Verifying Reachability-Logic Properties on Rewriting-Logic Specifications

Dorel Lucanu, Vlad Rusu, Andrei Arusoaie, David Nowak

Verifying Reachability-Logic Properties on Rewriting-Logic Specifications

Dorel Lucanu[1], Vlad Rusu[2], Andrei Arusoaie[2], and David Nowak[3]

[1] Faculty of Computer Science, Alexandru Ioan Cuza University, Iaşi, Romania
dlucanu@info.uaic.ro
[2] Inria Lille Nord Europe, France
Andrei.Arusoaie@inria.fr Vlad.Rusu@inria.fr
[3] CRIStAL, CNRS & University of Lille, France
david.nowak@univ-lille1.fr

Abstract. Rewriting Logic is a simply, flexible, and powerful framework for specifying and analysing concurrent systems. 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. Reachability Logic has its roots in a wider-spectrum framework, namely, in Rewriting Logic Semantics. In this paper we show how Reachability Logic can be adapted for stating properties of transition systems described by Rewriting-Logic specifications. We propose a 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

Since its original formulation [1] by José Meseguer, Rewriting Logic (RWL) has been defined both 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 have dynamic and strong development. A concurrent system is specified by a rewrite theory (Σ, E, R), where Σ defines the syntax of the system and of its states, E defines the states of the system as an algebraic data type, and R is a set of rewriting rules defining the local transitions of the system. RWL deduction consists of a set of inference rules used to prove sequents $t \to t'$, meaning that the term-pattern t "becomes" t' by concurrently applying the rewrite rules R modulo the equations E. Here the relation "becomes" refers the dynamics of the specified concurrent system.

Maude [2] 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.

Reachability Logic (hereafter, RL) was introduced in a series of papers [3,4,5,6] as means for specifying the operational semantics of programming languages and for stating reachability properties between states of program executions. Reachability Logic is a very promising framework. It has emerged from the Rewriting Logic Semantics project introduced by José Meseguer and Grigore Roşu [7]. Briefly, an RL formula $\varphi \Rightarrow \varphi'$ expresses *reachability relationships* between two sets of states of a system, denoted by the *patterns* φ and φ' 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 widely-used languages (including C [8], Java [9]) have been completely specified using an RL-based semantics in the $\mathbb{K}$ tool [10], and nontrivial programs have been proved using an implementation of RL[4].

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 [11] (hereafter, RWL). We propose a 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 [2].

Our contribution with respect to RL is a proved-sound verification procedure. Previous works [3,4,5,6] include sound and relatively complete proof systems for various versions of RL, but these 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 φ eventually reach a state in the set φ'. Both φ and φ' 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 or the Linear Temporal Logic (LTL) model checker [2]. 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. Specifically, the version of RL that we consider (the *all-paths* interpretation) corresponds to a subset of LTL interpreted on *finite paths*, whereas Maude's LTL model checker uses the standard (infinite-paths) interpretation of LTL; and both the model checker and the `search` command are bound to checking reachability properties starting from finitely many initial states by exploring finitely many execution paths.

Related work. We focus only on related verification approaches for RWL specifications. These fall under the usual classification of verification techniques: *algorithmic* ones, which essentially consist in using an automatic model checker; *deductive* ones, which involve an interaction with a theorem prover; and *abstraction-based* ones, which consist 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.

[4] Available at `http://www.matching-logic.org/index.php/Special:MatchCOnline`.

Algorithmic techniques include Maude's finite-state LTL model checker [12] with its more recent extensions to the *temporal logic of rewriting* [13], and a narrowing-based symbolic model-checker for handling classes of infinite-state systems [14]. Among the deductive techniques, [15,16] 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 [17] for defining the semantics of RWL. Among the abstraction-based techniques, equational abstractions [18], and algebraic simulations [19] 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 [20,21].

Outline. After preliminaries in Section 2, we introduce in Section 3 the notion of derivative, which 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 state its soundness. In Section 5, we adapt our approach to transition systems defined by RWL theories. We illustrate in Section 6 the usability of our procedure by applying it to a communication protocol described in Maude. Proofs of the technical results are included in an Appendix.

