplus B_\Omega$-unification algorithm which has a very simple description and therefore I include it here. Given two Ω-terms u and v a complete set of *constrained* $E_\Omega \uplus B_\Omega$-unifiers for the equation $u = v$ is obtained as follows: (i) we first compute their Ω_0-abstractions $(u^\circ, \alpha_u^{\Omega_0})$ and $(v^\circ, \alpha_v^{\Omega_0})$, and then (ii) the set

$$\{(\beta \mid (\widehat{\alpha}_u^{\Omega_0} \wedge \widehat{\alpha}_v^{\Omega_0})\beta) \mid \beta \in \mathit{Unif}_{E_{\Omega_1} \uplus B_{\Omega_1}}(u^\circ = v^\circ)\}$$

provides a complete set of constrained $E_\Omega \uplus B_\Omega$-unifiers, were we can of course discard those unifiers such that $(\widehat{\alpha}_u^{\Omega_0} \wedge \widehat{\alpha}_v^{\Omega_0})\alpha$ is unsatisfiable in $T_{\Sigma/E \cup B|\Sigma_0}$. The key observation is that when u and v are Ω_1-terms, their $E_\Omega \uplus B_\Omega$-unifiers *coincide* with their $E_{\Omega_1} \uplus B_{\Omega_1}$-unifiers. This is because then their Ω_0-abstractions have the form (u, id) and (v, id), with id the identity substitution, making the constraints $(\widehat{id} \wedge \widehat{id})\beta$ trivial.

6. Symbolic methods for generalized rewrite theories

This section presents sound and complete symbolic methods for reasoning about universal and existential reachability formulas in a generalized rewrite theory, and also for verifying invariants.

6.1. Universal reachability by generalized rewriting

Coherence of a generalized rewrite theory, plus absence of extra variables in the righthand sides and conditions of rules, plus existence of a B-matching algorithm, plus decidability of the QF formulas instantiating rule conditions, provide a semi-decision procedure based on *symbolic execution* by generalized rewriting for proving all *valid universal reachability formulas* of the form:

$$(\dagger)\ \ \mathcal{R} \models (\forall Y)\, t \rightarrow^* t'$$

where Y are the variables appearing in t, t'. Recall that the models of the generalized rewrite theory $\mathcal{R}$ are precisely the algebraic transition systems in the category $\mathbf{Trans}_{\mathcal{R}}$. By definition, $(\dagger)$ holds, iff for each $(A, \rightarrow_A) \in \mathbf{Trans}_{\mathcal{R}}$ $(A, \rightarrow_A) \models (\forall Y)\, t \rightarrow^* t'$, which, again by definition, means that for each $\alpha \in [X \rightarrow A]$, with $X \supset Y$ having countably many variables for each sort, we have $t\alpha \rightarrow_A^* t'\alpha$, where $\rightarrow_A^*$ is the reflexive-transitive closure of the transition relation $\rightarrow_A$.

The theorem below is very general. It reduces validity of universal reachability formulas to various rewrite relations on the canonical model $\mathcal{C}_{\mathcal{R}}(X)$ and on terms in $T_\Sigma(X)$. The statement of Theorem 8 does not require the absence of extra variables in the righthand sides and conditions of rules or, beyond coherence requirements, any of the other symbolic executability requirements mentioned above. The easy to automate case obtained by imposing those extra requirements is considered in Corollary 1.

Theorem 8. *Let $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$ be a generalized rewrite theory such that: (i) $(\Sigma, B, \vec{E})$ is a convergent, strongly deterministic rewrite theory; (ii) the rules R are strictly B-coherent; and (iii) the rewrite theory $\mathcal{R}$ is coherent. Then the following are equivalent:*

1. $\mathcal{R} \models (\forall Y)\, t \rightarrow^* t'$
2. $[t!_{\vec{E},B}]_B \rightarrow^*_{\mathcal{C}_{\mathcal{R}}(X)} [t'!_{\vec{E},B}]_B$
3. $t!_{\vec{E},B} \rightarrow^*_{R,B;!_{\vec{E},B}} w$ *and* $w =_B t'!_{\vec{E},B}$.

where, by definition, $u \rightarrow_{R,B;!_{\vec{E},B}} v$ iff there is a Σ-term v' such that $u \rightarrow_{R,B} v'$ and $v = v'!_{\vec{E},B}$.

Corollary 1. *Under the assumptions of Theorem 8, plus the further assumptions that: (iv) B has a matching algorithm yielding a finite and complete set of B-*matching substitutions[16] *for each matching problem; (v) each rule $l \to r$ if φ in R is such that $vars(r) \cup vars(\varphi) \subseteq vars(l)$; (vi) there is a subsignature $\Sigma_0 \subseteq \Sigma$ such that T-validity of QF Σ_0-formulas is decidable, and for each rule $l \to r$ if φ and Σ-substitution θ, $\varphi\theta$ is a QF Σ_0-formula; and (vii) the set R of rules is finite, breadth-first search from the term $t!_{\vec{E},B}$ with the computable relation $\to_{R,B;!_{\vec{E},B}}$ to find a term w such that $t!_{\vec{E},B} \to^*_{R,B;!_{\vec{E},B}} w$ and $w =_B t'!_{\vec{E},B}$ provides a semi-decision procedure for proving the $\mathcal{R}$-validity of the universal reachability formula $(\forall Y)\ t \to^* t'$.*

Note that the additional assumptions (iv)–(vii) in Corollary 1 are important to automate the relation $\to_{R,B;!_{\vec{E},B}}$. In particular, assumption (v) is important because otherwise, we would *not* be able to perform a step $u \to_{R,B;!_{\vec{E},B}} v$ based on a substitution β B-matching a subterm $u|_p$ against the lefthand side l of a rule in R. This is because we would have to *guess* the values of β on the extra variables in the rule's righthand side and condition to be able to *check* whether a rewrite step with $\to_{R,B;!_{\vec{E},B}}$ can be performed, and there may be an infinite number of such guesses. Furthermore, even assuming (v), assumption (vi) is needed for the condition's check to be effective. As we shall see in Section 6.2, assumption (v) is not needed at all when we are instead interested in proving the $\mathcal{R}$-validity of *existential* formulas of the form $(\exists Y)\ t \to^* t'$.

Example 9. Consider the sorting theory $\mathcal{R} = (\Sigma, E \uplus B, \overline{R}, T, \phi)$ of Example 4, where $\overline{R}$ are the strictly B-coherent rules obtained by adding to the original rule in R its B-extensions (1)–(3) in Example 4. Recall that $\mathcal{R}$ is also coherent. It is easy to check that, furthermore, it satisfies all other conditions (iv)–(vii) in Corollary 1. In particular, condition (vi) holds for Σ_0 the restriction of Σ to the sorts *Nat* and *Bool*, since the equational theory $(\Sigma, E \uplus B)$ is FVP, and the constructor subsignature for the sorts *Nat* and *Bool* is OS-compact [73], yielding a decision procedure for T-satisfiability (and therefore T-validity) of QF Σ_0-formulas by variant satisfiability [73]. Although not essential for our purposes, it is also not hard to see that the relation $\to_{\overline{R},B;!_{\vec{E},B}}$ is terminating. We can then prove, for example, the $\mathcal{R}$-validity of the universal formula:

$$(\forall \{n, m, k\})\ n + m + k + 1 + 1; n + m + 1; n \to^* n; n + m + 1; n + m + k + 1 + 1$$

where the variables n, m, k have sort *Nat*, by computing all terminating sequences of symbolic executions with the relation $\to_{\overline{R},B;!_{\vec{E},B}}$ from the term $n + m + k + 1 + 1; n + m + 1; m$. There are several such sequences. One of them is the sequence:

$$n + m + k + 1 + 1; n + m + 1; n \to_{\overline{R},B;!_{\vec{E},B}} n + m + k + 1 + 1; n; n + m + 1 \to_{\overline{R},B;!_{\vec{E},B}}$$
$$n; n + m + k + 1 + 1; n + m + 1 \to_{\overline{R},B;!_{\vec{E},B}} n; n + m + 1; n + m + k + 1 + 1.$$

6.2. Narrowing-based existential reachability analysis

The *existential reachability* problem, which in its simplest possible form can be posed, for given topmost generalized rewrite theory $\mathcal{R}$ and state terms t, t', as checking the satisfaction of[17]:

$$\mathcal{T}_{\mathcal{R}} \models \exists(t \to^* t')$$

where $\exists(t \to^* t')$ denotes the existential closure of the formula $t \to^* t'$, can be answered, for any ground-coherent and topmost generalized rewrite theory $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$ with $(\Sigma, B, \vec{E})$ convergent and strongly deterministic, and B having a unification algorithm. In plain English, we want to be able to prove that there is a ground substitution ρ such that $[t\rho]_{E \uplus B} \to^*_{\mathcal{R}} [t'\rho]_{E \uplus B}$ whenever this is the case for the given t, t'. Furthermore, we want to have a *constructive* proof that can exhibit such a ρ. However, in practice it is quite hard, and in general quite hopeless, to try to solve this problem *directly* on $\mathcal{R}$ itself, unless $\mathcal{R}$ satisfies very special and restrictive properties. Briefly put, this is because, although in principle *narrowing* with the rules R modulo $E \uplus B$ provides a sound and complete answer to existential reachability (see [68] for $\mathcal{R}$ unconditional), each narrowing step requires performing $E \uplus B$-unification, which is possible by convergence: (i) under the strong determinism assumption using the unification with constraints technique in [20], and (ii) for unconditional oriented equations $\vec{E}$ it can be performed in Maude by variant unification (without the FVP assumption). But, unfortunately, in general $E \uplus B$-unification is both infinitary and undecidable. This means that when narrowing a term with rules R modulo $E \uplus B$ we may: (i) experience infinite branching; (ii) loop forever without getting any $E \uplus B$-unifiers; and (iii) experience steep performance barriers to compute each $E \uplus B$-unifier, making the overall prospect quite dim.

One of the key goals of this paper is to solve this problem for the *same* topmost generalized rewrite theory $\mathcal{R}$ under mild assumptions, but to do so in a much simpler and much more efficient way, requiring only B_Ω-unification in narrowing steps, by posing it on the *semantically equivalent*, transformed rewrite theory $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$, under the mild assumptions given

[16] By definition, a Σ-term t *B-matches* the Σ-term u with matching substitution α iff $t =_B u\alpha$. A set M of substitutions B-matching t against u is *complete* iff for any substitution β that B-matches t against u there is a substitution $\alpha \in M$ such that for each variable x appearing in u we have $\beta(x) =_B \alpha(x)$.

[17] Thanks to the initiality of $\mathcal{T}_{\mathcal{R}}$, $\mathcal{T}_{\mathcal{R}} \models \exists(t \to^* t')$ holds iff $\mathcal{R} \models \exists(t \to^* t')$ holds. That is, existential reachability is just the satisfaction of the existential closure of an atomic formula (or of a conjunction of atomic formulas [68]) in the theory $\mathcal{R}$.

in Section 4.3, plus the further assumption that the constructors Ω are free[18] modulo axioms B_Ω. Furthermore, using the language of pattern predicates of Section 5, and the fact that for $\mathcal{R}$ ground coherent we have $\mathcal{T}_\mathcal{R} \cong \mathcal{C}_\mathcal{R} = \mathcal{C}_{\overline{\mathcal{R}}^\Omega_{\Sigma_1,l,r}}$, we can now pose on $\mathcal{R}$ considerably more sophisticated existential reachability problems of the form:

$$\mathcal{T}_\mathcal{R} \models \exists (A \rightarrow^* B)$$

where A and B are pattern predicates. And we can then use the transformed theory $\overline{\mathcal{R}}^\Omega_{\Sigma_1,l,r}$ to solve them in a much more efficient and simpler way. Rather than delving directly into narrowing technicalities, I first take a more abstract, and I think conceptually more enlightening, tack based on *predicate transformers* (in the classical sense, see, e.g., [32]). Narrowing will then appear as a natural *implementation* of such predicate transformers.

