A Constructor-Based Reachability Logic for Rewrite Theories

Stephen Skeirik, Andrei Stefanescu and José Meseguer

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

Abstract. Reachability logic has been applied to $\mathbb{K}$ rewrite-rule-based language definitions as a *language-generic* logic of programs. It has been proved successful in verifying a wide range of sophisticated programs in conventional languages. Here we study how reachability logic can be made not just language-generic, but *rewrite-theory-generic* to make it available not just for conventional program verification, but also to verify rewriting-logic-based programs and distributed system designs. A theory-generic reachability logic is presented and proved sound for a wide class of rewrite theories. Particular attention is given to increasing the logic's automation by means of *constructor-based* semantic unification, matching, and satisfiability procedures. The relationships to Hoare logic and LTL are discussed, new methods for proving invariants of possibly never terminating distributed systems are developed, and experiments with a prototype implementation illustrating the new methods are presented.
Keywords: reachability and rewriting logics, program verification.

1 Introduction

The main applications of reachability logic to date have been as a *language-generic* logic of programs [41,46,47]. In these applications, a $\mathbb{K}$ specification of a language's operational semantics by means of rewrite rules is assumed as the language's "golden semantic standard," and then a correct-by-construction reachability logic for a language so defined is automatically obtained [47]. This method has been shown effective in proving a wide range of programs in real programming languages specified within the $\mathbb{K}$ Framework.

Although the foundations of reachability logic are very general [46,47], such general foundations do not provide a straightforward answer to the following non-trivial questions: (1) Could a reachability logic be developed to verify not just conventional programs, but also *distributed system designs and algorithms* formalized as *rewrite theories* in rewriting logic [32,34]? And (2) if so, what would be the most natural way to conceive such a *rewrite-theory-generic* logic? Since $\mathbb{K}$ specifications are a special type of rewrite theories [35], a satisfactory answer to questions (1)–(2) would move the verification game from the level of verifying *code* to that of verifying *both code and distributed system designs*. Since the cost of design errors can be several orders of magnitude higher than that of coding errors, questions (1) and (2) are of practical software engineering interest.

Although a first step towards a reachability logic for rewrite theories has been taken in [28], as explained in Section 7 and below, that first step still leaves several important questions open. The most burning one is: how can we prove *invariants* of a distributed system? Since invariants are the most basic safety properties, support for proving invariants is a *sine qua non* requirement. As explained below and in Section 4.1, if we apply the standard foundations of reachability logic —so that the logic's transition relation is instantiated to the given theory's rewrite relation— the whole enterprise collapses before what we call the *invariant paradox*: we cannot verify in this manner *any* invariants of a never-terminating system such as, for example, a mutual exclusion protocol.

A second, important open question is how to best take advantage of the wealth of equational reasoning techniques such as matching, unification, and narrowing modulo an equational theory (Σ, E), e.g., [44,26,3,4,24,25,17,48], as well as recent results on decidable satisfiability (and validity) of quantifier-free formulas in initial algebras, e.g., [30,9,10,2,11,36,20,21,18,1,31] to *automate* as much as possible reachability logic deduction. In this regard, the very general foundations of reachability logic —which assume any Σ-algebra $\mathcal{A}$ with a first-order-definable transition relation— provide no help at all for automation. As shown in this work and its prototype implementation, if we assume instead that the model in question is the *initial model* $\mathcal{T}_{\mathcal{R}}$ of a rewrite theory $\mathcal{R}$ satisfying reasonable assumptions, large parts of the verification effort can be automated.

A third important issue is *simplicity*. Reachability logic has eight inference rules [46,47]. Could a reachability logic for rewrite theories be simpler? The main goal of this work is to tackle head on these three open questions to provide a general reachability logic and a prototype tool suitable for reasoning about properties of *both* distributed systems and programs based on their rewriting logic semantics. What all this really means requires some further explanations.

Rewriting Logic in a Nutshell. A distributed system can be designed and modeled as a *rewrite theory* $\mathcal{R} = (\Sigma, E, R)$ [32,34] in the following way: (i) the distributed system's *states* are modeled as elements of the initial algebra $T_{\Sigma/E}$ associated to the equational theory (Σ, E) with function symbols Σ and equations E; and (ii) the system's *concurrent transitions* are modeled by rewrite rules R, which are applied *modulo* E. Let us consider the QLOCK [19] mutual exclusion protocol, explained in detail in Section 2 and used later as a running example. QLOCK allows an unbounded number of processes, which can be identified by numbers. Such processes can be in one of three states: "normal" (doing their own thing), "waiting" for a resource, and "critical," i.e., using the resource. Waiting processes enqueue their identifier at the end of a waiting queue (a list), and can become critical when their name appears at the head of the queue. A QLOCK state can be represented as a tuple $< n \mid w \mid c \mid q >$ where n, resp. w, resp. c, denotes the set of identifiers for normal, resp. waiting, resp. critical processes, and q is the waiting queue. QLOCK can be naturally modeled as a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ where Σ contains operators to build natural numbers, multisets of natural numbers, like n, w, and c, and lists of natural numbers like q, plus the above tupling operator. The equations E include axioms such as

the associativity-commutativity of multiset union, and the associativity of list concatenation, and identity axioms for $\varnothing$ and *nil*. QLOCK's behavior is specified by a set R of five rewrite rules. For example, the rule *w2c* below specifies how a waiting process i can pass from waiting to critical

$$w2c : < n \mid w\ i \mid c \mid i; q > \rightarrow < n \mid w \mid c\ i \mid i; q > .$$

Reachability Logic in a Nutshell. A reachability logic formula has the form $A \rightarrow^{\circledast} B$, where A and B are state predicates. Consider the easier to explain case where the formula has no parameters, i.e., $vars(A) \cap vars(B) = \varnothing$. We interpret such a formula in the initial model $\mathcal{T}_{\mathcal{R}}$ of a rewrite theory $\mathcal{R} = (\Sigma, E, R)$, whose states are E-equivalence classes $[u]$ of ground Σ-terms, and where a state transition $[u] \rightarrow_{\mathcal{R}} [v]$ holds iff $\mathcal{R} \vdash u \rightarrow v$ according to the rewriting logic inference system [32,34] (computation = deduction). As a first approximation, $A \rightarrow^{\circledast} B$ is a Hoare logic *partial correctness* assertion of the form[1] $\{A\}\mathcal{R}\{B\}$, but with the slight twist that B need not hold of a terminating state, but just *somewhere along the way*. To be fully precise, $A \rightarrow^{\circledast} B$ holds in $\mathcal{T}_{\mathcal{R}}$ iff for each state $[u_0]$ satisfying A and each terminating sequence $[u_0] \rightarrow_{\mathcal{R}} [u_1] \ldots \rightarrow_{\mathcal{R}} [u_{n-1}] \rightarrow_{\mathcal{R}} [u_n]$ there is a j, $0 \leqslant j \leqslant n$ such that $[u_j]$ satisfies B. A key question is how to choose a good language of state predicates like A and B. Here is where the potential for increasing the logic's automation resides. We call our proposed logic *constructor-based*, because our choice is to make A and B positive (only $\vee$ and $\wedge$) combinations of what we call *constructor patterns* of the form $u \mid \varphi$, where u is a *constructor* term[2] and φ a quantifier-free (QF) Σ-formula. The state predicate $u \mid \varphi$ holds for a state $[u']$ iff there is a ground substitution ρ such that $[u'] = [u\rho]$ and $E \models \varphi\rho$. This is crucially important, because the initial algebra of constructor terms is typically *much simpler* than $T_{\Sigma/E}$, and this can be systematically exploited for matching, unification, narrowing, and satisfiability purposes to automate large portions of reachability logic.

The Invariant Paradox. This is all very well, but how can we *prove invariants* in such a reachability logic? For example, we would like to prove that for QLOCK a mutual exclusion invariant holds. But, paradoxically, we cannot! The simple reason is that QLOCK, like many other protocols, *never terminates*, that is, has no terminating sequences whatsoever. But this has the ludicrous trivial consequence that QLOCK's initial model $\mathcal{T}_{\mathcal{R}}$ vacuously satisfies *all* reachability formulas $A \rightarrow^{\circledast} B$. This of course means that it is in fact *impossible* to prove any invariants using reachability logic in the initial model $\mathcal{T}_{\mathcal{R}}$. But it does *not* mean that it is impossible using some other initial model. In Section 4.1 we give a systematic solution to this paradox by means of a *simple theory transformation* allowing us to prove any invariant in the original initial model $\mathcal{T}_{\mathcal{R}}$ by proving an equivalent reachability formula in the initial model of the transformed theory.

Our Contributions. Section 2 gathers preliminaries. The main theoretical contributions of a *simple* semantics and inference system for a rewrite-theory-generic

[1] The notation $\{A\}\mathcal{R}\{B\}$, and the relation to Hoare logic are explained in Section 4.2.

[2] That is, a term in a subsignature $\Omega \subseteq \Sigma$ such that each ground Σ-term is equal modulo E to a ground Ω-term.

reachability logic with just *two* inference rules and its soundness are developed in Sections 4 and 5. A systematic methodology to prove *invariants* by means of reachability formulas is developed in Section 4.1. The semantic relations of reachability logic to Hoare logic and to LTL are explained in Section 4.2. The goal of increasing the logic's potential for automation by making it constructor-based is advanced in Sections 3–5. A proof of concept of the entire approach is given by means of a Maude-based prototype tool and a suite of experiments verifying various properties of distributed system designs in Section 6. Related work and conclusions are discussed in Section 7. Proofs are relegated to Appendix A.

2 Order-Sorted Algebra and Rewriting Logic

We present some preliminaries on order-sorted algebra and rewriting logic. The material is adapted from [33,31,34]. The presentation is self-contained: we only assume the notions of many-sorted signature and many-sorted algebra, e.g., [16].

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

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

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

denotes the family of "subsort polymorphic" operators f. □

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

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

An order-sorted Σ-homomorphism *$h : A \to B$ is a many-sorted (S, Σ)-homomorphism such that whenever $[s] = [s']$ and $a \in A_s \cap A_{s'}$, then we have $h_s(a) = h_{s'}(a)$. This defines a category* $\mathbf{OSAlg}_\Sigma$. □

Theorem 1. *[33] The category $\mathbf{OSAlg}_\Sigma$ has an initial algebra. Furthermore, if Σ is sensible, then the term algebra T_Σ with:*

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

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

For $[s] \in \widehat{S}$, $T_{\Sigma,[s]}$ denotes the set $T_{\Sigma,[s]} = \bigcup_{s' \in [s]} T_{\Sigma,s'}$. T_Σ will (ambiguously) denote: (i) the term algebra; (ii) its underlying S-sorted set; and (iii) the set $T_\Sigma = \bigcup_{s \in S} T_{\Sigma,s}$. An OS signature Σ is said to *have non-empty sorts* iff for each $s \in S$, $T_{\Sigma,s} \neq \emptyset$. An OS signature Σ is called *preregular* [22] 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 $s \in S$. 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 as usual: for atoms, $A, \alpha \models t = t'$ iff $t\alpha = t'\alpha$; for Boolean connectives it is the corresponding Boolean combination of the satisfaction relations for subformulas; and for quantifiers: $A, \alpha \models (\forall x\!:\!s)\ \varphi$ (resp. $A, \alpha \models (\exists x\!:\!s)\ \varphi$) holds iff for all $a \in A_s$ (resp. some $a \in A_s$) we have $A, \alpha \uplus \{(x\!:\!s, a)\} \models \varphi$, where the assignment $\alpha \uplus \{(x\!:\!s, a)\}$ extends α by mapping $x\!:\!s$ to a. Finally, $A \models \varphi$ holds iff $A, \alpha \models \varphi$ holds for each $\alpha \in [Y \to A]$, where $Y = fvars(\varphi)$. We say that φ is *valid* (or *true*) in A iff $A \models \varphi$. We say that φ is *satisfiable* in A iff $\exists \alpha \in [Y \to A]$ such that $A, \alpha \models \varphi$, where $Y = fvars(\varphi)$. For a subsignature $\Omega \subseteq \Sigma$ and $A \in \mathbf{OSAlg}_\Sigma$, the *reduct* $A|_\Omega \in \mathbf{OSAlg}_\Omega$ agrees with A in the interpretation of all sorts and operations in Ω and discards everything in $\Sigma \backslash \Omega$. If $\varphi \in Form(\Omega)$ we have the equivalence $A \models \varphi \Leftrightarrow A|_\Omega \models \varphi$.

An OS *equational theory* is a pair $T = (\Sigma, E)$, with E a set of (possibly conditional) Σ-equations. $\mathbf{OSAlg}_{(\Sigma,E)}$ denotes the full subcategory of $\mathbf{OSAlg}_\Sigma$

with objects those $A \in \mathbf{OSAlg}_{\Sigma}$ such that $A \models E$, called the (Σ, E)-*algebras*. $\mathbf{OSAlg}_{(\Sigma,E)}$ has an *initial algebra* $T_{\Sigma/E}$ [33]. Given $T = (\Sigma, E)$ and $\varphi \in Form(\Sigma)$, we call φ T-*valid*, written $E \models \varphi$, iff $A \models \varphi$ for each $A \in \mathbf{OSAlg}_{(\Sigma,E)}$. We call φ T-*satisfiable* iff there exists $A \in \mathbf{OSAlg}_{(\Sigma,E)}$ with φ satisfiable in A. Note that φ is T-*valid* iff $\neg\varphi$ is T-*unsatisfiable*. The inference system in [33] 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 $E \vdash u = v \quad \Leftrightarrow \quad E \models u = v$. Deducibility $E \vdash u = v$ is abbreviated as $u =_E v$, called E-*equality*. 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 $u_i\sigma =_E v_i\sigma$, $1 \leqslant i \leqslant n$. 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.

Given a set of equations B used for deduction modulo B, a preregular OS signature Σ is called B-*preregular*[3] iff for each $u = v \in B$ and substitutions ρ, $ls(u\rho) = ls(v\rho)$.

We now recall some basic concepts about *rewriting logic*. The survey in [34] gives a fuller account. The key purpose of a rewrite theory $\mathcal{R}$ is to axiomatize a *distributed system*, so that concurrent computation is modeled as concurrent rewriting with the rules of $\mathcal{R}$ *modulo* the equations of $\mathcal{R}$.

Recall the notation for term positions, subterms, and term replacement from [13]: (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.

