A Constructor-Based Reachability Logic for Rewrite Theories

Stephen Skeirik(✉), Andrei Stefanescu, and José Meseguer

University of Illinois at Urbana-Champaign, Champaign, USA
{skeirik2,stefane1,meseguer}@illinois.edu

Abstract. Reachability logic has been applied to $\mathbb{K}$ rewrite-rule-based language definitions as a *language-generic* logic of programs. To be able to verify not just code but also *distributed system designs*, a new *rewrite-theory-generic* reachability logic is presented and proved sound for a wide class of rewrite theories. *Constructor-based* semantic unification, matching, and satisfiability procedures greatly increase the range of decidable background theories that can be used in reachability logic proofs. New methods for proving invariants of possibly never terminating distributed systems are developed, and experiments with a prototype implementation illustrating the new proof methods are presented.

Keywords: Program verification · Rewriting logic · Reachability logic

1 Introduction

The main applications of reachability logic to date have been as a *language-generic* logic of programs [14,15]. 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 a correct-by-construction reachability logic for a language so defined is automatically obtained [15]. This method has been effective in proving reachability properties for a wide range of programs.

Although the foundations of reachability logic are very general [14,15], the existing theory does not provide straightforward answers to the following 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 [8]? (2) If so, what would be the most natural way to conceive such a *rewrite-theory-generic* logic? 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, answering questions (1) and (2) is of practical software engineering interest.

Although a first step towards a reachability logic for rewrite theories has been taken in [6], as explained in Sect. 7 and below, that first step still leaves several

F. Fioravanti and J. P. Gallagher (Eds.): LOPSTR 2017, LNCS 10855, pp. 201–217, 2018.
https://doi.org/10.1007/978-3-319-94460-9_12

important questions open. The most burning one is how to prove *invariants*. Since they are the most basic safety properties, support for proving invariants is a *sine qua non* requirement. As explained below, a serious obstacle is 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 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), and of recent results on decidable satisfiability of quantifier-free formulas in initial algebras, e.g., [9] 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 [14,15]. Could a reachability logic for rewrite theories be simpler? This work tackles head on these three open questions to provide a general reachability logic and a prototype implementation suitable for reasoning about properties of *both* distributed systems and programs based on their rewriting logic semantics.

Rewriting Logic in a Nutshell. A distributed system can be designed and modeled as a *rewrite theory* $\mathcal{R} = (\Sigma, E, R)$ [8] 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 [5] mutual exclusion protocol, explained in detail in Sect. 2. 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 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 modeled as a rewrite theory $\mathcal{R} = (\Sigma, E, R)$, where E includes axioms such as associativity-commutativity of multiset union, list associativity, and identity axioms for $\emptyset$ and nil. QLOCK's behavior is specified by five rewrite rules R. Rule *w2c* below specifies a waiting process i becoming 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 $A \rightarrow^{\circledast} B$, with A and B state predicates (see Sect. 3). Assume for simplicity that $vars(A) \cap vars(B) = \emptyset$. Such a formula is then interpreted 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 [8] (computation = deduction). As a first approximation, $A \rightarrow^{\circledast} B$ is a Hoare logic *partial correctness* assertion of the form $\{A\}\mathcal{R}\{B\}$, 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, $0 \leq j \leq n$ 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[1] 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 $E \models \varphi\rho$.

The Invariant Paradox. How can we *prove invariants* in such a reachability logic? For example, mutual exclusion for QLOCK? Paradoxically, we cannot! This is because QLOCK, like many other protocols, *never terminates*, that is, has no terminating sequences whatsoever. And 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 Sect. 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 reachability logic with just *two* inference rules and its soundness are developed in Sects. 4 and 5. A systematic methodology to prove *invariants* by means of reachability formulas is developed in Sect. 4.1. The goal of increasing the logic's potential for automation by making it constructor-based is advanced in Sects. 3–5. A proof of concept of the entire approach is given by means of a Maude-based prototype implementation and a suite of experiments verifying various properties of distributed system designs in Sect. 6. Related work and conclusions are discussed in Sect. 7. Proofs can be found in [13].

