



**Verifying Reachability-Logic Properties  
on Rewriting-Logic Specifications  
(Extended Version)**

**Andrei Arusoaie, Dorel Lucanu,  
David Nowak, Vlad Rusu**

**TR 15-01, February 2015**

**ISSN 1224-9327**



**Universitatea “Alexandru Ioan Cuza” Iași  
Facultatea de Informatică**

Str. Berthelot 16, 6600-Iași, Romania  
Tel. +40-32-201090, email: [bibl@infoiasi.ro](mailto:bibl@infoiasi.ro)

# Verifying Reachability-Logic Properties on Rewriting-Logic Specifications (Extended Version)

Andrei Arusoaie<sup>1</sup>, Dorel Lucanu<sup>2</sup>, David Nowak<sup>3</sup>, and Vlad Rusu<sup>1</sup>

<sup>1</sup> Inria Lille Nord Europe, France

`Andrei.Arusoaie@inria.fr` `Vlad.Rusu@inria.fr`

<sup>2</sup> Faculty of Computer Science, Alexandru Ioan Cuza University, Iași, Romania  
`dlucanu@info.uaic.ro`

<sup>3</sup> CRISTAL, CNRS & University of Lille, France  
`david.nowak@univ-lille1.fr`

**Abstract.** Reachability Logic is a recently introduced formalism, which is currently used for defining the operational semantics of programming languages and for stating properties about program executions. In this paper we show how Reachability Logic can be adapted for stating properties of transition systems described by Rewriting-Logic specifications. We propose an automatic procedure for verifying Rewriting-Logic specifications against Reachability-Logic properties. We prove the soundness of the procedure and illustrate it by verifying a communication protocol specified in Maude.

## 1 Introduction

Reachability Logic (hereafter, RL) was introduced in a series of papers [1,2,3,4] as a means for specifying the operational semantics of programming languages and for stating reachability properties between states of program executions. Briefly, an RL formula  $\varphi \Rightarrow \varphi'$  expresses *reachability relationships* between two sets of states of a system, denoted by the *patterns*  $\varphi$  and  $\varphi'$  respectively. Depending on the interpretation of formulas, the relationships can express either programming-language semantic steps, or safety properties of programs in the languages in question. Several real language semantics (including C [5], Java [6]) have been completely specified in using an RL-based semantics in the  $\mathbb{K}$  tool [7], and nontrivial programs have been proved using an implementation of RL<sup>4</sup>.

*Contributions.* In this paper we show that RL can be used beyond its (by now, traditional) domain of programming languages. Specifically, we adapt RL for stating properties of systems described in Rewriting Logic [8] (hereafter, RWL). We propose an automatic procedure for proving RL properties of RWL specifications, we prove the soundness of our procedure, and illustrate its use by verifying RL properties of a communication protocol written in Maude [9].

<sup>4</sup> available at <http://www.matching-logic.org/index.php/Special:MatchCOnline>.

Our contribution with respect to RL is a proved-sound mechanical verification procedure. Previous works include sound and relatively complete proof systems for (the various versions of) RL, but these proof systems lack strategies for rule applications, making them unpractical for verification; our procedure can be seen as such a strategy.

With respect to RWL, our contribution is the adaptation of the above procedure for verifying RWL theories against *reachability* properties  $\varphi \Rightarrow \varphi'$ , which say that all terminating executions starting from a state in the set  $\varphi$  eventually reach a state in the set  $\varphi'$ . Both  $\varphi$  and  $\varphi'$  denote possibly infinite sets of states. We note that RL properties for RWL theories are different from the reachability properties that can be checked in Maude using the `search` command. The difference resides in the possibility of using first-order logic for constraining the initial and the final state terms, and in the interpretation of RL formulas.

*Related work.* We have already mentioned related work on RL. We now focus on related verification approaches for RWL specifications. These fall under the usual classification of verification techniques: *algorithmic* ones, which essentially consists in using an automatic model checker; *deductive* ones, which involve an interaction with a theorem prover; and *abstraction-based* ones, which consists in first reducing the state-space of a system from unmanageable (e.g., infinite/large) to manageable (e.g., finite/small), a step that typically involves human interaction, and then using a model checker on the reduced system.

Algorithmic techniques include Maude's finite-state LTL model checker [10], with its more recent extensions to the *LTL logic of rewriting* [11], and a narrowing-based symbolic model-checker for handling classes of infinite-state systems [12]. Among the deductive techniques, [13,14] propose two different approaches for reducing safety properties of RWL to equational reasoning, and then using equational reasoning tools for proving the resulting encoded properties. We note that the encoding of RWL into equational logic was proposed earlier in [15] for defining the semantics of RWL. Among the abstraction-based techniques, equational abstractions [16], and algebraic simulations [17] are key contributions.

Finally, our verification procedure uses an operation called *derivative* that consists in computing the symbolic successors of a given set of states (represented by a formula). This symbolic computation is inspired from [18,19].

*Outline.* After preliminaries in Section 2, we introduce in Section 3 the notion derivative that is essential for our approach. We then introduce in Section 4 our procedure for verifying RL properties on transition systems also defined by RL formulas, and prove its soundness. In Section 5, we adapt our approach to transition systems defined by RWL theories. We finally illustrate in Section 6 the usability of our procedure by applying it to a communication protocol described in Maude, and conclude in Section 7.

## 2 Preliminaries

### 2.1 Matching Logic

We recall the syntax and the semantics of Matching Logic (ML) as presented in [1]. Since ML is based on the many-sorted first-order logic (FOL), we recall first the basic definitions from FOL.

Given  $S$  a set of sorts, an  $S$ -sorted *first order signature*  $\Phi$  is a pair  $(\Sigma, \Pi)$ , where  $\Sigma$  is an algebraic  $S$ -sorted signature and  $\Pi$  is an indexed set of the form  $\{\Pi_w \mid w \in S^*\}$  whose elements are called *predicate symbols*, where  $\pi \in \Pi_w$  is said to have *arity*  $w$ .

A  $\Phi$ -*model* consists of a  $\Sigma$ -algebra  $M$  together with a subset  $M_p \subseteq M_{s_1} \times \dots \times M_{s_n}$  for each predicate  $p \in \Pi_w$ , where  $w = s_1 \dots s_n$ .

Next, we define the syntax of FOL formulas over a first order signature  $\Phi = (\Sigma, \Pi)$  and a possible infinite set of variables  $Var$ . We let  $S$  denote the set of sorts in  $\Phi$ ,  $Var$  an infinite  $S$ -indexed set of variable symbols (disjoint from  $\Sigma$ ), and  $T_\Sigma(Var)$  the algebra of  $\Sigma$ -terms with variables in  $Var$ .

The set of  $\Phi$ -*formulas* is defined by

$$\phi ::= \top \mid p(t_1, \dots, t_n) \mid \neg\phi \mid \phi \wedge \phi \mid (\exists V)\phi$$

where  $p$  ranges over predicate symbols  $\Pi$ , each  $t_i$  ranges over  $T_\Sigma(Var)$  of appropriate sort, and  $V$  over finite subsets of  $Var$ .

Given a first order  $\Phi$ -model  $M$ , a  $\Phi$ -formula  $\phi$ , and a valuation  $\rho : Var \rightarrow M$ , the satisfaction relation  $\rho \models \phi$  is defined as follows:

1.  $\rho \models \top$ ;
2.  $\rho \models p(t_1, \dots, t_n)$  iff  $(\rho(t_1), \dots, \rho(t_n)) \in M_p$ ;
3.  $\rho \models \neg\phi$  iff  $\rho \models \phi$  does not hold;
4.  $\rho \models \phi_1 \wedge \phi_2$  iff  $\rho \models \phi_1$  and  $\rho \models \phi_2$ ;
5.  $\rho \models (\exists V)\phi$  iff there is  $\rho' : Var \rightarrow M$  with  $\rho'(X) = \rho(X)$ , for all  $X \notin V$ , such that  $\rho' \models \phi$ .

A formula  $\phi$  is *valid in*  $M$ , denoted by  $M \models \phi$ , if it is satisfied by all valuations  $\rho$ .

We recall below the ML concepts and results used in this paper. Their presentation is based on [1].

**Definition 1 (ML Formulas).** *An ML signature  $\Phi = (\Sigma, \Pi, State)$  is a first-order signature  $(\Sigma, \Pi)$  together with a distinguished sort  $State$  for states. The set of ML-formulas over  $\Phi$  is defined by*

$$\varphi ::= \pi \mid \top \mid p(t_1, \dots, t_n) \mid \neg\varphi \mid \varphi \wedge \varphi \mid (\exists V)\varphi$$

where the basic pattern  $\pi$  ranges over  $T_{\Sigma, State}(Var)$ ,  $p$  ranges over predicate symbols  $\Pi$ , each  $t_i$  ranges over  $T_\Sigma(Var)$  of appropriate sorts, and  $V$  over finite subsets of  $Var$ .

The sort  $State$  is intended to model system states. The free occurrence of variables in ML formulas is defined as usual (i.e., like in FOL) and we let  $FreeVars(\varphi)$  denote the set of variables freely occurring in  $\varphi$ .

*Example 1.* Assume that  $S$  includes the sorts  $Nat, State$ ,  $\Sigma$  includes a binary operation symbol  $\langle \_, \_ \rangle : Nat \times Nat \rightarrow State$ , and  $\Pi$  the predicate symbols  $div$  and  $\_ > \_$ , with arguments of sort  $Nat$ . Then  $\varphi \triangleq \langle x, y \rangle \wedge (\exists z)((z > 1) \wedge div(z, x) \wedge div(z, y))$  is an ML formula. We have  $FreeVars(\varphi) = \{x, y\}$ .

**Definition 2 (ML satisfaction relation).** Given  $\Phi = (\Sigma, \Pi, State)$  an ML signature,  $M$  a  $(\Sigma, \Pi)$ -model,  $\varphi$  an ML formula,  $\gamma \in M_{State}$  a state, and  $\rho : Var \rightarrow M$  a valuation, the satisfaction relation  $(\gamma, \rho) \models \varphi$  is defined as follows:

1.  $(\gamma, \rho) \models \pi$  iff  $\rho(\pi) = \gamma$ ;
2.  $(\gamma, \rho) \models \top$ ;
3.  $(\gamma, \rho) \models p(t_1, \dots, t_n)$  iff  $(\rho(t_1), \dots, \rho(t_n)) \in M_p$ ;
4.  $(\gamma, \rho) \models \neg \varphi$  iff  $(\gamma, \rho) \not\models \varphi$  does not hold;
5.  $(\gamma, \rho) \models \varphi_1 \wedge \varphi_2$  iff  $(\gamma, \rho) \models \varphi_1$  and  $(\gamma, \rho) \models \varphi_2$ ; and
6.  $(\gamma, \rho) \models (\exists V)\varphi$  iff there is  $\rho' : Var \rightarrow M$  with  $\rho'(X) = \rho(X)$ , for all  $X \notin V$ , such that  $(\gamma, \rho') \models \varphi$ .

We let  $\llbracket \varphi \rrbracket$  denote the set  $\{\gamma \in M_{State} \mid (\exists \rho : Var \rightarrow M)(\gamma, \rho) \models \varphi\}$ .

*Example 2.* Let  $M$  be a model defined such that  $M_{Nat}$  is the set of natural numbers,  $M_{<}$  is the inequality "greater than" over natural numbers, and  $M_{div}(m, n)$  holds iff  $m$  divides  $n$ . Let  $\varphi$  denote the ML formula  $\langle x, y \rangle \wedge ((\exists z)(z > 1) \wedge div(z, x) \wedge div(z, y))$ . If we consider  $\rho(x) = 4$  and  $\rho(y) = 6$  then we have  $(\langle 4, 6 \rangle, \rho) \models \varphi$  because  $\rho(\langle x, y \rangle) = \langle 4, 6 \rangle$  and  $\rho \models ((\exists z)(z > 1) \wedge div(z, x) \wedge div(z, y))$ . We do not have  $(\langle 3, 5 \rangle, \rho) \models \varphi$  because  $\rho(\langle x, y \rangle) \neq \langle 3, 5 \rangle$ . Even if we consider  $\rho'(x) = 3$  and  $\rho'(y) = 5$  we still have  $(\langle 3, 5 \rangle, \rho') \not\models \varphi$  because there is no  $m$  greater than 1 such that  $M_{div}(m, 3)$  and  $M_{div}(m, 5)$  hold.