Definition 3. *A* rewrite theory *is a 3-tuple* $\mathcal{R} = (\Sigma, E \cup B, R)$ *with* $(\Sigma, E \cup B)$ *an OS equational theory and* R *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 quantifier-free* Σ*-formula.*[4]

We further assume 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*. Furthermore,* Σ *is* B-preregular *(in the broader sense of Footnote 3).*

[3] 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 preregular; (ii) Σ is B_0-preregular in the standard sense that $ls(u\rho) = ls(v\rho)$ for all $u = v \in B_0$ and substitutions ρ; and (iii) the axioms U oriented as rules $\vec{U}$ are *sort-decreasing* in the sense of Definition 3 below. Maude automatically checks B-preregularity of an OS signature Σ in this broader sense [8].

[4] Usually, ϕ is assumed to be a *conjunction* of Σ-equations. We give here this more general definition for two reasons: (i) often, using *equationally-defined equality predicates* [23], a quantifier-free formula can be transformed into a conjunction of equalities; and (ii) the more general notion is particularly useful for symbolic methods.

2. *The (possibly conditional) equations E, when oriented as rewrite rules $\vec{E} = \{u \to v \text{ if } \psi \mid u = v \text{ if } \psi \in E\}$, are* convergent *modulo B, that is, sort-decreasing, strictly coherent, confluent, and operationally terminating as rewrite rules modulo B [29].*
3. *The rules R are* ground coherent *with the equations E modulo B [14].*

We refer to [34,29,14] for more details, but give here an intuitive high-level explanation of what the above conditions mean in practice. Conditions (1)–(2) ensure that the initial algebra $T_{\Sigma/E\cup B}$ is isomorphic to the *canonical term algebra* $C_{\Sigma/E,B}$, whose elements are B-equivalence classes of $\vec{E}, B$-irreducible ground Σ-terms.

Define the *one-step R, B-rewrite relation* $t \to_{R,B} t'$ between ground terms as follows. For $t, t' \in T_{\Sigma_{[s]}}$, $[s] \in \widehat{S}$, $t \to_{R,B} t'$ holds iff there is a rewrite rule $l \to r \text{ if } \phi \in R$, a ground substitution $\sigma \in [Y \to T_\Sigma]$ with Y the rule's variables, and a term position p in t such that $t|_p =_B l\sigma$, $t' = t[r\sigma]_p$, and $E \cup B \models \phi\sigma$. In the context of (1)–(2), condition (3) ensures that "computing $\vec{E}, B$-canonical forms before performing R, B-rewriting" is a *complete* strategy. That is, if $t \to_{R,B} t'$ and $u = t!_{E,B}$, i.e., $t \to^*_{\vec{E},B} u$ with u in $\vec{E}, B$-canonical form (abbreviated in what follows to $u = t!$), then there exists a u' such that $u \to_{R,B} u'$ and $t'! =_B u'!$. Note that $vars(r) \subseteq vars(l)$ *is nowhere assumed* for rules $l \to r \text{ if } \phi \in R$. This means that $\mathcal{R}$ can specify an *open system*, in the sense of [40], that interacts with an external, non-deterministic environment.

Conditions (1)–(3) allow a simple and intuitive description of the *initial reachability model* $\mathcal{T}_\mathcal{R}$ [5] of $\mathcal{R}$ as the *canonical reachability model* $\mathcal{C}_\mathcal{R}$ whose states are the elements of the *canonical term algebra* $C_{\Sigma/E,B}$, and where the one-step transition relation $[u] \to_\mathcal{R} [v]$ holds iff $u \to_{R,B} u'$ and $[u'!] = [v]$. Furthermore, if $u \to_{R,B} u'$ has been performed with a rewrite rule $l \to r \text{ if } \phi \in R$ and a ground substitution $\sigma \in [Y \to T_\Sigma]$, then, assuming B-equality is decidable, checking whether condition $E \cup B \models \phi\sigma$ holds is *decidable* by reducing the terms in $\phi\sigma$ to $\vec{E}, B$-canonical form.

A Running Example. Consider the following rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ modeling a dynamic version of the QLOCK mutual exclusion protocol [19], 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 $F = \{0 : \to Nat,\ s\_ : Nat \to Nat,\ \varnothing : \to MSet,\ nil : \to List,\ \_\_ : MSet\ MSet \to MSet,\ \_;\_ : List\ List \to List, dupl : MSet \to Pred, tt : \to Pred, \_|\_|\_|\_ : MSet\ MSet\ MSet\ List \to Conf, < \_ > : Conf \to State\}$, where underscores denote operator argument placement. The axioms B are the associativity-commutativity of the multiset union $\_\_$ with identity $\varnothing$, and the associativity of list concatenation $\_;\_$ with identity nil. The only equation in E is $dupl(s\ i\ i) = tt$. It defines the $dupl$ predicate by detecting a duplicated element i in the multiset $s\ i\ i$ (where s could be empty). The *states* of QLOCK are B-equivalence classes of ground terms of sort $State$.

QLOCK [19] is a mutual exclusion protocol where the number of processes is unbounded. Furthermore, in the *dynamic* version of QLOCK presented below, such a number can grow or shrink. Each process is identified by a number. The system configuration has three sets of processes (normal, waiting, and critical) plus a waiting queue. To ensure mutual exclusion, a normal process must first register its name at the end of the waiting queue. When its name appears at the front of the queue, it is allowed to enter the critical section. The first three rewrite rules in R below specify how a *normal* process i first transitions to a *waiting* process, then to a *critical* process, and back to normal. The last two rules in R specify how a process can dynamically join or exit the system.

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

where $\phi = \mathit{dupl}(n\, i\, w\ c) \neq \mathit{tt}$, 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. Using [7] it is easy to check that $(\Sigma, E \cup B)$ satisfies the finite variant property [12], and that $\mathcal{R} = (\Sigma, E \cup B, R)$ satisfies requirements (1)–(3). Note that *join* makes QLOCK an *open* system in the sense explained above.

3 Constrained Constructor Pattern Predicates

Given an OS equational theory $(\Sigma, E \cup B)$, the *atomic state predicates* appearing in the constructor-based reachability logic formulas of Section 4 will be pairs $u \mid \varphi$, called *constrained constructor patterns*, with u a term in a subsignature $\Omega \subseteq \Sigma$ of constructors, and φ a quantifier-free Σ-formula. Intuitively, $u \mid \varphi$ is a pattern describing the set of states that are $E_\Omega \cup B_\Omega$-equal to ground terms of the form $u\rho$ for ρ a ground constructor substitution such that $E \cup B \models \varphi\rho$. Therefore, $u \mid \varphi$ can be used as a *symbolic description* of a, typically infinite, *set of states* in the canonical reachability model $\mathcal{C}_\mathcal{R}$ of a rewrite theory $\mathcal{R}$.

Often, the signature Σ on which $T_{\Sigma/E\cup B}$ is defined has a natural decomposition as a disjoint union $\Sigma = \Omega \uplus \Delta$, where the elements of the canonical term algebra $C_{\Sigma/E,B}$ 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.

A *decomposition* of an order-sorted equational theory $(\Sigma, E \cup B)$ is a triple $(\Sigma, B, \vec{E})$ such that the rules $\vec{E}$ are convergent modulo B. $(\Sigma, B, \vec{E})$ is called *sufficiently complete* with respect to the *constructor subsignature* Ω iff for each $t \in T_\Sigma$ we have: (i) $t!_{\vec{E},B} \in T_\Omega$, and (ii) if $u \in T_\Omega$ and $u =_B v$, then $v \in T_\Omega$. This ensures that for each $[u]_B \in C_{\Sigma/E,B}$ we have $[u]_B \subseteq T_\Omega$. Sufficient completeness is closely related to the notion of a *protecting* inclusion of decompositions.

Definition 4. *Let* $(\Sigma_0, E_0 \cup B_0) \subseteq (\Sigma, E \cup B)$ *be a theory inclusion such that* $(\Sigma_0, B_0, \vec{E_0})$ *and* $(\Sigma, B, \vec{E})$ *are respective decompositions of* $(\Sigma_0, E_0 \cup B_0)$ *and* $(\Sigma, E \cup B)$. *We then say that the decomposition* $(\Sigma, B, \vec{E})$ protects $(\Sigma_0, B_0, \vec{E_0})$ *iff (i) 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)* $C_{\Sigma_0/E_0, B_0} = C_{\Sigma/E,B}|_{\Sigma_0}$.

$(\Omega, B_\Omega, \vec{E_\Omega})$ *is a* constructor decomposition *of* $(\Sigma, B, \vec{E})$ *iff (i)* $(\Sigma, B, \vec{E})$ *protects* $(\Omega, B_\Omega, \vec{E_\Omega})$, *and (ii)* $(\Sigma, B, \vec{E})$ *is sufficiently complete with respect to the constructor subsignature* Ω. *Furthermore,* Ω *is called a subsignature of* free constructors modulo B_Ω *iff* $E_\Omega = \emptyset$, *so that* $C_{\Omega/E_\Omega, B_\Omega} = T_{\Omega/, B_\Omega}$.

We are now ready to define constrained constructor pattern predicates and their semantics. In what follows, X will always denote the countably infinite S-sorted set of variables used in the language of Σ-formulas.

Definition 5. *Let* $(\Omega, B_\Omega, \vec{E_\Omega})$ *be a* constructor decomposition *of* $(\Sigma, B, \vec{E})$. *A* constrained constructor pattern *is an expression* $u \mid \varphi$ *with* $u \in T_\Omega(X)$ *and* φ *a QF* Σ*-formula. The set PatPred*(Ω, Σ) *of* constrained constructor pattern predicates *contains* $\bot$ *and the set of constrained constructor patterns, and is closed under disjunction* ($\vee$) *and conjunction* ($\wedge$). *Capital letters* $A, B, \ldots, P, Q, \ldots$ *range over PatPred*(Ω, Σ). *The* semantics *of a constrained constructor pattern predicate* A *is a subset* $[\![A]\!] \subseteq C_{\Sigma/E,B}$ *defined inductively as follows:*

1. $[\![\bot]\!] = \emptyset$
2. $[\![u \mid \varphi]\!] = \{[(u\rho)!]_{B_\Omega} \in C_{\Sigma/E,B} \mid \rho \in [X \to T_\Omega] \ \wedge \ E \cup B \models \varphi\rho\}$.
3. $[\![A \vee B]\!] = [\![A]\!] \cup [\![B]\!]$
4. $[\![A \wedge B]\!] = [\![A]\!] \cap [\![B]\!]$.

Note that for any constructor pattern predicate A, if σ is a (sort-preserving) bijective renaming of variables we always have $[\![A]\!] = [\![A\sigma]\!]$. Given constructor patterns $u \mid \varphi$ and $v \mid \psi$ with $vars(u \mid \varphi) \cap vars(v \mid \psi) = \emptyset$, we say that $u \mid \varphi$ *subsumes* $v \mid \psi$ iff there is a substitution α such that: (i) $v =_{E_\Omega \cup B_\Omega} u\alpha$, and (ii) $\mathcal{T}_{E \cup B} \models \psi \Rightarrow (\varphi\alpha)$. It then follows easily from the above definition of $[\![u \mid \varphi]\!]$ that if $u \mid \varphi$ subsumes $v \mid \psi$, then $[\![v \mid \psi]\!] \subseteq [\![u \mid \varphi]\!]$. Likewise, $\bigvee_{i \in I} u_i \mid \varphi_i$ *subsumes* $v \mid \psi$ iff there is a $k \in I$ such that $u_k \mid \varphi_k$ subsumes $v \mid \psi$.

Pattern Predicate Example. Recall that QLOCK states have the general form $< n \mid w \mid c \mid q >$ with n, w, c multisets of process identifiers and q an associative list of process identifiers. From the five rewrite rules defining QLOCK, it is easy to prove that if $< n \mid w \mid c \mid q > \to^* < n' \mid w' \mid c' \mid q' >$ and $n\,w\,c$ is a set (has no repeated elements), then $n'\,w'\,c'$ is also a set. Of course, it seems very reasonable to assume that these process identifier multisets are, in fact, sets, since otherwise we could, for example, have a process i which is *both* waiting and critical at the *same* time. We can rule out such ambiguous states by means of the pattern predicate $< n \mid w \mid c \mid q > \mid dupl(n\ w\ c) \neq tt$.

Note that, assuming that $E_\Omega \cup B_\Omega$ has a finitary unification algorithm, any constrained constructor pattern predicate A is semantically equivalent to a finite disjunction $\bigvee_i u_i \mid \varphi_i$ of constrained constructor patterns. This is because: (i)

by (3)–(4) in Def. 5 we may assume A in disjunctive normal form; and (ii) it is easy to check that $[\![(u \mid \varphi) \wedge (v \mid \phi)]\!] = \bigcup_{\alpha \in Unif_{E_\Omega \cup B_\Omega}(u,v)} [\![u\alpha \mid (\varphi \wedge \phi)\alpha]\!]$, were we assume without loss of generality that $vars(u \mid \varphi) \cap vars(v \mid \psi) = \varnothing$, and that all variables in $ran(\alpha)$ are *fresh*.

In the above discussion of intersections we assumed that the variables in the two constructor patterns are disjoint. But this may not always be what we want. Consider constrained patterns $u \mid \varphi$ and $v \mid \phi$ with $Y = vars(u \mid \varphi) \cap vars(v \mid \phi) = vars(u) \cap vars(v)$. The sharing of variables Y may be intentional as *parameters* common to both $u \mid \varphi$ and $v \mid \phi$. This can be illustrated with an example of two patterns describing triples of natural numbers, namely, $\langle 0, y, z\rangle \mid \top$ and $\langle x, s(y), 1\rangle \mid \top$ with shared parameter y. We can understand these patterns *parametrically* as describing the $\mathbb{N}$-*indexed families* of sets: $\{\{\langle 0, n, z\rangle \mid z \in \mathbb{N}\}\}_{n\in\mathbb{N}}$ and $\{\{\langle x, s(n), 1\rangle \mid x \in \mathbb{N}\}\}_{n\in\mathbb{N}}$. Their $\mathbb{N}$-*indexed intersection* $\{\{\langle 0, n, z\rangle \mid z \in \mathbb{N}\} \cap \{\langle x, s(n), 1\rangle \mid z \in \mathbb{N}\}\}_{n\in\mathbb{N}} = \{\varnothing\}_{n\in\mathbb{N}}$ can then be symbolically described by $\bot$, because the terms $\langle 0, y, z\rangle$ and $\langle x, s(y), 1\rangle$ have *no unifier*, although by renaming $\langle x, s(y), 1\rangle$ to $\langle x, s(y'), 1\rangle$ they *can be unified* into the term $\langle 0, s(y''), 1\rangle$, so that $[\![\langle 0, y, z\rangle \mid \top]\!] \cap [\![\langle x, s(y), 1\rangle \mid \top]\!] = [\![\langle 0, s(y''), 1\rangle \mid \top]\!]$. This suggests the notion of a Y-*parameterized intersection* defined as: $[\![u \mid \varphi]\!] \cap_Y [\![v \mid \phi]\!] = \bigcup_{\alpha \in Unif_{E_\Omega \cup B_\Omega}(u,v)} [\![u\alpha \mid (\varphi \wedge \phi)\alpha]\!]$, assuming $Y = vars(u \mid \varphi) \cap vars(v \mid \phi) = vars(u) \cap vars(v)$, and that all variables in $ran(\alpha)$ are *fresh*. Under the same assumption there is also a natural notion of Y-*parameterized subsumption* of $v \mid \phi$ by $u \mid \varphi$, denoted $v \mid \phi \subseteq_Y u \mid \varphi$, namely, such subsumption holds iff there is a substitution α such that: (i) $(\forall y \in Y)\ \alpha(y) = y$, (ii) $v =_{E_\Omega \cup B_\Omega} u\alpha$, and (iii) $\mathcal{T}_{E\cup B} \models \phi \Rightarrow (\varphi\alpha)$. This ensures that for all ground substitutions ρ of the variables Y we have $[\![(v \mid \phi)\rho]\!] \subseteq [\![(u \mid \varphi)\rho]\!]$. These notions will be used in Section 4.1 to reason about *parameterized* invariants and co-invariants.