Rewrite Rules as Predicate Transformers. Let $\overline{\mathcal{R}}^\Omega_{\Sigma_1,l,r}$ be the transformed theory of a ground coherent, topmost generalized rewrite theory $\mathcal{R}$ defined in Section 4.3. Recall that $\mathcal{T}_\mathcal{R} \cong \mathcal{C}_\mathcal{R} = \mathcal{C}_{\overline{\mathcal{R}}^\Omega_{\Sigma_1,l,r}}$, and that for states, by the assumption that the constructors Ω are free modulo axioms B_Ω, we have: $C_{\Sigma/\vec{E},B}|_\Omega = T_{\Omega/B_\Omega}$. Therefore, we have, $\mathcal{P}(C_{\Sigma/E,B,State}) = \mathcal{P}(T_{\Omega/B_\Omega,State})$. This means that the rewrite relation $\rightarrow_{\mathcal{C}_\mathcal{R}}$ is a binary relation $\rightarrow_{\mathcal{C}_\mathcal{R}} \subseteq \mathcal{P}(T_{\Omega/B_\Omega,State})^2$. But since: (i) the functor $\mathcal{P} : \mathbf{Set} \rightarrow \mathbf{Set}$ is the (functor part of) the monad of upper ($\bigvee$) complete semilattices (see, e.g., [67]), and (ii) the category of relations and relation composition is isomorphic to the Kleisli category (see, e.g., [66,67]) for such a monad when we view a relation $Q \subseteq A \times B$ as a function $\widetilde{Q} \equiv \lambda a \in A.\, \{b \in B \mid (a,b) \in Q\} \in \mathcal{P}(B)$, the function $\widetilde{\rightarrow_{\mathcal{C}_\mathcal{R}}} : T_{\Omega/B_\Omega,State} \rightarrow \mathcal{P}(T_{\Omega/B_\Omega,State})$ uniquely extends to an upper complete semilattice homomorphism which, to simplify notation, I denote by:

$$R! : \mathcal{P}(T_{\Omega/B_\Omega,State}) \rightarrow \mathcal{P}(T_{\Omega/B_\Omega,State})$$

i.e., $R! \equiv \lambda V \in \mathcal{P}(T_{\Omega/B_\Omega,State}).\, \{[w] \mid \exists [v] \in V \ s.t.\ [v] \rightarrow_{\mathcal{C}_\mathcal{R}} [w]\} \in \mathcal{P}(T_{\Omega/B_\Omega,State})$. Let us call $R!$ the *predicate transformer* associated to the generalized rewrite theory $\mathcal{R}$. In particular, $R!$ is a $\{\vee,\bot\}$-*endomorphism* of $\mathcal{P}(T_{\Omega/B_\Omega,State})$. Of course, $\mathcal{P}(T_{\Omega/B_\Omega,State})$ will typically have the power of the continuum, so any talk of making the predicate transformer $R!$ computable is utter nonsense. But recall that, if we restrict ourselves to the image subalgebra $[\![PatPred(\Omega,\Sigma)]\!] \subseteq \mathcal{P}(T_{\Omega/B_\Omega,State})$, the prospects for a *computable*, symbolic representation of the predicate transformer $R!$ look much brighter.

Let me put the question more precisely. We have already *lifted* the $\{\vee,\wedge,\bot\}$ operations on $[\![PatPred(\Omega,\Sigma)]\!]$ to symbolic operations in the computable $\{\vee,\wedge,\bot\}$-algebra $\vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee$ by means of the $\{\vee,\wedge,\bot\}$-homomorphism $\vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee \xrightarrow{[\![\_]\!]} \mathcal{P}(T_{\Omega/B_\Omega,State})$. Can we likewise *lift* the predicate transformer $R!$ to a computable, symbolic version as a $\{\vee,\bot\}$-*endomorphism*

$$R! : \vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee \rightarrow \vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee$$

in such a way that we have the identity $[\![\_]\!]; R! = R!; [\![\_]\!]$? The answer is *yes!* It relies in an essential manner on the $\mathcal{R} \mapsto \overline{\mathcal{R}}^\Omega_{\Sigma_1,l,r}$ transformation.

So, let us do it! First of all, recall that: (i) $\vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee$ is the free $\{\vee,\bot\}$-algebra modulo $Ax_\vee$ on the set $AtPatPred(\Omega,\Sigma)/\approx_B$, and (ii) $\mathcal{P}(T_{\Omega/B_\Omega,State})$ trivially satisfies the axioms $Ax_\vee$. Therefore, since $[\![\_]\!]$ is a $\{\vee,\wedge,\bot\}$-homomorphism, and in particular a $\{\vee,\bot\}$-homomorphism, by the *freeness* of $\vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee$, all we need to do to define the computable, symbolic predicate transformer $R!$ and prove the equality $[\![\_]\!]; R! = R!; [\![\_]\!]$ is: (a) to define a computable function $R^\circ! : AtPatPred(\Omega,\Sigma)/\approx_B \rightarrow \vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee$ (which uniquely extends to a computable, symbolic $\{\vee,\bot\}$-endomorphism $R!$); and (b) *check* that $[\![\_]\!]|_{AtPatPred(\Omega,\Sigma)/\approx_B}; R! = R^\circ!; [\![\_]\!]$.

It is in the definition of the computable function $R^\circ!$ that the transformation $\mathcal{R} \mapsto \overline{\mathcal{R}}^\Omega_{\Sigma_1,l,r}$ becomes crucial. Recall that the rewrite rules in $R^\Omega_{\Sigma_1,l,r}$ have the form $l \rightarrow r \ if \ \varphi$, with $l, r \in T_\Omega(X)_{State}$. To be able to *name* each of these rules, I give to each of them a *different* label $\gamma \in \Gamma$, and display them as follows: $\gamma : l \rightarrow r \ if \ \varphi$. Rather than defining the function $R^\circ!$ directly, I define, for each rule $\gamma : l \rightarrow r \ if \ \varphi$ in $R^\Omega_{\Sigma_1,l,r}$, a function $\gamma : AtPatPred(\Omega,\Sigma)/\approx_B \rightarrow \vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee$. Then $R^\circ!$ is defined as the function: $R^\circ! \equiv \lambda [v \mid \psi] \in AtPatPred(\Omega,\Sigma)/\approx_B .\ \bigvee_{\gamma \in \Gamma} \gamma([v \mid \psi]) \in \vee.PatPred(\Omega,\Sigma)/\approx_B \cup Ax_\vee$. Here is the definition of the function γ for a rule $\gamma : l \rightarrow r \ if \ \varphi$ in $R^\Omega_{\Sigma_1,l,r}$:

$$\gamma \equiv \lambda [v \mid \psi].\ \bigvee_{\alpha \in Unif_{B_\Omega}(l=v)} [(r \mid \psi \wedge \varphi)\alpha],$$

where w.l.o.g. we assume that $v \mid \psi$ and $l \rightarrow r \ if \ \varphi$ *share no variables*. All we have left to do to get our desired *computable* predicate transformer $R!$ is to prove:

[18] As explained in Section 5.3, all this generalizes to the case of constructor theories $(\Omega, E_\Omega \cup B_\Omega)$ where $E_\Omega \cup B_\Omega$ is FVP, or even to the more general case of the constructor theories $(\Omega, E_\Omega \uplus B_\Omega) = (\Omega_1 \uplus \Omega_0, E_{\Omega_1} \uplus E_{\Omega_0} \uplus B_{\Omega_1} \uplus B_{\Omega_0})$ of Section 5.3, since both have finitary $E_\Omega \cup B_\Omega$-unification algorithms. To simplify the exposition I treat here the very common case of free constructors modulo B_Ω and, *also but implicitly*, the case where $(\Omega, E_\Omega \uplus B_\Omega) = (\Omega_1 \uplus \Omega_0, E_{\Omega_0} \uplus B_{\Omega_1} \uplus B_{\Omega_0})$. In this second case, the only assumptions needed are that: (i) in the (transformed) rewrite rules $l \rightarrow r \ if \ \varphi$, l and r are Ω_1-terms; and (ii), as explained in Section 5.3, atomic pattern predicates always have the form $u \mid \varphi$ with u a Ω_1-term.

Lemma 3. *For each* $[v \mid \psi] \in AtPatPred(\Omega, \Sigma)/\approx_B$, *we have the set equality* $R![\![v \mid \psi]\!] = [\![R^\circ!([v \mid \psi])]\!]$.

Constrained Narrowing in $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$. Let us now come back to our original goal. Assume, as before, a coherent topmost rewrite theory $\mathcal{R}$ with free constructors modulo B_Ω, so that $C_{\Sigma/\vec{E},B}|_\Omega = T_{\Omega/B_\Omega}$. How can we use the transformation $\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ to solve reachability problems of the form $\mathcal{T}_{\mathcal{R}} \models \exists(A \rightarrow^* B)$, with A and B pattern predicates? Of course, $\mathcal{T}_{\mathcal{R}} \models \exists(A \rightarrow^* B)$, where we assume $vars(A) \cap vars(B) = \emptyset$, just means that there are $[v] \in [\![A]\!]$ and $[w] \in [\![B]\!]$ such that $[v] \rightarrow^*_{\mathcal{C}_{\mathcal{R}}} [w]$. I claim that, in rough outline, *we already know the answer!* Indeed:

1. Since we have defined $\_ \wedge \_$ in a computable way in terms of $\_ \vee \_$, we may assume w.l.o.g. that A and B are finite disjunctions of atomic pattern predicates. In particular, $A \equiv u_1 \mid \varphi_1 \vee \ldots \vee u_n \mid \varphi_n$.
2. By the very definition of the predicate transformer $R! : \mathcal{P}(T_{\Omega/B_\Omega,State}) \rightarrow \mathcal{P}(T_{\Omega/B_\Omega,State})$, $\mathcal{T}_{\mathcal{R}} \models \exists(A \rightarrow^* B)$ holds iff there is a natural number n such that $R!^n([\![A]\!]) \cap [\![B]\!] \neq \emptyset$, where $R!^0$ is the identity function on $\mathcal{P}(T_{\Omega/B_\Omega,State})$, and $R!^{n+1} = R!^n; R!$.
3. But we can use the *computable* predicate transformer $R! : \vee.PatPred(\Omega, \Sigma)/\approx_B \cup Ax_\vee \rightarrow \vee.PatPred(\Omega, \Sigma)/\approx_B \cup Ax_\vee$ to answer this reachability problem symbolically: it will have a positive answer iff there is a natural number n such that $R!^n([A]) \wedge [B]$ denotes a non-empty set. Since $R!^n([A]) \wedge [B]$ will always symbolically evaluate to either $\bot$ or to a finite disjunction of the form $v_1 \mid \psi_1 \vee \ldots \vee v_m \mid \psi_m$, $m \geq 1$, it is all a matter of checking whether we do not get $\bot$ as an answer and there is a j, $1 \leq j \leq m$, such that $[\![v_j \mid \psi_j]\!] \neq \emptyset$.
4. The hardness of answering a question of the form $[\![v_j \mid \psi_j]\!] \neq \emptyset$ will depend on the specific atomic predicate $v_j \mid \psi_j$ and on the background theory T. The best possibility is when T can answer whether φ is *satisfiable* in the initial algebra $T_{\Sigma/E \uplus B}$, but this may not always be decidable.
5. Assuming that we *do* find a j such that $[\![v_j \mid \psi_j]\!] \neq \emptyset$, then we *do* know that there is a concrete rewrite sequence of length n in $\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}$ witnessing that $\mathcal{T}_{\mathcal{R}} \models \exists(A \rightarrow^* B)$ holds; but of course we would like to have an easy way to exhibit such a witness rewrite sequence.

Although points (1)–(5) give us a high-level answer to how to solve symbolic reachability problems under mild assumptions on a topmost $\mathcal{R}$ using the predicate transformer $R!$, several issues remain. The most basic one is how to best *implement* $R!$. Of course, since $R!$ is a $\{\vee, \bot\}$-endomorphism, it is enough for us to implement $R^\circ!$. Recall that, by definition, $R^\circ!$ is just the function:

$$R^\circ! \equiv \lambda[v \mid \psi]. \bigvee_{l \rightarrow r \text{ if } \varphi \in R^{\Omega}_{\Sigma_1,l,r}} \bigvee_{\alpha \in Unif_{B_\Omega}(l=v)} [(r \mid \psi \wedge \varphi)\alpha].$$

This definition wears a very direct answer to the implementation question on its sleeve: *by constrained narrowing!*

Definition 10. The following narrowing relations can be defined:

1. Let (Σ, B, R, ϕ) be a standard rewrite theory where B are regular and linear axioms having a B-unification algorithm and R is a set of strictly B-coherent rewrite rules. The *narrowing modulo* B relation between two terms $u, v \in T_\Sigma(X)$, denoted $u \rightsquigarrow_{R,B} v$, or $u \overset{\alpha}{\rightsquigarrow}_{R,B} v$, holds iff there exists a non-frozen position p in u, a rule $l \rightarrow r$ in R — which we assume *renamed* if necessary to ensure $vars(t) \cap vars(l \rightarrow r) = \emptyset$ — and a B-unifier $\alpha \in Unif_B(l = u|_p)$ such that $v = (u[r]_p)\alpha$.
2. Let $\mathcal{R} = (\Sigma, B, R, T, \phi)$ be a generalized rewrite theory where B are regular and linear axioms having a B-unification algorithm and R is a set of strictly B-coherent rewrite rules. The *constrained narrowing modulo* B relation between two *constrained terms* $u \mid \psi$ and $v \mid \chi$ — where $u, v \in T_\Sigma(X)$, and $\psi, \chi \in QFForm(\Sigma)$ — denoted $u \mid \psi \rightsquigarrow_{R,B} v \mid \chi$, or $u \mid \psi \overset{\alpha}{\rightsquigarrow}_{R,B} v \mid \chi$, holds iff there exists a non-frozen position p in u, a rule $l \rightarrow r \text{ if } \varphi$ in R — which we assume *renamed* if necessary to ensure $vars(u \mid \psi) \cap vars(l \rightarrow r \text{ if } \varphi) = \emptyset$ — and a B-unifier $\alpha \in Unif_B(l = u|_p)$ such that $v \mid \chi = (u[r]_p \mid \psi \wedge \varphi)\alpha$.
3. Let $\mathcal{R} = (\Sigma, E \uplus B, R, T, \phi)$ be a topmost rewrite theory where $(\Sigma, B, \vec{E})$ is convergent and strongly deterministic, has a constructor subtheory $(\Sigma, B_\Omega, \emptyset)$ with B_Ω regular and linear axioms having a B_Ω-unification algorithm, and satisfies the additional requirements in Section 4.3, so that $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ is its ground semantically equivalent transformed theory. The *constrained narrowing modulo* B_Ω relation between two *constrained constructor patterns* $u \mid \psi$ and $v \mid \chi$, where $u, v \in T_\Omega(X)_{State}$, denoted $u \mid \psi \rightsquigarrow_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} v \mid \chi$, or $u \mid \psi \overset{\alpha}{\rightsquigarrow}_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} v \mid \chi$, holds iff there is a rule $l \rightarrow r \text{ if } \varphi$ in $R^{\Omega}_{\Sigma_1,l,r}$ — which we assume *renamed* if necessary to ensure $vars(u \mid \psi) \cap vars(l \rightarrow r \text{ if } \varphi) = \emptyset$ — and a B_Ω-unifier $\alpha \in Unif_{B_\Omega}(l = u)$ such that $v \mid \chi = (r \mid \psi \wedge \varphi)\alpha$.