**Definition 3 (FOL encoding of ML).** If  $\varphi$  is an ML-formula then  $\varphi^{=?}$  is the FOL formula  $(\exists z)\varphi^=$ , where  $\varphi^=$  is obtained from  $\varphi$  by replacing each basic pattern occurrence  $\pi$  with  $z = \pi$ , and  $z$  is a variable that does not occur in  $\varphi$ .

*Example 3.* Here are a few examples of ML formulas and their FOL encodings:

$\varphi$	$\varphi^{=?}$
$(\pi_1 \wedge \phi_1) \vee (\pi_2 \wedge \phi_2)$	$(\exists z)((z = \pi_1 \wedge \phi_1) \vee (z = \pi_2 \wedge \phi_2))$
$\neg \pi$	$(\exists z)\neg(z = \pi)$
$\pi_1 \wedge \neg \pi_2$	$(\exists z)((z = \pi_1) \wedge \neg(z = \pi_2))$
$\pi \vee \neg \pi$	$(\exists z)(z = \pi \vee \neg(z = \pi))$

The relationship between ML formulas and their FOL encodings is given by the following result.

**Proposition 1.**  $\rho \models \varphi^{=?}$  iff there is  $\gamma$  such that  $(\gamma, \rho) \models \varphi$ .

*Proof.* We use the notation convention from Definition 3.

$\Leftarrow$ . If  $(\gamma, \rho) \models \varphi$  then we consider  $\rho'$  such that  $\rho'(z) = \gamma$  and  $\rho'(x) = \rho(x)$  for all  $x \neq z$ . We obtain  $\rho' \models \varphi^=$ , which implies  $\rho \models (\exists z)\varphi^=$ .

$\Rightarrow$ . Let  $\square$  be a fresh variable ( $\square \notin \text{Var}$ ) of sort *State* and  $\varphi^\square$  defined in the same way like  $\varphi^\equiv$ , but using  $\square$  instead of  $z$ . Note that  $\varphi^\square$  is defined on an extended signature. If  $\rho : \text{Var} \rightarrow M$  and  $\gamma \in M_{\text{State}}$ , then let  $\rho^\gamma : \text{Var} \cup \{\square\} \rightarrow M$  denote the extension of  $\rho$  with  $\rho(\square) = \gamma$ . By Proposition 1 in [1] we have  $\rho^\gamma \models \varphi^\square$  iff  $(\gamma, \rho) \models \varphi$ . Assume that  $\rho \models \varphi^\equiv$ . It follows that for any extension  $\rho'$  of  $\rho$  to  $\text{Var} \cup \{\square\}$  we have  $\rho' \models \varphi^\equiv$ . Since  $\varphi^\equiv$  can be obtained from  $(\exists \square)\varphi^\square$  by alpha conversion, we obtain  $\rho' \models (\exists \square)\varphi^\square$ . It follows that there exists  $\rho'' : \text{Var} \cup \{\square\} \rightarrow M$  such that  $\rho''(x) = \rho'(x)$  for  $x \neq \square$  and  $\rho'' \models \varphi^\square$ . Since  $\rho'$  extends  $\rho$ , we obtain  $\rho''(x) = \rho(x)$  for  $x \in \text{Var}$ . If we take  $\gamma = \rho''(\square)$ , then  $\rho'' = \rho^\gamma$  and hence  $(\gamma, \rho) \models \varphi$ .  $\square$

**Proposition 2.** *If  $\phi$  is a FOL (structureless) formula, then  $(\phi \wedge \varphi)^\equiv$  is equivalent to  $\phi \wedge \varphi^\equiv$ .*

*Proof.* The FOL formula  $\phi^\equiv$  is the same with  $\phi$  because  $\phi$  has no basic patterns, and hence  $\phi^\equiv$  is equivalent to  $\phi$  because the existential variable  $z$  does not occur in  $\phi$ . It follows that  $(\phi \wedge \varphi)^\equiv \triangleq (\exists z)(\phi \wedge \varphi)^\equiv$  is equivalent  $(\exists z)(\phi^\equiv \wedge \varphi^\equiv)$ , which in turn is equivalent to  $\phi \wedge (\exists z)\varphi^\equiv$ , which is the same with  $\phi \wedge \varphi^\equiv$ .

## 2.2 Reachability Logic

In this section we recall reachability-logic formulas, the transition systems that they induce, and their all-paths interpretation [4]. We consider a fixed ML signature  $\Phi = (\Sigma, \Pi, \text{State})$ , a set of variables  $\text{Var}$ , and a fixed  $\Phi$ -model  $M$ .

**Definition 4 (RL Formulas).** *An RL formula is a pair  $\varphi \Rightarrow \varphi'$  of ML-formulas.*

**Definition 5 (Transition-System Specification).** *An RL transition-system specification is a set  $\mathcal{S}$  of RL-formulas. The transition system defined by  $\mathcal{S}$  over  $M$  is  $(M_{\text{State}}, \Rightarrow_{\mathcal{S}})$ , where  $\Rightarrow_{\mathcal{S}} = \{(\gamma, \gamma') \mid (\exists \varphi \Rightarrow \varphi' \in \mathcal{S})(\exists \rho)(\gamma, \rho) \models \varphi \wedge (\gamma', \rho) \models \varphi'\}$ . We write  $\gamma \Rightarrow_{\mathcal{S}} \gamma'$  for  $(\gamma, \gamma') \in \Rightarrow_{\mathcal{S}}$ .*

*Example 4.* The following set of rules is meant to compute the gcd of two natural numbers:

$$\mathcal{S} = \{ \langle x, y \rangle \wedge x > y \wedge y > 0 \Rightarrow (\exists k)\langle y, x - k * y \rangle \wedge x \geq k * y \wedge k > 0, \\ \langle x, y \rangle \wedge y \geq x \Rightarrow \langle y, x \rangle \}$$

We further assume that  $M$  interprets  $+$  and  $*$  as being the usual operations over natural numbers;  $m - n$  is the difference between  $m$  and  $n$ , if  $m > n$ , or 0, otherwise. Examples of transitions are:  $\langle 8, 10 \rangle \Rightarrow_{\mathcal{S}} \langle 10, 8 \rangle$  and  $\langle 8, 2 \rangle \Rightarrow_{\mathcal{S}} \langle 2, 0 \rangle$  as instances of the first rule, and  $\langle 2, 2 \rangle \Rightarrow_{\mathcal{S}} \langle 2, 2 \rangle$  as an instance of the second rule.

In the sequel we consider a fixed transition system  $(M_{\text{State}}, \Rightarrow_{\mathcal{S}})$ .

**Definition 6 (Execution Paths).** *An execution path is a (possibly infinite) sequence of transitions  $\tau \triangleq \gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1 \Rightarrow_{\mathcal{S}} \dots$ .*

*If  $i \geq 0$  then  $\tau|_{i..}$  is the execution path consisting of the (possibly infinite) subsequence starting from  $\gamma_i$ , if any. An execution path is complete iff it is not a strict prefix of another execution path.*

A pair  $(\tau, \rho)$ , consisting of an execution path  $\tau \triangleq \gamma_0 \Rightarrow_S \dots$  and a valuation  $\rho$ , starts from an ML formula  $\varphi$  if  $(\gamma_0, \rho) \models \varphi$ .

*Example 5.* Examples of executions are  $\tau \triangleq \langle 8, 10 \rangle \Rightarrow_S \langle 10, 8 \rangle \Rightarrow_S \langle 8, 2 \rangle \Rightarrow_S \langle 2, 0 \rangle$  and  $\tau' \triangleq \langle 8, 10 \rangle \Rightarrow_S \langle 10, 8 \rangle \Rightarrow_S \langle 8, 2 \rangle \Rightarrow_S \langle 2, 2 \rangle \Rightarrow_S \langle 2, 2 \rangle \Rightarrow_S \dots$ . Both executions are complete.

Note that infinite execution paths are complete and hence the above definition is slightly different from that given in [4].

**Definition 7 (All-Paths Interpretation of RL formulas).** We say that a pair  $(\tau, \rho)$  satisfies an RL formula  $\varphi \Rightarrow \varphi'$ , written  $(\tau, \rho) \models \varphi \Rightarrow \varphi'$ , iff  $(\tau, \rho)$  starts from  $\varphi$  and one of the following two conditions holds: there exists  $i \geq 0$  such that  $(\gamma_i, \rho) \models \varphi'$  or  $\tau$  is infinite. We say that  $\Rightarrow_S$  satisfies  $\varphi \Rightarrow \varphi'$ , written  $\Rightarrow_S \models \varphi \Rightarrow \varphi'$ , iff  $(\tau, \rho) \models \varphi \Rightarrow \varphi'$  for all  $(\tau, \rho)$  starting from  $\varphi$ .

We let  $\llbracket \varphi \Rightarrow \varphi' \rrbracket \triangleq \{ \tau \mid (\exists \rho)(\tau, \rho) \models \varphi \Rightarrow \varphi' \}$ . If  $F$  is a set of RL formulas, then  $\llbracket F \rrbracket = \bigcup_{\varphi \Rightarrow \varphi' \in F} \llbracket \varphi \Rightarrow \varphi' \rrbracket$ .

*Example 6.* An RL formula specifying that any execution path satisfying it computes the greatest common divisor (gcd) is  $\langle x, y \rangle \Rightarrow (\exists x', y') \langle x', y' \rangle \wedge gcd(x', x, y)$ , where  $gcd$  is a predicate symbol with the interpretation:  $M_{gcd}(d, m, n)$  holds iff  $d$  is the gcd of  $m$  and  $n$ . If  $\rho(x) = 10, \rho(y) = 8$ ,  $\tau$  and  $\tau'$  are the execution paths defined in Example 5, then both  $(\tau, \rho)$  and  $(\tau', \rho)$  satisfy the given formula.

### 2.3 Rewrite Theories

Since its original formulation [21], Rewriting Logic (RWL) was defined as a *semantical framework*, suitable for describing concurrent and distributed systems, and as a *logical framework*, i.e. a meta-logic where other logics can be naturally represented. Both directions are having a dynamic and strong development.

Maude [9] is the most well-known and most used implementation of RWL. Maude is accompanied by a set of tools for analysing rewrite theories and the systems they describe.

In this paper we use RWL as a semantical framework. Here we briefly recall the definition for (a particular case of) rewrite theories and their rewriting relation. A *rewrite theory*  $\mathcal{R} = (\Sigma, E \cup A, R)$  consists of a signature  $\Sigma$ , a set of equations  $E$ , a set of axioms  $A$ , e.g., associativity, commutativity, identity or combinations of these, a set of rewrite rules  $R$  of the form  $l \rightarrow r$  **if**  $b$ , where  $l$  and  $r$  are terms with variables and  $b$  is a term of a distinguished sort *Bool*. We further assume that there is a special constant *true* of sort *Bool*.

In this paper we shall consider only rewrite theories  $\mathcal{R} = (\Sigma, E \cup A, R)$  with a distinguished sort *State* such that  $R$  is topmost w.r.t. *State*. We use the standard notation for RWL artefacts:  $=_{E \cup A}$  denotes the equality modulo the equations given by  $E$  and  $A$ ,  $T_{\Sigma, E \cup A}(X)$  denotes the set of  $=_{E \cup A}$ -equivalences classes of  $\Sigma$ -terms with variables in  $X$ ,  $T_{\Sigma, E \cup A} \triangleq T_{\Sigma, E \cup A}(\emptyset)$  is the set of  $=_{E \cup A}$ -equivalences classes of ground  $\Sigma$ -terms, and  $FreeVars(t)$  denotes the set of variables occurring

in the term  $t^5$ . The relation  $\rightarrow_{\mathcal{R}}$  denotes the one-step rewriting relation defined by applying a rule from  $R$  modulo axioms  $E \cup A$  over ground terms of sort *State*:  $[u] \rightarrow_{\mathcal{R}} [v]$  iff there are a rule  $l \rightarrow r$  in  $R$  and a (ground) substitution  $\sigma : \text{FreeVars}(l, r, b) \rightarrow T_{\Sigma, E \cup A}$  such that  $\sigma(l) =_{E \cup A} \sigma(u)$ ,  $\sigma(r) =_{E \cup A} \sigma(v)$ , and  $\sigma(b) =_{E \cup A} \text{true}$ .

### 3 Derivatives of ML and RL Formulas

The notion of derivative is essential for our approach. Roughly speaking, the derivative of a formula specifies states/execution paths obtained from those satisfying the initial formula after executing one step. For the remaining part of this section we consider a fixed transition system specification  $\mathcal{S}$  and its associated transition system  $(M_{\text{State}}, \Rightarrow_{\mathcal{S}})$  over a fixed model  $M$ .