2 Many-Sorted Algebra and Rewriting Logic

We present some preliminaries on many-sorted algebra and rewriting logic. For a more general treatment using order-sorted algebra see [13]. Readers familiar with many-sorted logic may go directly to Definition 1. We assume familiarity with the following basic concepts and notation that are explained in full detail in, e.g., [10]: (i) *many-sorted (MS) signature* as a pair $\Sigma = (S, \Sigma)$ with S a set of *sorts*

[1] 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.

and Σ an $S^* \times S$-indexed family $\Sigma = \{\Sigma_{w,s}\}_{(w,s)\in S^*\times S}$ of function symbols, where $f \in \Sigma_{s_1\ldots s_n,s}$ is displayed as $f : s_1 \ldots s_n \to s$; (ii) *Σ-algebra* A as a pair $A = (A, \_A)$ with $A = \{A_s\}_{s\in S}$ an S-indexed family of sets, and $\_A$ a mapping interpreting each $f : s_1 \ldots s_n \to s$ as a function in the set $[A_{s_1} \times \ldots \times A_{s_n} \to A_s]$. (iii) *$\Sigma$-homomorphism* $h : A \to B$ as an S-indexed family of functions $h = \{h_s : A_s \to B_s\}_{s\in S}$ preserving the operations in Σ; (iv) the term Σ-algebra T_Σ and its initiality in the category $\mathbf{MSAlg}_\Sigma$ of Σ-algebras when Σ is unambiguous.

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 $T_\Sigma(X)$ means that for each $A \in \mathbf{MSAlg}_\Sigma$ 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.

We also assume familiarity with many-sorted first-order logic including: (i) the first-order language of *Σ-formulas* for Σ a signature (in our case Σ has only function symbols and the $=$ predicate); (ii) 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* $A, \alpha \models \varphi$ (iii) the notions of a formula $\varphi \in Form(\Sigma)$ being *valid*, denoted $A \models \varphi$, resp. *satisfiable* in a Σ-algebra A. For a subsignature $\Omega \subseteq \Sigma$ and $A \in \mathbf{MSAlg}_\Sigma$, the *reduct* $A|_\Omega \in \mathbf{MSAlg}_\Omega$ agrees with A in the interpretation of all sorts and operations in Ω and discards everything in $\Sigma \setminus \Omega$. If $\varphi \in Form(\Omega)$ we have the equivalence $A \models \varphi \Leftrightarrow A|_\Omega \models \varphi$.

An MS *equational theory* is a pair $T = (\Sigma, E)$, with E a set of (possibly conditional) Σ-equations. $\mathbf{MSAlg}_{(\Sigma,E)}$ denotes the full subcategory of $\mathbf{MSAlg}_\Sigma$ with objects those $A \in \mathbf{MSAlg}_\Sigma$ such that $A \models E$, called the *(Σ, E)-algebras*. $\mathbf{MSAlg}_{(\Sigma,E)}$ has an *initial algebra* $T_{\Sigma/E}$ [10]. The inference system in [10] is *sound and complete* for MS equational deduction, i.e., for any MS equational theory (Σ, E), and Σ-equation $u = v$ we have an equivalence $E \vdash u = v \Leftrightarrow E \models u = v$. For the sake of simpler inference we assume *non-empty sorts*, i.e., $\forall s \in S\ T_\Sigma, s \neq \emptyset$. 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 $u_i\sigma =_E v_i\sigma$, $1 \leq i \leq n$. An *E-unification algorithm* for (Σ, E) is an algorithm generating a *complete set* of E-unifiers $Unif_E(\phi)$ for any system of Σ equations ϕ, where "complete" means that for any E-unifier σ of ϕ there is a $\tau \in Unif_E(\phi)$ and a substitution ρ such that $\sigma =_E (\tau\rho)|_{dom(\sigma)\cup dom(\tau)}$, where $=_E$ here means

that for any variable x we have $x\sigma =_E x(\tau\rho)|_{dom(\sigma)\cup dom(\tau)}$. The algorithm is *finitary* if it always terminates with a *finite set* $Unif_E(\phi)$ for any ϕ.

We recall some basic concepts about *rewriting logic*. The survey in [8] gives a fuller account. A rewrite theory $\mathcal{R}$ axiomatizes 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 also the following notation from [3]: (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 1. *A* rewrite theory *is a 3-tuple* $\mathcal{R} = (\Sigma, E \cup B, R)$ *with* $(\Sigma, E \cup B)$ *an MS equational theory and* R *a set of conditional* Σ-rewrite rules $l \to r$ *if* ϕ, *with* $l, r \in T_\Sigma(X)_s$ *for some* $s \in S$, *and* ϕ *a quantifier-free* Σ*-formula. 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 either* u *or* v. *(2) The equations* E, *when oriented as conditional rewrite rules* $\vec{E} = \{u \to v \text{ if } \psi \mid u = v \text{ if } \psi \in E\}$, *are* convergent *modulo* B, *i.e., strictly coherent, confluent, and operationally terminating as rewrite rules modulo* B [7]. *(3) The rules* R *are* ground coherent *with the equations* E *modulo* B [4].