The following, basic observations about the above definitions may be helpful:

- The *standard* narrowing relation $u \leadsto_R v$ is just the special case of (1) where $B = \emptyset$ and $\phi(f) = \emptyset$ for each $f \in \Sigma$, and in many approaches the even more special case where, furthermore, Σ is unsorted.
- The special instance of (2) where $\mathcal{R} = (\Sigma, B, \vec{E}, \emptyset)$ is a *strongly deterministic, convergent, conditional theory* has been studied in detail in [20]. This work is of special interest for symbolic equational reasoning, since it provides a narrowing-based *constrained $E \uplus B$-unification* algorithm.

That the constrained narrowing relation $\leadsto_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega}$ is a *correct implementation* of the function $R^{\circ}!$ is now obvious, since we can give the following, alternative definition of $R^{\circ}!$:

$$(\ddagger)\quad R^{\circ}! \equiv \lambda[v \mid \psi].\ \bigvee_{v|\psi \overset{\alpha}{\leadsto}_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega}(r|\psi\wedge\varphi)\alpha} [(r \mid \psi \wedge \varphi)\alpha].$$

The exact relationship between the constrained narrowing relation $\leadsto_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega}$ and the rewrite relation $\rightarrow_{\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}}$ on B_Ω-equivalence classes of constructor terms of sort *State* is clarified by the following

Lemma 4. *(Lifting Lemma). Let $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ be the transformed theory of a ground coherent, topmost generalized rewrite theory $\mathcal{R}$ defined in Section 4.3 and such that its constructors Ω are free modulo axioms B_Ω. Then:*

1. *(Completeness). For any $w, w' \in T_{\Omega_{State}}$ such that: (i) $[w] \in [\![v \mid \psi]\!]$, and (ii) $[w] \rightarrow_{\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}} [w']$, there is a constrained narrowing step $v \mid \psi \overset{\alpha}{\leadsto}_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} (r \mid \psi \wedge \varphi)\alpha$ such that $[w'] \in [\![(r \mid \psi \wedge \varphi)\alpha]\!]$.*
2. *(Soundness). For any constrained narrowing step $v \mid \psi \overset{\alpha}{\leadsto}_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} (r \mid \psi \wedge \varphi)\alpha$ such that $[\![(r \mid \psi \wedge \varphi)\alpha]\!] \neq \emptyset$ and each $[w'] \in [\![(r \mid \psi \wedge \varphi)\alpha]\!]$ there exist $[w] \in [\![v \mid \psi]\!]$ such that $[w] \rightarrow_{\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}} [w']$.*

As an immediate consequence of the above Lifting Lemma, we can characterize existential reachability in what might be called a local, "piecemeal" way, as opposed to the global way provided by the predicate transformer $R!^n$. But before doing so we need a slight technicality. To avoid variable capture nonsense, the reflexive-transitive closure $\leadsto^*_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega}$ of the constrained narrowing relation $\leadsto_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega}$ should only use narrowing sequences of the form:

$$v_1 \overset{\alpha_1}{\leadsto}_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} v_2, \ldots, v_n \overset{\alpha_n}{\leadsto}_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} v_{n+1}$$

where for each substitution α_i, $1 \leq i \leq n$, all variables in $ran(\alpha_i)$ are *fresh*, i.e., they are disjoint from all other variables used earlier in the sequence. We then write $v_1 \overset{\alpha\ *}{\leadsto}_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} v_{n+1}$, where $\alpha \equiv \alpha_1 \ldots \alpha_n$ is called the *accumulated substitution* along the sequence.

Theorem 9. *(Existential Reachability Theorem). Let $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ be the transformed theory of a ground coherent, topmost generalized rewrite theory $\mathcal{R}$ defined in Section 4.3 and such that its constructors Ω are free modulo axioms B_Ω. Then, $\mathcal{T}_{\mathcal{R}} \models \exists(A \rightarrow^* B)$ holds, say, with $A \equiv u_1 \mid \varphi_1 \vee \ldots \vee u_m \mid \varphi_m$, iff there is an i, $1 \leq i \leq m$, and a narrowing sequence $u_i \mid \varphi_i \leadsto^*_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} v \mid \psi$ such that $[\![(v \mid \psi) \wedge B]\!] \neq \emptyset$. Furthermore, using the rules and B_Ω-unifiers enabling each constrained narrowing step in $u_i \mid \varphi_i \leadsto^*_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} v \mid \psi$, and the method described in the proof of Lemma 3, we can effectively find for each $[w'] \in [\![v \mid \psi \wedge B]\!]$ an element $[w] \in [\![u_i \mid \varphi_i]\!]$ such that $[w] \rightarrow^*_{\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}} [w']$.*

Practical Issues. There are several practical issues that need to be addressed in order for reachability analysis by constrained narrowing to get real traction. The discussion below makes no pretense of completeness; but all issues treated are surely relevant. The first big issue is how to perform the non-emptiness check $[\![(v \mid \psi) \wedge B]\!] \neq \emptyset$. This is closely intertwined with a second, very practical issue, namely, *state space reduction*. To begin with, whenever possible we should try to detect when in a narrowing sequence $u_i \mid \varphi_i \leadsto^*_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} v \mid \psi$ we have $[\![v \mid \psi]\!] = \emptyset$, so that we can stop it. Such emptiness can be decided if: (i) there is a subsignature $\Sigma_0 \subseteq \Sigma$ such that background theory T is such that T-satisfiability of QF Σ_0-formulas corresponds to their satisfiability in $T_{\Sigma/E\uplus B}$ and is decidable, (ii) all Ω-substitution instances of QF Σ_0-formulas are QF Σ_0-formulas, (iii) all conditions in the rewrite rules are QF Σ_0-formulas, and (iv) all QF formulas appearing in the reachability problem $\mathcal{T}_{\mathcal{R}} \models \exists(A \rightarrow^* B)$ are Σ_0-formulas. This makes the non-emptiness problem $[\![(v \mid \psi) \wedge B]\!] \neq \emptyset$ decidable.

The million dollar question is how to check: (i) $[\![v \mid \psi]\!] = \emptyset$ in a narrowing sequence $u_i \mid \varphi_i \leadsto^*_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} v \mid \psi$, and (ii) the non-emptiness $[\![(v \mid \psi) \wedge B]\!] \neq \emptyset$ when satisfiability of QF formulas is undecidable. Although formally these two problems

are closely related, pragmatically, the methods used to solve them can, and should, be quite different. To check $[\![v \mid \psi]\!] = \emptyset$ we should use relatively *inexpensive formula simplification methods*, including SAT-solving, rewriting with $\vec{E}$ modulo B, and also rewriting with $\vec{E^{\equiv}}$ modulo B in the equality enrichment of $(\Sigma, E \uplus B)$ [53]. Instead, to check the non-emptiness $[\![(v \mid \psi) \wedge B]\!] \neq \emptyset$ we should: (i) first of all perform the cheap, B_Ω-unification-based check that $v \mid \psi \wedge B \neq \bot$, and (ii) in case this holds, we must have $(v \mid \psi) \wedge B = w_1 \mid \psi_1 \vee \ldots \vee w_k \mid \psi_k$, and we need to check if there is a j, $1 \leq j \leq k$, such that ψ_j is satisfiable in $T_{\Sigma/E \uplus B}$. But this is equivalent to $T_{\Sigma/E \uplus B} \models \exists \psi_j$, and therefore to $T_{\Sigma/E \uplus B} \not\models \forall \neg \psi_j$. Therefore, *if* $[\![(v \mid \psi) \wedge B]\!] \neq \emptyset$, we will have $T_{\Sigma/E \uplus B} \not\models \forall \neg \psi_j$ for some j, which can be automatically checked by a refutationally complete inductive theorem prover, such as those based on various forms of superposition, e.g., [17,27].

The search space provided by constrained narrowing can be understood as a *narrowing tree* (a narrowing forest when the initial symbolic state A is a disjunction of atomic predicates). Further *state space reduction* in the narrowing search process, which is a form of infinite state symbolic model checking, can be obtained by various ways of *merging* states. The simplest one is the *folding*, i.e., *subsumption*, of symbolic states [9]. In the context of constrained constructor patterns, folding can happen if we have two nodes $v \mid \psi$ and $v' \mid \psi'$ in the narrowing tree such that $v \mid \psi \sqsubseteq_{B_\Omega} v' \mid \psi'$. We can then "fold" or "merge" the less general symbolic state $v \mid \psi$ into the more general one $v' \mid \psi'$. More generally, even more powerful symbolic state merging methods, such as those proposed in [11], can be used when satisfiability of the given QF formulas is decidable.

Combining the two ideas of: (i) *formula simplification*, e.g., the elimination of a symbolic state $v \mid \psi$ by some simplification methods when we can show that $[\![v \mid \psi]\!] = \emptyset$, or just expressing $v \mid \psi$ by a simpler, equivalent formula; and (ii) *state space reduction*, for example, by subsumption, so that a symbolic state is "folded" into a more general one, we arrive at a useful notion of a *state space reduction equivalence*, which is the reflexive, symmetric and transitive closure $\approx_r$ of a given family of simplification and state space reduction steps according to some chosen techniques such as those discussed above. Note that both formula simplification and state space reduction steps are *formula transformations*, where some pattern formula A is transformed into a simpler, semantically equivalent one A'. For example, subsumption is the conditional formula transformation:

$$(v \mid \psi) \vee (v' \mid \psi) \rightarrow (v' \mid \psi') \;\; \textit{if} \;\; v \mid \psi \sqsubseteq_{B_\Omega} v' \mid \psi'.$$

Of course, we will apply those steps in a state-space reducing way, say, beginning with a symbolic state space description as a pattern predicate D and obtaining an equivalent but more compact description D' such that $D \approx_r D'$. The crucial property about the relation $\approx_r$ is that it is *semantics-preserving*, in the sense that $D \approx_r D' \Rightarrow [\![D]\!] = [\![D']\!]$.

Besides increasing performance by allowing a possibly much more compact description as a pattern predicate of the narrowing forest explored so far, equivalences of the form $\approx_r$ may be crucial in the following sense. The set of *all* states reachable from $[\![A]\!]$ can be symbolically described as the *infinite* disjunction:

$$\bigvee_{i \in \mathbb{N}} R!^i(A).$$

However, the space of all states reachable from $[\![A]\!]$ may be *finitely describable* if we can reach a *fixpoint* after finitely many steps in the following sense:

Definition 11. Given a pattern predicate A, the symbolic predicate transformer $R!$ has a *fixpoint*[19] for A after finitely many iterations relative to the state space reduction equivalence $\approx_r$ iff there is a $k \in \mathbb{N}$ and pattern predicate formulas C and D such that:

$$R!^{k+1}(A) \approx_r C \sqsubseteq_{B_\Omega} D \approx_r \bigvee_{0 \leq i \leq k} R!^i(A).$$

By repeated application of Lemma 3, the formula $\bigvee_{0 \leq i \leq k} R!^i(A)$ describes the set of all states reachable from $[\![A]\!]$ in k *or fewer steps*. Indeed, each atomic disjunct in this formula is exactly one of the symbolic states reachable by constrained narrowing in k or fewer steps from some atomic disjunct in A. D is therefore a compact representation of the symbolic state space describing all states reachable from A in k or fewer steps. But when D is a fixpoint of $R!$ we get the set-theoretic equality $[\![D]\!] = [\![\bigvee_{0 \leq i \leq k} R!^i(A)]\!] = \bigcup_{i \in \mathbb{N}} [\![R!^i(A)]\!]$, and therefore a *finite* symbolic description of the set of *all* states *reachable* from states in $[\![A]\!]$.

Last, but not least, we should consider the eminently practical issue of how to best implement constrained narrowing search from an initial pattern predicate to a pattern predicate goal for a topmost rewrite theory $\mathcal{R}$ using the transformed theory $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1, l, r}$. For example, Maude does already have an implementation of *unconditional narrowing* modulo B (Case (1) in Definition 10). But how should we implement Case (3) in Definition 10? The extremely good news is that there is no need for a special implementation of Case (3), because it can be reduced to Case (1) by a simple theory transformation.

[19] Strictly speaking, the fixpoint is that of the monotonic function $\lambda[A].\, [A] \vee R!([A])$, but this is just a technicality.