**Assumption 1** *In what follows we consider only ML formulas  $\varphi$  with the following property: if  $\varphi$  does not occur as a member of a rule in  $\mathcal{S}$  and  $\varphi_l \Rightarrow \varphi_r \in \mathcal{S}$  then  $\text{FreeVars}(\varphi) \cap \text{FreeVars}(\varphi_l, \varphi_r) = \emptyset$ . This is not a real restriction since the free variable in rules can always be renamed.*

**Definition 8 (Semantic Definition of Derivatives for RL Formulas).** *We say that  $\varphi_1 \Rightarrow \varphi'$  is a  $\mathcal{S}$ -derivative of  $\varphi \Rightarrow \varphi'$  if for all  $(\tau_1, \rho) \models \varphi_1 \Rightarrow \varphi'$  there is  $(\tau, \rho) \models \varphi \Rightarrow \varphi'$  such that  $\tau_1 = \tau|_{1..}$ .*

*Example 7.* An  $\mathcal{S}$ -derivative for  $\langle x, y \rangle \Rightarrow (\exists x', y') \langle x', y' \rangle \wedge \text{gcd}(x', x, y)$  is the following formula:  $\langle y, x - y \rangle \wedge x > y \wedge y > 0 \Rightarrow (\exists x', y') \langle x', y' \rangle \wedge \text{gcd}(x', x, y)$ .

**Definition 9 (Complete Sets of Derivatives).** *A set  $D$  of  $\mathcal{S}$ -derivatives of  $\varphi \Rightarrow \varphi'$  is complete iff  $\llbracket \varphi_1 \Rightarrow \varphi' \rrbracket \subseteq \llbracket D \rrbracket$  for each  $\mathcal{S}$ -derivative  $\varphi_1 \Rightarrow \varphi'$  of  $\varphi \Rightarrow \varphi'$ .*

*Example 8.* The set

$$\{(\exists k) \langle y, x - k * y \rangle \wedge y > 0 \wedge x \geq k * y \wedge k > 0 \Rightarrow (\exists x', y') \langle x', y' \rangle \wedge \text{gcd}(x', x, y), \\ \langle y, x \rangle \wedge y \geq x \Rightarrow (\exists x', y') \langle x', y' \rangle \wedge \text{gcd}(x', x, y)\}$$

is a complete set of  $\mathcal{S}$ -derivatives for  $\langle x, y \rangle \Rightarrow (\exists x', y') \langle x', y' \rangle \wedge \text{gcd}(x', x, y)$ .

The next definition and lemmas provide us with syntactical means of computing complete sets of derivatives for RL formulas.

**Definition 10 (Syntactic Definition of Derivative for RL Formulas).** *If  $\varphi$  is an ML formula then*

$$\Delta_{\mathcal{S}}(\varphi) \triangleq \{(\exists \text{FreeVars}(\varphi_l, \varphi_r))(\varphi_l \wedge \varphi)^{=?} \wedge \varphi_r \mid \varphi_l \Rightarrow \varphi_r \in \mathcal{S}\}.$$

*If  $\varphi \Rightarrow \varphi'$  is an RL-formula then*

$$\Delta_{\mathcal{S}}(\varphi \Rightarrow \varphi') \triangleq \{\varphi_1 \Rightarrow \varphi' \mid \varphi_1 \in \Delta_{\mathcal{S}}(\varphi)\}.$$

<sup>5</sup> For the sake of uniformity, we keep the notation  $\text{FreeVars}(t)$  for the set of variables occurring in the term  $t$ . This is a consistent notation since all occurrences of variables in term are considered as being free.  $\text{FreeVars}(t_1, t_2)$  is  $\text{FreeVars}(t_1) \cup \text{FreeVars}(t_2)$ .



**Lemma 1.** *If  $\varphi_1 \in \Delta_{\mathcal{S}}(\varphi)$  then  $\varphi_1 \Rightarrow \varphi'$  is a  $\mathcal{S}$ -derivative of  $\varphi \Rightarrow \varphi'$ .*

*Proof.* Assume that  $\varphi_1$  is  $(\exists \text{FreeVars}(\varphi_l, \varphi_r))(\varphi_l \wedge \varphi)^{=?} \wedge \varphi_r$  for some  $\varphi_l \Rightarrow \varphi_r \in \mathcal{S}$ . If  $\llbracket \varphi_1 \rrbracket = \emptyset$  then  $\varphi_1 \Rightarrow \varphi'$  is a  $\mathcal{S}$ -derivative of  $\varphi \Rightarrow \varphi'$  by Definition 8. Assume  $\llbracket \varphi_1 \rrbracket \neq \emptyset$ , i.e. there exists  $(\tau_1, \rho_1)$  starting from  $\varphi_1$ , with  $\tau_1 \triangleq \gamma_1 \Rightarrow_{\mathcal{S}} \dots$ . Then

$$\begin{array}{ll}
(\gamma_1, \rho_1) \models \varphi_1 & \longleftrightarrow \\
(\exists \rho)(\gamma_1, \rho) \models ((\varphi_l \wedge \varphi)^{=?} \wedge \varphi_r) & \longleftrightarrow \\
(\exists \rho)(\rho \models (\varphi_l \wedge \varphi)^{=?} \wedge (\gamma_1, \rho) \models \varphi_r) & \longleftrightarrow \\
(\exists \rho)((\exists \gamma_0)(\gamma_0, \rho) \models (\varphi_l \wedge \varphi) \wedge (\gamma_1, \rho) \models \varphi_r) & \longleftrightarrow \\
(\exists \rho)((\exists \gamma_0)((\gamma_0, \rho_1) \models \varphi \wedge (\gamma_0, \rho) \models \varphi_l) \wedge (\gamma_1, \rho) \models \varphi_r) & \longleftrightarrow \\
(\exists \gamma_0)(\gamma_0, \rho_1) \models \varphi \wedge (\exists \rho)((\gamma_0, \rho) \models \varphi_l \wedge (\gamma_1, \rho) \models \varphi_r) & \longrightarrow \\
(\exists \gamma_0)(\gamma_0, \rho_1) \models \varphi \wedge \gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1 & 
\end{array}$$

where, by Assumption 1, we may w.l.o.g. choose  $\rho$  such that  $\rho(x) = \rho_1(x)$  for all  $x \notin \text{FreeVars}(\varphi_l, \varphi_r)$ . Hence there is  $\tau = \gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1 \Rightarrow_{\mathcal{S}} \dots$  such that  $\tau|_{1..} = \tau_1$  and  $(\gamma_0, \rho_1) \models \varphi$ . If  $\tau_1$  is infinite then  $\tau$  is infinite. If  $(\exists i \geq 1)(\gamma_i, \rho_1) \models \varphi'$  then  $(\exists i \geq 0)(\gamma_i, \rho_1) \models \varphi'$ . So, finally, we obtain that  $(\tau_1, \rho_1) \models \varphi_1 \Rightarrow \varphi'$  implies  $(\tau, \rho_1) \models \varphi \Rightarrow \varphi'$ . Since  $\gamma_1$  and  $\rho_1$  have been chosen arbitrary, we may conclude that  $\varphi_1 \Rightarrow \varphi'$  is a  $\mathcal{S}$ -derivative of  $\varphi \Rightarrow \varphi'$ .  $\square$

**Lemma 2.** *Let  $\varphi_1 \Rightarrow \varphi'$  be a  $\mathcal{S}$ -derivative of  $\varphi \Rightarrow \varphi'$ ,  $\tau_1$  be an execution path and  $\rho$  a valuation. If  $(\tau_1, \rho) \models \varphi_1 \Rightarrow \varphi'$  then there is  $\varphi'_1 \in \Delta_{\mathcal{S}}(\varphi)$  such that  $(\tau_1, \rho) \models \varphi'_1 \Rightarrow \varphi'$ .*

*Proof.* Suppose that  $(\tau_1, \rho) \models \varphi_1 \Rightarrow \varphi'$  and  $\tau_1 \triangleq \gamma_1 \Rightarrow_{\mathcal{S}} \dots$ . Then  $(\tau_1, \rho)$  starts from  $\varphi$  and one of the following two claims holds: a) there exists  $i \geq 1$  such that  $(\gamma_i, \rho) \models \varphi'$  or b)  $\tau$  is infinite. So, to prove that  $(\tau_1, \rho) \models \varphi'_1 \Rightarrow \varphi'$  it is enough to prove that  $(\tau_1, \rho)$  starts from some  $\varphi'_1 \in \Delta_{\mathcal{S}}(\varphi)$ . The pair  $(\tau_1, \rho)$  can be extended to  $(\tau, \rho)$  such that  $\tau|_{1..} = \tau_1$  and  $(\tau, \rho) \models \varphi \Rightarrow \varphi'$  by the definition of the  $\mathcal{S}$ -derivative. It follows that there is  $\gamma_0$  such that  $\tau \triangleq \gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1 \Rightarrow_{\mathcal{S}} \dots$ ,  $(\gamma_0, \rho) \models \varphi$ , and  $(\gamma_1, \rho) \models \varphi_1$ . There is  $\varphi_l \Rightarrow \varphi_r \in \mathcal{S}$  and  $\rho'$  such that  $(\gamma_0, \rho') \models \varphi_l$  and  $(\gamma_1, \rho') \models \varphi_r$  by the definition of  $\Rightarrow_{\mathcal{S}}$ . By Assumption 1, we may w.l.o.g. choose  $\rho'$  such that  $\rho'(x) = \rho(x)$  for all  $x \notin \text{FreeVars}(\varphi_l, \varphi_r)$ . Hence  $(\gamma_0, \rho') \models \varphi \wedge \varphi_l$ . We take  $\varphi'_1 \triangleq (\exists \text{FreeVars}(\varphi_l, \varphi_r))(\varphi \wedge \varphi_l)^{=?} \wedge \varphi_r$ . We obviously have  $\varphi'_1 \in \Delta_{\mathcal{S}}(\varphi)$  and  $(\gamma_1, \rho) \models \varphi'_1$  because there exists  $\rho'$  (defined above) such that  $(\gamma_1, \rho') \models (\varphi \wedge \varphi_l)^{=?} \wedge \varphi_r$ . Since  $\tau_1 = \gamma_1 \Rightarrow_{\mathcal{S}} \dots$ , it follows that  $(\tau_1, \rho)$  starts from  $\varphi'_1 \in \Delta_{\mathcal{S}}(\varphi)$ , which implies  $(\tau_1, \rho) \models \varphi'_1 \Rightarrow \varphi'$ . Since  $(\tau_1, \rho)$  has been chosen arbitrary, the conclusion of the lemma follows.  $\square$

**Lemma 3.**  *$\Delta_{\mathcal{S}}(\varphi \Rightarrow \varphi')$  is a complete set of  $\mathcal{S}$ -derivatives for  $\varphi \Rightarrow \varphi'$ .*

*Proof.* The conclusion directly follows from Lemma 1 and Lemma 2.

*Example 9.*

$$\begin{aligned}
\Delta_{\mathcal{S}}(\langle x, y \rangle \wedge y \geq 0) = & \{ (\exists x', y', k) \langle y', x' - k * y' \rangle \wedge \langle x', y' \rangle = \langle x, y \rangle \wedge y' > 0 \wedge x' > y' \\
& \wedge x' \geq k * y' \wedge k > 0 \wedge y \geq 0, \\
& (\exists x', y') \langle y', x' \rangle \wedge \langle x', y' \rangle = \langle x, y \rangle \wedge y' \geq x' \wedge y \geq 0 \},
\end{aligned}$$

which can be simplified to

$$\Delta_{\mathcal{S}}(\langle x, y \rangle \wedge y \geq 0) = \{(\exists k)(\langle y, x - k * y \rangle \wedge y > 0 \wedge x \geq k * y \wedge k > 0, \langle y, x \rangle \wedge y \geq x \wedge y \geq 0)\},$$

using the implications  $M \models \langle x', y' \rangle = \langle x, y \rangle \longrightarrow (x = x' \wedge y = y')$ ,  $M \models (x \geq k * y \wedge k > 0) \longrightarrow x > y$  and  $M \models y > 0 \longrightarrow y \geq 0$ , where  $M$  is the model defined in the previous examples.

$\Delta_{\mathcal{S}}(\langle x, y \rangle \Rightarrow (\exists x', y')(\langle x', y' \rangle \wedge gcd(x', x, y)))$  is the set given in Example 8.