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 $t, t' \in T_{\Sigma,s}$, $s \in S$, $t \to_{R,B} t'$ holds iff there is a rewrite rule $l \to r$ *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 $l \to r$ *if* $\phi \in R$. This means that $\mathcal{R}$ can specify an *open system*, in the sense of [11], that interacts with an external, non-deterministic environment such as, for example, a thermostat.

Conditions (1)–(3) allow a simple description of the *initial reachability model* $\mathcal{T}_\mathcal{R}$ [8] of $\mathcal{R}$ as the *canonical reachability model* $\mathcal{C}_\mathcal{R}$ whose states belong to the *canonical term algebra* $C_{\Sigma/E,B}$, and 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 $l \to r$ *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 [5], where (Σ, B) defines the protocol's states, involving natural numbers, lists, and multi-sets over natural numbers. Σ has sorts $S = \{Nat, List, MSet, Conf, State, Pred\}$

with subsorts[2] $Nat < List$ and $Nat < MSet$ and operators $F = \{0 : \rightarrow Nat,\ s\_ : Nat \rightarrow Nat,\ \emptyset : \rightarrow MSet,\ nil : \rightarrow List,\ \_\,\_ : MSet\ MSet \rightarrow MSet,\ \_;\_ : List\ List \rightarrow List, dupl : MSet \rightarrow Pred, tt : \rightarrow Pred, < \_ > : Conf \rightarrow State, \_|\_|\_|\_ : MSet\ MSet\ MSet\ List \rightarrow Conf\}$, where underscores denote operator argument placement. The axioms B are the associativity-commutativity of the multiset union $\_\,\_$ with identity $\emptyset$, and the associativity of list concatenation $\_;\_$ with identity nil. The only equation in E is $dupl(s\ i\ i) = tt$. It defines the $dupl$ predicate by detecting a duplicated element i in the multiset $s\ i\ i$ (s could be empty). *States* of QLOCK are B-equivalence classes of ground terms of sort $State$.

QLOCK [5] is a mutual exclusion protocol where the number of processes is unbounded. Furthermore, in the *dynamic* version of QLOCK presented below, such a number can grow or shrink. Each process is identified by a number. The system configuration has three sets of processes (normal, waiting, and critical) plus a waiting queue. To ensure mutual exclusion, a normal process must first register its name at the end of the waiting queue. When its name appears at the front of the queue, it is allowed to enter the critical section. The first three rewrite rules in R below specify how a *normal* process i first transitions to a *waiting* process, then to a *critical* process, and back to normal. The last two rules in R specify how a process can dynamically join or exit the system.

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

where $\phi \equiv dupl(n\ i\ w\ c) \neq tt$, i is a number, n, w, and c are, respectively, normal, waiting, and critical process identifier sets, and q is a queue of process identifiers. It is easy to check 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 MS equational theory $(\Sigma, E \cup B)$, the *atomic state predicates* appearing in the constructor-based reachability logic formulas of Sect. 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 $\mathcal{R}$.

[2] As pointed out at the beginning of Sect. 2, [13] treats the more general *order-sorted* case, where sorts form a poset $(S, \leq)$ with $s \leq s'$ interpreted as set containment $A_s \subseteq A_{s'}$ in a Σ-algebra A.

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 a MS 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 2. *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.

Definition 3. *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 $PatPred(\Omega, \Sigma)$ 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 $PatPred(\Omega, \Sigma)$. 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. Letting n, w, c be multisets of process identifiers and q be an associative list of process identifiers, recall that QLOCK states have the form $< n \mid w \mid c \mid q >$. From the five rewrite rules defining QLOCK, it is easy to prove that if $< n \mid w \mid c \mid q > \rightarrow^* < n' \mid w' \mid c' \mid q' >$ 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 $< n \mid w \mid c \mid q > \mid dupl(n\ w\ c) \neq tt$.