Such a theory transformation was already implicit in the discussion of B-equality between constrained constructor patterns, $(u\sigma \mid \varphi\sigma) =_B (u' \mid \varphi')$, in Section 5, where we viewed $u \mid \varphi$ as a *term* in an extended signature consisting of $\mid$-separated pairs of terms, whose first components are Σ-terms, and whose second components are QF Σ-formulas. The point is that, using an extended equational theory defining such pairs, we can transform the *conditional* theory $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r} = (\Sigma, E \cup B, R^{\Omega}_{\Sigma_1,l,r}, T, \phi)$ into an *unconditional* one $(\Sigma^{\bullet}, E \cup B, R^{\bullet,\Omega}_{\Sigma_1,l,r})$, where $\Sigma^{\bullet}$ is the extended signature of $\mid$-separated pairs, and where for each rule $l \to r$ *if* φ in $R^{\Omega}_{\Sigma_1,l,r}$ we add an unconditional rule $l \mid Q \to r \mid Q \wedge \varphi$ to $R^{\bullet,\Omega}_{\Sigma_1,l,r}$, where the variable Q ranges over the sort *QFForm* of QF formulas in $\Sigma^{\bullet}$.

6.3. Narrowing-based invariant verification

Invariants are the most basic *safety properties* of a system. They make sense for any transition system, so I start by giving some basic definitions about invariants and coinvariants for any *transition system* $\mathcal{Q} = (Q, \to_{\mathcal{Q}})$, that is, for any set Q of *states* together with a *transition relation* $\to_{\mathcal{Q}} \subseteq Q^2$.

Definition 12. (Invariants, Coinvariants, Stable and Costable Sets). Given a transition system $\mathcal{Q} = (Q, \to_{\mathcal{Q}})$ and a subset $Q_0 \subseteq Q$ of *initial states*, a subset $I \subseteq Q$ (resp. $C \subseteq Q$) is called an *invariant* (resp. a *coinvariant*) for $\mathcal{Q}$ from Q_0 iff for each $a \in Q_0$ and $b \in Q$, $a \to^*_{\mathcal{Q}} b$ implies $b \in I$ (resp. $b \notin C$), where $\to^*_{\mathcal{Q}}$ denotes the reflexive-transitive closure of $\to_{\mathcal{Q}}$. Note that I is an invariant (resp. C a coinvariant) from Q_0 iff $Q \setminus I$ (resp. $Q \setminus C$) is a co-invariant (resp. invariant) from Q_0.

A subset $A \subseteq Q$ is called *stable* (resp. *costable*) in $\mathcal{Q}$ iff for each $a \in A$ and $b \in Q$, $a \to_{\mathcal{Q}} b$ (resp. $b \to_{\mathcal{Q}} a$) implies $b \in A$. An invariant I (resp. a coinvariant C) for $\mathcal{Q}$ from Q_0 is called *inductive* iff I (resp. C) is stable (resp. costable). Note that inductiveness is equivalent to I stable and $Q_0 \subseteq I$ (resp. C costable and $Q_0 \cap C = \emptyset$).

In particular, for a topmost rewrite theory $\mathcal{R}$ satisfying the requirements in Section 4.3 for applying the $\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ transformation, we have the transition system $(T_{\Omega,B_\Omega,State}, \to_{\mathcal{C}_{\mathcal{R}}})$. Taking pattern predicates as the formal language to symbolically specify subsets of $T_{\Omega,B_\Omega,State}$ we can then use constrained narrowing with the rules in the transformed theory $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ to reason about invariants and coinvariants in $(T_{\Omega,B_\Omega,State}, \to_{\mathcal{C}_{\mathcal{R}}})$, that is, in the initial model $\mathcal{C}_{\mathcal{R}}$ of $\mathcal{R}$.

Before going any further, a very simple, yet crucial observation is in order, namely, the "invertibility" of the rules in the transformed theory $\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ already pointed out in Section 4.3. That is, any rule $l' \to r'$ *if* $(\varphi\gamma)!_{\vec{E},B}$ in $R^{\Omega}_{\Sigma_1,l,r}$ is such that *both* l' and r' are Ω-terms, so that the "inverse rule" $r' \to l'$ *if* $(\varphi\gamma)!_{\vec{E},B}$ shares the same property of rewriting constructor terms to constructor terms. In fact, it follows immediately from Proposition 1 that for any $[u], [v] \in T_{\Omega,B_\Omega,State}$, (i) $[u] \to_{\mathcal{C}_{\mathcal{R}}} [v]$ iff (ii) there is a $\vec{E}_\Omega, B_\Omega$-normalized ground substitution δ and a rewrite rule $l' \to r'$ *if* $(\varphi\gamma)!_{\vec{E},B}$ in $R^{\Omega}_{\Sigma_1,l,r}$ such that $u =_{B_\Omega} l'\delta$, $v =_{B_\Omega} r'\delta$, and $T \models (\varphi\gamma)!_{\vec{E},B}\delta$, so that $u \to_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} r'\delta$, with $v =_{B_\Omega} r'\delta$, iff (iii) using the inverse rule $r' \to l'$ *if* $(\varphi\gamma)!_{\vec{E},B}$ we get an inverse rewrite step $v \to_{R^{-1\,\Omega}_{\Sigma_1,l,r},B_\Omega} l'\delta$, with $u =_{B_\Omega} l'\delta$, where

$$R^{-1\,\Omega}_{\Sigma_1,l,r} = \{r' \to l' \text{ if } (\varphi\gamma)!_{\vec{E},B} \mid l' \to r' \text{ if } (\varphi\gamma)!_{\vec{E},B} \in R^{\Omega}_{\Sigma_1,l,r}\}$$

iff (iv) $[v](\to_{\mathcal{C}_{\mathcal{R}}})^{-1}[u]$. That is, the inverse relation $(\to_{\mathcal{C}_{\mathcal{R}}})^{-1}$ of the transition relation $\to_{\mathcal{C}_{\mathcal{R}}}$ on $T_{\Omega,B_\Omega,State}$ is precisely the transition relation associated to the initial model of the "inverse" rewrite theory $\overline{\mathcal{R}^{-1}}^{\Omega}_{\Sigma_1,l,r} = (\Sigma, E \cup B, R^{-1\,\Omega}_{\Sigma_1,l,r}, T, \phi)$. Therefore, we can use the constrained narrowing relation $\rightsquigarrow_{R^{-1\,\Omega}_{\Sigma_1,l,r},B_\Omega}$ to symbolically reason *backwards*, i.e., with $(\to_{\mathcal{C}_{\mathcal{R}}})^{-1}$, about transitions in the initial model $\mathcal{C}_{\mathcal{R}}$ of $\mathcal{R}$.

Throughout I of course assume that $\mathcal{R}$ satisfying the requirements in Section 4.3. Since to reason about invariants in $(T_{\Omega,B_\Omega,State}, \to_{\mathcal{C}_{\mathcal{R}}})$ we will use pattern predicates to symbolically specify subsets, our chosen set of initial states will be specified by a pattern predicate Q_0, and the chosen invariant (resp. coinvariant) by pattern predicates I (resp. C). Constrained narrowing can then be used to reason about invariants and coinvariants in the following sense:

Forwards Reachability Analysis. Suppose that we wish to verify that in the transition system $(T_{\Omega,B_\Omega,State}, \to_{\mathcal{C}_{\mathcal{R}}})$ the pattern predicate C correctly specifies a coinvariant from initial states specified by the pattern predicate Q_0. By definition, this exactly means that

$$\bigcup_{i \in \mathbb{N}} [\![R!^i(Q_0)]\!] \cap [\![C]\!] = \emptyset.$$

But this holds if and only if for each atomic pattern predicate $u \mid \varphi$ in Q_0 and each constrained narrowing sequence $u \mid \varphi \rightsquigarrow^*_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} v \mid \psi$ we have $[\![(v \mid \psi) \wedge C]\!] = \emptyset$. Therefore, "all we have to do" is to perform breadth first search from all atomic pattern predicates in Q_0, performing the emptiness check $[\![(v \mid \psi) \wedge C]\!] = \emptyset$ for each atomic predicate $v \mid \psi$ thus reached. To do this, all the practical considerations at the end of Section 6.2 should be taken into account. For example, we should use a state space reduction equivalence $\approx_r$ to keep the symbolic state space already explored in as compact a form as possible. Essentially, three things can then happen:

1. If C is *not* a coinvariant from Q_0, it must be the case that there is a number k and a narrowing sequence $u \mid \varphi \leadsto^k_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} v \mid \psi$ from an atomic predicate $u \mid \varphi$ in Q_0 such that $[\![(v \mid \psi) \wedge C]\!] \neq \emptyset$. Furthermore, thanks to Theorem 9, a *counterexample* ground computation can be constructed witnessing the violation of the invariant implicitly defined by C. As pointed out at the end of Section 6.2, if indeed $[\![(v \mid \psi) \wedge C]\!] \neq \emptyset$ holds, it should be possible to establish this by using a refutationally complete superposition-based inductive theorem prover.
2. If $R!$ has a fixpoint for Q_0 after k iterations relative to $\approx_r$ and we have been able to show for each atomic pattern predicate $u \mid \varphi$ in Q_0 and each constrained narrowing sequence $u \mid \varphi \leadsto^n_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} v \mid \psi$ with $0 \leq n \leq k$ we have $[\![(v \mid \psi) \wedge C]\!] = \emptyset$, then we have *proved* that C is indeed a coinvariant from Q_0.
3. If no fixpoint of $R!$ for Q_0 is found, even though C may be a coinvariant from Q_0, only *bounded model checking* up to a given depth k can be effectively performed for this property following this method.

Backwards Reachability Analysis. To verify that in the transition system $(T_{\Omega,B_\Omega,State}, \rightarrow_{\mathcal{C}_\mathcal{R}})$ the pattern predicate C correctly specifies a coinvariant from initial states specified by the pattern predicate Q_0 means verifying that $\mathcal{R} \not\models \exists(Q_0 \rightarrow^* C)$, which is equivalent to proving $\overline{\mathcal{R}^{-1}}^{\Omega}_{\Sigma_1,l,r} \not\models \exists(C \rightarrow^* Q_0)$. But this exactly means that

$$\bigcup_{i \in \mathbb{N}} [\![R^{-1}!^i(C)]\!] \cap [\![Q_0]\!] = \emptyset,$$

where $R^{-1}!$ denotes the predicate transformer associated to the rules ${R^{-1}}^{\Omega}_{\Sigma_1,l,r}$ in the theory $\overline{\mathcal{R}^{-1}}^{\Omega}_{\Sigma_1,l,r}$. As for forwards narrowing verification, we try to verify this by breadth first search from all atomic pattern predicates in C, but now using the inverse constrained narrowing relation $\leadsto_{{R^{-1}}^{\Omega}_{\Sigma_1,l,r},B_\Omega}$; and for each atom $v \mid \psi$ thus reached we perform the emptiness check $[\![(v \mid \psi) \wedge Q_0]\!] = \emptyset$. Similarly to the forwards case, three things can happen:

1. If C is *not* a coinvariant from Q_0, it must be the case that there is a number k and a narrowing sequence $u \mid \varphi \leadsto^k_{{R^{-1}}^{\Omega}_{\Sigma_1,l,r},B_\Omega} v \mid \psi$ from an atomic predicate $u \mid \varphi$ in Q_0 such that $[\![(v \mid \psi) \wedge C]\!] \neq \emptyset$. Furthermore, a counterexample ground computation can be constructed witnessing the violation of the invariant implicitly defined by C.
2. If $R^{-1}!$ has a fixpoint for C after k iterations relative to $\approx_r$ and we have been able to show for each atomic pattern predicate $u \mid \varphi$ in C and each constrained narrowing sequence $u \mid \varphi \leadsto^n_{{R^{-1}}^{\Omega}_{\Sigma_1,l,r},B_\Omega} v \mid \psi$ with $0 \leq n \leq k$ that $[\![(v \mid \psi) \wedge Q_0]\!] = \emptyset$, then we have *proved* that C is indeed a coinvariant from Q_0.
3. If no fixpoint of $R^{-1}!$ for C is found, even though C may be a coinvariant from Q_0, only *backwards bounded model checking* up to a given depth k can be effectively performed for this property following this method.

This backwards narrowing method is indeed the one utilized in the Maude-NPA tool [35], where $\mathcal{R}$ specifies the behavior of honest principals and a Dolev-Yao intruder in a cryptographic protocol, C is an "attack state" where a malicious intruder has broken the protocol, and Q_0 specifies the initial states where no protocol steps have yet been taken and the intruder only knows public information. The remarkable effectiveness with which Maude-NPA can reach a fixpoint in many practical cases is due to its quite sophisticated state space reduction equivalence $\approx_r$ (see [36]). The formal setting for Maude-NPA is close but not exactly the same as the one adopted here. However, as here, some constraints are also carried along when performing (backwards) narrowing steps [37].

Verifying Inductive Invariants. Finding an invariant I from Q_0 that is furthermore *inductive* can be nontrivial. However, *if* we are able to specify such an invariant I, *then* verifying that it is indeed an inductive invariant can be relatively easy to do by constrained narrowing, because, unlike in the just-described forwards and backwards methods, we do *not* need to rely on reaching a fixpoint. Indeed, I will be an inductive invariant from Q_0 iff we have the following two inclusions: $[\![R!(I)]\!] \subseteq [\![I]\!]$, and $[\![Q_0]\!] \subseteq [\![I]\!]$. But a sufficient condition for this is to find pattern predicates I', I'', A, Q_0' such that:

$$R!(I) \approx_r A \sqsubseteq_{B_\Omega} I' \approx_r I \quad \textit{and} \quad Q_0 \approx_r Q_0' \sqsubseteq_{B_\Omega} I'' \approx_r I.$$

The great advantage in this case is that $R!(I)$ can be computed as the disjunction of atomic pattern predicates obtained by performing all *one-step* constrained narrowing steps with $\leadsto_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega}$ from the atomic predicates in I.