The following definition of  $\mathcal{S}$ -derivability is used in our verification procedure for RL formulas. The lemma following it gives an equivalent characterisation in terms of FOL, which enables the checking of  $\mathcal{S}$ -derivability using SMT solvers.

**Definition 11 ( $\mathcal{S}$ -derivability of ML-formulas).** An ML formula  $\varphi$  is  $\mathcal{S}$ -derivable iff there is at least a transition starting from it, i.e., there exist a model  $(\gamma, \rho) \models \varphi$  and a transition  $\gamma \Rightarrow_{\mathcal{S}} \gamma_1$ .

**Lemma 4.**  $\varphi$  is  $\mathcal{S}$ -derivable iff  $\bigvee_{\varphi_1 \in \Delta_{\mathcal{S}}(\varphi)} \varphi_1^{=?}$  is satisfiable.

*Proof.* The following equivalences hold:

$$\begin{aligned} \bigvee_{\varphi_1 \in \Delta_{\mathcal{S}}(\varphi)} \varphi_1^{=?} \text{ is satisfiable} & \iff \\ (\exists \rho_1) \rho_1 \models \bigvee_{\varphi_1 \in \Delta_{\mathcal{S}}(\varphi)} \varphi_1^{=?} & \iff \\ (\exists \rho_1)(\exists \varphi_1 \in \Delta_{\mathcal{S}}(\varphi)) \rho_1 \models \varphi_1^{=?} & \iff \\ (\exists \rho_1)(\exists \varphi_l \Rightarrow \varphi_r \in \mathcal{S}) \rho_1 \models ((\exists FreeVars(\varphi_l, \varphi_r))(\varphi_l \wedge \varphi)^{=?} \wedge \varphi_r)^{=?} & \iff \\ (\exists \rho_1)(\exists \varphi_l \Rightarrow \varphi_r \in \mathcal{S})(\exists \gamma_1)(\gamma_1, \rho_1) \models (\exists FreeVars(\varphi_l, \varphi_r))(\varphi_l \wedge \varphi)^{=?} \wedge \varphi_r & \end{aligned}$$

Then, we have:

$$\begin{aligned} (\gamma_1, \rho_1) \models (\exists FreeVars(\varphi_l, \varphi_r))(\varphi_l \wedge \varphi)^{=?} \wedge \varphi_r & \iff \\ (\exists \rho)(\gamma_1, \rho) \models (\varphi_l \wedge \varphi)^{=?} \wedge \varphi_r & \iff \\ (\exists \rho) \rho \models (\varphi_l \wedge \varphi)^{=?} \wedge (\gamma_1, \rho) \models \varphi_r & \iff \\ (\exists \rho)(\exists \gamma)(\gamma, \rho) \models (\varphi_l \wedge \varphi) \wedge (\gamma_1, \rho) \models \varphi_r & \iff \\ (\exists \rho)(\exists \gamma)(\gamma, \rho) \varphi \wedge (\gamma, \rho) \models \varphi_l \wedge (\gamma_1, \rho) \models \varphi_r & \iff \\ (\exists (\gamma, \rho))(\gamma, \rho) \models \varphi \wedge \gamma \Rightarrow_{\mathcal{S}} \gamma_1 & \end{aligned}$$

where, by the definition of  $\models$ ,  $\rho$  satisfies  $\rho(x) = \rho_1(x)$  for all  $x \notin FreeVars(\varphi_l, \varphi_r)$ . We obtained that  $\bigvee_{\varphi_1 \in \Delta_{\mathcal{S}}(\varphi)} \varphi_1^{=?}$  is satisfiable iff there exists  $(\gamma, \rho)$  such that  $(\gamma, \rho) \models \varphi$  and there exists  $\gamma_1$  such that  $\gamma \Rightarrow_{\mathcal{S}} \gamma_1$ , i.e., iff  $\varphi$  is  $\mathcal{S}$ -derivable. The proof of the lemma is finished now.  $\square$

Lemma 4 also shows the strong relationship between the  $\mathcal{S}$ -derivability of a ML formula  $\varphi$  and the  $\mathcal{S}$ -derivatives of RL-formulas  $\varphi \Rightarrow \varphi'$ . Hence it does make sense to name the elements of the set  $\Delta_{\mathcal{S}}(\varphi)$  as being  $\mathcal{S}$ -derivatives of  $\varphi$ .

The notion of totality, defined below, is essential for the soundness of our verification procedure. Intuitively, a transition-system specification  $\mathcal{S}$  is total if its rules cover all models  $(\gamma, \rho)$  of any  $\mathcal{S}$ -derivable formula  $\varphi$ . For instance, if  $\langle x, y \rangle \wedge y \neq 0 \Rightarrow \langle y, x \% y \rangle \in \mathcal{S}$  then in order to be total  $\mathcal{S}$  must also include a rule for the case  $y = 0$ .

**Definition 12.**  $\mathcal{S}$  is total iff for each  $\mathcal{S}$ -derivable  $\varphi$  and each pair  $(\gamma, \rho)$  such that  $(\gamma, \rho) \models \varphi$ , there is  $\gamma_1$  such that  $\gamma \Rightarrow_{\mathcal{S}} \gamma_1$ .

Note the difference between  $\mathcal{S}$ -derivability and totality:  $\mathcal{S}$ -derivability requires to have at least one transition starting from  $\varphi$  and the totality requires to have at least one transition starting from  $\gamma$  for any model  $(\gamma, \rho)$  of  $\varphi$ .

The next result enables the use of SMT solvers for checking totality.

**Proposition 3.**  $\mathcal{S}$  is total iff for each  $\varphi$  derivable,

$$M \models \varphi^{=?} \longrightarrow \bigvee_{\varphi_1 \in \Delta_{\mathcal{S}}(\varphi)} \varphi_1^{=?}.$$

*Proof.* We use the notation convention in Definition 3.

$$\begin{aligned} M \models \varphi^{=?} &\longrightarrow \bigvee_{\varphi_1 \in \Delta_{\mathcal{S}}(\varphi)} \varphi_1^{=?} && \longleftrightarrow \\ (\forall \rho) \rho \models \varphi^{=?} &\longrightarrow \rho \models \bigvee_{\varphi_1 \in \Delta_{\mathcal{S}}(\varphi)} \varphi_1^{=?} && \longleftrightarrow \\ (\forall \rho)(\exists \gamma)(\gamma, \rho) \models \varphi &\longrightarrow && \\ (\exists \varphi_l \Rightarrow \varphi_r \in \mathcal{S}) \rho \models (\exists \text{Free Vars}(\varphi_l, \varphi_r))((\varphi_l \wedge \varphi)^{=?} \wedge \varphi_r)^{=?} && \longleftrightarrow \\ (\forall \rho)(\exists \gamma)(\gamma, \rho) \models \varphi &\longrightarrow && \\ (\exists \varphi_l \Rightarrow \varphi_r \in \mathcal{S}) \rho \models (\exists \text{Free Vars}(\varphi_l, \varphi_r))(\varphi_l \wedge \varphi)^{=?} \wedge \varphi_r^{=?} && \longleftrightarrow \quad (1) \\ (\forall \rho)(\exists \gamma)(\gamma, \rho) \models \varphi &\longrightarrow && \\ (\exists \varphi_l \Rightarrow \varphi_r \in \mathcal{S})(\exists \rho') \rho' \models (\varphi_l \wedge \varphi)^{=?} \wedge (\exists \gamma_1)(\gamma_1, \rho') \models \varphi_r && \longleftrightarrow \quad (2) \\ (\forall \rho)(\exists \gamma)(\gamma, \rho) \models \varphi &\longrightarrow && \\ (\exists \varphi_l \Rightarrow \varphi_r \in \mathcal{S})(\exists \rho')(\gamma, \rho') \models (\varphi_l \wedge \varphi) \wedge (\exists \gamma_1)(\gamma_1, \rho') \models \varphi_r && \longleftrightarrow \quad (3) \\ (\forall \rho)(\exists \gamma)(\gamma, \rho) \models \varphi &\longrightarrow (\exists \gamma_1) \gamma \Rightarrow_{\mathcal{S}} \gamma_1 \end{aligned}$$

where, by Assumption 1, we may w.l.o.g. choose  $\rho'$  such that  $\rho'(x) = \rho(x)$  for all  $x \notin \text{Free Vars}(\varphi_l, \varphi_r)$ , which implies  $(\gamma', \rho') \models \varphi$  iff  $\gamma' = \gamma$  and  $(\gamma, \rho) \models \varphi$ . Therefore in the equivalence (2)  $\longleftrightarrow$  (3) we could take  $(\gamma, \rho') \models (\varphi_l \wedge \varphi)$ . The equivalence (1) follows by Proposition 2.  $\square$

## 4 A Procedure for Verifying RL Properties

We now introduce our procedure for verifying RL properties on transition systems also defined by RL formulas. We assume given an ML signature  $\Phi$  and  $\Phi$ -model  $M$ .

**procedure** `prove`( $\mathcal{S}, G_0, G$ )

```

1: if  $G = \emptyset$  then return success
2:   else choose  $\varphi \Rightarrow \varphi' \in G$ 
3:     if  $M \models \varphi \longrightarrow \varphi'$  then return prove( $\mathcal{S}, G_0, G \setminus \{\varphi \Rightarrow \varphi'\}$ )
4:     else if there is  $\varphi_c \Rightarrow \varphi'_c \in G_0$  s. t.  $M \models \varphi \longrightarrow (\exists \text{FreeVars}(\varphi_c))\varphi_c$  then
5:       return prove( $\mathcal{S}, G_0, G \setminus \{\varphi \Rightarrow \varphi'\} \cup \Delta_{\varphi_c \Rightarrow \varphi'_c}(\varphi \Rightarrow \varphi')$ )
6:     else if  $\varphi$  is  $\mathcal{S}$ -derivable then
7:       return prove( $\mathcal{S}, G_0, G \setminus \{\varphi \Rightarrow \varphi'\} \cup \Delta_{\mathcal{S}}(\varphi \Rightarrow \varphi')$ )
8:     else return failure.

```

**Fig. 1.** RL verification procedure.

The soundness result, stated below, says that if the procedure returns **success** when presented with a given input (consisting of a transition system RL specification  $\mathcal{S}$ , a set of goals  $G_0$ , and the  $\mathcal{S}$ -derivatives of  $G_0$ ), then the transition system  $\Rightarrow_{\mathcal{S}}$  satisfies all the goals. We shall need the following lemma.

**Lemma 5 (Coverage by Derivatives).** *Any computation  $\tau \triangleq \gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1 \Rightarrow_{\mathcal{S}} \dots$  with  $(\tau, \rho)$  starting from  $\varphi$  is "covered" by derivatives, i.e., there exists a sequence  $\varphi_0, \varphi_1, \dots$  of ML formulas such that*

1.  $\varphi_0 = \varphi$
2.  $\varphi_{i+1} \in \Delta_{\mathcal{S}}(\varphi_i)$ ,  $i = 0, 1, \dots$
3.  $(\gamma_i, \rho) \models \varphi_i$ ,  $i = 0, 1, \dots$

*Proof.* We proof by induction on  $i$  the following property:

$$(\forall i \geq 0)(\gamma_i, \rho) \models \varphi_i \longrightarrow (\exists \varphi_{i+1} \in \Delta_{\mathcal{S}}(\varphi_i)) \wedge (\gamma_{i+1}, \rho) \models \varphi_{i+1} \quad (4)$$

Assuming  $\varphi_i$  defined we set  $\varphi_{i+1} \triangleq (\exists \text{FreeVars}(\varphi_l, \varphi_r))(\varphi_i \wedge \varphi_l) \stackrel{?}{=} \wedge \varphi_r \in \Delta_{\mathcal{S}}(\varphi_i)$ , where  $\varphi_l \Rightarrow \varphi_r \in \mathcal{S}$  is the rule used in  $\gamma_i \Rightarrow_{\mathcal{S}} \gamma_{i+1}$ . We show that  $(\gamma_i, \rho) \models \varphi_i$  implies  $(\gamma_{i+1}, \rho) \models \varphi_{i+1}$ . There is  $\rho'$  such that  $(\gamma_i, \rho') \models \varphi_l$  and  $(\gamma_{i+1}, \rho') \models \varphi_r$  by the definition of  $\Rightarrow_{\mathcal{S}}$ . Since  $\text{FreeVars}(\varphi_l, \varphi_r) \cap \text{FreeVars}(\varphi_i) = \emptyset$  by Assumption 1, we may choose w.l.o.g.  $\rho'$  such that  $\rho'(x) = \rho(x)$  for  $x \notin \text{FreeVars}(\varphi_l, \varphi_r)$ . This implies that  $(\gamma_i, \rho) \models (\exists \text{FreeVars}(\varphi_l, \varphi_r))(\varphi_i \wedge \varphi_l)$  and  $(\gamma_{i+1}, \rho) \models (\exists \text{FreeVars}(\varphi_l, \varphi_r))\varphi_r$ . Hence  $(\gamma_{i+1}, \rho) \models (\exists \text{FreeVars}(\varphi_l, \varphi_r))(\varphi_i \wedge \varphi_l) \stackrel{?}{=} \wedge \varphi_r$ , i.e.  $(\gamma_{i+1}, \rho) \models \varphi_{i+1}$ . This finishes the proof by induction.