If $E_\Omega \cup B_\Omega$ 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 Definition 3 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 that $vars(u \mid \varphi) \cap vars(v \mid \psi) = \emptyset$, and that all variables in $ran(\alpha)$ are *fresh*. Pattern intersection can also be defined when $u \mid \varphi$ and $v \mid \phi$ share *parameters* $Y = vars(u \mid \varphi) \cap vars(v \mid \phi) = vars(u) \cap vars(v)$. [13] defines in detail the notions of *parametric intersection* $[\![u \mid \varphi]\!] \cap_Y [\![v \mid \phi]\!]$ and of *parametric subsumption* $v \mid \phi \subseteq_Y u \mid \varphi$ of patterns. These notions are very useful to reason about *parameterized* invariants and co-invariants (see Sect. 4.1 and [13]).

4 Constructor-Based Reachability Logic

The constructor-based reachability logic we 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" intuitively means all rewrites must occur at the top of the term.[3] Many rewrite theories of interest, including those specifying distributed object-oriented systems or the semantics of (possibly concurrent) programming languages, can be easily made topmost by a theory transformation (see, e.g., [16]). Formally, we require $\mathcal{R} = (\Sigma, E \cup B, R)$, besides satisfying the requirements in Definition 1, also satisfies:

1. $(\Sigma, E \cup B)$ has a sort *State*, a decomposition $(\Sigma, B, \vec{E})$, and a constructor decomposition $(\Omega, B_\Omega, \vec{E_\Omega})$ where: (i) $\forall u \in T_\Omega(X)_{State}, vars(u) = vars(u!)$; (ii) B_Ω are linear and regular with a finitary $E_\Omega \cup B_\Omega$-unification algorithm.
2. Rules in R have the form $l \rightarrow r$ *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 $[u] \rightarrow_\mathcal{R} [v]$ holds, then the R, B-rewrite $u \rightarrow_{R,B} u'$ such that $[u'!] = [v]$ *happens at the top* of u, i.e., uses a rewrite rule $l \rightarrow r$ *if* $\varphi \in R$ and a ground substitution $\sigma \in [Y \rightarrow T_\Omega]$, with Y the rule's variables, such that $u =_{B_\Omega} l\sigma$ and $u' = r\sigma$.

[3] Topmost theories have reachability completeness for narrowing [16]. Our inference system uses narrowing to symbolically compute successor states in $\mathcal{C}_\mathcal{R}$.

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 of $PatPred(\Omega, \Sigma)$ 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: $A \rightarrow^{\circledast} B$, with $A, B \in PatPred(\Omega, \Sigma)_{State}$. The *parameters* Y of $A \rightarrow^{\circledast} B$ are the variables in the set $Y = vars(A) \cap vars(B)$, and $A \rightarrow^{\circledast} B$ is called *unparameterized* iff $Y = \emptyset$.

The reachability logic in [14,15] is based on *terminating* sequences of state transitions; when there are no terminating states, *all* reachability formulas are *vacuously true*. Our purpose is to extend the logic in order to verify properties of general distributed systems specified as rewrite theories $\mathcal{R}$ which *may never terminate*. For this, as explained in Sect. 4.1, we *generalize* the *all-paths* satisfaction relation in [15], which for a theory $\mathcal{R}$ we denote by $\mathcal{R} \models^{\forall} A \rightarrow^{\circledast} B$, to a *relativized* satisfaction relation $\mathcal{R} \models^{\forall}_{T} A \rightarrow^{\circledast} B$, where T is a constrained pattern predicate such that $[\![T]\!]$ is a set of terminating states. That is, let $Term_{\mathcal{R}} = \{[u] \in \mathcal{C}_{\mathcal{R},State} \mid (\nexists [v])\ [u] \rightarrow_{\mathcal{R}} [v]\}$. We then require $[\![T]\!] \subseteq Term_{\mathcal{R}}$. The standard relation $\mathcal{R} \models^{\forall} A \rightarrow^{\circledast} B$ is then recovered as the special case where $[\![T]\!] = Term_{\mathcal{R}}$. Call $[u] \rightarrow^{*}_{\mathcal{R}} [v]$ a *T-terminating sequence* iff $[v] \in [\![T]\!]$.