4 Constructor-Based Reachability Logic

The constructor-based reachability logic we shall define is a logic to reason about reachability properties of the canonical reachability model $\mathcal{C}_\mathcal{R}$ of a topmost rewrite theory $\mathcal{R}$, where "topmost" captures the intuitive idea that all rewrites with the rules R in $\mathcal{R}$ happen at the top of the term. Many rewrite theories of interest, including theories specifying distributed object-oriented systems and rewriting logic specifications of (possibly concurrent) programming languages, can be easily specified as topmost rewrite theories by a simple theory transformation (see, e.g., [48]). The rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$, besides satisfying the requirements in Definition 3, satisfies two additional requirements:

1. $(\Sigma, E \cup B)$ has a sort *State*, the top sort of a connected component $[State]$, a decomposition $(\Sigma, B, \vec{E})$, and a constructor decomposition $(\Omega, B_\Omega, \vec{E_\Omega})$ such that: (i) $\forall u \in T_\Omega(X)_{State}$, $vars(u) = vars(u!)$; and (ii) the axioms B_Ω are linear and regular and have a finitary $E_\Omega \cup B_\Omega$-unification algorithm.

2. Rules in R have the form $l \to r$ *if* φ with $l \in T_\Omega(X)$. Furthermore, they are *topmost* in the sense that: (i) for all such rules, l and r have sort *State*, and (ii) for any $u \in T_\Omega(X)_{State}$ and any non-empty position p in u, $u|_p \notin T_\Omega(X)_{State}$.

Requirements (1)–(2) ensure that in the canonical reachability model $\mathcal{C}_\mathcal{R}$ if $[u] \to_\mathcal{R} [v]$ holds, then the R, B-rewrite $u \to_{R,B} u'$ such that $[u'!] = [v]$ *happens at the top* of u, i.e., uses a rewrite rule $l \to r$ *if* $\varphi \in R$ and a ground substitution $\sigma \in [Y \to T_\Omega]$, with Y the rule's variables, such that $u =_{B_\Omega} l\sigma$ and $u' = r\sigma$.

We are now ready to define the formulas of our constructor-based reachability logic for $\mathcal{R}$ satisfying above requirements (1)–(2). Let $PatPred(\Omega, \Sigma)_{State}$ denote the subset $PatPred(\Omega, \Sigma)$ determined by those pattern predicates A such that, for all atomic constrained constructor predicates $u \mid \varphi$ appearing in A, u has sort *State*. Reachability logic formulas then have the form: $A \to^{\circledast} B$, with $A, B \in PatPred(\Omega, \Sigma)_{State}$, where $(\Omega, B_\Omega, \vec{E_\Omega})$ is the constructor decomposition of $(\Sigma, B, \vec{E})$. By definition, the *parameters* Y of $A \to^{\circledast} B$ are the variables in the set $Y = vars(A) \cap vars(B)$, and $A \to^{\circledast} B$ is called *unparameterized* iff $Y = \varnothing$.

The presentation of reachability logic in [47] considers *two* different semantics: (i) a *one-path* semantics, which we denote $\mathcal{R} \models^1 A \to^{\circledast} B$, and (ii) an *all-paths* semantics, which we denote $\mathcal{R} \models^\forall A \to^{\circledast} B$. Since the all-paths semantics is the most general and expressive, and the one-path semantics applies mostly to sequential systems, in this work we focus on the all-paths semantics.

The reachability logic in [46,47] is based on *terminating* sequences of state transitions and is such that all reachability formulas are *vacuously true* when there are no terminating states. Our purpose is to extend reachability logic so as to be able to verify properties of general distributed systems specified as rewrite theories $\mathcal{R}$ which *may never terminate*. For this, as further explained in Section 4.1, we need to *generalize* the satisfaction relation $\mathcal{R} \models^\forall A \to^{\circledast} B$ to a *relativized* satisfaction relation $\mathcal{R} \models^\forall_T A \to^{\circledast} B$, where T is a constrained pattern predicate such that $[\![T]\!]$ is a set of terminating states. That is, let $Term_\mathcal{R} = \{[u] \in \mathcal{C}_{\mathcal{R},State} \mid (\nexists [v])\ [u] \to_\mathcal{R} [v]\}$. We then require $[\![T]\!] \subseteq Term_\mathcal{R}$. The standard relation $\mathcal{R} \models^\forall A \to^{\circledast} B$ is then recovered as the special case where $[\![T]\!] = Term_\mathcal{R}$. Call $[u] \to^*_\mathcal{R} [v]$ a *T-terminating sequence* iff $[v] \in [\![T]\!]$.

Definition 6. *Given T with $[\![T]\!] \subseteq Term_\mathcal{R}$, the* all-paths *satisfaction relation $\mathcal{R} \models^\forall_T u \mid \varphi \to^{\circledast} \bigvee_{j \in J} v_j \mid \phi_j$ asserts the satisfaction of the formula $u \mid \varphi \to^{\circledast} \bigvee_{j \in J} v_j \mid \phi_j$ in the canonical reachability model $\mathcal{C}_\mathcal{R}$ of a rewrite theory $\mathcal{R}$ satisfying topmost requirements (1)–(2). It is defined as follows:*

For $u \mid \varphi \to^{\circledast} \bigvee_{j \in J} v_j \mid \phi_j$ unparameterized, $\mathcal{R} \models^\forall_T u \mid \varphi \to^{\circledast} \bigvee_{j \in J} v_j \mid \phi_j$ holds iff for each T-terminating sequence $[u_0] \to_\mathcal{R} [u_1] \ldots [u_{n-1}] \to_\mathcal{R} [u_n]$ with $[u_0] \in [\![u \mid \varphi]\!]$ there exist k, $0 \leqslant k \leqslant n$ and $j \in J$ such that $[u_k] \in [\![v_j \mid \phi_j]\!]$. For $u \mid \varphi \to^{\circledast} \bigvee_{j \in J} v_j \mid \phi_j$ with parameters Y, $\mathcal{R} \models^\forall_T u \mid \varphi \to^{\circledast} \bigvee_{j \in J} v_j \mid \phi_j$ holds if $\mathcal{R} \models^\forall_T (u \mid \varphi)\rho \to^{\circledast} (\bigvee_{j \in J} v_j \mid \phi_j)\rho$ holds for each $\rho \in [Y \to T_\Omega]$.

Since a constrained pattern predicate is equivalent to a disjunction of atomic ones, we can define satisfaction on general reachability logic formulas as follows: $\mathcal{R} \models^\forall_T \bigvee_{1 \leqslant i \leqslant n} u_i \mid \varphi_i \to^{\circledast} A$ iff $\bigwedge_{1 \leqslant i \leqslant n} \mathcal{R} \models^\forall_T u_i \mid \varphi_i \to^{\circledast} A$, assuming same parameters $Y_i = vars(u_i \mid \varphi_i) \cap vars(A)$, i.e., $Y_i = Y_{i'}$ for $1 \leqslant i < i' \leqslant n$.

$\mathcal{R} \models^{\forall}_{T} A \rightarrow^{\circledast} B$ is a *path-universal partial correctness assertion*: If state $[u]$ satisfies "precondition" A, then "postcondition" B is satisfied *somewhere* along *each* T-terminating sequences from $[u]$, generalizing a Hoare formula $\{A\}\mathcal{R}\{B\}$.

Recall that in requirement (2) for our rewrite theory $\mathcal{R}$ we assumed topmost rewrite rules of the form $l \rightarrow r$ *if* ϕ with $l \notin T_\Omega(X)$. For symbolic reasoning purposes it will be very useful to also require that $r \notin T_\Omega(X)$. This can be done without any real loss of generality by means of a theory transformation $\mathcal{R} \mapsto \hat{\mathcal{R}}$ defined as follows. If $\mathcal{R} = (\Sigma, E \cup B, R)$, then $\hat{\mathcal{R}} = (\Sigma, E \cup B, \hat{R})$, where the rules $\hat{R}$ are obtained from the rules R by transforming each $l \rightarrow r$ *if* ϕ in R into the rule $l \rightarrow r'$ *if* $\phi \wedge \hat{\theta}$, where: (i) r' is the *Ω-abstraction* of r obtained by replacing each length-minimal position p of r such that $t|_p \notin T_\Omega(X)$ by a fresh variable x_p whose sort is the least sort of $t|_p$, (ii) $\hat{\theta} = \bigwedge_{p \in P} x_p = t_p$, where P is the set of all length-minimal positions in r such that $t|_p \notin T_\Omega(X)$. The key semantic property about this transformation can be expressed as follows:

Lemma 1. *The canonical reachability models $\mathcal{C}_\mathcal{R}$ and $\mathcal{C}_{\hat{\mathcal{R}}}$ are* identical.

4.1 Invariants, Co-Invariants, and Never-Terminating Systems

The notion of an *invariant* makes sense for any transition system $\mathcal{S}$, that is, for any pair $\mathcal{S} = (S, \rightarrow_\mathcal{S})$ with S its set of *states* and $\rightarrow_\mathcal{S} \subseteq S \times S$ its *transition relation*. Given a set of "initial states" $S_0 \subseteq S$, the set $Reach(S_0)$ of states *reachable* from S_0 is defined as $Reach(S_0) = \{s \in S \mid (\exists s_0 \in S_0)\ s_0 \rightarrow^*_\mathcal{S} s\}$, where $\rightarrow^*_\mathcal{S}$ denotes the reflexive-transitive closure of $\rightarrow_\mathcal{S}$. An invariant is a safety property about $\mathcal{S}$ with initial states S_0 and can be specified in two ways: (i) by a "good" property $P \subseteq S$, the *invariant*, that *always holds* from S_0, i.e., such that $Reach(S_0) \subseteq P$, or (ii) as a "bad" property $Q \subseteq S$, the *co-invariant*, that *never holds* from S_0, i.e., such that $Reach(S_0) \cap Q = \varnothing$. Obviously, P is an invariant iff $S \backslash P$ is a co-invariant. Sometimes it is easier to specify an invariant *positively*, as P, and sometimes *negatively*, as its co-invariant $S \backslash P$.

All this is particularly relevant for the transitions system $(\mathcal{C}_{\mathcal{R},State}, \rightarrow_\mathcal{R})$ associated to the canonical model $\mathcal{C}_\mathcal{R}$ of a rewrite theory $\mathcal{R}$. Here is an obvious question with a non-obvious answer. Suppose we have specified a distributed system as the canonical model $\mathcal{C}_\mathcal{R}$ of a rewrite theory $\mathcal{R}$ satisfying topmost requirements (1)–(2). Suppose further that we have specified constrained pattern predicates S_0 and P (resp. and Q) and we want to prove that $[\![P]\!]$ (resp. $[\![Q]\!]$) is an *invariant* (resp. *co-invariant*) of the system $(\mathcal{C}_{\mathcal{R},State}, \rightarrow_\mathcal{R})$ from $[\![S_0]\!]$. Can we specify such invariant or co-invariant by means of *reachability formulas* and use the inference system of Section 5 to try to prove such formulas?

The answer to the above question is not obvious. Suppose $\mathcal{R}$ specifies a *never-terminating system*, i.e., a system such that $Term_\mathcal{R} = \varnothing$. Many distributed systems are never-terminating. For example, QLOCK and other mutual exclusion protocols are never-terminating. Then, no reachability formula can characterize and invariant (resp. co-invariant) holding by means of the satisfaction relation $\mathcal{R} \models^{\forall}_{T} A \rightarrow^{\circledast} B$. The reason for this impossibility is that, since $Term_\mathcal{R} = \varnothing$, $\mathcal{R} \models^{\forall}_{T} A \rightarrow^{\circledast} B$ holds vacuously for *all* reachability formulas $A \rightarrow^{\circledast} B$.

Is then reachability logic useless to prove invariants? Definitely *not*. It can indeed be very useful for proving invariants and co-invariants of distributed systems, regardless of whether they are: (a) terminating, (b) sometimes terminating, or (c) never terminating: we just need to first perform a simple *theory transformation*. Call an invariant *specifiable by constrained pattern predicates* S_0 and P if $[\![P]\!]$ is an *invariant* of $(\mathcal{C}_{\mathcal{R},State}, \rightarrow_{\mathcal{R}})$ from $[\![S_0]\!]$. To ease the exposition, we explain the transformation for the case where Ω has a single state constructor operator, say, $\langle \_, \ldots, \_ \rangle : s_1, \ldots, s_n \rightarrow State$. The extension to several such operators is straightforward. The theory transformation is of the form $\mathcal{R} \mapsto \mathcal{R}_{stop}$, where $\mathcal{R}_{stop}$ is obtained from $\mathcal{R}$ by just adding: (1) a new state constructor operator $[\_, \ldots, \_] : s_1, \ldots, s_n \rightarrow State$ to Ω, and (2) a new rewrite rule $stop : \langle x_1 : s_1, \ldots, x_n : s_n \rangle \rightarrow [x_1 : s_1, \ldots, x_n : s_n]$ to R. Also, let $[\,]$ denote the pattern predicate $[x_1 : s_1, \ldots, x_n : s_n] \mid \top$. Likewise, for any atomic constrained pattern predicate $B = \langle u_1, \ldots, u_n \rangle \mid \varphi$ we define the pattern predicate $[B] = [u_1, \ldots, u_n] \mid \varphi$ and extend this notation to any union Q of atomic predicates. Since $\langle \_, \ldots, \_ \rangle : s_1, \ldots, s_n \rightarrow State$ is the only state constructor, we can assume without loss of generality that any atomic constrained pattern predicate in $\mathcal{R}$ is semantically equivalent to one of the form $\langle u_1, \ldots, u_n \rangle \mid \varphi$. Likewise, any pattern predicate will be semantically equivalent to a union of atomic predicates of such form, called in *standard form*. Here is the main theorem:

Theorem 3. *For $S_0, P \in PatPred(\Omega, \Sigma)$ constrained pattern predicates in standard form with $vars(S_0) \cap vars(P) = \varnothing$, $[\![P]\!]$ is an invariant of $(\mathcal{C}_{\mathcal{R},State}, \rightarrow_{\mathcal{R}})$ from $[\![S_0]\!]$ iff $\mathcal{R}_{stop} \models^{\forall}_{[\,]} S_0 \rightarrow^{\circledast} [P]$.*

The notion of a *parametric invariant* can be reduced to the unparameterized one: if $Y = vars(S_0) \cap vars(P)$, then $[\![P]\!]$ is an *invariant* of $(\mathcal{C}_{\mathcal{R},State}, \rightarrow_{\mathcal{R}})$ from $[\![S_0]\!]$ *with parameters* Y iff $\mathcal{R}_{stop} \models^{\forall}_{[\,]} S_0 \rightarrow^{\circledast} [P]$. That is, iff $[\![P\rho]\!]$ is an (unparameterized) invariant of $(\mathcal{C}_{\mathcal{R},State}, \rightarrow_{\mathcal{R}})$ from $[\![S_0\rho]\!]$ for each $\rho \in [Y \rightarrow T_\Omega]$. In this way, Theorem 3 extends seamlessly to parametric invariants.