The conclusion of the lemma follows by observing that for  $i = 0$  the properties required by (4).  $\square$

**Definition 13 (Digraph Associated to prove).** *The digraph  $\mathcal{G}$  associated to a successful run of the procedure `prove`, given in Figure 1, is defined as follows:*

- the vertices consists of  $G_0$  together with all sets  $G$  collected during the recursive calls of the procedure `prove`;
- there are arcs from the current formula  $\varphi \Rightarrow \varphi'$  to the each derivative in  $\Delta_{\mathcal{S}}(\varphi_c \Rightarrow \varphi'_c)$  whenever the condition on the line 4 holds;

- there is an arc from the current formula  $\varphi \Rightarrow \varphi'$  to each formula in  $\Delta_{\mathcal{S}}(\varphi \Rightarrow \varphi')$  added by the recursive call at line 7, whenever the condition on line 4 does not hold and that on the line 6 holds.

One can see that all formulas in  $\mathcal{G}$  belong to cycles, except for the ones satisfying the condition on line 3 which are sinks in the graph. Moreover, any maximal walk starting from a source, i.e. a formula in  $G_0$ , either is infinite or ends in a sink.

We first prove the next lemma that states that any sequence of derivatives, starting from the left-hand side of a initial goal, is covered by a walk in  $\mathcal{G}$ .

**Lemma 6.** *Assume that the call  $\text{prove}(\mathcal{S}, G_0, \Delta_{\mathcal{S}}(G_0))$  successfully finishes for  $G_0$  such that each  $\varphi_c \Rightarrow \varphi'_c \in G_0$  is  $\mathcal{S}$ -derivable and satisfies  $\text{FreeVars}(\varphi'_c) \subseteq \text{FreeVars}(\varphi_c)$ . Let  $\mathcal{G}$  the digraph associated to this run (cf. Definition 13).*

*Then any sequence of ML-formulas  $\varphi_0, \varphi_1, \dots$ , with  $\varphi_0 \Rightarrow \varphi'_0 \in G_0$  and  $\varphi_{i+1} \in \Delta_{\mathcal{S}}(\varphi_i)$  for  $i = 0, 1, \dots$ , is "covered" by  $\mathcal{G}$ , i.e., there exists a walk  $\bar{\varphi}_0 \Rightarrow \varphi'_0, \bar{\varphi}_1 \Rightarrow \varphi'_1, \dots$  in  $\mathcal{G}$  such that*

- there is  $j$  with  $\bar{\varphi}_j \Rightarrow \varphi'_0$  a sink, i.e.  $M \models \bar{\varphi}_j \longrightarrow \varphi'_0$ , and  $M \models \varphi_i \longrightarrow (\exists \text{FreeVars}(\bar{\varphi}_i))\bar{\varphi}_i$  for all  $i \leq j$ , or
- $M \models \varphi_i \longrightarrow (\exists \text{FreeVars}(\bar{\varphi}_i))\bar{\varphi}_i$  for all  $i \geq 0$ .

*Proof.* We prove first the following property  $\diamond$ :

if  $\bar{\varphi}_0 \Rightarrow \varphi'_0, \dots, \bar{\varphi}_n \Rightarrow \varphi'_0$  is a walk in  $\mathcal{G}$  such that  $M \models \varphi_i \longrightarrow (\exists \text{FreeVars}(\bar{\varphi}_i))\bar{\varphi}_i$ , for all  $i \leq n$ , then one of the following claims holds:

- $M \models \bar{\varphi}_n \longrightarrow \varphi'_0$  or
- there exists  $\bar{\varphi}_{n+1}$  and an arrow from  $\bar{\varphi}_n$  to  $\bar{\varphi}_{n+1}$  in  $\mathcal{G}$  such that  $M \models \varphi_{n+1} \longrightarrow (\exists \text{FreeVars}(\bar{\varphi}_{n+1}))\bar{\varphi}_{n+1}$ .

By the definition of  $\mathcal{G}$ , we distinguish three cases corresponding to the three **if** statements in the procedure **prove**:

A)  $\bar{\varphi}_n \Rightarrow \varphi'_0$  is a sink, i.e.,  $M \models \bar{\varphi}_n \longrightarrow \varphi'_0$ . Then 1. in  $\diamond$  holds.

B)  $M \models \bar{\varphi}_n \longrightarrow (\exists \text{FreeVars}(\varphi_c))\varphi_c$  for some  $\varphi_c \Rightarrow \varphi'_c \in G_0$ , and there is an arc from  $\bar{\varphi}_n \Rightarrow \varphi'_0$  to each  $\mathcal{S}$ -derivative of  $\varphi_c \Rightarrow \varphi'_c$ . Let  $\bar{\varphi}_{n+1} \Rightarrow \varphi'_c$  be the derivative of  $\varphi_c \Rightarrow \varphi'_c$  corresponding to the rule  $\varphi_l \Rightarrow \varphi_r \in \mathcal{S}$  used in the definition of  $\varphi_{n+1}$ . We have  $\varphi_{n+1} \triangleq (\exists \text{FreeVars}(\varphi_l, \varphi_r))(\varphi_n \wedge \varphi_l)^{=?} \wedge \varphi_r$  and  $\bar{\varphi}_{n+1} \triangleq (\exists \text{FreeVars}(\varphi_l, \varphi_r))(\bar{\varphi}_n \wedge \varphi_l)^{=?} \wedge \varphi_r$ .

In  $\Delta_{\varphi_c \Rightarrow \varphi'_c}(\varphi \Rightarrow \varphi')$  the RL formula  $\varphi_c \Rightarrow \varphi'_c$  is used in the same way the rules  $\mathcal{S}$  are used, so obviously it respects the same constraints as the rules  $\mathcal{S}$ , i.e.  $\text{FreeVars}(\varphi) \cap \text{FreeVars}(\varphi_c) = \emptyset$ . In particular, we have  $\text{FreeVars}(\bar{\varphi}_n) \cap \text{FreeVars}(\varphi_c) = \emptyset$ .

We prove now that  $M \models \varphi_n \longrightarrow (\exists \text{FreeVars}(\bar{\varphi}_n))\bar{\varphi}_n$ , which holds by the hypothesis of  $\diamond$ , and  $M \models \bar{\varphi}_n \longrightarrow (\exists \text{FreeVars}(\varphi_c))\varphi_c$ , which holds by the above assumption, imply  $M \models \varphi_n \longrightarrow (\exists \text{FreeVars}(\varphi_c))\varphi_c$  ( $\clubsuit$ ). If  $\llbracket \varphi_n \rrbracket = \emptyset$  then ( $\clubsuit$ ) obviously holds. Assume that there is  $(\gamma, \rho) \models \varphi_n$ . Since  $M \models \varphi_n \longrightarrow (\exists \text{FreeVars}(\bar{\varphi}_n))\bar{\varphi}_n$ , there is  $\rho'$  such that  $(\gamma, \rho') \models \bar{\varphi}_n$  and  $\rho'(x) = \rho(x)$ , for all  $x \notin \text{FreeVars}(\bar{\varphi}_n)$ . Since  $M \models \bar{\varphi}_n \longrightarrow (\exists \text{FreeVars}(\varphi_c))\varphi_c$ , there is  $\rho''$  such that

$(\gamma, \rho'') \models \varphi_c$  and  $\rho''(x) = \rho'(x)$ , for all  $x \notin \text{FreeVars}(\varphi_c)$ . It follows that  $\rho''(x) = \rho(x)$ , for all  $x \notin \text{FreeVars}(\overline{\varphi}_n, \varphi_c)$  and hence  $(\gamma, \rho) \models (\exists \text{FreeVars}(\overline{\varphi}_n, \varphi_c))\varphi_c$ . But, the formula  $(\exists \text{FreeVars}(\overline{\varphi}_n, \varphi_c))\varphi_c$  is equivalent to  $(\exists \text{FreeVars}(\varphi_c))\varphi_c$  because  $\text{FreeVars}(\overline{\varphi}_n) \cap \text{FreeVars}(\varphi_c) = \emptyset$ . Now  $\clubsuit$  is proved. We obtain that  $M \models \varphi_{n+1} \longrightarrow (\exists \text{FreeVars}(\varphi_l, \varphi_r, \varphi_c))(\varphi_c \wedge \varphi_l) \stackrel{?}{=} \wedge \varphi_r$  using  $\clubsuit$ , i.e.,  $M \models \varphi_{n+1} \longrightarrow (\exists \text{FreeVars}(\overline{\varphi}_{n+1}))\overline{\varphi}_{n+1}$ . It follows that 2. in  $\diamond$  holds.

C)  $\overline{\varphi}_n$  is  $\mathcal{S}$ -derivable and there is an arc from from  $\overline{\varphi}_n \Rightarrow \overline{\varphi}'_0$  to each derivative in  $\Delta_{\mathcal{S}}(\overline{\varphi}_n \Rightarrow \overline{\varphi}'_0)$ . Then we set  $\overline{\varphi}_{n+1} \Rightarrow \overline{\varphi}'_0$  to be the derivative of  $\overline{\varphi}_n \Rightarrow \overline{\varphi}'_0$  corresponding to the rule  $\varphi_l \Rightarrow \varphi_r \in \mathcal{S}$  used in the definition of  $\varphi_{n+1}$ . We obtain  $M \models \varphi_{n+1} \longrightarrow (\exists \text{FreeVars}(\overline{\varphi}_{n+1}))\overline{\varphi}_{n+1}$  in a similar way to that of the case B, which implies that 2. in  $\diamond$  holds.

Now  $\diamond$  is proved. We construct the required sequence  $\overline{\varphi}_0 \Rightarrow \overline{\varphi}'_0, \overline{\varphi}_1 \Rightarrow \overline{\varphi}'_0, \dots$  in  $\mathcal{G}$  as follows:

- The initial step  $n = 0$ : We set  $\overline{\varphi}_0 \Rightarrow \overline{\varphi}'_0 \triangleq \varphi_0 \Rightarrow \varphi'_0$ , which is in  $\mathcal{G}$  by the definition of  $\mathcal{G}$ . We obviously have  $M \models \varphi_0 \longrightarrow (\exists \text{FreeVars}(\overline{\varphi}_0))\overline{\varphi}_0$ .
- The iterative step: we assume that the sequence  $\overline{\varphi}_0 \Rightarrow \overline{\varphi}'_0, \dots, \overline{\varphi}_n \Rightarrow \overline{\varphi}'_0$  has been constructed, and  $M \models \varphi_i \longrightarrow (\exists \text{FreeVars}(\overline{\varphi}_i))\overline{\varphi}_i$  for all  $i \leq n$ . If the sequence  $(\varphi_i)$  has the length  $n$ , the process is finished and the built sequence satisfies the second item of the conclusion of the lemma. If 1. in  $\diamond$  holds, then the constructed sequence satisfies the first item of the conclusion of the lemma for  $j = n$ . and the construction of the sequence is finished. If 2. in  $\diamond$  holds, then the sequence of length  $n + 1$  is obtained by adding  $\overline{\varphi}_{n+1} \Rightarrow \overline{\varphi}'_0$  to the above one.