Definition 4. *Given T with $[\![T]\!] \subseteq Term_{\mathcal{R}}$, the* all-paths *satisfaction relation $\mathcal{R} \models^{\forall}_{T} u \mid \varphi \rightarrow^{\circledast} \bigvee_{j \in J} v_j \mid \phi_j$ asserts the satisfaction of the formula $u \mid \varphi \rightarrow^{\circledast} \bigvee_{j \in J} v_j \mid \phi_j$ 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 $u \mid \varphi \rightarrow^{\circledast} \bigvee_{j \in J} v_j \mid \phi_j$ unparameterized, $\mathcal{R} \models^{\forall}_{T} u \mid \varphi \rightarrow^{\circledast} \bigvee_{j \in J} v_j \mid \phi_j$ holds iff for each T-terminating sequence $[u_0] \rightarrow_{\mathcal{R}} [u_1] \ldots [u_{n-1}] \rightarrow_{\mathcal{R}} [u_n]$ with $[u_0] \in [\![u \mid \varphi]\!]$ there exist k, $0 \leq k \leq n$ and $j \in J$ such that $[u_k] \in [\![v_j \mid \phi_j]\!]$. For $u \mid \varphi \rightarrow^{\circledast} \bigvee_{j \in J} v_j \mid \phi_j$ with parameters Y, $\mathcal{R} \models^{\forall}_{T} u \mid \varphi \rightarrow^{\circledast} \bigvee_{j \in J} v_j \mid \phi_j$ holds if $\mathcal{R} \models^{\forall}_{T} (u \mid \varphi)\rho \rightarrow^{\circledast} (\bigvee_{j \in J} v_j \mid \phi_j)\rho$ holds for each $\rho \in [Y \rightarrow T_{\Omega}]$.

Since a constrained pattern predicate is equivalent to a disjunction of atomic ones, we can define satisfaction on general reachability logic formulas as follows: $\mathcal{R} \models^{\forall}_{T} \bigvee_{1 \leq i \leq n} u_i \mid \varphi_i \rightarrow^{\circledast} A$ iff $\bigwedge_{1 \leq i \leq n} \mathcal{R} \models^{\forall}_{T} u_i \mid \varphi_i \rightarrow^{\circledast} A$, assuming same parameters $Y_i = vars(u_i \mid \varphi_i) \cap vars(A)$, i.e., $Y_i = Y_{i'}$ for $1 \leq i < i' \leq n$.

$\mathcal{R} \models^{\forall}_{T} A \rightarrow^{\circledast} B$ is a *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 $\{A\}\mathcal{R}\{B\}$ [13].

Recall that rewrite rules $l \rightarrow r$ *if* ϕ are assumed to have $l \in T_{\Omega}(X)$. For symbolic reasoning purposes it will be very useful to also require that $r \in T_{\Omega}(X)$. This can be achieved by a theory transformation $\mathcal{R} \mapsto \hat{\mathcal{R}}$. Stated formally, 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 is:

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* applies to any transition system $\mathcal{S} = (S, \rightarrow_{\mathcal{S}})$ with *states* S and *transition relation* $\rightarrow_{\mathcal{S}} \subseteq S \times S$. The set $Reach(S_0)$ of states *reachable* from $S_0 \subseteq S$ 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 about $\mathcal{S}$ with initial states S_0 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 $Reach(S_0) \cap Q = \emptyset$. Obviously, P is an invariant iff $S \setminus P$ is a co-invariant.

Suppose we have specified a distributed system by a topmost rewrite theory $\mathcal{R}$, and constrained pattern predicates S_0 and P, and we want to prove that $[\![P]\!]$ is an *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 Sect. 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., $Term_{\mathcal{R}} = \emptyset$. For example, QLOCK and other mutual exclusion protocols are never-terminating. Then, no reachability formula can characterize and 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 $Term_{\mathcal{R}} = \emptyset$, $\mathcal{R} \models^{\forall}_{T} A \rightarrow^{\circledast} B$ holds vacuously for *all* reachability formulas $A \rightarrow^{\circledast} B$.

Is then reachability logic useless to prove invariants? Definitely *not*. We 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*.

Theorem 1. *For* $S_0, P \in PatPred(\Omega, \Sigma)$ *constrained pattern predicates in standard form with* $vars(S_0) \cap vars(P) = \emptyset$, $[\![P]\!]$ *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 1 extends to parametric invariants.