Verifying Inductive Coinvariants. The method is entirely similar to that for verifying inductive invariants. C will be an inductive coinvariant from Q_0 iff we have: $[\![R^{-1}!(C)]\!] \subseteq [\![C]\!]$, and $[\![Q_0]\!] \cap [\![C]\!] = \emptyset$. But a sufficient condition for this is to find pattern predicates C', A such that $R^{-1}!(C) \approx_r A \sqsubseteq_{B_\Omega} C' \approx_r C$ and proving $[\![Q_0]\!] \cap [\![C]\!] = \emptyset$. And, of course, $R^{-1}!(C)$ can be computed as the disjunction of atomic pattern predicates obtained from some atomic predicate in C by performing all *one-step* (backwards) constrained narrowing steps with $\leadsto_{{R^{-1}}^{\Omega}_{\Sigma_1,l,r},B_\Omega}$.

7. Related work and conclusions

Related work falls into three main areas: (i) rewriting techniques; (ii) symbolic model checking; and (iii) Constraint Logic Programming (CLP) and theorem proving.

Related Work on Rewriting Techniques. As mentioned in the Introduction, one of the main goals of proposing generalized rewrite theories is to unify various forms of conditional rewriting with constraints in their conditions that have appeared in somewhat different fashions. Perhaps the work most closely related to this one is that on rewriting modulo SMT [87,7] and the closely related work [7]. In rewriting modulo SMT, rewriting happens modulo axioms B, and there are no other equations, except those defining a Σ_0-reduct of the theory's initial algebra with decidable QF-satisfiability in the sense explained in Section 5.3. Furthermore, the rewrite rules can always be put in the format $l \to r$ *if* φ, where l and r are constructor terms not involving any Σ_0-subterms and φ is a QF Σ_0-formula. Rewriting modulo SMT provides a sound and complete method for reasoning about *existential reachability* problems, very much like it is done in Section 6.2. Perhaps the best way of understanding the relationships of this work with rewriting modulo SMT is by seeing rewriting modulo SMT as a restricted form of constrained narrowing. Specifically, by realizing that: (i) rewriting modulo SMT can be viewed as the special case of constrained narrowing where the B-unification problem $u = l$ between the term u in an atomic pattern predicate $u \mid \varphi$ and the lefthand side of a rule $l \to r\ \varphi$ can be solved by B-matching u against l, so that $u =_B l\alpha$, and (ii) since in pattern predicates and in rules we only consider constraints φ that are QF Σ_0-formulas, whose satisfiability in the Σ_0-reduct of the theory's initial algebra is assumed decidable, we are in an easier situation to reason about emptiness, non-emptiness and subsumption of pattern predicates than in the more general case considered in Section 6.2, where such constraints can be arbitrary QF Σ-formulas. Observations (i) and (ii) clearly indicate that rewriting modulo SMT can be implemented more efficiently than constrained narrowing but has a more limited scope of applicability, both in terms of the problems that it can solve and of the rewrite theories to which it can be applied. In fact, these two techniques complement each other.

There is a second body of work in which the notion of constrained rewriting has a completely different meaning and semantics than that of rewriting modulo SMT, namely, a *universal* meaning exactly in the sense of Section 6.1 in this paper. This work includes, for example, the work of Ayala-Rincón [8], Falke and Kapur [42,43,40], and Kop and Nishida [60,61]. Something common to all these approaches is that, in addition to the standard notion of rewriting $u \to_{\mathcal{R}} v$ essentially coinciding with that in Section 6.1 except for minor technical differences, an additional notion of what might be called *constrained contextual rewriting* is also used. This notion is also universal. It allows rewriting a constrained term $u \mid \varphi$ using φ as a "context" that can be assumed, so that a rewrite $u \mid \psi \to_{\mathcal{R}} v \mid \psi$ exists if there is a rule $l \to r$ *if* φ and a subterm of u matching l with substitution α (were decidable satisfiability of φ, ψ and their instances in a given reduct model is assumed) such that the formula $\psi \Rightarrow \varphi\alpha$ is valid in the given reduct model and v is the result of rewriting u with such a rule and substitution at that position. At least in [8] and [42,43,40], the semantics given to such rewrite theories is clearly *equational*. Therefore, the goal is to achieve a more powerful, symbolic form of equational reasoning, so that notions such as termination and confluence become crucial and are studied for these theories. For example, conditional termination for them is studied in, e.g., [42,40], and confluence in [43,40]. The work of Kop and Nishida [60,61] is similar, but it seems to go beyond equational reasoning in some cases by allowing extra variables in conditions and righthand sides that can model "open system behavior" such as, for example, input-output behavior in a programming language's symbolic semantics. As I discuss later, something common to all the works in [8,42,43,40,60,61] is a shared interest in inductive theorem proving applications.

Related Work on Symbolic Model Checking. By describing sets of states symbolically by formulas or by suitable automata, symbolic methods can be applied to model checking verification. Traditionally, sets of states have been encoded as Boolean formulas, and BDDs or SAT solvers are used for model checking such systems. However, such boolean representations are not enough to deal with unbounded data structures and with nontrivial control structures such as recursive function calls. In order to cope with these, SMT-based model checking, where sets of states and transitions are symbolically described using logical formulas in decidable theories that can be handled by SMT solvers, has been introduced, e.g., [82,30,88,79,49,50,6, 29,45,46,77,95,99,18]. Many SMT-based model checking techniques have been developed for a wide range of applications, including array-based systems, e.g., [49,50], hardware systems, e.g., [99], real-time and hybrid systems, e.g., [94,21], and programing languages, e.g., [6,29,13,77,56,52].

Instead of using formulas solvable by an SMT solver to describe states, one can describe sets of states that are languages effectively specified using some kind of automata. In this way, a style of automata-based infinite-state model checking has also been developed in work such as, e.g., [1,16,14,15,48,80,5,4,3].

Both the SMT-based and automata-based approaches to symbolic model checking are unified in the constrained narrowing approach presented in Sections 6.2–6.3. Let me explain what the exact relationship is. The key point is that atomic pattern predicates $u \mid \varphi$ provide a very expressive way of symbolically describing sets of states *without prescribing a fixed format for state representation*. SMT-based model checking approaches typically correspond to specific formats for state representation, where the state pattern u is either a fixed-length *vector* of so-called "state variables," or an array pattern. But for systems with nontrivial state structures such as, for example, many distributed systems, such fixed formats for states can easily become a straight jacket.

Another way in which constrained narrowing extends SMT-based model checking is by dropping the requirement that satisfiability of the constraint φ is decidable. Of course, in the decidable case one enjoys many advantages. But this comes at the price of lack of extensibility: as soon as some functions in a system's state push it outside the case where satisfiability of constraints is decidable, one is out of luck. Instead, the optimistic approach taken in constrained narrowing allows symbolic model checking with constraints whose satisfiability may not be decidable by implicitly relying on a more intimate combination of symbolic model checking and inductive theorem proving, where an inductive theorem prover becomes an *oracle* that is consulted for dealing with constraints outside the decidable fragment. The good news is that there is a wealth of experience and useful techniques in SMT-based model checking that can be leveraged to make this more general form of symbolic model checking effective in practice.

I now explain how constrained narrowing is related to symbolic model checking approaches based on standard finite automata or on tree automata. Since finite automata are just the special case of Σ-tree automata where the operators in Σ are unary, it is enough to consider tree-automata-based symbolic model checking. The key point is that, as explained in [25,76], tree automata and order-sorted signatures *are exactly the same thing*, and that the specification language of free (i.e., $B_\Omega = \emptyset$) order-sorted constructor patterns u that are *linear*, i.e., have no repeated variables, is *closed under all Boolean operations* [76], just as tree automata are, and, as explained in [76], linear patterns have essentially the same expressive power as tree automata. Specifically, they are essentially co-extensive with the sort expressions in [26]. In fact, the set $[\![u]\!]$ defined by any linear pattern is always a regular tree language, and any regular tree language is always describable by the linear pattern $x{:}s$, where s is a sort/state in an order-sorted signature/tree-automaton Σ. The upshot of these observations is that constrained narrowing provides an alternative form of tree-automata-based symbolic model checking in which: (i) patterns are free and linear; (ii) all constraints are the trivial constraint $\top$, and (iii) rewrite rules describing state transitions are unconditional and linear. Three advantages of the pattern-based approach are that: (i) it is more natural and direct to describe a set of states by patterns than by tree automata, (ii) the infinite set of linear patterns u on a given order-sorted signature Σ describe not just the finite set of regular tree languages defined by the sorts/states of Σ itself, but the *infinite class of languages* associated to the regular tree automata *implicitly defined* by the patterns u themselves, and (iii) by allowing non-linear patterns and rewrite rules and giving up on closure under negation, we still remain effectively closed under unions and intersections of sets of states as explained in Section 5, and the entire approach is naturally extended beyond tree-automata-based model checking.

Last, but not least, the work on symbolic model checking most closely related to the ideas in Sections 6.2–6.3 is the work on narrowing-based LTL model checking of rewrite theories [38,9,10] and on the Maude-NPA [35]. In all these works the rewrite theory $\mathcal{R}$ is assumed to be *unconditional* and its equational theory $(\Sigma, E \cup B)$ is assumed to be FVP, so that there is a finitary $E \cup B$-unification algorithm that can be used to perform narrowing steps. In relation to all that work, this work's new contribution is to make narrowing-based symbolic model checking much more widely available by: (i) drastically expanding the class of, now conditional, rewrite theories to which it can be applied thanks to the $\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ theory transformation; (ii) making the symbolic language for describing sets of states substantially more expressive by allowing constrained pattern predicates as opposed to just patterns, and (iii) making such symbolic model checking possible even when satisfiability of constraints is undecidable. One important difference is that, for the moment, the support for LTL and LTLR symbolic model checking provided in [38,9,10] has not yet been extended to the constrained narrowing setting. This is obviously important future work.

Relationship to CLP and Theorem Proving. We can naturally understand a CLP program $\mathcal{P}$ (see the survey [54] on CLP and its bibliography up to 1994) as a generalized rewrite theory whose set P of rules contains rules[20] of the form:

$$p(\overline{u}) \rightarrow p_1(\overline{u_1}), \ldots, p_n(\overline{u_n}) \; \textit{if} \; \varphi$$

where: (i) when $n = 0$, the rule's righthand side is the constant *true* and then the rule is called a *fact*, (ii) $\mathcal{P}$ has signature $\Sigma = \Omega \uplus \Sigma_0$ and the terms $p(\overline{u}), p_1(\overline{u_1}), \ldots, p_n(\overline{u_n})$ are Ω-terms of a privileged sort *Pred* having a constant *true* and a binary associative-commutative symbol $\_,\_$ with identity *true*, (iii) φ is a Σ_0-formula[21] belonging to a conjunction-closed class $\mathcal{C}$ of Σ_0-formulas whose satisfiability in $T_{\Sigma/E \uplus B|\Sigma_0}$ is decidable (where $E \uplus B$ are the equations in $\mathcal{P}$), and (iv) for any Ω-substitution θ, if $\varphi \in \mathcal{C}$ then $\varphi\theta \in \mathcal{C}$.

A *query* for $\mathcal{P}$ is a constrained Ω-term $q_1(\overline{v_1}), \ldots, q_k(\overline{v_k}) \mid \psi$ with $\psi \in \mathcal{C}$ and its *solutions* are the *constrained substitutions* $\theta \mid \phi$ such that ϕ is satisfiable in $T_{\Sigma/E \uplus B|\Sigma_0}$ and there is a constrained narrowing sequence:

$$q_1(\overline{v_1}), \ldots, q_k(\overline{v_k}) \mid \psi \overset{\theta\,*}{\leadsto}_{P, B_\Omega} \textit{true} \mid \phi.$$

That is, solving the query $q_1(\overline{v_1}), \ldots, q_k(\overline{v_k}) \mid \psi$ exactly corresponds to solving the existential reachability problem:

$$\mathcal{P} \models \exists (q_1(\overline{v_1}), \ldots, q_k(\overline{v_k}) \mid \psi \rightarrow^* \textit{true} \mid \top).$$

[20] Logically, rules are *constrained Horn clauses* $p(\overline{u}) \Leftarrow p_1(\overline{u_1}) \wedge \ldots \wedge p_n(\overline{u_n}) \mid \varphi$. Constrained narrowing then coincides with *constrained resolution* deduction.
[21] Recall Footnote 5, explaining that a rule's condition φ need not be QF in a broader understanding of generalized rewrite rule.

The above three-thousand-feet-high description of CLP hides a multitude of useful engineering details. For example, the accumulated constraint along a narrowing path, say φ, is called the *store* and is split into an "active part" φ_1 and a "passive" one φ_2. Also, various state space reduction techniques are used to detect *failures*, i.e., symbolic states which will never reach the goal $true \mid \top$.

The perhaps not so obvious point about this brief comparison with CLP is that it opens a two-way street between CLP and generalized rewrite theories. On the one hand, useful CLP techniques can be naturally adapted to make constrained narrowing more efficient; on the other hand, various extensions of CLP can be developed such as, for example, (i) reasoning modulo axioms B; (ii) order-sorted typing and subtype polymorphism; and (iii) combinations of CLP with equational-style functional programming in the extended setting (i)–(ii).