Specifying Invariants for QLOCK. We illustrate how to specify invariants as reachability formulas using the QLOCK specification from Sections 2 and 3. Note that not only is QLOCK nonterminating: it is also *never* terminating. Thus, specifying any invariants as reachability formulas in the original theory is impossible. However, we can apply the theory transformation suggested by Theorem 3 by adding a fresh operator $[\_] : Conf \rightarrow State$ and a rule $stop : < t > \rightarrow [t]$ for $t : Conf$. Define the set of initial states having only normal processes by the pattern predicate $S_0 = < n' \mid \varnothing \mid \varnothing \mid nil > \mid dupl(n') \neq tt$. Since QLOCK states have the form $< n \mid w \mid c \mid q >$, mutual exclusion means $|c| \leqslant 1$, which is expressible by the pattern predicate $< n \mid w \mid i \mid i \; ; \; q > \vee < n \mid w \mid \varnothing \mid q >$. But we need also to ensure our multisets are actually *sets*. Thus, the pattern predicate $P = \big(< n \mid w \mid i \mid i \; ; \; q > \mid dupl(n \; w \; i) \neq tt\big) \vee \big(< n \mid w \mid \varnothing \mid q > \mid dupl(n \; w) \neq tt\big)$ specifies mutual exclusion. By Theorem 3, QLOCK ensures mutual exclusion from $[\![S_0]\!]$ iff $\mathcal{R}_{stop} \models^{\forall}_{[\,]} S_0 \rightarrow^{\circledast} [P]$.

The following easy corollary can be very helpful in proving invariants. It can, for example, be applied to prove the mutual exclusion of QLOCK.

Corollary 1. *Let $S_0, P \in PatPred(\Omega, \Sigma)$ be constrained pattern predicates in standard form with $vars(S_0) \cap vars(P) = Y$. Then $[\![P]\!]$ is an invariant of $(\mathcal{C}_{\mathcal{R},State}, \rightarrow_{\mathcal{R}})$ from $[\![S_0]\!]$ with parameters Y if: (i) $S_0 \subseteq_Y P$ (see Section 3), and (ii) $\mathcal{R}_{stop} \models^{\forall}_{[]} P \rightarrow^{\circledast} [P\sigma]$, where σ is a sort-preserving bijective renaming of variables such that σ is the identity on Y and $vars(P) \cap vars(P\sigma) = Y$.*

Let us now turn to the case of co-invariants. Suppose we have specified constrained pattern predicates S_0 and Q and we want to prove that $[\![Q]\!]$ is a *co-invariant* of the system $(\mathcal{C}_{\mathcal{R},State}, \rightarrow_{\mathcal{R}})$ from $[\![S_0]\!]$. Can this be expressed by some reachability formula or formulas? The answer is yes! By using the rules of $\mathcal{R}$ *backwards*. Assume without loss of generality that $\mathcal{R} = \hat{\mathcal{R}}$. Then, if $\mathcal{R} = (\Sigma, E \cup B, R)$, define $\mathcal{R}^{-1} = (\Sigma, E \cup B, R^{-1})$, where $R^{-1} = \{r \rightarrow l \text{ if } \varphi \mid (l \rightarrow r \text{ if } \varphi) \in R\}$. Then note that if $\mathcal{R}$ satisfies the topmost conditions (1)–(2) and, assuming the rules R^{-1} are ground coherent with the equations E modulo B, then $\mathcal{R}^{-1}$ also satisfies conditions (1)–(2). Here is the key result.

Theorem 4. *Under the above assumptions on $\mathcal{R}^{-1}$, if $S_0, Q \in PatPred(\Omega, \Sigma)$ are constrained pattern predicates in standard form with $vars(S_0) \cap vars(Q) = \varnothing$, then $[\![Q]\!]$ is a co-invariant of $(\mathcal{C}_{\mathcal{R},State}, \rightarrow_{\mathcal{R}})$ from $[\![S_0]\!]$ if: (i) $[\![Q]\!] \cap [\![S_0]\!] = \varnothing$, and (ii) $(\mathcal{R}^{-1})_{stop} \models^{\forall}_{[]} Q \rightarrow^{\circledast} [Q\sigma]$, where σ is a bijective renaming of variables such that $vars(Q) \cap vars(Q\sigma) = \varnothing$.*

The reduction of parametric invariants to unparameterized ones applies *mutatis mutandis*, to parametric co-invariants. For example, in the parametric version of the above theorem, where $Y = vars(S_0) \cap vars(Q)$, the checking of (i) now becomes checking $[\![Q]\!] \cap_Y [\![S_0]\!] = \varnothing$ (see Section 3); and for (ii), proving the reachability formula $Q \rightarrow^{\circledast} [Q\sigma]$, where σ is a bijective renaming of variables such that $vars(Q) \cap vars(Q\sigma) = Y$ and $\sigma(y) = y$ for each $y \in Y$.

4.2 Relationships to Hoare Logic and Universally Quantified LTL

It is both natural and helpful to compare the reachability logic formalism to other commonly used notations such as Hoare logic or linear time temporal logic (LTL). Let us begin with Hoare logic. A Hoare logic is usually associated to a programming language; but the desired comparison should apply not just to programming languages but to any systems specifiable by topmost rewrite theories. This suggests defining Hoare logic in this more general setting.

Definition 7. *(Hoare Logic). Let $\mathcal{R} = (\Sigma, E \cup B, R)$ satisfy the requirements in Definition 3 and the topmost requirements (1)–(2), and let Ω be its constructor subsignature. A* Hoare triple *for $\mathcal{R}$ is then a triple of the form:*

$$\{A\}\ \mathcal{R}\ \{B\}$$

where $A, B \in PatPred(\Omega, \Sigma)_{State}$. Let $Y = vars(A) \cap vars(B)$. By definition, when $Y = \varnothing$, a Hoare triple $\{A\}\ \mathcal{R}\ \{B\}$ is satisfied by the initial model $\mathcal{T}_{\mathcal{R}}$, denoted $\mathcal{T}_{\mathcal{R}} \models \{A\}\ \mathcal{R}\ \{B\}$, iff for each $[u] \in [\![A]\!]$ and each terminating sequence $[u] \rightarrow_{\mathcal{R}}! [v]$, $[v] \in [\![B]\!]$. If $Y \neq \varnothing$, then $\mathcal{T}_{\mathcal{R}} \models \{A\}\ \mathcal{R}\ \{B\}$ iff $\mathcal{T}_{\mathcal{R}} \models \{A\rho\}\ \mathcal{R}\ \{B\rho\}$ for each $\rho \in [X \rightarrow T_{\Omega}]$.

Since the rewriting logic semantics of a programming language $\mathcal{L}$ can be specified by a topmost rewrite theory $\mathcal{R}_{\mathcal{L}}$, the *standard* Hoare logic for $\mathcal{L}$ becomes the special case where in the above notation we represent a Hoare triple $\{\varphi\}\ p\ \{\psi\}$ as the Hoare triple $\{\langle p : \mathit{init}\rangle \mid \widetilde{\varphi}\}\ \mathcal{R}_{\mathcal{L}}\ \{\langle \mathit{skip} : S\rangle \mid \widetilde{\psi}\}$, where *init* is the initial program state, of sort *ProgState*, and where *configurations* of a program (or, more generally, a continuation) p and a program state S are represented as pairs $\langle p : S\rangle$. Explaining how the QF $\Sigma_{\mathcal{L}}$-formulas $\widetilde{\varphi}$ and $\widetilde{\psi}$ are derived from the original φ and ψ is essentially straightforward, but becomes complicated by the systematic confusion of *program* variables with *mathematical* variables in φ and ψ. This can be best illustrated with an example. Consider the Hoare triple $\{n \geqslant 0\}$ `x := n ; factp` $\{\mathtt{y} = n!\}$, which specifies that a factorial program `factp` with its variable `x` initialized to the integer $n \geqslant 0$ will have upon termination the value $n!$ stored in its variable `y`. For $\mathcal{R}_{\mathcal{L}}$ this can be expressed as the Hoare triple $\{\langle \mathtt{x := } n \mathtt{\ ; factp} : \mathit{init}\rangle \mid n \geqslant 0\}\ \mathcal{R}_{\mathcal{L}}\ \{\langle \mathit{skip} : S\rangle \mid S[\mathtt{y}] = n!\}$, where S is a variable of sort *ProgState* and $S[v]$ is an auxiliary function extracting the value in state S of program variable v. Of course, conversely, a Hoare triple $\{A\}\ \mathcal{R}\ \{B\}$ has also in a sense an equivalent *standard* interpretation, since we can view $\mathcal{R}$ as a *program* in a rewriting logic language such as Maude.

The comparison with reachability logic is now straightforward: Hoare logic is essentially a sublogic of reachability logic, namely, a Hoare triple $\{A\}\ \mathcal{R}\ \{B\}$ is just syntactic sugar for the reachability formula[5] $A \to^{\circledast} (B \wedge T)$, where $[\![T]\!] = \mathit{Term}_{\mathcal{R}}$, and we assume $\mathit{vars}(T) \cap Y = \varnothing$. Indeed, we then have:

$$\mathcal{T}_{\mathcal{R}} \models \{A\}\ \mathcal{R}\ \{B\} \quad \Leftrightarrow \quad \mathcal{R} \models^{\forall} \ A \to^{\circledast} (B \wedge T).$$

The comparison with LTL requires making explicit the atomic predicates and the Kripke structure $\mathcal{K}_{\mathcal{R}}$ associated to $\mathcal{R} = (\Sigma, E \cup B, R)$ on which the comparison is based. The atomic predicates are $\mathit{PatPred}(\Omega, \Sigma)_{\mathit{State}}$, and $\mathcal{K}_{\mathcal{R}}$ is the Kripke structure $\mathcal{K}_{\mathcal{R}} = (C_{\Sigma/E,B,\mathit{State}}, (\to_{\mathcal{R}})^{\bullet}, L_{\mathcal{R}})$, where $(\to_{\mathcal{R}})^{\bullet}$ is the totalization of the one-step rewrite relation and $L_{\mathcal{R}}$ is the labeling function:

$$C_{\Sigma/E,B,\mathit{State}} \ni [u] \mapsto \{A \in \mathit{PatPred}(\Omega, \Sigma)_{\mathit{State}} \mid [u] \in [\![A]\!]\} \in \mathcal{P}(\mathit{PatPred}(\Omega, \Sigma)_{\mathit{State}}).$$

Note the useful fact that $\mathcal{K}_{\mathcal{R}}$ can give semantics not only to *propositional* LTL formulas φ, but also to *universal quantifications* $(\forall Y)\ \varphi$ of propositional LTL formulas φ, where Y is a (possibly empty) finite set of variables typed in the signature Σ of $\mathcal{R}$. Indeed, we can *define*, for each $[u] \in C_{\Sigma/E,B,\mathit{State}}$,

$$\mathcal{K}_{\mathcal{R}}, [u] \models_{\mathit{LTL}} (\forall Y)\ \varphi \quad \Leftrightarrow \quad \forall \rho \in [Y \to T_{\Omega}]\ \mathcal{K}_{\mathcal{R}}, [u] \models_{\mathit{LTL}} \varphi\rho.$$

The comparison with LTL then also becomes straightforward: reachability logic is essentially a sublogic of quantified LTL: a reachability formula $A \to^{\circledast} B$ with parameters Y is syntactic sugar for the LTL formula $(\forall Y)\ A \to \mathit{en}_{\mathcal{R}}\, \mathcal{W}\, B$,

[5] Admittedly, the pattern predicate $B \wedge T$ is not a disjunction of constrained patterns. However, by handling substitutions α in disjoint unifiers as extra conjunctions of equalities $\hat{\alpha}$, we can transform $B \wedge T$ into a disjunction of constrained patterns without affecting the parameters of $A \to^{\circledast} (B \wedge T)$.

where $\mathcal{W}$ is the "weak until" operator, and if $R = \{l_i \to r_i \text{ if } \varphi_i\}_{i\in I}$, then $en_{\mathcal{R}}$ is the "enabledness" pattern predicate $en_{\mathcal{R}} = \bigvee_{i\in I} l_i \mid \varphi_i$. Indeed, we have:

$$\mathcal{R} \models^{\forall} A \to^{\circledast} B \;\Leftrightarrow\; \mathcal{K}_{\mathcal{R}} \models_{LTL} (\forall Y)\, A \to en_{\mathcal{R}}\, \mathcal{W}\, B.$$

Of course, when the semantics of $A \to^{\circledast} B$ is relativized to a pattern predicate T of terminating states, we get instead the LTL formula $(\forall Y)\, A \to (\neg T)\, \mathcal{W}\, B$.