Specifying Invariants for QLOCK. Consider the QLOCK specification from Sects. 2 and 3. QLOCK is *never* terminating. However, we can apply the theory transformation in Theorem 1 by adding an operator $[\_] : Conf \rightarrow State$ and a rule $stop : < t > \rightarrow [t]$ for $t{:}Conf$. Define the set of initial states by the pattern predicate $S_0 = < n' \mid \emptyset \mid \emptyset \mid nil > \mid dupl(n') \neq tt$. Since QLOCK states have the form $< n \mid w \mid c \mid q >$, mutual exclusion means $|c| \leq 1$, which is expressible by the pattern predicate $< n \mid w \mid i \mid i \ ; \ q > \vee < n \mid w \mid \emptyset \mid q >$. But we need also to ensure our multisets are actually *sets*. Thus, the pattern predicate $P = \big(< n \mid w \mid i \mid i \ ; \ q > \mid dupl(n \ w \ i) \neq tt\big) \vee \big(< n \mid w \mid \emptyset \mid q > \mid dupl(n \ w) \neq tt\big)$ specifies mutual exclusion. By Theorem 1, 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* $vars(S_0) \cap vars(P) = \emptyset$. $[\![P]\!]$ *is an invariant of* $(\mathcal{C}_{\mathcal{R},State}, \rightarrow_{\mathcal{R}})$ *from* $[\![S_0]\!]$ *if: (i)* $S_0 \subseteq P$, *and (ii)* $\mathcal{R}_{stop} \models^{\forall}_{[]} P \rightarrow^{\circledast} [P\sigma]$, *where* σ *is a sort-preserving bijective renaming of variables such that* $vars(P) \cap vars(P\sigma) = \emptyset$.

Corollary 1 can be extended to parametric invariants (see [13]). The treatment of co-invariants is similar and can also be found in [13].

5 A Sound Inference System

We present our inference system for all-path reachability for any $\mathcal{R}$ satisfying topmost requirements (1)–(2), with rules $R = \{l_j \rightarrow r_j \text{ if } \phi_j\}_{j \in J}$ 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 $\textsc{Step}^{\forall} + \textsc{Subsumption}$ 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 $[\emptyset,\ \mathcal{C}]\ \vdash_T\ \mathcal{C}$, which is a shorthand for the set of goals $\{[\emptyset,\ \mathcal{C}]\ \vdash_T\ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i \ \mid\ (u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i) \in \mathcal{C}\}$. 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 feature is that sequents $[\emptyset,\ \mathcal{C}]\ \vdash_T\ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$, whose formulas $\mathcal{C}$ have been *postulated* (as the conjectures to be proved), are transformed by $\text{STEP}^{\forall}$ + SUBSUMPTION into sequents of the form $[\mathcal{C},\ \emptyset]\ \vdash_T u' \mid \varphi' \longrightarrow^{\circledast} \bigvee_i v'_i \mid \psi'_i$, 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 $vars(P) \cap vars(P\sigma) = \emptyset$. But, since for QLOCK, P is a disjunction, in our inference system this means proving from $\mathcal{R}_{stop}$ that $[\emptyset, \mathcal{C}] \vdash_{[]} \mathcal{C}$, where $\mathcal{C}$ are the conjectures:

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

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

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

Before explaining the $\text{STEP}^{\forall}$ + 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:

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

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.[4] Consider now the rule:

[4] In the current prototype implementation (see Sect. 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 $\text{STEP}^{\forall}$ + SUBSUMPTION or AXIOM rules below: they will be either over-approximated, or will become proof obligations to be discharged by an inductive theorem prover.

STEP$^{\forall}$ + SUBSUMPTION

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

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 4, $u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ is semantically valid iff $u \mid \varphi' \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ 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 performs narrowing of $u \mid \varphi'$ with rules R [16].

AXIOM

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

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 $\text{STEP}^{\forall}$ + SUBSUMPTION plus AXIOM is now the theorem:

Theorem 2 *(Soundness). Let $\mathcal{R}$ be a rewrite theory, and $\mathcal{C}$ a finite set of reachability formulas. If $\mathcal{R}$ proves $[\emptyset,\ \mathcal{C}]\ \vdash_T\ \mathcal{C}$ then $\mathcal{R} \models^{\forall}_T \mathcal{C}$.*

Investigating completeness of the logic is left as future work.

6 Prototype Implementation and Experiments

We have implemented the reachability logic proof system in Maude [1]. Our prototype takes as input (i) a 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.

To mechanize the two proof rules we use a finitary B-unification algorithm as well as an SMT solver to discharge $E \cup B$ constraints. For SMT solving we use variant satisfiability [9,12], which allows 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 [2] and protects a constructor subtheory which we assume consists only of commutative and/or AC and/or identity axioms B_Ω. Thus, both validity and satisfiability of QF formulas in the initial algebra $\mathcal{T}_{\Sigma/E \cup B}$ are decidable [9]. Future implementations will support more general rewrite theories, add other decision procedures, and use an inductive theorem prover backend.

We have verified properties for a suite of examples of rewrite theories specifying distributed systems such as communication or mutual exclusion protocols and real-time systems. Table 1 summarizes these experiments. For further details plus runnable code see http://maude.cs.illinois.edu/tools/rltool/.