In the theorem proving area, the work by C. Kirchner, H. Kirchner, and M. Rusinowitch on equational theorem proving with constraints [59] extended superposition theorem proving with notions such as constrained formulas, formula simplification with constrained equations oriented as rewrite rules, and constrained superposition (of which the constrained Horn clause resolution used in CLP is a special case). These notions have influenced subsequent work on superposition theorem proving with constraints, e.g., [47,2]; and in the combination of symbolic constraint solvers using constrained rewriting proposed by H. Kirchner and C. Ringeissen [58].

The constrained rewriting work in [57,41–43,40,60], has had as one of its main goals the application of (universal) constrained rewriting to *inductive theorem proving*, in two closely related ways. On the one hand, constrained rewriting becomes part of an inference system to prove theorems in initial algebras; on the other hand, constrained rewrite rules are used to give semantic definitions for an imperative programming language, and to then obtain a programming-language-specific theorem prover for such a language. The work in [97,96,61] can be seen as yet another application of constrained rewriting to the verification of imperative programs.

The work on verification of *inductive* invariants in Section 6.3 is closely related to the work on deductive verification of safety properties of rewrite theories in [83,85,84]. They both share the idea of using one-step narrowing to verify inductive invariants, but Section 6.3 revisits this topic in the new setting of constrained narrowing and state properties specified by means of pattern predicates. Furthermore, it deals with both variants and co-invariants, which can help the verification effort in practice, since sometimes expressing a coinvariant as a pattern predicate may be easier than expressing its complement invariant set.

Another body of inductive theorem proving work, perhaps the most closely related to the present one, is that on *reachability logic* [92,93,64,91,78,63,23]. In equational inductive theorem proving one reasons about validity of formulas in an initial algebra $T_{\Sigma/E \uplus B}$, whereas in reachability logic one reasons about validity of reachability formulas in the initial model $\mathcal{T}_{\mathcal{R}}$ of a generalized rewrite theory $\mathcal{R}$. Although originally developed for purposes of verifying properties of programs in a programming language from their rewriting logic semantics [92,93], it has later been extended to verify reachability properties in general rewrite theories [64,91,63,23]. This subsequent development is the one most closely related to the present work. In particular, the ideas in [91] are the most closely related to this work since: (i) the language of constrained pattern predicates based on constructors was first identified in [91] as a suitable language of state predicates on which to base reachability logic formulas so that large portions of the logic can be mechanized; and (ii) constrained narrowing is implicitly used as the key symbolic technique in the $\textsc{Step}^{\forall} + \textsc{Subsumption}$ inference rule in [91].

Conclusions. I have presented a new notion of *generalized rewrite theory* that unifies various previous approaches to constrained rewriting. Since such theories are very general and combine both equations and rules, the issue of their symbolic executability is nontrivial. The first main effort has been to find suitable *theory transformations* that can bring a class of generalized rewrite theories as wide as possible into the executable fold. The theory transformation $\mathcal{R} \mapsto \overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}$ is the culmination of this effort. A second main effort has been to develop symbolic techniques for reasoning about generalized rewrite theories. To begin with, an expressive language of state predicates based on constructor pattern predicates has been studied. Then, symbolic techniques such as universal constrained rewriting, sound and complete solution of existential reachability problems by constrained narrowing, and applications to invariant verification have been studied. As the discussion on related work makes clear, these techniques live within, and can contribute to, a larger ecosystem of *symbolic verification* techniques, including both symbolic model checking and theorem proving. In fact, the borderline between these two areas becomes more and more tenuous as time goes on, and what this work explicitly proposes is their synergistic combination.

This is of course easier said than done. Much work remains ahead, both in implementing these ideas, experimenting with them, and engineering mature tools that can verify challenging case studies. I am looking forward to tackling these challenges and hope that this work will stimulate the interest of other researchers, so that we can collectively bring these ideas to fruition.

Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgements

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.

Appendix A. Proofs of theorems

Proof of Lemma 1.

Proof. First of all, since for any QF formula φ and substitutions θ, γ we have the implication $(T \models \varphi\theta) \Rightarrow (T \models \varphi\theta\gamma)$, it is easy to show, using the definition of $u \rightarrow_{\mathcal{R}} v$, that both $\mathcal{T}_{\mathcal{R}}$ and $\mathcal{T}_{\mathcal{R}}(X)$ are objects of $\mathbf{Trans}_{\mathcal{R}}$. Consider now any $(A, \rightarrow_A) \in \mathbf{Trans}_{\mathcal{R}}$. Uniqueness of a homomorphism $h : \mathcal{T}_{\mathcal{R}} \rightarrow (A, \rightarrow_A)$ in $\mathbf{Trans}_{\mathcal{R}}$ is assured, since, if it exists, it must be the unique Σ-homomorphism $h : T_{\Sigma/G} \rightarrow A$. To show existence, note that any $\alpha \in [X \rightarrow A]$ extends to a homomorphism $\_\alpha : \mathcal{T}_{\mathcal{R}}(X) \rightarrow (A, \rightarrow_A)$ in $\mathbf{Trans}_{\mathcal{R}}$. Furthermore, since transitions in $\mathcal{T}_{\mathcal{R}}$ are just the special ground case of transitions in $\mathcal{T}_{\mathcal{R}}(X)$, the unique Σ-homomorphism $T_{\Sigma/G} \hookrightarrow T_{\Sigma/G}(X)$ also extends to a homomorphism $\mathcal{T}_{\mathcal{R}} \hookrightarrow \mathcal{T}_{\mathcal{R}}(X)$. But then, our desired homomorphism is just the composition:

$$\mathcal{T}_{\mathcal{R}} \hookrightarrow \mathcal{T}_{\mathcal{R}}(X) \xrightarrow{\_\alpha} (A, \rightarrow_A). \quad \square$$

Proof of Theorem 4.

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

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

But since we always have $u \rightarrow_{R/B} v \Rightarrow u \rightarrow_{\mathcal{R}} v$, we get $(\ddagger) \Rightarrow (\dagger)$. So we only need to prove $(\dagger) \Rightarrow (\ddagger)$. But, by definition, $u \rightarrow_{\mathcal{R}} v$ iff there exists a term $u' \in T_\Sigma(X)$, an *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) $u =_{E \uplus B} u' = u'[l\theta]_p$; and (iii) $u'[r\theta]_p =_{E \uplus B} v$. Letting $w = u'[r\theta]_p$ this means that $u =_{E \uplus B} u'$, $u' \rightarrow_{R/B} w$, and $w =_{E \uplus B} v$, which by (†) implies that there is a $w' \in T_\Sigma(X)$ such that $u'!_{\vec{E},B} \rightarrow_{R/B} w'$ 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 $u!_{\vec{E},B} \rightarrow_{R/B} w'$ 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 $(\ddagger)$. $\square$

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. $\square$

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 $u \to_{\mathcal{R}} v$ with rule $l \to r$ *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 $u \to_{\mathcal{R}_{\Sigma_1}} v$ with rule $l \to r'$ *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 $u \to_{\mathcal{R}_{\Sigma_1}} v$ with rule $l \to r'$ *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 $u \to_{\mathcal{R}} v$. □

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 $\to_{\mathcal{R}}|_{T^2_\Sigma} = \to_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}|_{T^2_\Sigma}$. But any rewrite step (ground or not) $u \to_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}|_{T^2_\Sigma} v$, say with rule $l' \to r'$ *if* $(\varphi\gamma)!_{\vec{E},B}$ where $(l \to r \text{ if } \varphi) \in R$ 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 $u \to_{\mathcal{R}} v$ with substitution $\gamma\theta$. Therefore, $\to_{\mathcal{R}}|_{T^2_\Sigma} \supseteq \to_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}|_{T^2_\Sigma}$. Two show $\to_{\mathcal{R}}|_{T^2_\Sigma} \subseteq \to_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}|_{T^2_\Sigma}$, assume $u \to_{\mathcal{R}} v$ with u, v ground terms. Since $\mathcal{R}$ is topmost, this means that we have a substitution θ and a rule $l \to r$ *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 are 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 $u \to_{\mathcal{R}} v$ 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 $u \to_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}|_{T^2_\Sigma} v$ with rule $l' \to r'$ *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 $\to_{\mathcal{R}}|_{T^2_\Sigma} \subseteq \to_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}|_{T^2_\Sigma}$ and (even more closely) the proof of ground coherence of $\overline{\mathcal{R}}_l$ in Theorem 5. It is left to the reader. □

Proof of Proposition 1.

Proof. Let $[u], [v] \in C_{\Sigma/\vec{E},B}$ be such that $[u] \to_{\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}} [v]$ holds. Because of the isomorphism $\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}} \cong \mathcal{T}_{\mathcal{R}}$ this means that there is a substitution μ and a rule $l \to r$ *if* φ in R such that: (i) $u =_B (l\mu)!_{\vec{E},B}$, (ii) $T \models \varphi\mu$, and (iii) $v =_B (r\mu)!_{\vec{E},B}$. Furthermore, reasoning as in the proof of Theorem 7, we may assume without loss of generality that μ is a $\vec{E}_\Omega, B_\Omega$-normalized constructor *ground* substitution. Therefore, by Definition 3 and the ground assumption, we have $u =_{B_\Omega} (l\mu)!_{\vec{E}_1,B_1}$ and $v =_{B_\Omega} (r\mu)!_{\vec{E}_1,B_1}$. But this means that $(\langle u, v\rangle, \mu)$ is a ground constructor variant of $\langle l, r\rangle$. Therefore, there is a constructor variant $(\langle l', r'\rangle, \gamma) \in [\![\langle l, r\rangle]\!]^{\Omega}_{\vec{E}_1,B_1}$ and a $\vec{E}_\Omega, B_\Omega$-normalized constructor ground substitution δ such that $\langle l', r'\rangle\delta =_{B_1} \langle u, v\rangle$ and $(\gamma\delta)|_{vars(\langle l,r\rangle)} =_{B_1} \mu|_{vars(\langle l,r\rangle)}$. Let $Z_0 = vars(\varphi) \setminus vars(\langle l, r\rangle)$. We can choose δ so that $\delta|_{Z_0} = \mu|_{Z_0}$ and $\mu =_{B_1} \gamma\delta$. Therefore, $T \models \varphi\mu$ iff $T \models \varphi\gamma\delta$ iff $T \models (\varphi\gamma)!_{\vec{E},B}\delta$. But, by Definition 3 and the $\vec{E}_\Omega, B_\Omega$-normalized constructor ground substitution assumption on δ, we also have $u =_{B_\Omega} (l\mu)!_{\vec{E}_1,B_1} =_{B_\Omega} l'\delta$ and $v =_{B_\Omega} (r\mu)!_{\vec{E}_1,B_1} =_{B_\Omega} r'\delta$. Therefore, using the rule $l' \to r'$ *if* $(\varphi\gamma)!_{\vec{E},B}$ in $R^{\Omega}_{\Sigma_1,l,r}$ we obtain a rewrite step $u \to_{R^{\Omega}_{\Sigma_1,l,r},B_\Omega} r'\delta$, with $v =_{B_\Omega} r'\delta$, as desired. □

Proof of Proposition 2.

Proof. Since for each $1 \leq l \leq n$ and each such α we have $[\![(u_{k_1} \mid \bigwedge_{1 \leq l \leq n} \varphi_{k_l})\alpha]\!] \subseteq [\![(u_{k_l} \mid \varphi_{k_l})\alpha]\!] = [\![(u_{k_l} \mid \varphi_{k_l})\alpha]\!] \subseteq [\![u_{k_l} \mid \varphi_{k_l}]\!]$, the $\supseteq$ containment is obvious. To see that the $\subseteq$ containment also holds, let $[v] \in \bigcap_{1 \leq l \leq n} [\![u_{k_l} \mid \varphi_{k_l}]\!]$. By the variable disjointness assumption this means that there are disjoint ground substitutions ρ_l, $1 \leq l \leq n$, such that $[v] = [u_{k_l}\rho_l] \in [\![u_{k_l} \mid \varphi_{k_l}]\!]$, $1 \leq l \leq n$. But this means that the substitution $\rho = \rho_1 \uplus \ldots \uplus \rho_n$ B_Ω-unifies the set $\{u_{k_1}, \ldots, u_{k_n}\}$, so there is a $\beta \in Unif_{B_\Omega}(\{u_{k_1}, \ldots, u_{k_n}\})$ and a ground substitution γ such that $\rho =_{B_\Omega} \beta\gamma$, and, furthermore, $T_{\Sigma/E \cup B} \models (\bigwedge_{1 \leq l \leq n} \varphi_{k_l})\beta\gamma$. Therefore, $[v] \in [\![(u_{k_1} \mid \bigwedge_{1 \leq l \leq n} \varphi_{k_l})\beta]\!] \subseteq \bigcup_{\alpha \in Unif_{B_\Omega}(\{u_{k_1}, \ldots, u_{k_n}\})} [\![(u_{k_1} \mid \bigwedge_{1 \leq l \leq n} \varphi_{k_l})\alpha]\!]$, as desired. □

Proof of Theorem 8.