If the above process is finite then obviously the built sequence satisfies the conclusion of the lemma. If it is infinite, then it builds an infinite sequence that satisfies the second item of the conclusion of the lemma. This finishes the proof of the lemma.  $\square$

**Theorem 1 (Soundness).** *Let  $\text{prove}$  be the procedure given in Figure 1. Assume that  $\mathcal{S}$  is total. Let  $G_0$  be such that each  $\varphi_c \Rightarrow \varphi'_c \in G_0$  is  $\mathcal{S}$ -derivable and satisfies  $\text{FreeVars}(\varphi'_c) \subseteq \text{FreeVars}(\varphi_c)$ . If  $\text{prove}(\mathcal{S}, G_0, \Delta_{\mathcal{S}}(G_0))$  returns success then  $\Rightarrow_{\mathcal{S}} \models G_0$ .*

*Proof.* Let  $\tau \triangleq \gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1 \Rightarrow_{\mathcal{S}} \dots$  be a complete execution path, and let the valuation  $\rho$  be such that  $(\tau, \rho)$  starts from  $\varphi_0$  with  $\varphi_0 \Rightarrow \varphi'_0 \in G_0$ . There is a sequence of ML-formulas  $\varphi_0, \varphi_1, \dots$  that covers  $\tau$  cf. Lemma 5. There is a walk  $\overline{\varphi}_0 \Rightarrow \overline{\varphi}'_0, \overline{\varphi}_1 \Rightarrow \overline{\varphi}'_0, \dots$  in  $\mathcal{G}$  that covers  $\varphi_0, \varphi_1, \dots$  cf. Lemma 6.

If the walk ends in a sink  $\overline{\varphi}_j \Rightarrow \overline{\varphi}'_0$  then  $(\gamma_j, \rho) \models \overline{\varphi}'_0$ , and therefore  $(\tau, \rho) \models \varphi_0 \Rightarrow \overline{\varphi}'_0$ .

If the walk is infinite then  $\tau$  is also infinite, which also implies  $(\tau, \rho) \models \varphi_0 \Rightarrow \overline{\varphi}'_0$ .

It remains to investigate the case when the walk is finite and does not end in a sink. In that case  $\tau$  is finite as well and let  $\gamma_j$  be the last element of  $\tau$ . We have  $(\gamma_j, \rho) \models \varphi_j$  and  $M \models \varphi_j \longrightarrow (\exists \text{FreeVars}(\overline{\varphi}_j))\overline{\varphi}_j$  by Lemma 5 and Lemma 6, respectively, which implies  $(\gamma_j, \rho) \models (\exists \text{FreeVars}(\overline{\varphi}_j))\overline{\varphi}_j$ , i.e., there is

$\rho'$  such that  $(\gamma_j, \rho') \models \bar{\varphi}_j$ . By the definition of  $\mathcal{G}$ , we have two cases when the current goal is the non-sink vertex  $\bar{\varphi}_j \Rightarrow \bar{\varphi}'_j$ : either it is processed by line 6 in **prove**, case in which  $\bar{\varphi}_j$  is  $\mathcal{S}$ -derivable, or it is processed by line 4 in **prove**, case in which we have  $M \models \bar{\varphi}_j \rightarrow (\exists \text{FreeVars}(\varphi_c))\varphi_c$  for some  $\varphi_c \Rightarrow \varphi'_c \in G_0$ . In the later case, there is  $\rho''$  such that  $(\gamma_j, \rho'') \models \varphi_c$  and  $\varphi_c$  is  $\mathcal{S}$ -derivable by the hypotheses of the theorem. So, in both cases we have  $(\gamma_j, \rho'') \models \varphi$  for certain  $\rho''$  and  $\mathcal{S}$ -derivable  $\varphi$ . Since  $\mathcal{S}$  is total, there is  $\gamma_j \Rightarrow_{\mathcal{S}} \gamma_{j+1}$ , which contradicts that  $\tau$  is complete.  $\square$

*Remark 1.* A *sound approximated check* for validity statements  $M \models \varphi$  is an automatic procedure that, when presented with  $M$  and  $\varphi$ , if it answers *true* then  $M \models \varphi$ . We note that Theorem 1 still holds when one uses sound approximated checks for the various validity statements occurring in the procedure (including  $\mathcal{S}$ -derivability and totality of  $\mathcal{S}$ , which amount to validity, cf. the previous section). Approximated checks, such as those implemented in SMT solvers, are required here since exact checks do not exist due to undecidability issues.

*Remark 2.* Theorem 1 says nothing about executions of the procedure that return **failure** or that do not terminate. Such outcomes may mean either  $\Rightarrow_{\mathcal{S}} \not\models G_0$ , or  $\Rightarrow_{\mathcal{S}} \models G_0$  but the information contained in the goals  $G_0$  is not enough for proving them, or, again, that the approximation induced by the validity checkers was too coarse. It is the user's burden to come up with a set of goals containing enough information such that the procedure terminates successfully.

This is similar to proving loop invariants in imperative programs, which requires users to provide a strong-enough invariant (i.e., one that can be proved).

*Example 10.* Let  $S$  be the RL specification defined in Example 4 and  $G_0 \triangleq \{\varphi_0 \Rightarrow \varphi'_0\}$ , where  $\varphi_0 \Rightarrow \varphi'_0$  is

$$\langle x, y \rangle \wedge y \geq 0 \Rightarrow (\exists x', y') \langle x', y' \rangle \wedge \text{gcd}(x', x, y).$$

We illustrate in a step-by-step manner the procedure call **prove**( $\mathcal{S}, G_0, G$ ), where  $G$  is initially  $\Delta_{\mathcal{S}}(G_0) = \{\varphi_1 \Rightarrow \varphi'_0, \varphi_2 \Rightarrow \varphi'_0\}$ , and

$$\varphi_1 \triangleq (\exists k) \langle y, x - k * y \rangle \wedge y > 0 \wedge x \geq k * y \wedge k > 0 \Rightarrow \varphi'_0,$$

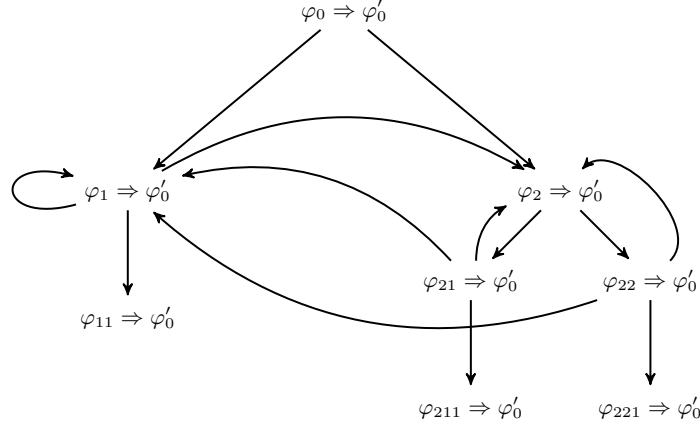
$$\varphi_2 \triangleq \langle y, x \rangle \wedge y \geq x \wedge y \geq 0 \Rightarrow \varphi'_0.$$

Let us consider that  $\varphi_1 \Rightarrow \varphi'_0$  is the current chosen goal from  $G$ . Obviously,  $M \models \varphi_1 \rightarrow \varphi'_0$  does not hold. Since  $M \models x \geq k * y \rightarrow x - k * y \geq 0$ , we obtain  $M \models \varphi_1 \rightarrow (\exists x', y') \langle x', y' \rangle \wedge y' \geq 0$  (i.e. the condition of the **if** statement at line 4 holds) and hence the goal  $\varphi_1 \Rightarrow \varphi'_0$  is replaced with  $\varphi_{11} \Rightarrow \varphi'_0$  by the statement in line 5, where

$$\begin{aligned} \varphi_{11} \triangleq & (\exists x_1, y_1, x'_1, y'_1, k) \langle x'_1, y'_1 \rangle \wedge \text{gcd}(x'_1, x_1, y_1) \wedge \langle y, x - k * y \rangle = \langle x_1, y_1 \rangle \\ & \wedge y > 0 \wedge x \geq k * y \wedge k > 0 \end{aligned}$$

Since  $M \models (\text{gcd}(x', y, x - k * y) \wedge k > 0 \wedge x \geq k * y) \rightarrow \text{gcd}(x', x, y)$ , it follows that  $M \models \varphi_{11} \rightarrow \varphi'_0$  and the goal  $\varphi_{11} \Rightarrow \varphi'_0$  is removed from  $G$  by the **if** statement in line 2.

Now the only goal in  $G$  is  $\varphi_2 \Rightarrow \varphi'_0$ . It is easy to see that  $M \not\models \varphi_2 \rightarrow \varphi'_0$  and  $M \not\models \varphi_2 \rightarrow (\exists \text{FreeVars}(\varphi'_0))\varphi'_0$ , i.e. the conditions on then lines 3 and 4



**Fig. 2.** The graph  $\mathcal{G}$  corresponding to the `prove` procedure call for Example 10

do not hold. We have  $\Delta_{\mathcal{S}}(\varphi_2 \Rightarrow \varphi'_0) = \{\varphi_{21} \Rightarrow \varphi'_0, \varphi_{22} \Rightarrow \varphi'_0\}$ , where

$$\varphi_{21} \triangleq (\exists k)\langle x, y - k * x \rangle \wedge x > 0 \wedge y \geq k * x \wedge k > 0 \wedge y > x \Rightarrow \varphi'_0,$$

$$\varphi_{22} \triangleq \langle x, y \rangle \wedge x \geq y \wedge y \geq x \wedge y \geq 0 \Rightarrow \varphi'_0.$$

The  $\varphi_{21} \Rightarrow \varphi'_0$  is processed in the same way like  $\varphi_1 \Rightarrow \varphi'_0$ . Since  $M \models \varphi_{22} \longrightarrow (\exists x', y')\langle x', y' \rangle \wedge y' \geq 0$  and hence the goal  $\varphi_{22} \Rightarrow \varphi'_0$  is replaced with  $\varphi_{221} \Rightarrow \varphi'_0$  by the `if` statement in line 4, where

$$\varphi_{221} \triangleq (\exists x_1, y_1, x'_1, y'_1, k)\langle x'_1, y'_1 \rangle \wedge \text{gcd}(x'_1, x_1, y_1) \wedge \langle x, y \rangle = \langle x_1, y_1 \rangle \\ \wedge x \geq y \wedge y \geq x \wedge y \geq 0$$

It is easy to see that  $M \models \varphi_{221} \longrightarrow \varphi'_0$  and hence the goal  $\varphi_{221} \Rightarrow \varphi'_0$  is removed from  $G$  by the `if` statement in line 2.

Now the set of current goals  $G$  is empty and the execution of the procedure call `prove`( $\mathcal{S}, G_0, \Delta_{\mathcal{S}}(G_0)$ ) returns `success`. The execution of the procedure `prove` corresponding to this call is graphically represented in Figure 2.

## 5 Reachability Properties for Rewrite Theories

In this section we show that RL formulas can be used to specify properties of transition systems defined by RWL theories. This is achieved by extending the signature of an RWL theory  $\mathcal{R}$  to an ML-signature, which includes predicates that can be used to define RL properties of the transition system defined by  $\mathcal{R}$ .

We also show how the verification procedure in Section 4 can be adapted in order to take advantage of RWL-specific operations such as matching. More precisely, we prove that, under reasonable assumptions, a complete set of derivatives of an RL formula can be computed using standard matching-modulo algorithms.



We note that RL properties for rewrite theories are different from the reachability properties that can be checked in Maude using the `search` command. The difference is given by the possibility of using FOL for constraining the initial and the final state terms, and by the interpretation of RL formulas.

**Definition 14 (ML Extension of a Rewrite Theory).** Consider a rewrite theory  $\mathcal{R} = (\Sigma, E \cup A, R)$  with a distinguished sort *State* such that  $\mathcal{R}$  is topmost w.r.t. *State*. An ML extension of  $\mathcal{R}$  consists of an ML signature  $(\Sigma, \Pi, \text{State})$  together with an interpretation  $(T_{\Sigma, E \cup A})_p \subseteq T_{\Sigma, E \cup A, s_1} \times \cdots \times T_{\Sigma, E \cup A, s_n}$  for each predicate symbol  $p \in \Pi_{s_1, \dots, s_n}$ . In this way,  $T_{\Sigma, E \cup A}$  is a model of the ML extension of  $\mathcal{R}$ .

The above definition allows one to see the operations  $bop \in \Sigma_{s_1 \dots s_n, Bool}$  as predicates  $bop \in \Pi_{s_1 \dots s_n}$  with the interpretation  $([t_1], \dots, [t_n]) \in (T_{\Sigma, E \cup A})_{bop}$  iff  $bop(t_1, \dots, t_n) =_{E \cup A} true$ . Consequently, each term  $b$  of sort *Bool* defines a FOL formula such that, if  $\rho : \text{Var} \rightarrow T_{\Sigma, E \cup A}$ , then  $\rho \models b$  iff  $\rho(b) =_{E \cup A} true$ .

Any rewrite rule  $l \rightarrow r$  **if**  $b$  can be viewed as an RL formula  $l \wedge b \Rightarrow r$  and the transition relation  $\Rightarrow_R$  is exactly the same with the one-step topmost rewriting relation  $\rightarrow_{\mathcal{R}}$ . This allows one to naturally define when RL formulas specify properties for RWL theories.