Note, finally, that thanks to the results in Section 4.1, reachability logic can also express universal LTL *safety formulas* of the form: $(\forall Y)\ A \to \Box B$ (with $A, B \in PatPred(\Omega, \Sigma)_{State}$ and $Y = vars(A) \cap vars(B)$), since we have:

$$\mathcal{R}_{stop} \models^{\forall}_{[]} A \to^{\circledast} [B] \;\Leftrightarrow\; \mathcal{K}_{\mathcal{R}} \models_{LTL} (\forall Y)\, A \to \Box B.$$

5 A Sound Inference System

We present our inference system for all-path reachability, parametric on $\mathcal{R}$ satisfying topmost requirements (1)–(2), with rules $R = \{l_j \to r_j \text{ if } \phi_j\}_{j\in J}$ such that $l_j, r_j \in T_\Omega(X)$, $j \in J$. *Variables of rules in R are always assumed disjoint from variables in reachability formulas*; this can be ensured by renaming. The inference system has two proof rules. The Step$^{\forall}$ + Subsumption proof rule allows taking one step of (symbolic) rewriting along all paths according to the rules in $\mathcal{R}$. The Axiom proof rule allows the use of a trusted reachability formula to summarize multiple rewrite steps, and thus to handle repetitive behavior.

These proof rules derive sequents of the form $[\mathcal{A},\, \mathcal{C}] \;\vdash_T\; u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$, where $\mathcal{A}$ and $\mathcal{C}$ are finite sets of reachability formulas and T a pattern predicate defining a set of T-terminating ground states. Formulas in $\mathcal{A}$ are called *axioms* and those in $\mathcal{C}$ are called *circularities*. We furthermore assume that in all reachability formulas $u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ we have $vars(\psi_i) \subseteq vars(v_i) \cup vars(u \mid \varphi)$ for each i. According to the implicit quantification of the semantic relation $\models^{\forall}_T$ this means that any variable in ψ_i is either universally quantified and comes from the precondition $u \mid \varphi$, or is existentially quantified and comes from v_i only. This property is an invariant preserved by the two inference rules.

Proofs always begin with a set $\mathcal{C}$ of formulas that we want to *simultaneously* prove, so that the proof effort only succeeds if *all* formulas in $\mathcal{C}$ are eventually proved. $\mathcal{C}$ contains the main properties we want to prove as well as any auxiliary lemmas that may be needed to carry out the proof. The initial set of goals we want to prove is $[\varnothing,\, \mathcal{C}] \;\vdash_T\; \mathcal{C}$, which is a shorthand for the set of goals $\{[\varnothing,\, \mathcal{C}] \;\vdash_T\; u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i \;\big|\; (u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i) \in \mathcal{C}\}$. Thus, we start *without any axioms* $\mathcal{A}$, but we shall be able to use the formulas in $\mathcal{C}$ as axioms in their own derivation *after* taking at least on step with the rewrite rules in $\mathcal{R}$.

A very useful key feature is that sequents $[\varnothing,\, \mathcal{C}] \;\vdash_T\; u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$, whose formulas $\mathcal{C}$ have been *postulated* (as the conjectures we want to prove) but not yet justified, are transformed by Step$^{\forall}$ + Subsumption into sequents of the form $[\mathcal{C},\, \varnothing] \;\vdash_T\; u' \mid \varphi' \longrightarrow^{\circledast} \bigvee_i v'_i \mid \psi'_i$, where now the formulas in $\mathcal{C}$ *can be assumed valid*, and can be used in derivations with the Axiom rule.

Verifying QLOCK's Mutual Exclusion. By Corollary 1, QLOCK's mutual exclusion can be verified by: (i) using pattern subsumption to check the trivial inclusion $[\![S_0]\!] \subseteq [\![P]\!]$, and (ii) proving $\mathcal{R}_{stop} \models^{\forall}_{[\,]} P\sigma \rightarrow^{\circledast} [P]$, where σ is a sort-preserving bijective renaming of variables such that $vars(P) \cap vars(P\sigma) = \varnothing$. But since for QLOCK P is a disjunction, in our inference system this means proving from $\mathcal{R}_{stop}$ that $[\varnothing, \mathcal{C}] \vdash_{[]} \mathcal{C}$, where $\mathcal{C}$ are the conjectures:

$$< n' \mid w' \mid i' \mid i' \;;\; q' > \mid \varphi' \rightarrow^{\circledast} [< n \mid w \mid i \mid i \;;\; q > \mid \varphi \vee < n \mid w \mid \varnothing \mid q > \mid \psi]$$

$$< n' \mid w' \mid \varnothing \mid q' > \mid \psi' \rightarrow^{\circledast} [< n \mid w \mid i \mid i \;;\; q > \mid \varphi \vee < n \mid w \mid \varnothing \mid q > \mid \psi].$$

where $\varphi = dupl(n\ w\ i) \neq tt$, $\psi = dupl(n\ w) \neq tt$, and φ', ψ' are their obvious renamings.

Before explaining the STEP$^{\forall}$ + SUBSUMPTION proof rule we introduce some notational conventions. Assume T is the pattern predicate $T = \bigvee_j t_j \mid \chi_j$, with $vars(\chi_j) \subseteq vars(t_j)$, and let $R = \{l_j \rightarrow r_j \text{ if } \phi_j\}_{j \in J}$, we then define:

$$\text{MATCH}(u,\ \{v_i\}_{i \in I}) \subseteq \{(i, \beta) \mid \beta \in [vars(v_i) \backslash vars(u) \rightarrow T_\Omega(X)] \text{ s.t. } u =_{E_\Omega \cup B_\Omega} v_i\beta\}$$

a *complete* set of (parameter-preserving) $E_\Omega \cup B_\Omega$-matches of u against the v_i,

$$\text{UNIFY}(u \mid \varphi',\ R) \equiv \{(j, \alpha) \mid \alpha \in Unif_{E_\Omega \cup B_\Omega}(u, l_j) \text{ and } (\varphi' \wedge \phi_j)\alpha \text{ satisfiable in } \mathcal{T}_{\Sigma/E \cup B}\}$$

a complete set of $E_\Omega \cup B_\Omega$-unifiers of a pattern $u \mid \varphi'$ with the lefthand-sides of the rules in R with satisfiable associated constraints.[6]

Consider now the rule:

STEP$^{\forall}$ + SUBSUMPTION

$$\frac{\bigwedge_{(j,\alpha) \in \text{UNIFY}(u \mid \varphi',\ R)} [\mathcal{A} \cup \mathcal{C},\ \varnothing]\ \vdash_T\ (r_j \mid \varphi' \wedge \phi_j)\alpha \longrightarrow^{\circledast} \bigvee_i (v_i \mid \psi_i)\alpha}{[\mathcal{A},\ \mathcal{C}]\ \vdash_T\ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i}$$

where $\varphi' \equiv \varphi \wedge \bigwedge_{(i,\beta) \in \text{MATCH}(u,\ \{v_i\})} \neg(\psi_i\beta)$. This inference rule allows us to take one step with the rules in $\mathcal{R}$. Intuitively, $u \mid \varphi'$ characterizes the states satisfying $u \mid \varphi$ that are not subsumed by any $v_i \mid \psi_i$; that is, states in the lefthand side of the current goal that have not yet reached the righthand side. Note that, according to Definition 6, $u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ is semantically valid iff

[6] In the current version of the tool (see Section 6), variant satisfiability makes constraint checking decidable. Future versions will only assume $\vec{E}$ convergent modulo B for the equational part $E \cup B$ of $\mathcal{R}$, so that satisfiability of such constraints will in general be undecidable. Unifiers whose associated constraints cannot be proved unsatisfiable will then be included in $\text{UNIFY}(u \mid \varphi',\ R)$ as a safe over-approximation. The same approach will apply to the, in general undecidable, checking of satisfiability/validity for other constraints involved in the application of the STEP$^{\forall}$ + SUBSUMPTION or AXIOM rules below: they will be either over-approximated, or will become explicit proof obligations to be discharged by an inductive theorem prover backend.

$u \mid \varphi' \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ is valid. Thus, this inference rule only unifies $u \mid \varphi'$ with the lefthand sides of rules in R. We impose on this inference rule a side condition that $\bigvee_{j,\gamma \in Unif_{E_\Omega \cup B_\Omega}(u,t_j)}(\varphi' \wedge \chi_j)\gamma$ is unsatisfiable in $\mathcal{T}_{\Sigma/E \cup B}$, where $T = \bigvee_j t_j \mid \chi_j$ is the pattern predicate characterizing the chosen T-terminating states. This condition ensures that any state in $u \mid \varphi'$ has an $\mathcal{R}$-successor. Thus, a state in $u \mid \varphi'$ reaches on all T-terminating paths a state in $\bigvee_i v_i \mid \psi_i$ if all its successors do so. Each $\mathcal{R}$-successor is covered by one of $(r_j \mid \varphi' \wedge \phi_j)\alpha$. As an optimization, we check that $(\varphi' \wedge \phi_j)\alpha$ is satisfiable and we drop the ones which are not. Finally, we also assume that $vars((u \mid \varphi)\alpha) \cap vars((\bigvee_i v_i \mid \psi_i)\alpha) = vars((r_j \mid \varphi' \wedge \phi_j)\alpha) \cap vars((\bigvee_i v_i \mid \psi_i)\alpha)$. This parameter preservation condition ensures correct implicit quantification. Note that formulas in $\mathcal{C}$ are added to $\mathcal{A}$, so that from now on they can be used by Axiom. By using $E_\Omega \cup B_\Omega$-unification, this inference rule is actually performing narrowing of $u \mid \varphi'$ with rules R [48].

Axiom

$$\frac{\bigwedge_j [\{u' \mid \varphi' \longrightarrow^{\circledast} \bigvee_j v'_j \mid \psi'_j\} \cup \mathcal{A},\ \varnothing] \ \vdash_T \ v'_j\alpha \mid \varphi \wedge \psi'_j\alpha \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i}{[\{u' \mid \varphi' \longrightarrow^{\circledast} \bigvee_j v'_j \mid \psi'_j\} \cup \mathcal{A},\ \varnothing] \ \vdash_T \ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i}$$

if $\exists \alpha$ such that $u =_{E_\Omega \cup B_\Omega} u'\alpha$ and $\mathcal{T}_{\Sigma/E \cup B} \models \varphi \Rightarrow \varphi'\alpha$. This inference rule allows us to use a trusted formula in $\mathcal{A}$ to summarize multiple transition steps. This is similar to how several transition steps would apply to a ground term, except that for ground terms we would check that $\varphi'\alpha$ is valid, whereas here we check that the condition φ implies $\varphi'\alpha$. Since φ is stronger than $\varphi'\alpha$, we add φ to $(v'_j \mid \psi'_j)\alpha$ (the result of using axiom $u' \mid \varphi' \longrightarrow^{\circledast} \bigvee_j v'_j \mid \psi'_j$). We assume that $u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ and $u' \mid \varphi' \longrightarrow^{\circledast} \bigvee_j v'_j \mid \psi'_j$ do not share variables, which can always be guaranteed by renaming. For correct implicit quantification, as in Step$^{\forall}$ + Subsumption, we assume for each j the parameter preservation condition $vars(u \mid \varphi) \cap vars(\bigvee_i v_i \mid \psi_i) = vars(v'_j\alpha \mid \varphi \wedge \psi'_j\alpha) \cap vars(\bigvee_i v_i \mid \psi_i)$. On a practical note, in order to be able to find the α, our implementation requires that $vars(\varphi') \subseteq vars(u')$, so that all the variables in $vars(\varphi')$ are matched.

The soundness of Step$^{\forall}$ + Subsumption plus Axiom is now the theorem:

Theorem 5. *(Soundness) Let $\mathcal{R}$ be a rewrite theory, and $\mathcal{C}$ a finite set of reachability formulas. If $\mathcal{R}$ proves $[\varnothing,\ \mathcal{C}] \ \vdash_T \ \mathcal{C}$ then $\mathcal{R} \models^{\forall}_T \mathcal{C}$.*

QLOCK Proof Details. Using our implementation of the proof system (see Section 6), we can extract traces of completed proofs to help understand how such proofs work in practice. Recall that for QLOCK we had to prove $[\varnothing, \mathcal{C}] \vdash_{[]} \mathcal{C}$, where $\mathcal{C}$ were the two already-discussed reachability formulas with respective preconditions the renamed disjuncts P'_i, $1 \leqslant i \leqslant 2$ in the invariant $P = P_1 \vee P_2$, and postcondition $[P]$, where $P_1 = < n \mid w \mid i \mid i \ ; \ q > \mid \varphi$ and $P_2 = < n \mid w \mid \varnothing \mid q > \mid \psi$. As an example, we examine a branch of the proof tree for the sequent $[\varnothing, \mathcal{C}] \vdash_{[]} P'_1 \rightarrow^{\circledast} [P]$, where the dots represent omitted proof branches.

$$\dfrac{\dfrac{\dfrac{\dfrac{\varnothing}{\cdots\ [\mathcal{C},\varnothing]\vdash_{[]}[n''|\ w''\mid\varnothing\mid q'']\,|\,\varphi'\wedge\psi''\to^{\circledast}[P]\ \cdots}\ \text{subsume}}{\cdots\ [\mathcal{C},\varnothing]\vdash_{[]}<n'\ i'\mid w'\mid\varnothing\mid q'>|\,\varphi'\to^{\circledast}[P]\ \cdots}\ \text{axiom: } P_2'\to^{\circledast}[P]}{[\varnothing,\mathcal{C}]\vdash_{[]}<n'\mid w'\mid i'\mid i\ ;\ q'>|\,\varphi'\to^{\circledast}[P]}\ \text{step: } c2n$$

5.1 The Split and Case Analysis Auxiliary Rules

The following Split rule is an auxiliary proof rule that uses a formula ϕ to split a goal into two. Split is a *validity-preserving* rule transforming a set $\mathcal{G}$ of reachability logic goals to be proved (understood as a *conjunction*) into a *semantically equivalent* set of goals $\mathcal{G}'$, so that $\mathcal{R}\models^{\forall}_{T}\mathcal{G}\ \Leftrightarrow\ \mathcal{R}\models^{\forall}_{T}\mathcal{G}'$. This means that Split *does not affect soundness.*

Split

$$\frac{[\mathcal{A},\ \mathcal{C}]\ \vdash_T\ u\mid\varphi\wedge\phi\longrightarrow^{\circledast}A\qquad[\mathcal{A},\ \mathcal{C}]\ \vdash_T\ u\mid\varphi\wedge\neg\phi\longrightarrow^{\circledast}A}{[\mathcal{A},\ \mathcal{C}]\ \vdash_T\ u\mid\varphi\longrightarrow^{\circledast}A}$$

subject to the condition $vars(u\mid\varphi)\cap vars(A)=vars(u\mid\varphi\wedge\phi)\cap vars(A)$.