Proof. By the definition of $\to_{\mathcal{C}_{\mathcal{R}}}$ and assumption (ii), (2) and (3) are equivalent. Note that (1) $\Rightarrow$ (2) because, by assumption (iii), we have an isomorphism $\gamma : \mathcal{C}_{\mathcal{R}}(X) \cong \mathcal{T}_{\mathcal{R}}(X)$, and therefore $\mathcal{C}_{\mathcal{R}}(X) \in \mathbf{Trans}_{\mathcal{R}}$. But this means that, assuming $X \supset Y$, for the map $\iota \in [X \to C_{\Sigma/\vec{E},B}(X)]$ such that $(\forall x \in X)\, \iota(x) = [x]_B$, we have $t\iota = [t!_{\vec{E},B}]_B$, and $t'\iota = [t'!_{\vec{E},B}]_B$. But since $\mathcal{C}_{\mathcal{R}}(X) \models (\forall Y)\, t \to^* t'$, this then forces (2), as desired. The proof that (2) $\Rightarrow$ (1) is an easy induction on the length of the sequence

$[t!_{\vec{E},B}]_B \rightarrow^*_{\mathcal{C}_{\mathcal{R}}(X)} [t'!_{\vec{E},B}]_B$ using the fact that for each $(A, \rightarrow_A) \in \mathbf{Trans}_{\mathcal{R}}$ and $\alpha \in [X \rightarrow A]$ we have a Σ-homomorphism of algebraic transition systems:

$$\mathcal{C}_{\mathcal{R}}(X) \overset{\gamma}{\cong} \mathcal{T}_{\mathcal{R}}(X) \xrightarrow{\alpha} (A, \rightarrow_A),$$

and γ gives us the equivalence $[u!_{\vec{E},B}]_B \rightarrow_{\mathcal{C}_{\mathcal{R}}} [v!_{\vec{E},B}]_B \Leftrightarrow [u]_{E \uplus B} \rightarrow_{\mathcal{R}} [v]_{E \uplus B}$. □

Proof of Corollary 1.

Proof. By the equivalence (1) ⇔ (3) in Theorem 8 we will be done of we show that if a term w exists such that $t!_{\vec{E},B} \rightarrow^*_{R,B;!_{\vec{E},B}} w$ and $w =_B t'!_{\vec{E},B}$, then such a term can be effectively found after a finite number of steps. But if such a w exists, there will be a smallest length n possible for a rewrite sequence $t!_{\vec{E},B} \rightarrow^*_{R,B;!_{\vec{E},B}} w$ such that $w =_B t'!_{\vec{E},B}$. Therefore, since, by assumptions (iv)–(vii), there is only a finite number of possible rules and associated B-matching substitutions allowing a rewrite step with $\rightarrow_{R,B}$, and such substitutions can be computed in finite time; and by assumption (vi) whether any of those B-matching substitutions satisfies or not the corresponding rule's condition can also be determined in finite time, we can effectively compute a finitely branching search tree with $t!_{\vec{E},B}$ as its root node and edges corresponding to $\rightarrow_{R,B;!_{\vec{E},B}}$ rewrite steps, so that our desired term w will appear at depth n in such a tree. □

Proof of Lemma 3.

Proof. To see the $\subseteq$ containment, let $[w] \in R!([\![v \mid \psi]\!])$. This means that there is a $[v\rho] \in [\![v \mid \psi]\!]$ such that $[v\rho] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [w]$. But since $\rightarrow_{\mathcal{C}_{\mathcal{R}}} = \rightarrow_{\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}}$, by Proposition 1 this just means that there is a rule $\gamma : l \rightarrow r \text{ if } \varphi$ in $R^{\Omega}_{\Sigma_1,l,r}$ and a ground constructor substitution δ such that $[v\rho] = [l\delta]$, $[w] = [r\delta]$, and $T \models \varphi\delta$, which, by condition (iv)-(c) in Definition 6, is equivalent to $T_{\Sigma/E \uplus B} \models \varphi\delta$. But, by the variable disjointness assumption, ρ and δ are disjoint and $\rho \uplus \delta$ B_{Ω}-unify $v = l$. Therefore, there is a B_{Ω}-unifier $\alpha \in Unif_{B_{\Omega}}(l = v)$ and a ground constructor substitution η such that $\rho \uplus \delta =_{B_{\Omega}} \alpha\eta$. But this means that $[w] \in [\![(r \mid \psi \wedge \varphi)\alpha]\!] \subseteq [\![R^{\circ}!([\![v \mid \psi]\!])]\!]$, as desired. Let us now prove the $\supseteq$ containment. Suppose $[w] \in [\![R^{\circ}!([\![v \mid \psi]\!])]\!]$. This means that there is a rule $\gamma : l \rightarrow r \text{ if } \varphi$ in $R^{\Omega}_{\Sigma_1,l,r}$ and a B_{Ω}-unifier $\alpha \in Unif_{B_{\Omega}}(l = v)$ such that $[w] = [r\alpha\rho] \in [\![(r \mid \psi \wedge \varphi)\alpha]\!]$. Let $Y = vars(l\alpha\rho)$ and choose any ground substitution $\tau \in [Y \rightarrow T_{\Omega}]$. Then, since $(\psi \wedge \varphi)\alpha(\tau \uplus \rho) = (\psi \wedge \varphi)\alpha\rho$, $T_{\Sigma/E \uplus B} \models (\psi \wedge \varphi)\alpha\rho$, which by condition (iv)-(c) in Definition 6 is equivalent to $T \models (\psi \wedge \varphi)\alpha\rho$, $[v\alpha(\tau \uplus \rho)] = [l\alpha(\tau \uplus \rho)]$, and $[w] = [r\alpha\rho] = [r\alpha(\tau \uplus \rho)]$ we have $[v\alpha(\tau \uplus \rho)] \in [\![v \mid \psi]\!]$ such that $[v\alpha(\tau \uplus \rho)] \rightarrow_{\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}} [w]$, and therefore $[w] \in R!([\![v \mid \psi]\!])$, as desired. □

Proof of Lemma 4.

Proof. Follows immediately from the definition (‡) of $R^{\circ}!$ and the proof of Lemma 3. Specifically, the proof of Soundness is a rephrasing of the proof of the $\subseteq$ containment in Lemma 3, and the proof of Completeness rephrases the proof of the $\supseteq$ containment in Lemma 3. □

Proof of Theorem 9.

Proof. $\mathcal{T}_{\mathcal{R}} \models \exists(A \rightarrow^* B)$ holds iff there is an n, $n \geq 0$, such that $[\![R!^n([A]) \wedge [B]]\!] \neq \emptyset$. But, by the definition of $R^{\circ}!$ in (‡), this holds iff there is an i and a narrowing sequence $u_i \mid \varphi_i \rightsquigarrow^*_{R^{\Omega}_{\Sigma_1,l,r},B_{\Omega}} v \mid \psi$ such that $[\![v \mid \psi \wedge B]\!] \neq \emptyset$. Furthermore, by repeated application of the Soundness part of the Lifting Lemma 4, which uses the constructive method described in the proof of Lemma 3, given $[w'] \in [\![v \mid \psi \wedge B]\!]$ we can effective find a $[w] \in [\![u_i \mid \varphi_i]\!]$ such that $[w] \rightarrow^*_{\mathcal{C}_{\overline{\mathcal{R}}^{\Omega}_{\Sigma_1,l,r}}} [w']$. □

References

[1] P. Abdulla, B. Jonsson, P. Mahata, J. d'Orso, Regular tree model checking, in: Computer Aided Verification, Springer, 2002, pp. 452–466.
[2] E. Althaus, E. Kruglov, C. Weidenbach, Superposition modulo linear arithmetic SUP(LA), in: S. Ghilardi, R. Sebastiani (Eds.), FroCos, in: Lecture Notes in Computer Science, vol. 5749, Springer, 2009, pp. 84–99.
[3] R. Alur, C. Courcoubetis, T.A. Henzinger, P.H. Ho, Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems, in: R. Grossman, A. Nerode, A. Ravn, H. Rischel (Eds.), Workshop on Theory of Hybrid Systems, in: LNCS, vol. 739, Springer, 1993, pp. 209–229.
[4] R. Alur, D.L. Dill, A theory of timed automata, Theor. Comput. Sci. 126 (2) (1994) 183–235.
[5] R. Alur, P. Madhusudan, Adding nesting structure to words, J. ACM 56 (3) (2009).
[6] A. Armando, J. Mantovani, L. Platania, Bounded model checking of software using SMT solvers instead of SAT solvers, in: A. Valmari (Ed.), Model Checking Software, Proceedings of the 13th International SPIN Workshop, Vienna, Austria, March 30 - April 1, 2006, in: Lecture Notes in Computer Science, vol. 3925, Springer, 2006, pp. 146–162.
[7] A. Arusoaie, D. Lucanu, V. Rusu, Symbolic execution based on language transformation, Comput. Lang. Syst. Struct. 44 (2015) 48–71.
[8] M. Ayala-Rincón, Expressiveness of Conditional Equational Systems with Built-in Predicates, Ph.D. thesis, Universität Kaiserslauten, 1993.

[9] K. Bae, S. Escobar, J. Meseguer, Abstract logical model checking of infinite-state systems using narrowing, in: Rewriting Techniques and Applications, RTA'13, in: LIPIcs, vol. 21, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2013, pp. 81–96.
[10] K. Bae, J. Meseguer, Infinite-state model checking of LTLR formulas using narrowing, in: Proc. WRLA 2014, in: LNCS, vol. 8663, Springer, 2014, pp. 113–129.
[11] K. Bae, C. Rocha, Guarded terms for rewriting modulo SMT, in: Formal Aspects of Component Software – Proceedings of the 14th International Conference, FACS 2017, Braga, Portugal, October 10-13, 2017, 2017, pp. 78–97.
[12] J. Bergstra, J. Tucker, Characterization of computable data types by means of a finite equational specification method, in: J.W. de Bakker, J. van Leeuwen (Eds.), Automata, Languages and Programming, Seventh Colloquium, in: LNCS, vol. 81, Springer-Verlag, 1980, pp. 76–90.
[13] D. Beyer, M.E. Keremoglu, Cpachecker: a tool for configurable software verification, in: Proc CAV 2011, in: LNCS, vol. 6806, Springer, 2011, pp. 184–190.
[14] A. Bouajjani, Languages, rewriting systems, and verification of infinite-state systems, Autom. Lang. Program. (2001) 24–39.
[15] A. Bouajjani, J. Esparza, Rewriting models of boolean programs, in: Term Rewriting and Applications, 2006, pp. 136–150.
[16] A. Bouajjani, B. Jonsson, M. Nilsson, T. Touili, Regular model checking, in: Computer Aided Verification, Springer, 2000, pp. 403–418.
[17] A. Bouhoula, M. Rusinowitch, Implicit induction in conditional theories, J. Autom. Reason. 14 (1995) 189–235.
[18] M. Bozzano, R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, S. Tonetta, nuXmv 1.0 User Manual, Tech. Rep., FBK, 2014.
[19] R. Bruni, J. Meseguer, Semantic foundations for generalized rewrite theories, Theor. Comput. Sci. 360 (1–3) (2006) 386–414.
[20] A. Cholewa, S. Escobar, J. Meseguer, Constrained narrowing for conditional equational theories modulo axioms, Sci. Comput. Program. 112 (2015) 24–57.
[21] A. Cimatti, A. Griggio, S. Mover, S. Tonetta, HyComp: an SMT-based model checker for hybrid systems, in: Proc. TACAS 2015, in: LNCS, vol. 9035, Springer, 2015, pp. 52–67.
[22] Ștefan Ciobâcă, A. Arusoaie, D. Lucanu, Unification modulo builtins, in: Logic, Language, Information, and Computation – Proceedings of the 25th International Workshop, WoLLIC 2018, Bogota, Colombia, July 24-27, 2018, in: LNCS, vol. 10944, Springer, 2018, pp. 179–195.
[23] Ștefan Ciobâcă, D. Lucanu, A coinductive approach to proving reachability properties in logically constrained term rewriting systems, in: Proc. IJCAR 2018, in: Lecture Notes in Computer Science, vol. 10900, Springer, 2018, pp. 295–311.
[24] M. Clavel, F. Durán, S. Eker, J. Meseguer, P. Lincoln, N. Martí-Oliet, C. Talcott, All About Maude – A High-Performance Logical Framework, LNCS, vol. 4350, Springer, 2007.
[25] H. Comon, Equational formulas in order-sorted algebras, in: Proc. ICALP'90, in: LNCS, vol. 443, Springer, 1990, pp. 674–688.
[26] H. Comon, C. Delor, Equational formulae with membership constraints, Inf. Comput. 112 (2) (1994) 167–216.
[27] H. Comon, R. Nieuwenhuis, Induction=i-axiomatization+first-order consistency, Inf. Comput. 159 (1–2) (2000) 151–186.
[28] H. Comon-Lundth, S. Delaune, The finite variant property: how to get rid of some algebraic properties, in: Proc RTA'05, in: LNCS, vol. 3467, Springer, 2005, pp. 294–307.
[29] L. Cordeiro, B. Fischer, J. Marques-Silva, SMT-based bounded model checking for embedded ANSI-c software, in: ASE, IEEE, 2009, pp. 137–148.
[30] G. Delzanno, A. Podelski, Constraint-based deductive model checking, Int. J. Softw. Tools Technol. Transf. 3 (3) (2001) 250–270.
[31] N. Dershowitz, J.P. Jouannaud, Rewrite systems, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Vol. B, North-Holland, 1990, pp. 243–320.
[32] E. Dijkstra, C. Scholten, Petri Nets, Springer-Verlag, 1990.
[33] F. Durán, J. Meseguer, On the Church-Rosser and coherence properties of conditional order-sorted rewrite theories, J. Log. Algebraic Program. 81 (2012) 816–850.
[34] H. Ehrig, B. Mahr, Fundamentals of Algebraic Specification 1, Springer, 1985.
[35] S. Escobar, C. Meadows, J. Meseguer, Maude-NPA: cryptographic protocol analysis modulo equational properties, in: Foundations of Security Analysis and Design V, FOSAD 2007/2008/2009 Tutorial Lectures, in: LNCS, vol. 5705, Springer, 2009, pp. 1–50.
[36] S. Escobar, C. Meadows, J. Meseguer, S. Santiago, State space reduction in the Maude-NRL protocol analyzer, Inf. Comput. 238 (2014) 157–186.
[37] S. Escobar, C. Meadows, J. Meseguer, S. Santiago, Symbolic protocol analysis with disequality constraints modulo equational theories, in: Programming Languages with Applications to Biology and Security – Essays in Honour of Pierpaolo Degano on the Occasion of His 65th Birthday, in: LNCS, vol. 9465, Springer, 2015, pp. 238–261.
[38] S. Escobar, J. Meseguer, Symbolic model checking of infinite-state systems using narrowing, in: Proc. RTA, in: Lecture Notes in Computer Science, vol. 4533, 2007, pp. 153–168.
[39] S. Escobar, R. Sasse, J. Meseguer, Folding variant narrowing and optimal variant termination, J. Log. Algebraic Program. 81 (2012) 898–928.
[40] S. Falke, Term Rewriting with Built-In Numbers and Collection Data Structures, Ph.D. thesis, University of New, Mexico, 2009.
[41] S. Falke, D. Kapur, Dependency pairs for rewriting with built-in numbers and semantic data structures, in: 19th International Conference on Rewriting Techniques and Applications, in: Lecture Notes in Computer Science, vol. 5117, Springer, Berlin, Heidelberg, 2008, pp. 94–109.
[42] S. Falke, D. Kapur, Operational termination of conditional rewriting with built-in numbers and semantic data structures, Electron. Notes Theor. Comput. Sci. 237 (2009) 75–90.
[43] S. Falke, D. Kapur, Rewriting induction + linear arithmetic = decision procedure, in: Proc. IJCAR 2012, in: LNCS, vol. 7364, Springer, 2012, pp. 241–255.
[44] K. Futatsugi, Fostering proof scores in CafeOBJ, in: Proc. ICFEM 2010, in: LNCS, vol. 6447, Springer, 2010, pp. 1–20.
[45] M. Ganai, A. Gupta, Accelerating high-level bounded model checking, in: 2006 IEEE/ACM International Conference on Computer Aided Design, Nov 2006, pp. 794–801.
[46] M. Ganai, A. Gupta, Completeness in SMT-based BMC for software programs, in: DATE, IEEE, 2008, pp. 831–836.
[47] H. Ganzinger, R. Nieuwenhuis, Constraints and theorem proving, in: Constraints in Computational Logics: Theory and Applications, International Summer School, CCL'99, Gif-sur-Yvette, France, September 5-8, 1999, Revised Lectures, in: Lecture Notes in Computer Science, vol. 2002, Springer, 1999, pp. 159–201.
[48] T. Genet, V. Tong, Reachability analysis of term rewriting systems with Timbuk, in: Logic for Programming, Artificial Intelligence, and Reasoning, Springer, 2001, pp. 695–706.
[49] S. Ghilardi, E. Nicolini, S. Ranise, D. Zucchelli, Towards SMT model checking of array-based systems, in: Automated Reasoning, Proceedings of the 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, in: Lecture Notes in Computer Science, vol. 5195, Springer, 2008, pp. 67–82.
[50] S. Ghilardi, S. Ranise, MCMT: a model checker modulo theories, in: Proc. Automated Reasoning, 5th International Joint Conference, IJCAR 2010, in: Lecture Notes in Computer Science, vol. 6173, Springer, 2010, pp. 22–29.
[51] J. Goguen, J. Meseguer, Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations, Theor. Comput. Sci. 105 (1992) 217–273.
[52] A. Gurfinkel, T. Kahsai, A. Komuravelli, J.A. Navas, The seahorn verification framework, in: Proc. CAV 2015, Part I, in: LNCS, vol. 9206, Springer, 2015, pp. 343–361.
[53] R. Gutiérrez, J. Meseguer, C. Rocha, Order-sorted equality enrichments modulo axioms, Sci. Comput. Program. 99 (2015) 235–261.
[54] J. Jaffar, M.J. Maher, Constraint logic programming: a survey, J. Log. Program. 19/20 (1994) 503–581.
[55] J.P. Jouannaud, H. Kirchner, Completion of a set of rules modulo a set of equations, SIAM J. Comput. 15 (November 1986) 1155–1194.