**Definition 15 (RL Properties for RWL Theories).** An RL property for  $\mathcal{R} = (\Sigma, E \cup A, R)$  is an RL formula  $\varphi \Rightarrow \varphi'$ , where  $\varphi$  and  $\varphi'$  are ML formulas defined over an ML extension of  $\mathcal{R}$  (cf. Definition 14). We say that  $\mathcal{R}$  satisfies  $\varphi \Rightarrow \varphi'$  iff  $\rightarrow_{\mathcal{R}} \models \varphi \Rightarrow \varphi'$  (cf. Definition 7).

We next focus on adapting the `prove` procedure for verifying RL properties of RWL theories. Specifically, we show the derivatives can be computed using matching algorithms (under certain assumptions). We give first a technical definition.

**Definition 16 (Ground  $(E \cup A)$ -unifier).** Consider a rewrite theory  $\mathcal{R} = (\Sigma, E \cup A, R)$ . Two  $\Sigma$ -terms  $t$  and  $t'$  are ground  $(E \cup A)$ -unifiable if there is  $\sigma : \text{FreeVars}(t, t') \rightarrow T_{\Sigma}$  such that  $\sigma(t) =_{E \cup A} \sigma(t')$ . The substitution  $\sigma$  is called  $(E \cup A)$ -unifier of  $t$  and  $t'$ .

The following assumptions are required for computing derivatives using matching. The first assumption restricts the class of RL formulas that can be used as properties to those that are useful in practice (i.e., having more than one pattern, or having negations of patterns, in either side of formulas is not really useful).

**Assumption 2** We assume that  $\text{FreeVars}(b) \subseteq \text{FreeVars}(l, r)$  for each rule  $l \rightarrow r$  **if**  $b$  in  $R$ . In this paper we also assume that the RL properties of RWL theories  $\mathcal{R}$  are of particular form  $(\exists X)(\pi \wedge \phi) \Rightarrow (\exists X')(\pi' \wedge \phi')$  with  $\text{FreeVars}(\phi) \subseteq \text{FreeVars}(\pi)$  (and hence  $X \subseteq \text{FreeVars}(\pi)$ ).

The second assumption relates ground unifiers to matching substitutions.

**Assumption 3** We assume that for each  $l \rightarrow r$  **if**  $b$  in  $R$  with  $l$  and  $\pi$  ground  $(E \cup A)$ -unifiable, there is a set of matching substitutions  $\text{match}(l, \pi)$  such that

- $\sigma_0(l) = \pi$  for each  $\sigma_0 \in \text{match}(l, \pi)$ , and
- for each ground  $E \cup A$ -unifier  $\sigma$  of  $l$  and  $\pi$  there are  $\sigma_0 \in \text{match}(l, \pi)$  and  $\sigma' : \text{FreeVars}(\pi) \rightarrow T_\Sigma$  satisfying  $\sigma =_{E \cup A} \sigma' \circ \sigma_0$ <sup>6</sup>.

Assumption 3 holds under reasonable constraints, cf. the *Matching Lemma* [19]. The next results show that the matching substitutions can be used to compute the derivatives of ML formulas and, consequently, the derivatives of RL formulas.

**Lemma 7.** *Let  $\varphi \Rightarrow \varphi'$  be an RL property for  $\mathcal{R} = (\Sigma, E \cup A, R)$ , where  $\varphi \triangleq (\exists X)\pi \wedge \phi$ . Then, for each derivative  $\varphi_1 \in \Delta_R(\varphi)$  there exists a rewrite rule  $l \rightarrow r$  **if**  $b$  in  $R$  such that*

$$T_{\Sigma, E \cup A} \models \varphi_1 \iff \bigvee_{\sigma_0 \in \text{match}(l, \pi)} (\exists X \cup \text{FreeVars}(r) \setminus \text{FreeVars}(l))\sigma_0(r) \wedge \sigma_0(b) \wedge \phi.$$

*Proof.* By the definition of  $\Delta_R(\varphi)$ ,  $\varphi_1$  is  $(\exists \text{FreeVars}(l, r))(l \wedge b \wedge (\exists X)(\pi \wedge \phi)) \stackrel{?}{=} \wedge r$  for some rewrite rule  $l \rightarrow r$  **if**  $b \in R$ . By Assumption 1,  $\text{FreeVars}(l, r) \cap X = \emptyset$  and hence  $\varphi_1$  is equivalent to  $(\exists X \cup \text{FreeVars}(l, r))(l = \pi) \wedge b \wedge \phi \wedge r$ . We have:

$$\begin{aligned} (\gamma_1, \rho_1) \models \varphi_1 & \iff \\ (\gamma_1, \rho_1) \models (\exists X \cup \text{FreeVars}(l, r))(l = \pi) \wedge b \wedge \phi \wedge r & \iff \\ (\exists \rho)(\rho(l) = \rho(\pi)) \wedge \rho \models (b \wedge \phi) \wedge (\rho(r) = \gamma_1) & \iff \\ (\exists \sigma)(\sigma(l) =_{E \cup A} \sigma(\pi)) \wedge \rho \models b \wedge \rho_1 \models \phi \wedge (\sigma(r) \in \gamma_1) & \iff \\ (\exists \sigma_0)(\exists \sigma'')(\sigma_0(l) =_{E \cup A} \pi) \wedge \rho \models b \wedge \rho_1 \models \phi \wedge (\sigma''(\sigma'_0(r))) \in \gamma_1 & \iff \\ (\exists \sigma_0 \in \text{match}(l, \pi))(\exists \sigma'')\rho \models b \wedge \rho_1 \models \phi \wedge (\sigma'' \uplus \sigma'_0(r)) \in \gamma_1 & \iff \\ \bigvee_{\sigma_0 \in \text{match}(l, \pi)} (\exists \sigma'')\rho \models b \wedge \rho_1 \models \phi \wedge (\sigma'' \uplus \sigma'_0(r)) \in \gamma_1 & \iff \\ \bigvee_{\sigma_0 \in \text{match}(l, \pi)} (\exists \rho_0)\rho_0 \models \sigma_0(b) \wedge \rho_0 \models \phi \wedge \rho_0(\sigma_0(r)) = \gamma_1 & \iff \\ \bigvee_{\sigma_0 \in \text{match}(l, \pi)} (\exists \rho_0)(\gamma_1, \rho_0) \models (\sigma_0(b) \wedge \phi \wedge \sigma_0(r)) & \iff \\ \bigvee_{\sigma_0 \in \text{match}(l, \pi)} (\gamma_1, \rho_1) \models (\exists X \cup \text{FreeVars}(r) \setminus \text{FreeVars}(l))(\sigma_0(b) \wedge \phi \wedge \sigma_0(r)) & \iff \\ (\gamma_1, \rho_1) \models \bigvee_{\sigma_0 \in \text{match}(l, \pi)} (\exists X \cup \text{FreeVars}(r) \setminus \text{FreeVars}(l))(\sigma_0(b) \wedge \phi \wedge \sigma_0(r)) & \iff \end{aligned}$$

where

- by Assumption 1, we may assume w.l.o.g. that  $\rho(x) = \rho_1(x)$  for all  $x \notin X \cup \text{FreeVars}(l, r)$ ,
- $\sigma : \text{FreeVars}(l, r, \phi) \rightarrow T_\Sigma$  with  $[\sigma(x)] = \rho(x)$ ,

<sup>6</sup>  $\sigma_1 =_{E \cup A} \sigma_2$  iff  $\text{dom}(\sigma_1) = \text{dom}(\sigma_2)$  and  $(\forall x \in \text{dom}(\sigma_1))\sigma_1(x) =_{E \cup A} \sigma_2(x)$ .

- the substitutions  $\sigma_0 : \text{FreeVars}(l) \rightarrow \text{FreeVars}(\pi)$  and  $\sigma' : \text{FreeVars}(\pi) \rightarrow T_\Sigma$  are given by Assumption 2, i.e.,  $\sigma|_{\text{FreeVars}(l,\pi)} = \sigma' \circ \sigma_0$
- $\sigma'' = \sigma|_{\text{FreeVars}(r) \setminus \text{FreeVars}(l)}$ ,
- $\sigma'' \uplus \sigma'(x) = \sigma''(x)$  if  $x \in \text{FreeVars}(r) \setminus \text{FreeVars}(l)$ , and  $\sigma'' \uplus \sigma'(x) = \sigma'(x)$  if  $x \in \text{FreeVars}(\sigma_0(l))$ ,
- $\rho_0(x) = [\sigma'(x)]$ , for  $x \in \text{FreeVars}(\pi)$ ,  $\rho_0(x) = [\sigma''(x)]$ , for  $x \in \text{FreeVars}(r) \setminus \text{FreeVars}(l)$ , and  $\rho_0(x) = \rho(x)$  in the rest (hence  $\rho_0(x) = \rho_1(x)$  for  $x \notin X \cup \text{FreeVars}(r) \setminus \text{FreeVars}(l)$ ), and
- $\rho \models b$  iff  $\rho_0 \models \sigma_0(b)$ .

□

**Theorem 2.** *For each  $\varphi_1 \Rightarrow \varphi' \in \Delta_R(\varphi \Rightarrow \varphi')$  with  $\varphi \triangleq (\exists X)\pi \wedge \phi$ , there is  $l \rightarrow r$  **if**  $b \in R$  such that*

$$\llbracket \varphi_1 \Rightarrow \varphi' \rrbracket = \llbracket \bigvee_{\sigma_0 \in \text{match}(l,\pi)} (\exists X \cup \text{FreeVars}(r) \setminus \text{FreeVars}(l))\sigma_0(r) \wedge \sigma_0(b) \wedge \phi \Rightarrow \varphi' \rrbracket.$$

*Proof.* The proof directly follows from Lemma 7. □

*Symbolic Rewrite Rules.* We now show that Theorem 2 enables the use of *symbolic rewrite rules* to efficiently compute derivatives and hence to implement the procedure `prove` in RWL. For  $\varphi \triangleq (\exists X)(\pi \wedge \phi)$ , let  $\Delta_R^{\text{match}}(\varphi \Rightarrow \varphi')$  be the set

$$\{(\exists X \cup \text{FreeVars}(r) \setminus \text{FreeVars}(l))\sigma_0(r) \wedge \sigma_0(b) \wedge \phi \Rightarrow \varphi' \mid l \rightarrow r \text{ **if** } b \in R, \sigma_0 \in \text{match}(l,\pi)\} \quad (5)$$

We have  $\llbracket \Delta_R^{\text{match}}(\varphi \Rightarrow \varphi') \rrbracket = \llbracket \Delta_R(\varphi \Rightarrow \varphi') \rrbracket$  by Theorem 2, which implies that  $\Delta_R^{\text{match}}(\varphi \Rightarrow \varphi')$  is a complete set of  $\mathcal{R}$ -derivatives. This allows us to use of  $\Delta_R^{\text{match}}$  in the procedure `prove` instead of  $\Delta_R$ . Next, we note that formulas  $\varphi_1 \Rightarrow \varphi'$  in  $\Delta_R^{\text{match}}(\varphi \Rightarrow \varphi')$  (where  $\varphi \triangleq (\exists X)(\pi \wedge \phi)$ ) can be computed by applying a *symbolic rewrite rule* of the form  $l \wedge \psi \Rightarrow r \wedge b \wedge \psi$  to the left-hand side of  $\varphi \Rightarrow \varphi'$ , where  $\psi$  is a fresh variable of sort *Bool*, with a matching substitution  $\sigma_0$  such that  $\sigma_0(\psi) = \phi$ . Moreover,  $\varphi$  is  $R$ -derivable iff there are a rule  $l \rightarrow r$  **if**  $b$  in  $R$  and  $\sigma_0 \in \text{match}(l,\pi)$  such that  $(\exists X \cup \text{FreeVars}(r) \setminus \text{FreeVars}(l))\sigma_0(b) \wedge \phi$  is satisfiable, by Lemma 7. This is equivalent to saying that  $\varphi$  is  $R$ -derivable iff the symbolic rewrite rule  $l \wedge \psi \Rightarrow r \wedge b \wedge \psi$  is applicable to  $\varphi$ .

Overall,  $R$ -derivatives of RL formulas  $\varphi \Rightarrow \varphi'$  (where  $\varphi \triangleq (\exists X)(\pi \wedge \phi)$ ) can be computed by transforming each rule  $l \rightarrow r$  **if**  $b \in R$  into a symbolic rewrite rule  $l \wedge \psi \Rightarrow r \wedge b \wedge \psi$  and by applying the symbolic rewrite rule. This is how derivatives are computed in our RWL adaptation of the `prove` procedure.