Lemma 2. *In the above* Split *rule,* $\mathcal{R}\models^{\forall}_{T}\mathcal{G}\ \Leftrightarrow\ \mathcal{R}\models^{\forall}_{T}\mathcal{G}'$*, where* $\mathcal{G}$ *is the premise and* $\mathcal{G}'$ *the conclusion.*

This still leaves open the question of when it would be advantageous to use the Split rule and with what choice of ϕ. One attractive possibility is to use Split to increase success in application attempts for the Axiom rule. Suppose that we have tried to apply Axiom with a substitution α such that $u=_{E_\Omega\cup B_\Omega}u'\alpha$, but the condition $\mathcal{T}_{\Sigma/E\cup B}\models\varphi\Rightarrow(\varphi'\alpha)$ does not hold. Suppose, however, that $\varphi\wedge(\varphi'\alpha)$ is satisfiable in $\mathcal{T}_{\Sigma/E\cup B}$, and that $vars(u\mid\varphi)\cap vars(\bigvee_i v_i\mid\psi_i)=vars(u\mid\varphi\wedge(\varphi'\alpha))\cap vars(\bigvee_i v_i\mid\psi_i)$. In such a case, we can first apply Split to split $u\mid\varphi\to^{\circledast}\bigvee_i v_i\mid\psi_i$ into $u\mid\varphi\wedge(\varphi'\alpha)\to^{\circledast}\bigvee_i v_i\mid\psi_i$ and $u\mid\varphi\wedge\neg(\varphi'\alpha)\to^{\circledast}\bigvee_i v_i\mid\psi_i$, and then apply Axiom (checking parameter preservation) to the first of these two reachability formulas.

The second, also validity-preserving, auxiliary rule is a Case Analysis rule. It allows us to reason by cases by decomposing a variable $x{:}s$ of sort s into a complete covering of it by constructor patterns. Call $\{u_1,\ldots,u_k\}\subseteq T_\Omega(X)_s$ a *pattern set* for sort s iff $T_{\Omega,s}=\bigcup_{1\leqslant i\leqslant k}\{u_i\rho\mid\rho\in[X{\to}T_\Omega]\}$. We assume throughout that $i\neq i'\Rightarrow vars(u_i)\cap vars(u_{i'})=\varnothing$, and that all variables in the pattern set are *fresh* variables not appearing in any current goal.

Case Analysis

$$\frac{\bigwedge_{1\leqslant i\leqslant k}[\mathcal{A},\ \mathcal{C}]\ \vdash_T\ (u\mid\varphi)\{x{:}s\mapsto u_i\}\longrightarrow^{\circledast}A\{x{:}s\mapsto u_i\}}{[\mathcal{A},\ \mathcal{C}]\ \vdash_T\ u\mid\varphi\longrightarrow^{\circledast}A}$$

where $x{:}s\in vars(u)$ and $\{u_1,\ldots,u_k\}$ is a pattern set for s.

Lemma 3. *In the above* Case Analysis *rule,* $\mathcal{R}\models^{\forall}_{T}\mathcal{G}\ \Leftrightarrow\ \mathcal{R}\models^{\forall}_{T}\mathcal{G}'$*, where* $\mathcal{G}$ *is the premise and* $\mathcal{G}'$ *the conclusion.*

6 Prototype Implementation and Experiments

We have implemented the reachability logic proof system in Maude [8]. We exploit the fact that rewriting logic is reflective, so that concepts such as terms, rewrite rules, signatures, and theories are directly expressible as data in the logic. This is supported by Maude's `META-LEVEL` library. Our prototype tool takes as input (i) a reflected rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R, \phi)$ and (ii) a set of reachability formulas $\mathcal{C} = \{A_i \rightarrow^{\circledast} B_i\}_{i \in I}$ to be simultaneously proved.

The state of a reachability proof is represented as a set of proof sequents with associative-commutative union, as defined in Section 5, plus some global state information (for example, theory $\mathcal{R}$). Given goal set $\mathcal{C}$, the initial proof state will be $\left\{[\emptyset,\ \mathcal{C}]\ \vdash_T\ A_i \longrightarrow^{\circledast} B_i\right\}_{i \in I}$, that is, one sequent for each goal. Given the simplicity of the proof system, we need only perform a very simple proof search strategy: until there are no pending goals, first apply Axiom as much as possible and then apply $\text{Step}^{\forall}$ + Subsumption if possible.

We of course need to mechanize the two proof rules. Internally, each proof rule is represented by a corresponding metalevel rewrite rule. Rule application is controlled by a metadata flag that indicates which rule to apply next. Our implementation further requires a finitary B-unification algorithm as well as an SMT solver to discharge $E \cup B$ constraints. Maude can perform unification modulo commutativity and associativity/commutativity with or without identity and in some cases associativity without commutativity. For SMT solving we use the variant satisfiability techniques in [31,45], which allow us to handle any rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ satisfying topmost requirements (1)–(2) and such that the equational theory $(\Sigma, E \cup B)$ has a convergent decomposition satisfying the finite variant property [12] and protects a constructor subtheory which we assume consists only of axioms B_Ω of the above-described kind. Note that this means that both validity and satisfiability of QF formulas in the initial algebra $\mathcal{T}_{\Sigma/E \cup B}$ are decidable [31]. Future versions of the tool will add other decision procedures and will support more general classes of rewrite theories $\mathcal{R} = (\Sigma, E \cup B, R)$. Furthermore, besides further automation, an inductive theorem prover backend will be added to reason about validity of formulas in $\mathcal{T}_{\Sigma/E \cup B}$.

In addition to the issue of proof representation, several other issues must be addressed. First, to ensure correct applications of unification, we uniquely rename all variables in rules in the theory $\mathcal{R}$ and in goals $\mathcal{C}$. Second, recall that we assume that the rewrite theory $\mathcal{R}$ has been Ω-abstracted as $\hat{\mathcal{R}}$. Therefore, we have implemented a method that can Ω-abstract many theories $\mathcal{R}$ in practice. Third, an important practical consideration during any tool development is a user interface which is flexible and usable enough to express real theories and problems that users may wish to reason about. To that end, we have developed a `FULL-MAUDE` based user-interface[15] in Maude that provides commands to input goals and invariants, solve pattern predicate subsumption/intersection queries, and specify theories plus the corresponding terminating state pattern predicates of interest. The full command grammar is given in Appendix B.

To validate the feasibility of our approach we have verified properties for a collection of examples including various rewrite theories specifying distributed

systems such as communication or mutual exclusion protocols and cyber-physical systems. Table 1 summarizes these experiments.

Table 1. Examples Verified by the Tool

Example	Description of the System/Property
Choice	Nondeterministically throws away elements from a multiset/eventually only one element left
Comm. Protocol	Fault-tolerant communication protocol/packets are eventually delivered in-order
Dijkstra	Dijkstra's classic mutual exclusion alg./mutual exclusion
Fixed-Size Token Ring	2-Token ring mutual exclusion alg./mutual exclusion
QLOCK	QLOCK mutual exclusion alg./mutual exclusion
Readers/Writers	Mutual exclusion alg. for readers-writers/mutual exclusion
Lamport's Bakery	Unbounded Lamport's bakery/mutual exclusion
Thermostat	Open system that dynamically responds to temperature data/temperature stays within given bounds

7 Related Work and Conclusions

Reachability logic [43,42,46,47] is a language-generic approach to program verification, parametric on the operational semantics of a programming language. Both Hoare logic and separation logic can be naturally mapped into reachability logic [43,42]. This work extends reachability logic from a programming-language-generic logic of programs to a rewrite-theory-generic logic to reason about *both* distributed system designs and programs, based on their rewriting logic semantics. This extension is non-trivial and requires a number of new concepts and results, including: (i) relativization of terminating sequences to a chosen subset $[\![T]\!]$ of terminating states; (ii) solving the "invariant paradox," to reason about invariants and co-invariants of possibly non-terminating systems, and characterizing such invariants by means of reachability formulas through a theory transformation; and (iii) making it possible to achieve higher levels of automation by systematically basing the state predicates on positive Boolean combination of constrained constructor patterns of the form $u \mid \varphi$ with u a constructor term.

In contrast, standard reachability logic [46,47] uses matching logic, which assumes a first-order model $\mathcal{M}$ and its satisfaction relation $\mathcal{M} \models \varphi$ as the basis of the reachability logic proof system, and further assumes a matching-logic-definable transition relation on $\mathcal{M}$. As discusses in Section 3, we choose $T_{\Sigma/E \cup B}$ as the model and $\rightarrow_{\mathcal{R}}$ for transitions, rather than some general $\mathcal{M}$ with definable transitions, and systematically exploit the isomorphism $T_{\Sigma/E \cup B}|_{\Omega} \cong T_{\Omega/E_{\Omega} \cup B_{\Omega}}$, allowing us to use unification, matching, narrowing, and satisfiability procedures based on the typically much simpler initial algebra of constructors $T_{\Omega/E_{\Omega} \cup B_{\Omega}}$. This has the advantage that we can explicitly give the complete details of our

inference rules (e.g. how STEP$^{\forall}$ + SUBSUMPTION checks the subsumption, or ensures that states have at least a successor), instead of relying on a general satisfaction relation $\models$ on some $\mathcal{M}$ with definable transitions. The result is a simpler inference system with only two rules (instead of the eight in reachability logic). On the practical side, reachability logic has been previously implemented as part of the $\mathbb{K}$ framework, and has only been instantiated with operational semantics of programming languages and used for the purpose of program verification. In particular, the implementation in $\mathbb{K}$ has several hand-crafted heuristics for reasoning about specific features of programming language, such as dynamically allocated memory (the "heap"). In spite of the fact that similar heuristics have not yet been added to the current prototype described in Section 6, the potential for automation of the constructor-based reachability logic approach has been demonstrated by the tool's capacity to prove safety properties for a representative suite of distributed system designs, including various communication protocols, mutual exclusion protocols, and real-time systems. Of course, this is just a proof of concept: adding reasoning heuristics and further theorem proving support will be crucial to handle a much wider range of applications.

As mentioned in the Introduction, we have been inspired by the work in [28]. We agree on the common goal of making reachability logic rewrite-theory-generic, but differ on the methods used and their applicability. Main differences include: (1) the authors in [28] do not give an inference system but a verification algorithm manipulating goals, which makes it hard to compare both logics. (2) the theories to which the methods in [28] apply seem more restricted than the ones presented here. Roughly, (see their **Assumption 3**) [28] assumes restrictions akin to those imposed in [40] to allow "rewriting modulo SMT," which limits the equational theories (Σ, E) that can be handled. (3) Matching is used throughout in [28] instead of unification. This means that, unless a formula has been sufficiently instantiated, no matching rule may exist, whereas unification with some rule is always possible in our case. (4) No method for proving invariants is given in [28]; solving the "invariant paradox" provides such a method.

Two recent further developments that add coinductive reasoning capabilities to reachability logic are also worth mentioning, namely, Moore's Ph.D. dissertation [37], and the coinductive approach by Lucanu et al. in [27]. Investigating how the approach in this paper can be related to such coinductive approaches seems an interesting topic for future research.

Finally, there is a close connection between this work and various rewriting-based symbolic methods, including: (i) unification modulo FVP theories [17]; (ii) decidable satisfiability (and validity) of quantifier-free formulas in initial algebras [30,9,10,2,11,36,20,21,18,1,31]; (iii) narrowing-based reachability analysis [48]; (iv) narrowing-based proof of safety properties [38,39]; (v) patterns and constrained-based conditional narrowing [36,6]; and (vi) rewriting modulo SMT [40]. Exploiting such connections, particularly with [17,31,45], has been essential to achieve the goals of this work.

In conclusion, this work advances the goal of making reachability logic available as a rewrite-theory-generic verification logic. The goals of wide applicability,

invariant verification, simplicity, and mechanization of inference rules have been substantially advanced, but much work remains ahead. The feasibility of the approach has been validated with a prototype implementation using a suite of representative examples; but building a robust and highly effective reachability logic tool for rewrite theories specifying both distributed systems and programming languages is a considerably more ambitious goal. This, together with experimentation with a wider class of examples and case studies, will be the main focus of future research.

Acknowledgments. We thank Grigore Roşu, Dorel Lucanu and Vlad Rusu for their very helpful comments on an earlier draft of this work. This research has been partially supported by NSF Grants CNS 13-19109 and CNS 14-09416, and AFOSR Contract FA8750-11-2-0084.

References

1. Aoto, T., Stratulat, S.: Decision procedures for proving inductive theorems without induction. In: Proc. PPDP2014. pp. 237–248. ACM (2014)
2. Baader, F., Schulz, K.U.: Combination techniques and decision problems for disunification. Theor. Comput. Sci. 142(2), 229–255 (1995)
3. Baader, F., Siekmann, J.H.: Unification theory. In: Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 2, pp. 41–126. Oxford University Press (1994)
4. Baader, F., Snyder, W.: Unification theory. In: Handbook of Automated Reasoning (in 2 volumes), pp. 445–532. Elsevier and MIT Press (2001)
5. Bruni, R., Meseguer, J.: Semantic foundations for generalized rewrite theories. Theor. Comput. Sci. 360(1-3), 386–414 (2006)
6. Cholewa, A., Escobar, S., Meseguer, J.: Constrained narrowing for conditional equational theories modulo axioms. Science of Computer Programming 112, 24–57 (2015)
7. Cholewa, A., Meseguer, J., Escobar, S.: Variants of variants and the finite variant property. Tech. rep., CS Dept. University of Illinois at Urbana-Champaign (February 2014), available at `http://hdl.handle.net/2142/47117`
8. Clavel, M., Durán, F., Eker, S., Meseguer, J., Lincoln, P., Martí-Oliet, N., Talcott, C.: All About Maude – A High-Performance Logical Framework. Springer LNCS Vol. 4350 (2007)
9. Comon, H., Lescanne, P.: Equational problems and disunification. Journal of Symbolic Computation 7, 371–425 (1989)
10. Comon, H.: Complete axiomatizations of some quotient term algebras. Theor. Comput. Sci. 118(2), 167–191 (1993)
11. Comon, H., Delor, C.: Equational formulae with membership constraints. Inf. Comput. 112(2), 167–216 (1994)
12. Comon-Lundth, H., Delaune, S.: The finite variant property: how to get rid of some algebraic properties, in Proc *RTA'05*, Springer LNCS 3467, 294–307, 2005
13. Dershowitz, N., Jouannaud, J.P.: Rewrite systems. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Vol. B, pp. 243–320. North-Holland (1990)
14. Durán, F., Meseguer, J.: On the church-rosser and coherence properties of conditional order-sorted rewrite theories. J. Log. Algebr. Program. 81(7-8), 816–850 (2012)