Acknowledgments This paper is to celebrate the 65th birthday of Professor José Meseguer. His seminal achievements, together with his warm and professional advices often guided and inspired the research activity of the first author.

The second author has spent his postdoc a couple of offices away from José's. At the time he was working on another topic and did not really understand what rewriting logic and Maude were about. He became aware of both of them several years later, and has been inspired by them and enjoying them ever since.

2 Preliminaries

2.1 Matching Logic

We recall the syntax and the semantics of Matching Logic (ML) as presented in [3]. 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* Φ is a pair (Σ, Π), where Σ is an algebraic S-sorted signature and Π 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 *Φ-model* consists of a Σ-algebra M together with a subset $M_p \subseteq M_{s_1} \times \cdots \times M_{s_n}$ 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 S-indexed set of variables Var. We let S denote the set of sorts in Φ and $T_\Sigma(Var)$ the algebra of Σ-terms with variables in Var.

The set of *Φ-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 Π, each t_i ranges over $T_\Sigma(\mathit{Var})$ of appropriate sort, and V over finite subsets of Var.

Given a first order Φ-model M,a Φ-formula ϕ, and a valuation $\rho : \mathit{Var} \to M$, the satisfaction relation $\rho \models \phi$ is defined as follows:

1. $\rho \models \top$;
2. $\rho \models p(t_1, \ldots, t_n)$ iff $(\rho(t_1), \ldots, \rho(t_n)) \in M_p$;
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 $\rho' : \mathit{Var} \to M$ with $\rho'(X) = \rho(X)$, for all $X \notin V$, such that $\rho' \models \phi$.

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

We recall below the ML concepts and results used in this paper. Their presentation is based on [3].

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

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

where the basic pattern π *ranges over* $T_{\Sigma,\mathit{State}}(\mathit{Var})$, p *ranges over predicate symbols* Π, *each* t_i *ranges over* $T_\Sigma(\mathit{Var})$ *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 $\mathit{FreeVars}(\varphi)$ denote the set of variables freely occurring in φ. We often use particular ML formulas $\varphi \triangleq \pi \wedge \phi$, where π represents a state and ϕ is a FOL formula used for constraining this state.

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

Definition 2 (ML satisfaction relation). *Given* $\Phi = (\Sigma, \Pi, \mathit{State})$ *an* ML *signature,* M *a* (Σ, Π)*-model,* φ *an* ML *formula over* Φ, $\gamma \in M_{\mathit{State}}$ *a state, and* $\rho : \mathit{Var} \to M$ *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. $(\gamma, \rho) \models p(t_1, \ldots, t_n)$ *iff* $(\rho(t_1), \ldots, \rho(t_n)) \in M_p$;
4. $(\gamma, \rho) \models \neg\varphi$ *iff* $(\gamma, \rho) \models \varphi$ *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* $\rho' : \mathit{Var} \to M$ *with* $\rho'(X) = \rho(X)$, *for all* $X \notin V$, *such that* $(\gamma, \rho') \models \varphi$.

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

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

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

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

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

The following proposition is needed later for our soundness result.

Proposition 2. *If ϕ is a FOL (i.e. structureless) formula, then $(\phi \wedge \varphi)^{=?}$ is equivalent to $\phi \wedge \varphi^{=?}$. Moreover, if $\rho \models \phi$ and $(\gamma, \rho) \models \varphi$ then $(\gamma, \rho) \models \phi \wedge \varphi$.*

Example 4. We consider first an ML formula including two state terms. Let φ denote the ML formula $\langle x,y\rangle \wedge x < 5 \wedge \langle u,v\rangle \wedge 8 < v$. Then $\varphi^{=?}$ is equivalent to $(\exists z)z = \langle x,y\rangle \wedge z = \langle u,v\rangle \wedge x < 5 \wedge 8 < v$, which in turn is equivalent to $\langle x,y\rangle \wedge = \langle u,v\rangle \wedge x < 5 \wedge 8 < v$. We have $\rho \models \varphi^{=?}$ iff $\rho(\langle x,y\rangle) = \rho(\langle u,v\rangle) \wedge \rho(x) < 5 \wedge \rho(v) > 8$ iff $(\gamma, \rho) \models \varphi$, where $\gamma = \rho(\langle x,y\rangle) = \rho(\langle u,v\rangle)$.