Table 1. Examples verified in the prototype implementation

Example	Description of the system/property
Choice	Nondeterministically throws away elements from a multiset/eventually only one element left
Comm. Protocol 1	Simple communication protocol/received data is always a prefix of the data to be sent
Comm. Protocol 2	Fault-tolerant communication protocol/all data is eventually received in-order
Dijkstra	Dijkstra's 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	Readers-writers mutual exclusion alg./mutual exclusion
Lamport's bakery	Unbounded Lamport's bakery/mutual exclusion
Thermostat	Open system that dynamically responds to temperature/temperature remains in preset bounds

$$T_1 \equiv \left\{ \frac{}{[\mathcal{C},\emptyset] \vdash_{[]} [n^3 \mid w^3 \mid \emptyset \mid q^3] \mid dupl(n'' \, w' \, p) \neq tt \wedge dupl(n^3 \, w^3) \neq tt \rightarrow^{\circledast} [P_1] \vee [P_2]} \; \mathrm{sub}(P_1, \alpha) \right.$$

$$T_2 \equiv \left\{ \frac{}{[\mathcal{C},\emptyset] \vdash_{[]} [n^3 \mid w^3 \mid i^3 \mid i^3; q^3] \mid dupl(n'' \, w' \, p) \neq tt \wedge dupl(n^3 \, w^3 \, i^3) \neq tt \rightarrow^{\circledast} [P_1] \vee [P_2]} \; \mathrm{sub}(P_2, \alpha) \right.$$

$$\frac{\dfrac{T_1 \qquad\qquad T_2}{\ldots \; [\mathcal{C},\emptyset] \vdash_{[]} < n'' \mid w' \; p \mid \emptyset \mid q' > \mid dupl(n'' \, w' \, p) \neq tt \; \ldots \; \rightarrow^{\circledast} [P_1] \vee [P_2]} \; \mathrm{axiom}(G_2, \alpha)}{[\emptyset,\mathcal{C}] \vdash_{[]} < n' \mid w' \mid \emptyset \mid q' > \mid dupl(n' \; w') \neq tt \; \rightarrow^{\circledast} [P_1] \vee [P_2]} \; \mathrm{step}(n2w, \theta)$$

Fig. 1. Partial proof tree for QLOCK

To illustrate how the tool works in practice, Fig. 1 shows a partial derivation of a sequent. Recall that for QLOCK we had to prove $[\emptyset, \mathcal{C}] \vdash_{[]} \mathcal{C}$, where $\mathcal{C}$ was two already-discussed reachability formulas $G_i \equiv P_i' \rightarrow [P_1] \vee [P_2]$ for $i \in \{1, 2\}$ with respective preconditions the renamed disjuncts P_i', $1 \leq i \leq 2$ in invariant $P_1 \vee P_2$, and postcondition $[P_1] \vee [P_2]$, where $P_1 \equiv < n \mid w \mid i \mid i \; ; \; q > \mid dupl(n \; w \; i) \neq tt$ and $P_2 \equiv < n \mid w \mid \emptyset \mid q > \mid dupl(n \; w) \neq tt$. Now, consider $[\emptyset, \mathcal{C}] \vdash_{[]} P_2' \rightarrow^{\circledast} [P]$. In the proof fragment below, the initial sequent must apply the step rule. The result of $\mathrm{step}(n2w, \theta)$ is the goal resulting from unifying the head of the sequent with the lefthand side of the rule $n2w$ using the unifier $\theta = \{n \mapsto n''p, w \mapsto w', c \mapsto \emptyset, q \mapsto q'\}$. The next inference $\mathrm{axiom}(G_2, \alpha)$ applies axiom G_2 using the substitution $\alpha \supseteq \{n \mapsto n^3, w \mapsto w^3, i \mapsto i^3, q \mapsto q^3\}$. Since G_2 has two constrained patterns in its succedent, we derive two new goals, represented by proof trees T_1 and T_2. In either case, we can immediately subsume by noting that our reachability formula's antecedent is an instance of either $[P_1]$ or $[P_2]$ using substitution α, thus terminating the proof.

7 Related Work and Conclusions

This work extends reachability logic [14,15] to a rewrite-theory-generic logic to reason about *both* distributed system designs and programs. This extension is non-trivial. It requires: (i) relativizing terminating sequences to a chosen subset $[\![T]\!]$ of terminating states; (ii) solving the "invariant paradox," to reason about invariants and co-invariants and characterizing them by 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 patterns of the form $u \mid \varphi$ with u a constructor term.