15. Durán, F., Ölveczky, P.C.: A guide to extending full maude illustrated with the implementation of real-time maude. Electronic Notes in Theoretical Computer Science 238(3), 83 – 102 (2009)
16. Ehrig, H., Mahr, B.: Fundamentals of Algebraic Specification 1. Springer (1985)
17. Escobar, S., Sasse, R., Meseguer, J.: Folding variant narrowing and optimal variant termination. J. Algebraic and Logic Programming 81, 898–928 (2012)
18. Falke, S., Kapur, D.: Rewriting induction + linear arithmetic = decision procedure. In: Proc. IJCAR 2012. vol. 7364, pp. 241–255. Springer LNCS (2012)
19. Futatsugi, K.: Fostering proof scores in CafeOBJ. In: Proc. ICFEM 2010. vol. 6447, pp. 1–20. Springer LNCS (2010)
20. Giesl, J., Kapur, D.: Decidable classes of inductive theorems. In: Proc. IJCAR 2001. vol. 2083, pp. 469–484. Springer LNCS (2001)
21. Giesl, J., Kapur, D.: Deciding inductive validity of equations. In: Proc. CADE 2003. vol. 2741, pp. 17–31. Springer LNCS (2003)
22. Goguen, J., Meseguer, J.: Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. Theoretical Computer Science 105, 217–273 (1992)
23. Gutiérrez, R., Meseguer, J., Rocha, C.: Order-sorted equality enrichments modulo axioms. Sci. Comput. Program. 99, 235–261 (2015)
24. Hullot, J.M.: Canonical forms and unification. In: Bibel, W., Kowalski, R. (eds.) Proceedings, Fifth Conference on Automated Deduction, pp. 318–334. Springer-Verlag (1980), lNCS, Volume 87
25. Jouannaud, J.P., Kirchner, C., Kirchner, H.: Incremental construction of unification algorithms in equational theories. In: Proc. ICALP'83. pp. 361–373. Springer LNCS 154 (1983)
26. Jouannaud, J.P., Kirchner, C.: Solving equations in abstract algebras: A rule-based survey of unification. In: Computational Logic - Essays in Honor of Alan Robinson. pp. 257–321. MIT Press (1991)
27. Lucanu, D., Rusu, V., Arusoaie, A.: A generic framework for symbolic execution: A coinductive approach. J. Symb. Comput. 80, 125–163 (2017)
28. Lucanu, D., Rusu, V., Arusoaie, A., Nowak, D.: Verifying reachability-logic properties on rewriting-logic specifications. In: Logic, Rewriting, and Concurrency - Essays dedicated to José Meseguer on the Occasion of His 65th Birthday. vol. 9200, pp. 451–474. Springer LNCS (2015)
29. Lucas, S., Meseguer, J.: Normal forms and normal theories in conditional rewriting. J. Log. Algebr. Meth. Program. 85(1), 67–97 (2016)
30. Maher, M.J.: Complete axiomatizations of the algebras of finite, rational and infinite trees. In: Proc. LICS '88. pp. 348–357. IEEE Computer Society (1988)
31. Meseguer, J.: Variant-based satisfiability in initial algebras. In: Artho, C., Ölveczky, P. (eds.) Proc. FTSCS 2015. pp. 1–32. Springer CCIS 596 (2016)
32. Meseguer, J.: Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science 96(1), 73–155 (1992)
33. Meseguer, J.: Membership algebra as a logical framework for equational specification. In: Proc. WADT'97. pp. 18–61. Springer LNCS 1376 (1998)
34. Meseguer, J.: Twenty years of rewriting logic. J. Algebraic and Logic Programming 81, 721–781 (2012)
35. Meseguer, J., Rosu, G.: The rewriting logic semantics project: A progress report. Inf. Comput. 231, 38–69 (2013)
36. Meseguer, J., Skeirik, S.: Equational formulas and pattern operations in initial order-sorted algebras. In: Falaschi, M. (ed.) Proc. LOPSTR 2015. vol. 9527, pp. 36–53. Springer LNCS (2015)

37. Moore, B.: Coinductive Program Verification. Ph.D. thesis, University of Illinois at Urbana-Champaign (2016), http://hdl.handle.net/2142/95372
38. Rocha, C., Meseguer, J.: Proving safety properties of rewrite theories (2011), in Proc. CALCO 2011, Springer LNCS 6859, 314-328
39. Rocha, C., Meseguer, J.: 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. Lecture Notes in Computer Science, vol. 8373, pp. 603–629. Springer (2014)
40. Rocha, C., Meseguer, J., Muñoz, C.A.: Rewriting modulo SMT and open system analysis. Journal of Logic and Algebraic Methods in Programming 86, 269–297 (2017)
41. Rosu, G., Serbanuta, T.: An overview of the K semantic framework. J. Log. Algebr. Program. 79(6), 397–434 (2010)
42. Rosu, G., Stefanescu, A.: Checking reachability using matching logic. In: Proc. OOPSLA 2012. pp. 555–574. ACM (2012)
43. Rosu, G., Stefanescu, A.: From Hoare logic to matching logic reachability. In: Giannakopoulou, D., Méry, D. (eds.) FM. Lecture Notes in Computer Science, vol. 7436, pp. 387–402. Springer (2012)
44. Siekmann, J.H.: Unification theory. J. Symb. Comput. 7(3/4), 207–274 (1989)
45. Skeirik, S., Meseguer, J.: Metalevel algorithms for variant-based satisfiability. In: Lucanu, D. (ed.) Proc. WRLA 2016. vol. 9942, pp. 167–184. Springer LNCS (2016)
46. Stefanescu, A., Ştefan Ciobâcă, Mereuta, R., Moore, B.M., Serbanuta, T., Rosu, G.: All-path reachability logic. In: Proc. RTA-TLCA 2014. vol. 8560, pp. 425–440. Springer LNCS (2014)
47. Stefanescu, A., Park, D., Yuwen, S., Li, Y., Rosu, G.: Semantics-based program verifiers for all languages. In: Proc. OOPSLA 2016. pp. 74–91. ACM (2016)
48. Thati, P., Meseguer, J.: Symbolic reachability analysis using narrowing and its application to the verification of cryptographic protocols. J. Higher-Order and Symbolic Computation 20(1–2), 123–160 (2007)

A Proofs of Lemmas and Theorems

Proof of Lemma 1.

Proof. If $[u] \to_{\mathcal{R}} [v]$ corresponds to the topmost R, B-rewrite $u \to_{R,B} u'$, performed with a rewrite rule $l \to r$ *if* $\phi \in R$ and a ground substitution $\sigma \in [Y \to T_\Sigma]$, with Y the rule's variables, and such that $u =_{B_\Omega} l\sigma$, $u' = r\sigma$, and $[u'!] = [v]$, this is also a rewrite with the rule $l \to r'$ *if* $\phi \wedge \hat{\theta}$, by extending σ to the fresh variables $X_P = \{x_p \mid p \in P\}$ with the assignments $x_p \mapsto (r\sigma)|_p$, so that we have $[u] \to_{\hat{\mathcal{R}}} [v]$.

Conversely, if $[u] \to_{\hat{\mathcal{R}}} [v]$ corresponds to the topmost $\hat{R}, B$-rewrite $u \to_{R,B} w$, performed with rewrite rule $l \to r'$ *if* $\phi \wedge \hat{\theta}$ in $\hat{R}$ and ground substitution $\rho \in [Y \uplus X_P \to T_\Sigma]$, so that $w = r'\rho$ and $[w!] = [v]$, then we can perform a corresponding rewrite with rule $l \to r$ *if* $\phi \in R$ and substitution $\rho|_Y$, because $E \cup B \models \phi\rho$. Furthermore, since $E \cup B \models \hat{\theta}\rho$, we must have $[w!] = [(r\rho)!] = [v]$, so that $[u] \to_{\mathcal{R}} [v]$. □

Proof of Theorem 3.

Proof. A state $[\langle u_1, \ldots, u_n \rangle] \in \mathcal{C}_{\mathcal{R},State}$ is reachable from $[\![S_0]\!]$ iff $[[u_1, \ldots, u_n]]$ is reachable from $[\![S_0]\!]$ in $\mathcal{C}_{\mathcal{R}_{stop}}$. Therefore, $[\![P]\!]$ is an invariant of $(\mathcal{C}_{\mathcal{R},State}, \rightarrow_{\mathcal{R}})$ from $[\![S_0]\!]$ iff $\mathcal{R}_{stop} \models^{\forall}_{[]} S_0 \rightarrow^{\circledast} [P]$. □

Proof of Corollary 1.

Proof. Suppose (i) and (ii) hold. Then, since $S_0 \subseteq_Y P$, for each $\rho \in [Y \rightarrow T_\Omega]$ we have $[\![S_0\rho]\!] \subseteq [\![P\rho]\!]$, and, of course, since $P\rho$ is a variable renaming of $P\sigma\rho$, we also have $[\![P\rho]\!] = [\![P\sigma\rho]\!]$. But by Theorem 3 this shows that $[\![P\sigma\rho]\!] = [\![P\rho]\!]$ is an invariant of $(\mathcal{C}_{\mathcal{R},State}, \rightarrow_{\mathcal{R}})$ from $[\![P\rho]\!]$ and, *a fortiori*, from $[\![S_0\rho]\!]$. □

Proof of Theorem 4.

Proof. Suppose $[\![Q]\!]$ is not a co-invariant from $[\![S_0]\!]$. This exactly means that there are $[u] \in [\![S_0]\!]$ and $[v] \in [\![Q]\!]$ such that $[u] \rightarrow^*_{\mathcal{R}} [v]$, or, equivalently, $[v] \rightarrow^*_{\mathcal{R}^{-1}} [u]$. But since, by (ii) and Theorem 3, $[\![Q\sigma]\!]$ is an invariant of $(\mathcal{C}_{\mathcal{R}^{-1},State}, \rightarrow_{\mathcal{R}^{-1}})$ from $[\![Q]\!] = [\![Q\sigma]\!]$, we must have $[u] \in [\![Q\sigma]\!]$, which is impossible by (i). □

Proof of Theorem 5.

Proof. We begin by introducing the following auxiliary notation

Definition 8. *Let $u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ be a reachability formula. By definition, $\mathcal{R} \models^{\forall,n}_T u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ iff for each $[u_0] = [u\rho!] \in [\![u \mid \varphi]\!]$ and for each T-terminating sequence $[u_0] \rightarrow_{\mathcal{R}} [u_1] \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} [u_k]$ with $k \leqslant n$, there exist $0 \leqslant j \leqslant k$, i, and $[w] = [v_i\tau!] \in [\![v_i \mid \psi_i]\!]$ such that $[u_j] = [w]$ and $\rho|_{Y \cap Z} =_{E_\Omega \cup B_\Omega} \tau|_{Y \cap Z}$, where $Y = vars(u \mid \varphi)$ and $Z = vars(v_i \mid \psi_i)$. By convention, $\mathcal{R} \models^{\forall,-1}_T u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ always holds.*

With this notation, we state the following auxiliary lemma:

Lemma 4. *Let $[\mathcal{A},\ \mathcal{C}]\ \vdash_T\ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ be a sequent derived by our inference system for $\mathcal{R}$. If $\mathcal{R} \models^{\forall,n}_T \mathcal{A}$ and $\mathcal{R} \models^{\forall,n-1}_T \mathcal{C}$, then $\mathcal{R} \models^{\forall,n}_T u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$.*

Proof. We prove the lemma by contradiction. Assume it does not hold, and let n_{min} be the minimal n for which the lemma does not hold. Further, let $[\mathcal{A},\ \mathcal{C}]\ \vdash_T\ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ be a sequent with a minimal proof tree $\mathcal{P}$ for which the lemma does not hold for n_{min}. Thus, the lemma holds for any $n < n_{min}$, and for any sequent derived by a sub-proof of $\mathcal{P}$ for $n \leqslant n_{min}$. Finally, let $[u_0] = [u\rho!] \in [\![u \mid \varphi]\!]$ and $[u_0] \rightarrow_{\mathcal{R}} [u_1] \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} [u_n]$ a T-terminating path with $n \leqslant n_{min}$. Next, we show that there exist $0 \leqslant j \leqslant n$ and $[w] = [v_i\tau!] \in [\![v_i \mid \psi_i]\!]$ such that $[u_j] = [w]$ and ρ and τ agree modulo $E_\Omega \cup B_\Omega$ on $vars(u \mid \varphi) \cap vars(v_i \mid \psi_i)$, which is a contradiction and completes the proof.

We distinguish the following cases according to the last proof rule applied.

STEP$^{\forall}$ + SUBSUMPTION. First, notice that

$$\varphi \Leftrightarrow (\varphi \wedge \bigvee_{(i,\beta)\in \text{MATCH}(u,\ \{v_i\})} \psi_i\beta) \vee \varphi'$$

If ρ satisfies the first part of the disjunction, then it follows that ρ satisfies $\psi_i\beta$ for some i and β. Since β is a matching substitution with domain $vars(v_i)\backslash vars(u)$, we can pick any τ extending $\rho|_{vars}(u \mid \varphi) \uplus \beta\rho$, and we have that τ satisfies ψ_i. Further, since $u_0 =_{E_\Omega \cup B_\Omega} u\rho$ and $u =_{E_\Omega \cup B_\Omega} v_i\beta$, we have that $u_0 =_{E_\Omega \cup B_\Omega} v_i\tau$. Thus, $[u_0] = [v_i\tau!] \in [\![v_i \mid \psi_i]\!]$, and in this case we are done. In the second case, it must be the case that $[u_0] = [u\rho!] \in [\![u \mid \varphi']\!]$.

We notice that $[u_0] \to_{\mathcal{R}} [u_1]$ by rule $l_j \to r_j$ if $\phi_j \in \mathcal{R}$ iff there exist a ground substitution τ such that $u_0 =_{E_\Omega \cup B_\Omega} l_i\tau$ and $E \cup B \models \phi_j\tau$, or equivalently, iff $[u_0] = [l_j\tau!] \in [\![l_j \mid \phi_j]\!]$. Since $u_0 =_{E_\Omega \cup B_\Omega} u\rho$ and $\rho|_{vars(u|\varphi')} \cap \tau|_{vars(l_j|\phi_j)} = \varnothing$, it follows that $[u_0] = [l_j\tau!] \in [\![l_j \mid \phi_j]\!]$ iff there exist $\alpha \in Unif_{E_\Omega \cup B_\Omega}(u, l_j)$ and a ground substitution θ such that $\alpha\theta$ agrees with ρ on $vars(u \mid \varphi')$ and with τ on $vars(l_j \mid \phi_j)$ modulo $E_\Omega \cup B_\Omega$. Then $u_0 =_{E_\Omega \cup B_\Omega} u\alpha\theta =_{E_\Omega \cup B_\Omega} l_j\alpha\theta =_{E_\Omega \cup B_\Omega} l_j\tau$, and, likewise, $r_j\tau =_{E_\Omega \cup B_\Omega} r_j\alpha\theta$. Since $[u_1] = [r_j\alpha\theta!]$, it follows that $[u_1] = [r_j\alpha\theta!] \in [\![r_j \mid \varphi' \wedge \phi_j]\!]$. Therefore, we can conclude that for any $[u_0] \to_{\mathcal{R}} [u_1]$, there exist some $(j, \alpha) \in \text{UNIFY}(u \mid \varphi',\ \mathcal{R})$ and some θ such that $[u_1] = [r_j\alpha\theta!] \in [\![(r_j \mid \varphi' \wedge \phi_j)\alpha]\!]$.