If φ_1 is an ML formula not including a state term (i.e., it is *structureless* according to ML terminology), then $\varphi_1^{=?}$ is the same with φ_1.

2.2 Reachability Logic

In this section we recall reachability-logic formulas, the transition systems that they induce, and their all-paths interpretation [6]. We consider a fixed ML signature $\Phi = (\Sigma, \Pi, State)$, a set of variables Var, and a fixed Φ-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 $(M_{State}, \Rightarrow_{\mathcal{S}})$, 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 5. The following set of rules is meant to compute the gcd of two natural numbers:

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

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: $\langle 8, 2\rangle \Rightarrow_{\mathcal{S}} \langle 2, 0\rangle$ as instance of the first rule, and $\langle 8, 10\rangle \Rightarrow_{\mathcal{S}} \langle 10, 8\rangle$ and $\langle 2, 2\rangle \Rightarrow_{\mathcal{S}} \langle 2, 2\rangle$ as instances of the second rule.

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

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

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

A pair (τ, ρ), consisting of an execution path $\tau \triangleq \gamma_0 \Rightarrow_{\mathcal{S}} \cdots$ and a valuation ρ, starts from *an* ML *formula φ if $(\gamma_0, \rho) \models \varphi$.*

Example 6. Examples of executions are $\tau \triangleq \langle 8, 10\rangle \Rightarrow_{\mathcal{S}} \langle 10, 8\rangle \Rightarrow_{\mathcal{S}} \langle 8, 2\rangle \Rightarrow_{\mathcal{S}} \langle 2, 0\rangle$ and $\tau' \triangleq \langle 8, 10\rangle \Rightarrow_{\mathcal{S}} \langle 10, 8\rangle \Rightarrow_{\mathcal{S}} \langle 8, 2\rangle \Rightarrow_{\mathcal{S}} \langle 2, 2\rangle \Rightarrow_{\mathcal{S}} \langle 2, 2\rangle \Rightarrow_{\mathcal{S}} \cdots$. Both executions are complete.

Since an infinite execution path cannot be the prefix of an another one, it follows that infinite execution paths are complete and hence the above definition is slightly different from that given in [6].

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

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

Example 7. An RL formula specifying that any execution path satisfying it computes the greatest common divisor (gcd) is $\langle x, y\rangle \Rightarrow (\exists x', y')\langle x', y'\rangle \wedge \mathit{gcd}(x', x, y)$, 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 $\rho(x) = 10, \rho(y) = 8$, τ and τ' are the execution paths defined in Example 6, then both (τ, ρ) and (τ', ρ) satisfy the given formula.

The definition of all-paths interpretation of ML formulas given above is slightly different from that given in [6], where $\Rightarrow_{\mathcal{S}} \models \varphi \Rightarrow \varphi'$ iff $(\tau, \rho) \models \varphi \Rightarrow \varphi'$ for all (τ, ρ) starting from φ with τ finite and complete. By contrast, our definition lets infinite paths satisfy formulas vacuously. The infinite paths are introduced from technical reasons,e.g., in order to prove $(\tau, \rho) \models \varphi \Rightarrow \varphi'$ we do not need to prove or know that τ is finite and complete.

2.3 Rewrite Theories

In this paper we propose a new approach for analysing rewrite theories. Our approach is dedicated to rewrite theories that model systems having at least *some* terminating executions, since nonterminating executions satisfy RL formulas vacuously. Communication protocols, such as the one we illustrate our approach on later in the paper, are an example of such systems. A counterexample are reactive systems, which (in the ideal case) execute forever in interaction with their environment.

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 Σ, 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 $l \to r$ **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 rewrite theories $\mathcal{R} = (\Sigma, E \cup A, R)$ with a distinguished sort *State* such that R is topmost w.r.t. *State*. Moreover, the actual theories $\mathcal{R}$ that we can analyse impose some additional technical restrictions on their components, which are briefly discussed in Section 5.