## 6 Verifying RL Properties of a Communication Protocol

We illustrate the theory on a simple communication protocol described in Maude.

*Protocol Description.* The protocol transmits a file between a sender and a receiver. The file is a sequence of records. The sender and receiver communicate through unidirectional, lossy channels, one of which carries messages (a record in the file, together with a sequence number) from sender to receiver, while the other one carries retransmission requests (natural numbers) from receiver to sender.

Both the sender and the receiver maintain a counter in order to keep track of the next record to be sent resp. received. The sender transmits the next record in the file together with the current value of its counter, which then is incremented by one. The receiver accepts a message only if the sequence number of the message is equal to the receiver's counter; if this is the case, the counter is incremented and the record from the message is saved. The receiver discards all other messages (i.e., whose sequence number is not the expected one). It may also (nondeterministically) request a retransmission, by sending the current value of its counter over the retransmission request channel. This nondeterminism can be seen as an abstraction of a timeout mechanism, not modeled here for simplicity.

Upon reception of a retransmission request, the sender ignores it if it is greater than or equal to its counter (indicating a wrong retransmission request). Otherwise the sender updates its counter to the number it received on retransmission request channel, in order to start resending messages from that number on.

The protocol's **State** structure is given as a constructor with five arguments:

```
<_,_,_,_,_> : Nat List{Pair} List{Nat} Nat List{Element} -> State,
```

where **Pair** is a sort of pairs consisting of an **Element** and a natural number, and **List{}** are parameterised lists. The file to be sent is modeled by a function `fileToSend : Nat -> Elements`, of size **Max**. The sender and receiver's rules are shown in Fig. 3. There are also rules for the channels losing elements, not shown here due to lack of space.

*Reachability Properties.* The protocol's initial state is `<0,nil,nil,0,nil>`. Its expected reachability property states that all terminating executions starting from the initial state should end up in a state of the form

```
< (s Max),nil,nil,(s Max),file>List{Element}>
```

where `file` should satisfy  $\forall j. 0 \leq j \leq \text{Max} \rightarrow \text{fileToSend}(j) = \text{file}[j]$ , where `_[_]` is a function returning an element at a given position in a list. (That is, the file received is the same as the file sent.) In order to specify the constraints on the final states we defined in Maude a subset of RL, so that the reachability property specifying the protocol is written as the following Maude rewrite rule:

```
< 0,nil,nil,0,nil > //\ True =>
Exists file : <(s Max),nil,nil,(s Max),file:>
//\ Forall j : ((0 <=? j And (j <=? Max)
               Implies (fileToSend(j) == file[j]))
```

The operation `//\` is the constructor for our defined subset of ML, which takes a term of sort **State** and term of sort FOL and builds a term of sort ML. Note that the Maude `search` command cannot be used to prove this RL formula:

```

--- sender
crl [send_to_R] :
< n, R, L, m, recv > => < s(n), (< fileToSend(n), n >: R), L, m, recv >
if n <= Max .

crl [update_request_from_L] :
< n, R, (resend : L), m, recv > => < resend, R, L, m, recv >
if resend < n .

crl [ignore_request_from_L] :
< n, R, (resend : L), m, recv > => < n, R, L, m, recv >
if resend >= n .

--- receiver
crl [accept_element_from_R] :
< n, (R : < e, nb >), L, m, recv > => < n, R, L, (s m), (recv : e) >
if m == nb .

crl [ignore_element_from_R] :
< n, (R : < e, nb >), L, m, recv > => < n, R, L, m, recv >
if m /= nb .

crl [send_request_to_L] :
< n, R, L, m, recv > => < n, R, L : m, m, recv >
if m <= Max .

```

**Fig. 3.** : Communication Protocol (excerpt).

to do so it would have to explore all terminating executions starting from the initial state, which are infinitely many (and can be arbitrarily long).

Thus, we use our verification procedure. Unsurprisingly, the above RL formula is not enough by itself for our procedure to succeed. For this to happen, a "helper" RL formula is required, whose right-hand side is the same as for the one above, but whose left-hand side describes an invariant (to hold for all states reachable from the initial states):

```

< n, R, L, m, file > //\\
Forall j : (0 <=? j And j <? m) Implies fileToSend(j) === file[j] And
Forall e : Forall nb : < e, nb > In? R Implies e === fileToSend(nb) And
size(file) + 1 === m
=>
Exists file : <(s Max), nil, nil, (s Max), file:> //\\
Forall j : (0 <=? j And j <=? Max) Implies (fileToSend(j) === file[j])

```

This formula says that the currently received file (whose size is  $m-1$ ) equals the portion of the file being sent (up to that size); and that all messages currently in transition from sender to receiver are records in the `fileToSend` file. It was obtained by trial-and-error, while applying the following verification technique.

*Verification.* We have implemented key functionality from our verification procedure (computing derivatives) at Maude's metalevel. Derivatives are computed based on matching and rewriting as described in Section 5. We also use Maude's metalevel to achieve the following executions of our verification procedure:

- for the first RL formula (the protocol's specification): deriving it with respect to the protocol's rules  $\mathcal{S}$ , then applying the second formula as a circularity;

- for the second RL formula (designated above as the "helper" formula): deriving it with respect to the protocol's rules, then applying itself as a circularity.

By requiring that these two executions return **success**, Maude generates several proof obligations: essentially, that the condition for applying circularities holds (at line 4 in our verification procedure), and that the condition for returning **success** (line 3) also holds. Several of those proof obligations are discharged automatically by simplification rules we included in Maude (e.g., that FOL disjunction is commutative). The remaining ones are axioms satisfied by the assumed model for the various elements in our problem domain (e.g., lossy channels, files consisting on records, and natural numbers). For example, one proof obligation says that if all messages in a channel contain records from the `fileToSend` file, then by losing a message all the remaining messages satisfy the same property.

There are four such proof obligations left after automatic simplification. We have (manually) checked that they hold (in the assumed model for our problem domain). The trial-and-error process for finding the helper RL formula consisted in examining the generated proof obligations, and noting that some do not hold unless more information is added about the problem domain.

## 7 Conclusion and Future Work

In this paper, we propose a procedure for verifying reachability properties on symbolic transition systems. While the reachability properties are stated as RL formulas, we allow symbolic transition systems to be described by either RL specifications or by RWL specifications. We prove that our procedure is sound. We show with a concrete example that our procedure works in practice.

The paper also contributes to establishing connections between RL and RWL. In [22] it is shown how RL specifications can be encoded as RWL theories. Here, we take an alternative approach, which consists in using RL as a property language for RWL. The proposed procedure adapted for RWL can be implemented in Maude, using reflection and the recently added support for sat checking and one-step rewriting modulo SMT using the CVC4 library (<http://cvc4.cs.nyu.edu/>).

In terms of future work there are several directions to follow. First, starting from our prototype, we shall develop a tool in the Maude environment that will efficiently implement our procedure; we envision that the tool will generate proof obligations to be discharged by Maude's inductive theorem prover ITP [23]. We also intend to formalise in the Coq proof assistant our procedure and its soundness proof in order to be able not only to verify properties but also to generate certificates. Finally, we will use the extraction mechanism of Coq to obtain certified OCaml code for our procedure and use it as a reference implementation.

## References

1. Grigore Roşu and Andrei Ştefănescu. Checking reachability using matching logic. In Gary T. Leavens and Matthew B. Dwyer, editors, *OOPSLA*, pages 555–574.

- ACM, 2012. Also available as technical report <http://hdl.handle.net/2142/33771>.
2. Grigore Rosu and Andrei Stefanescu. Towards a unified theory of operational and axiomatic semantics. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP'12)*, volume 7392 of *LNCS*, pages 351–363. Springer, 2012.
  3. Grigore Roşu, Andrei Ştefănescu, Ştefan Ciobăcă, and Brandon M. Moore. One-path reachability logic. In *Proceedings of the 28th Symposium on Logic in Computer Science (LICS'13)*, pages 358–367. IEEE, June 2013.
  4. Andrei Ştefănescu, Ştefan Ciobăcă, Radu Mereuţă, Brandon M. Moore, Traian Florin Şerbănuţă, and Grigore Roşu. All-path reachability logic. In *Proceedings of the Joint 25th International Conference on Rewriting Techniques and Applications and 12th International Conference on Typed Lambda Calculi and Applications (RTA-TLCA'14)*, volume 8560 of *LNCS*, pages 425–440. Springer, July 2014.
  5. Chucky Ellison and Grigore Roşu. An executable formal semantics of C with applications. In *Proceedings of the 39th Symposium on Principles of Programming Languages (POPL'12)*, pages 533–544. ACM, 2012.
  6. Denis Bogdănaş and Grigore Roşu. K-Java: A Complete Semantics of Java. In *Proceedings of the 42nd Symposium on Principles of Programming Languages (POPL'15)*, pages 445–456. ACM, January 2015.
  7. Grigore Roşu and Traian Florin Şerbănuţă. An overview of the K semantic framework. *Journal of Logic and Algebraic Programming*, 79(6):397–434, 2010.
  8. José Meseguer. Twenty years of rewriting logic. *The Journal of Logic and Algebraic Programming*, 81(7):721–781, 2012.
  9. Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *All about maude—a high-performance logical framework: how to specify, program and verify systems in rewriting logic*. Springer-Verlag, 2007.
  10. Steven Eker, José Meseguer, and Ambarish Sridharanarayanan. The maude ltl model checker. *Electronic Notes in Theoretical Computer Science*, 71:162–187, 2004.
  11. Kyungmin Bae and José Meseguer. A rewriting-based model checker for the linear temporal logic of rewriting. *Electronic Notes in Theoretical Computer Science*, 290:19–36, 2012.
  12. Santiago Escobar and José Meseguer. Symbolic model checking of infinite-state systems using narrowing. In *Term Rewriting and Applications*, pages 153–168. Springer, 2007.
  13. Camilo Rocha and José Meseguer. Proving safety properties of rewrite theories. In *Algebra and Coalgebra in Computer Science*, pages 314–328. Springer, 2011.
  14. Vlad Rusu. Combining theorem proving and narrowing for rewriting-logic specifications. In *Tests and Proofs*, pages 135–150. Springer, 2010.
  15. Roberto Bruni and José Meseguer. Semantic foundations for generalized rewrite theories. *Theoretical Computer Science*, 360(1):386–414, 2006.
  16. José Meseguer, Miguel Palomino, and Narciso Martí-Oliet. Equational abstractions. *Theoretical Computer Science*, 403(2):239–264, 2008.
  17. José Meseguer, Miguel Palomino, and Narciso Martí-Oliet. Algebraic simulations. *Journal of Logic and Algebraic Programming*, 79(2), 2009.
  18. Andrei Arusoiaie, Dorel Lucanu, and Vlad Rusu. A generic framework for symbolic execution. In *6th International Conference on Software Language Engineering*, volume 8225 of *LNCS*, pages 281–301. Springer Verlag, 2013. Also available as a technical report <http://hal.inria.fr/hal-00853588>.

19. Camilo Rocha, José Meseguer, and César A. Muñoz. Rewriting modulo SMT and open system analysis. In Santiago Escobar, editor, *Rewriting Logic and Its Applications - 10th International Workshop, WRLA 2014, Held as a Satellite Event of ETAPS, Grenoble, France, April 5-6, 2014, Revised Selected Papers*, volume 8663 of *Lecture Notes in Computer Science*, pages 247–262. Springer, 2014.
20. Andrei Arusoaie, Dorel Lucanu, David Nowak, and Vlad Rusu. Verifying Reachability-Logic Properties on Rewriting-Logic Specifications. Technical Report TR 15-01, “Al.I.Cuza” University of Iași, Faculty of Computer Science, 2015. <http://www.infoiasi.ro/~tr/tr.pl.cgi>.
21. José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73 – 155, 1992. Selected Papers of the 2nd Workshop on Concurrency and Compositionality.
22. Andrei Arusoaie, Dorel Lucanu, Vlad Rusu, Traian-Florin Serbanuta, Andrei Stefanescu, and Grigore Rosu. Language definitions as rewrite theories. In *WRLA*, volume 8663 of *Lecture Notes in Computer Science*, pages 97–113. Springer, 2014.
23. Joe Hendrix. *Decision Procedures for Equationally Based Reasoning*. PhD thesis, University of Illinois at Urbana Champaign, 2008.