In contrast, standard reachability logic [14,15] uses matching logic, which assumes a first-order model $\mathcal{M}$ and its satisfaction relation $\mathcal{M} \models \varphi$ in its reachability logic proof system. As discusses in Sect. 3, we choose $T_{\Sigma/E \cup B}$ as the

model and $\rightarrow_{\mathcal{R}}$ for transitions, rather than some general $\mathcal{M}$ 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 $\textsc{Step}^{\forall} + \textsc{Subsumption}$ checks the subsumption, or ensures that states have at least a successor), instead of relying on a general satisfaction relation $\models$ on some $\mathcal{M}$. The result is a simpler logic with only two rules (versus eight in [14,15]).

We agree with the work in [6] on the common goal of making reachability logic rewrite-theory-generic, but differ on the methods used. Main differences include: (1) [6] does not give an inference system but a verification algorithm. (2) the theories used in [6] assume restrictions like those in [11] for "rewriting modulo SMT," which limit the class of equational theories. (3) Matching is used in [6] instead of unification. Thus, 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 [6].

In conclusion, the goal of making reachability logic a rewrite-theory-generic verification logic has been advanced. Feasibility has been validated with a prototype and a suite of examples. Building a robust and highly effective reachability logic tool for rewrite theories is a more ambitious future goal.

Acknowledgements. Partially supported by NSF Grant CNS 14-09416.

References

1. Clavel, M., et al.: All About Maude - A High-Performance Logical Framework. LNCS, vol. 4350. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71999-1
2. Comon-Lundh, H., Delaune, S.: The finite variant property: how to get rid of some algebraic properties. In: Giesl, J. (ed.) RTA 2005. LNCS, vol. 3467, pp. 294–307. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-32033-3_22
3. Dershowitz, N., Jouannaud, J.-P.: Rewrite systems. In: Handbook of Theoretical Computer Science, vol. B, pp. 243–320. North-Holland (1990)
4. 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)
5. Futatsugi, K.: Fostering proof scores in CafeOBJ. In: Dong, J.S., Zhu, H. (eds.) ICFEM 2010. LNCS, vol. 6447, pp. 1–20. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16901-4_1
6. Lucanu, D., Rusu, V., Arusoaie, A., Nowak, D.: Verifying reachability-logic properties on rewriting-logic specifications. In: Martí-Oliet, N., Ölveczky, P.C., Talcott, C. (eds.) Logic, Rewriting, and Concurrency. LNCS, vol. 9200, pp. 451–474. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23165-5_21
7. Lucas, S., Meseguer, J.: Normal forms and normal theories in conditional rewriting. J. Log. Algebr. Meth. Program. **85**(1), 67–97 (2016)
8. Meseguer, J.: Twenty years of rewriting logic. J. Algebr. Logic Program. **81**, 721–781 (2012)

9. Meseguer, J.: Variant-based satisfiability in initial algebras. In: Artho, C., Ölveczky, P.C. (eds.) FTSCS 2015. CCIS, vol. 596, pp. 3–34. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29510-7_1
10. Meseguer, J., Goguen, J.: Initiality, induction and computability. In: Algebraic Methods in Semantics, pp. 459–541. Cambridge UP (1985)
11. Rocha, C., Meseguer, J., Muñoz, C.: Rewriting modulo SMT and open system analysis. J. Logic Algebr. Methods Program. **86**, 269–297 (2017)
12. Skeirik, S., Meseguer, J.: Metalevel algorithms for variant satisfiability. In: Lucanu, D. (ed.) WRLA 2016. LNCS, vol. 9942, pp. 167–184. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44802-2_10
13. Skeirik, S., Stefanescu, A., Meseguer, J.: A constructor-based reachability logic for rewrite theories. Technical report. http://hdl.handle.net/2142/95770
14. Ştefănescu, A., et al.: All-path reachability logic. In: Dowek, G. (ed.) RTA 2014. LNCS, vol. 8560, pp. 425–440. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08918-8_29
15. Stefanescu, A., Park, D., Yuwen, S., Li, Y., Rosu, G.: Semantics-based program verifiers for all languages. In: Proceedings of the OOPSLA 2016, pp. 74–91. ACM (2016)
16. Thati, P., Meseguer, J.: Symbolic reachability analysis using narrowing and its application to the verification of cryptographic protocols. J. High.-Order Symb. Comput. **20**(1–2), 123–160 (2007)