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

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

[5] For the sake of uniformity, we keep the notation $\mathit{FreeVars}(t)$ 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. $\mathit{FreeVars}(t_1, t_2)$ is $\mathit{FreeVars}(t_1) \cup \mathit{FreeVars}(t_2)$.

section we consider a fixed transition system specification $\mathcal{S}$ and its associated transition system $(M_{State}, \Rightarrow_{\mathcal{S}})$ over a fixed model M.

Assumption 1 *In what follows we consider only* ML *formulas φ with the following property: if φ does not occur as a member of a rule in $\mathcal{S}$ and $\varphi_l \Rightarrow \varphi_r \in \mathcal{S}$ then $\mathit{FreeVars}(\varphi) \cap \mathit{FreeVars}(\varphi_l, \varphi_r) = \emptyset$. 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 8. An $\mathcal{S}$-derivative for $\langle x, y\rangle \Rightarrow (\exists x', y')\langle x', y'\rangle \wedge \mathit{gcd}(x', x, y)$ is the following formula: $\langle y, x - y\rangle \wedge x > y \wedge y > 0 \Rightarrow (\exists x', y')\langle x', y'\rangle \wedge \mathit{gcd}(x', x, y)$.

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

Example 9. The set

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

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

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

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

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

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

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

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

Lemma 2. *Let $\varphi_1 \Rightarrow \varphi'$ be an $\mathcal{S}$-derivative of $\varphi \Rightarrow \varphi'$, τ_1 be an execution path and ρ a valuation. If $(\tau_1, \rho) \models \varphi_1 \Rightarrow \varphi'$ then there is $\varphi_1' \in \Delta_{\mathcal{S}}(\varphi)$ such that $(\tau_1, \rho) \models \varphi_1' \Rightarrow \varphi'$.*

From Lemma 1 and Lemma 2 we directly obtain:

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

Example 10.

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

which can be simplified to

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

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, and $\longrightarrow$ is the usual implication in FOL.

$\Delta_{\mathcal{S}}(\langle x,y\rangle \Rightarrow (\exists x',y')\langle x',y'\rangle \wedge gcd(x',x,y))$ is the set given in Example 9.

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 φ 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 3. *φ is $\mathcal{S}$-*derivable *iff $\bigvee_{\varphi_1 \in \Delta_{\mathcal{S}}(\varphi)} \varphi_1^{=?}$ is satisfiable.*

Lemma 3 also shows the strong relationship between the $\mathcal{S}$-derivability of a ML formula φ 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 φ.

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 (γ,ρ) of any $\mathcal{S}$-derivable formula φ. 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 for each $\mathcal{S}$-derivable φ and each pair (γ,ρ) such that $(\gamma,\rho) \models \varphi$, there is γ_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 φ and the totality requires to have at least one transition starting from γ for any model (γ,ρ) of φ.

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

Proposition 4. *$\mathcal{S}$ is* total *iff for each $\mathcal{S}$-derivable φ,*