Recall that we assume that $[u_0]$ is T-terminating iff $[u_0] = [t_j\tau!] \in [\![t_j \mid \chi_j]\!]$ for some j. Similar to the above reasoning, that holds iff there exists some $\gamma \in Unif_{E_\Omega \cup B_\Omega}(u, t_j)$ and a ground substitution θ such that $\gamma\theta$ agrees with ρ on $vars(u \mid \varphi')$ and with τ on $vars(t_j \mid \chi_j)$ modulo $E_\Omega \cup B_\Omega$. It would then follow that θ satisfies $(\varphi' \wedge \chi_j)\gamma$, which contradicts the side-condition of STEP$^{\forall}$+SUBSUMPTION, which states that $(\varphi' \wedge \chi_j)\gamma$ are unsatisfiable in $\mathcal{T}_{\Sigma/E\cup B}$. Therefore, we can assume that $[u_0]$ has a $\mathcal{R}$-successor and that $n_{min} \geqslant 1$.

Since $\mathcal{R} \models_T^{\forall, n_{min}} \mathcal{A}$ and $\mathcal{R} \models_T^{\forall, n_{min}-1} \mathcal{C}$, it follows that $\mathcal{R} \models_T^{\forall, n_{min}-1} \mathcal{A} \cup \mathcal{C}$. The lemma holds for $n_{min} - 1$, thus $\mathcal{R} \models_T^{\forall, n_{min}-1} (r_j \mid \varphi' \wedge \phi_j)\alpha \longrightarrow^{\circledast} \bigvee_i (v_i \mid \psi_i)\alpha$. Since $[u_1] \to_{\mathcal{R}} [u_1] \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} [u_n]$ is a T-terminating path of length at most $n_{min} - 1$, we have that there exist i and $1 \leqslant k \leqslant n$ and τ such that $[u_k] = [v_j\alpha\tau!] \in [\![(v_i \mid \psi_i)\alpha]\!]$ and, modulo $E_\Omega \cup B_\Omega$, τ agrees with θ on $vars((r_j \mid \varphi' \wedge \phi_j)\alpha) \cap vars((v_i \mid \psi_i)\alpha)$. It follows that $[u_k] = [v_j\alpha\tau!] \in [\![v_i \mid \psi_i]\!]$. Moreover, since STEP$^{\forall}$ + SUBSUMPTION guarantees that $vars((vars(u \mid \varphi) \cap vars(\bigvee_i v_i \mid \psi_i))\alpha) = vars((r_j \mid \varphi' \wedge \phi_j)\alpha) \cap vars(\bigvee_i (v_i \mid \psi_i)\alpha)$ and we have that τ agrees with θ on $vars((vars(u \mid \varphi) \cap vars(v_i \mid \psi_i))\alpha)$. We can conclude that, modulo $E_\Omega \cup B_\Omega$, $\alpha\theta$ agrees with $\alpha\tau$ on $vars(u \mid \varphi) \cap vars(v_i \mid \psi_i)$, and we are done.

AXIOM. Since $u =_{E_\Omega \cup B_\Omega} u'\alpha$ and $u_0 =_{E_\Omega \cup B_\Omega} u\rho$ we have that $u_0 =_{E_\Omega \cup B_\Omega} u'\alpha\rho$. Further, since $\mathcal{T}_{\Sigma/E\cup B} \models \varphi \Rightarrow \varphi'\alpha$ and $E \cup B \models \varphi\rho$, we have that $E \cup B \models \varphi'\alpha\rho$. Thus, $[u_0] = [u'\alpha\rho!] \in [\![u' \mid \varphi']\!]$. Since $\mathcal{R} \models_T^{\forall, n_{min}} u' \mid \varphi' \longrightarrow^{\circledast} \bigvee_j v'_j \mid \psi'_j$, there exists j and $0 \leqslant k \leqslant n$ and θ such that $[u_k] = [v'_j\alpha\theta!] \in [\![v'_j \mid \psi'_j]\!]$ and $\alpha\rho$ and $\alpha\theta$ agree on $vars(u' \mid \varphi') \cap vars(v'_j \mid \psi'_j)$ modulo $E_\Omega \cup B_\Omega$. Since $E \cup B \models \varphi\rho$, we can conclude that $[u_k] = [v'_j\alpha\theta!] \in [\![v'_j\alpha \mid \varphi \wedge \psi'_j\alpha]\!]$. We also have that ρ and θ agree on $vars((vars(u' \mid \varphi') \cap vars(v'_j \mid \psi'_j))\alpha)$ modulo $E_\Omega \cup B_\Omega$. Since the

proof tree deriving $v'_j\alpha \mid \varphi \wedge \psi'_j\alpha \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ is a subproof of $\mathcal{P}$, the lemma must hold. Thus, we have that there exists i and $k \leqslant m \leqslant n$ and τ such that $[u_m] = [v_i\tau!] \in [\![v_i \mid \psi_i]\!]$ and τ agrees with θ on $vars(v'_j\alpha \mid \varphi \wedge \psi'_j\alpha) \cap vars(v_i \mid \psi_i)$. Moreover, since Axiom guarantees that $vars(u \mid \varphi) \cap vars(\bigvee_i v_i \mid \psi_i) = vars(v'_j\alpha \mid \varphi \wedge \psi'_j\alpha) \cap vars(\bigvee_i v_i \mid \psi_i)$, we have that, modulo $E_\Omega \cup B_\Omega$, ρ agrees with θ on $vars(u \mid \varphi) \cap vars(v_i \mid \psi_i)$ and θ agrees with τ on $vars(u \mid \varphi) \cap vars(v_i \mid \psi_i)$, and we are done. □

Now we prove the main result (Theorem 5) using Lemma 4. Indeed, assume by contradiction that the theorem does not hold. Then, there must be a formula $u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i \in \mathcal{C}$ such that $[\varnothing,\ \mathcal{C}] \ \vdash_T \ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ is derived for $\mathcal{R}$ by our inference system, but $\mathcal{R} \not\models u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i \in \mathcal{C}$. Further, let n_{min} be the minimal natural number n for which $\mathcal{R} \not\models^{\forall}_n u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i \in \mathcal{C}$. Then $\mathcal{R} \models^{\forall, n_{min}-1}_T \mathcal{C}$ (recall that if $n_{min} = 0$, then $\mathcal{R} \models^{\forall,-1}_T \mathcal{C}$ holds by convention). Thus, by Lemma 4, we have that $\mathcal{R} \models^{\forall, n_{min}}_T \mathcal{C}$, and in particular, that $\mathcal{R} \models^{\forall, n_{min}}_T \varphi \longrightarrow^{\circledast} \bigvee_i \psi_i$. This is a contradiction with the definition of n_{min}, and it completes the proof. □

Proof of Lemma 2

Proof. Since φ is logically equivalent to $(\varphi \wedge \phi) \vee (\varphi \wedge \neg\phi)$ we have $[\![u \mid \varphi]\!] = [\![u \mid \varphi \wedge \phi]\!] \cup [\![u \mid \varphi \wedge \neg\phi]\!]$. The lemma then follows easily from Definition 6, using the parameter preservation condition. □

Proof of Lemma 3

Proof. Let Y be the parameters in $[\mathcal{A},\ \mathcal{C}] \ \vdash_T \ u \mid \varphi \longrightarrow^{\circledast} A$. We have two cases. (1) If $x : s \notin Y$, then $A\{x : s \mapsto u_i\} = A$, $1 \leqslant i \leqslant k$, and the result just follows from: (i) the parameters Y being the same in $[\mathcal{A},\ \mathcal{C}] \ \vdash_T \ u \mid \varphi \longrightarrow^{\circledast} A$ and in its k instances in the premise, and (ii) $[\![u \mid \varphi]\!] = \bigcup_{1 \leqslant i \leqslant k} [\![(u \mid \varphi)\{x : s \mapsto u_i\}]\!]$. (2) If $x : s \in Y$, then the parameters of each $[\mathcal{A},\ \mathcal{C}] \ \vdash_T \ (u \mid \varphi)\{x{:}s \mapsto u_i\} \longrightarrow^{\circledast} A\{x{:}s \mapsto u_i\}$ are $(Y - \{x : s\}) \cup vars(u_i)$. Observe that, by the definition of pattern set for s, $[Y \to T_\Omega] = \bigcup_{1 \leqslant i \leqslant k} \{\{x : s \mapsto u_i\}\tau_i \mid \tau_i \in [(Y - \{x{:}s\}) \cup vars(u_i) \to T_\Omega]\}$. Therefore, $\mathcal{R} \models^{\forall}_T [\mathcal{A},\ \mathcal{C}] \ \vdash_T \ u \mid \varphi \longrightarrow^{\circledast} A$ iff $\forall\ \rho \in [Y \to T_\Omega]\ \mathcal{R} \models^{\forall}_T ([\mathcal{A},\ \mathcal{C}] \ \vdash_T \ u \mid \varphi \longrightarrow^{\circledast} A)\rho$ iff $(\forall\ i,\ 1 \leqslant i \leqslant k)\ (\forall\ \tau_i \in [(Y - \{x{:}s\}) \cup vars(u_i) \to T_\Omega])\ \mathcal{R} \models^{\forall}_T ([\mathcal{A},\ \mathcal{C}] \ \vdash_T \ u \mid \varphi \longrightarrow^{\circledast} A)\{x{:}s \mapsto u_i\}\tau_i$ iff $\bigwedge_{1 \leqslant i \leqslant k} [\mathcal{A},\ \mathcal{C}] \ \vdash_T \ (u \mid \varphi)\{x{:}s \mapsto u_i\} \longrightarrow^{\circledast} A\{x{:}s \mapsto u_i\}$, as desired. □

B Command Grammar

Here we provide a BNF grammar of the commands which can be given as inputs to our prototype Maude tool. In the grammar below, **boldface words** represent themselves (i.e. terminals) while ⟨words in angle brackets⟩ represent non-terminals. A nonterminal surrounded by square brackets, e.g., [⟨number⟩], represents an optional argument. BNF grammar alternatives are separated by vertical bars (|). Whenever we use a reserved symbol as a terminal, we surround

it in double quotes, e.g. "`|`". The horizontal lines delimit the four basic categories of commands: (i) proof setup, (ii) adding invariants, (iii) adding goals, and (iv) applying proof steps or simple proof strategies.

⟨outer-cmd⟩ ::= **(** ⟨inner-cmd⟩ **.)**
⟨inner-cmd⟩ ::= **select** ⟨module-name⟩
| **declare-vars** ⟨var-set⟩
| **def-final-st** ⟨pattern-form⟩
| **start-proof**
| **quit**

| **normalize** ⟨pattern-form⟩
| **subsumed** ⟨norm-pattern-form⟩ `=<` ⟨norm-pattern-form⟩
| **intersect** ⟨norm-pattern-form⟩ `=?` ⟨norm-pattern-form⟩
| **inv wrap** ⟨op-id⟩ [**params** ⟨var-set⟩] **is** ⟨norm-pattern-form⟩

| **add-goal** ⟨reach-form⟩
| **add-lemma** ⟨reach-form⟩

| **step** [⟨number⟩]
| **step***
| **list-goals**
| **focus** ⟨goal-id⟩
| **case** ⟨goal-id⟩ **on** ⟨var-name⟩ **using** ⟨term-set⟩
| **split** ⟨goal-id⟩ **using** ⟨eqform⟩

Category (i) commands let the user select a module defining a rewrite relation we wish to reason over, to declare variables which can be used in commands of type (i) and (iii), and to start/stop proofs. The commands in category (ii) are also straightforward: **normalize** takes a pattern formula and returns a normalized pattern formula (a disjunction of constrained patterns) equivalent to it; **subsumed** and **intersect** perform parameterized substitution and intersection; **inv** takes a bracket operator-id ([]), a set of shared variables V, and a pattern form P and adds a goal to be solved of the form $P\sigma \rightarrow^{\circledast} [P]$ where $\sigma(v) = v \Leftrightarrow v \in V$. Commands of type (iii) are the basic primitives for adding goals and axioms to your proof. Finally, type (iv) commands let the user list the names of remaining goals and advance the state of the proof by applying one or several steps with proof rules (or trying to complete a proof with the **step*** strategy), performing case analysis on (or splitting the condition of) a goal. Focusing on a goal eliminates all other goals from the proof state; obviously, this is unsound. The intent, however, is not to continue the proof process, but to *restart it* after such focusing. The **focus** command enables the user to focus attention on some proof goals that seem to lead to looping so that, for example, the proof can be restarted with some additional lemmas (e.g., some strengthened invariants) to help its completion, or some bug in the original set of goals may be detected.

The grammar below defines the syntactic categories used by tool commands. Some non-terminals are marked as special. These non-terminals are handled by built-in parsers as part of the Maude runtime.

⟨reach-form⟩ ::= ⟨pattern-form⟩ `=>A` ⟨pattern-form⟩
⟨pattern-form⟩ ::= ⟨pattern-form⟩ `/\` ⟨pattern-form⟩
| ⟨pattern-form⟩ `\/` ⟨pattern-form⟩
| ⟨pattern⟩
⟨norm-pattern-form⟩ ::= ⟨norm-pattern-form⟩ `\/` ⟨norm-pattern-form⟩
| ⟨pattern⟩
⟨pattern⟩ ::= ⟨term⟩ "`|`" ⟨eqform⟩
⟨eqform⟩ ::= ⟨eqform⟩ `\/` ⟨eqform⟩ | ⟨eqform⟩ `/\` ⟨eqform⟩
| ⟨term⟩ `=` ⟨term⟩ | ⟨term⟩ `=/=` ⟨term⟩
⟨term-set⟩ ::= `(` ⟨term⟩ `)` ⟨term-set⟩ | `(` ⟨term⟩ `)`
⟨var-set⟩ ::= `(` ⟨var-name⟩ `)` ⟨var-set⟩ | `(` ⟨var-name⟩ `)`
⟨goal-id⟩ ::= ⟨nat⟩ ⟨goal-id⟩ | ⟨nat⟩
⟨op-id⟩ ::= special
⟨module-name⟩ ::= special
⟨var-name⟩ ::= special
⟨term⟩ ::= special
⟨nat⟩ ::= special