[56] T. Kahsai, C. Tinelli, PKind: a parallel k-induction based model checker, in: Proc. PDMC 2011, in: EPTCS, vol. 72, 2011, pp. 55–62.
[57] H. Kirchner, C. Ringeissen, Combining symbolic constraint solvers on algebraic domains, J. Symb. Comput. 18 (2) (1994) 113–155.
[58] H. Kirchner, C. Ringeissen, Constraint solving by narrowing in combined algebraic domains, in: Logic Programming, Proceedings of the Eleventh International Conference on Logic Programming, MIT Press, 1994, pp. 617–631.
[59] K. Kirchner, H. Kirchner, M. Rusinowitch, Deduction with symbolic constraints, Rev. Intell. Artif. 4 (3) (1990) 9–52.
[60] C. Kop, N. Nishida, Term rewriting with logical constraints, in: 9th International Symposium on Frontiers of Combining Systems, in: Lecture Notes in Computer Science, vol. 8152, Springer, 2013, pp. 343–358.
[61] C. Kop, N. Nishida, Automatic constrained rewriting induction towards verifying procedural programs, in: J. Garrigue (Ed.), Programming Languages and Systems – Proceedings of the 12th Asian Symposium, APLAS 2014, Singapore, November 17-19, 2014, in: Lecture Notes in Computer Science, vol. 8858, Springer, 2014, pp. 334–353.
[62] G. Kreisel, J. Krivine, Mathematical Logic, North-Holland, 1967.
[63] D. Lucanu, V. Rusu, A. Arusoaie, A generic framework for symbolic execution: a coinductive approach, J. Symb. Comput. 80 (2017) 125–163.
[64] D. Lucanu, V. Rusu, A. Arusoaie, D. Nowak, 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, in: LNCS, vol. 9200, Springer, 2015, pp. 451–474.
[65] S. Lucas, J. Meseguer, Normal forms and normal theories in conditional rewriting, J. Log. Algebraic Methods Program. 85 (1) (2016) 67–97.
[66] S. MacLane, Categories for the Working Mathematician, Springer-Verlag, 1971.
[67] E. Manes, Algebraic Theories, Graduate Texts in Mathematics, vol. 26, Springer, 1976.
[68] J. Meseguer, P. Thati, Symbolic reachability analysis using narrowing and its application to the verification of cryptographic protocols, High.-Order Symb. Comput. 20 (1–2) (2007) 123–160.
[69] J. Meseguer, General logics, in: H.D.E., et al. (Eds.), Logic Colloquium '87, North-Holland, 1989, pp. 275–329.
[70] J. Meseguer, Membership algebra as a logical framework for equational specification, in: Proc. WADT'97, in: LNCS, vol. 1376, Springer, 1998, pp. 18–61.
[71] J. Meseguer, Strict coherence of conditional rewriting modulo axioms, Theor. Comput. Sci. 672 (2017) 1–35.
[72] J. Meseguer, Generalized rewrite theories and coherence completion, in: V. Rusu (Ed.), Proc. Rewriting Logic and Its Applications – 12th International Workshop, WRLA 2018, in: Lecture Notes in Computer Science, vol. 11152, Springer, 2018, pp. 164–183.
[73] J. Meseguer, Variant-based satisfiability in initial algebras, Sci. Comput. Program. 154 (2018) 3–41.
[74] J. Meseguer, J. Goguen, Initiality, induction and computability, in: M. Nivat, J. Reynolds (Eds.), Algebraic Methods in Semantics, Cambridge University Press, 1985, pp. 459–541.
[75] J. Meseguer, M. Palomino, N. Martí-Oliet, Equational abstractions, Theor. Comput. Sci. 403 (2–3) (2008) 239–264.
[76] J. Meseguer, S. Skeirik, Equational formulas and pattern operations in initial order-sorted algebras, Form. Asp. Comput. 29 (3) (2017) 423–452.
[77] A. Milicevic, H. Kugler, Model checking using SMT and theory of lists, in: NASA 3rd International Symposium on Formal Methods, in: Lecture Notes in Computer Science, vol. 6617, Springer, 2011, pp. 282–297.
[78] B. Moore, Coinductive Program Verification, Ph.D. thesis, University of Illinois at Urbana-Champaign, 2016, http://hdl.handle.net/2142/95372.
[79] L.M. de Moura, S. Owre, H. Rueß, J.M. Rushby, N. Shankar, M. Sorea, A. Tiwari, SAL 2, in: Proc. CAV 2004, in: LNCS, vol. 3114, Springer, 2004, pp. 496–500.
[80] H. Ohsaki, H. Seki, T. Takai, Recognizing boolean closed a-tree languages with membership conditional rewriting mechanism, in: Rewriting Techniques and Applications, Springer, 2003, pp. 483–498.
[81] G.E. Peterson, M.E. Stickel, Complete sets of reductions for some equational theories, J. Assoc. Comput. Mach. 28 (2) (1981) 233–264.
[82] A. Podelski, Model checking as constraint solving, in: Static Analysis, Proceedings of the 7th International Symposium, SAS 2000, Santa Barbara, CA, USA, June 29 - July 1, 2000, in: Lecture Notes in Computer Science, vol. 1824, Springer, 2000, pp. 22–37.
[83] C. Rocha, J. Meseguer, Proving safety properties of rewrite theories, in: Proc. CALCO 2011, in: LNCS, vol. 6859, Springer, 2011, pp. 314–328.
[84] C. Rocha, Symbolic Reachability Analysis for Rewrite Theories, Ph.D. thesis, University of Illinois at Urbana-Champaign, 2012.
[85] C. Rocha, J. Meseguer, Mechanical analysis of reliable communication in the alternating bit protocol using the Maude invariant analyzer tool, in: Specification, Algebra, and Software – Essays Dedicated to Kokichi Futatsugi, in: Lecture Notes in Computer Science, vol. 8373, Springer, 2014, pp. 603–629.
[86] C. Rocha, J. Meseguer, C.A. Muñoz, Rewriting modulo SMT and open system analysis, in: Proc. Rewriting Logic and Its Applications – 10th International Workshop, WRLA 2014, in: Lecture Notes in Computer Science, vol. 8663, Springer, 2014, pp. 247–262.
[87] C. Rocha, J. Meseguer, C.A. Muñoz, Rewriting modulo SMT and open system analysis, J. Log. Algebraic Methods Program. 86 (2017) 269–297.
[88] T. Rybina, A. Voronkov, A logical reconstruction of reachability, in: Perspectives of Systems Informatics, 5th International Andrei Ershov Memorial Conference, Revised Papers, PSI 2003, Akademgorodok, Novosibirsk, Russia, July 9-12, 2003, in: Lecture Notes in Computer Science, vol. 2890, Springer, 2003, pp. 222–237.
[89] J.R. Shoenfield, Degrees of Unsolvability, North-Holland, 1971.
[90] S. Skeirik, J. Meseguer, Metalevel algorithms for variant satisfiability, J. Log. Algebraic Methods Program. 96 (2018) 81–110.
[91] S. Skeirik, A. Stefanescu, J. Meseguer, A constructor-based reachability logic for rewrite theories, in: Proc. Logic-Based Program Synthesis and Transformation – 27th International Symposium, LOPSTR 2017, in: Lecture Notes in Computer Science, vol. 10855, Springer, 2017, pp. 201–217.
[92] A. Stefanescu, Ştefan Ciobâcă, R. Mereuta, B.M. Moore, T. Serbanuta, G. Rosu, All-path reachability logic, in: Proc. RTA-TLCA 2014, in: LNCS, vol. 8560, Springer, 2014, pp. 425–440.
[93] A. Stefanescu, D. Park, S. Yuwen, Y. Li, G. Rosu, Semantics-based program verifiers for all languages, in: Proc. OOPSLA 2016, ACM, 2016, pp. 74–91.
[94] A. Tiwari, Hybrids al relational abstracter, in: Proc. CAV 2012, in: LNCS, vol. 7358, Springer, 2012, pp. 725–731.
[95] M. Veanes, N. Bjørner, A. Raschke, An SMT approach to bounded reachability analysis of model programs, in: 28th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems, Springer, 2008, pp. 53–68.
[96] G. Vidal, Closed symbolic execution for verifying program termination, in: IEEE 12th International Working Conference on Source Code Analysis and Manipulation, Sept 2012, pp. 34–43.
[97] G. Vidal, Symbolic execution as a basis for termination analysis, Sci. Comput. Program. 102 (2015) 142–157, http://www.sciencedirect.com/science/article/pii/S0167642315000271.
[98] P. Viry, Equational rules for rewriting logic, Theor. Comput. Sci. 285 (2002) 487–517.
[99] D. Walter, S. Little, C. Myers, Bounded model checking of analog and mixed-signal circuits using an SMT solver, in: 5th International Symposium on Automated Technology for Verification and Analysis, Springer, Berlin, Heidelberg, 2007, pp. 66–81.