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

We note that in general SMT solvers do not support theories for high-level algebraic structures. However, in practice, one can either introduce the theories in the solver, or use simplifications rules before sending the formulas to the solver.

```
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. $M \models \varphi \longrightarrow (\exists \mathit{FreeVars}(\varphi_c))\varphi_c$ 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.

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 Φ and Φ-model M.

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 note that this result is not a trivial consequence of the soundness of the RL proof system [6]; our initial attempts at proving soundness by reducing it to the soundness of the RL proof system showed that one step of our procedure corresponds to several (many) steps of the RL proof system. Thus, the soundness of several nontrivial derived rules of the RL proof system would have to be proved first before attempting to prove the soundness of our procedure. We thus opted for a direct proof.

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

Example 11. Let S be the RL specification defined in Example 5 and $G_0 \triangleq \{\varphi_0 \Rightarrow \varphi'_0\}$, where $\varphi_0 \Rightarrow \varphi'_0$ is

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

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

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

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

Let us consider that $\varphi_1 \Rightarrow \varphi'_0$ is the current chosen goal from G. Obviously, $M \models \varphi_1 \longrightarrow \varphi'_0$ does not hold. Since $M \models x \geq k * y \longrightarrow x - k * y \geq 0$, we obtain $M \models \varphi_1 \longrightarrow (\exists x', y')\langle x', y'\rangle \wedge y' \geq 0$ (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

$$\begin{tikzcd}
 & \varphi_0 \Rightarrow \varphi'_0 \arrow[dl] \arrow[drr] & & & \\
\varphi_1 \Rightarrow \varphi'_0 \arrow[ur, bend left] \arrow[d] & & & \varphi_2 \Rightarrow \varphi'_0 \arrow[dl] \arrow[dr] & \\
\varphi_{11} \Rightarrow \varphi'_0 & & \varphi_{21} \Rightarrow \varphi'_0 \arrow[uul, bend left] \arrow[d] & & \varphi_{22} \Rightarrow \varphi'_0 \arrow[uulll, bend right] \arrow[d] \\
 & & \varphi_{211} \Rightarrow \varphi'_0 & & \varphi_{221} \Rightarrow \varphi'_0
\end{tikzcd}$$

Fig. 2. The graph $\mathcal{G}$ corresponding to the `prove` procedure call for Example 11

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

Since $M \models (gcd(x', y, x - k * y) \wedge k > 0 \wedge x \geq k * y) \longrightarrow gcd(x', x, y)$, it follows that $M \models \varphi_{11} \longrightarrow \varphi'_0$ 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 $M \not\models \varphi_2 \longrightarrow \varphi'_0$ and $M \not\models \varphi_2 \longrightarrow (\exists \mathit{FreeVars}(\varphi'_0))\varphi'_0$, i.e. the conditions on then lines 3 and 4 do not hold. We have $\Delta_{\mathcal{S}}(\varphi_2 \Rightarrow \varphi'_0) = \{\varphi_{21} \Rightarrow \varphi'_0, \varphi_{22} \Rightarrow \varphi'_0\}$, where

$$\varphi_{21} \triangleq (\exists k)\langle x, y - k * x\rangle \wedge x > 0 \wedge y \geq k * x \wedge k > 0 \wedge y > x \Rightarrow \varphi'_0,$$
$$\varphi_{22} \triangleq \quad \langle x, y\rangle \wedge x \geq y \wedge y \geq x \wedge y \geq 0 \Rightarrow \varphi'_0\}.$$

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

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

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: the sinks correspond to the implications on line 3, the forward arrows correspond to the calls on line 7, and the backward arrows correspond to the calls on line 5 in the procedure. This graph covers the symbolic executions starting from φ_0.

Remark 1. A *sound approximated check* for validity statements $M \models \varphi$ is a procedure that, when presented with M and φ, if it answers *true* then $M \models \varphi$. We conjecture 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 sufficient 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).

Remark 3. Unlike the original proof system [6] we do not aim at (relative) completeness for our procedure. The relative completeness result is a very nice but essentially theoretical construction, which is based on strong assumptions (an oracle for deciding first-order theories) and is essentially of no practical use (it does not actually help in finding proofs).

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 13 (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 $(\Sigma, \Pi, State)$ 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 $p \in \Pi_{s_1,\ldots,s_n}$. 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 $bop \in \Sigma_{s_1\ldots s_n,Bool}$ as predicates $bop \in \Pi_{s_1\ldots s_n}$ with the interpretation $([t_1], \ldots, [t_n]) \in (T_{\Sigma,E\cup A})_{bop}$ iff $bop(t_1, \ldots, t_n) =_{E\cup A} true$. Consequently, each term b of sort *Bool* defines a FOL formula such that, if $\rho : Var \to T_{\Sigma,E\cup A}$, then $\rho \models b$ iff $\rho(b) =_{E\cup A} true$.

Any rewrite rule $l \to r$ **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 $\to_{\mathcal{R}}$. This allows one to naturally define when RL formulas specify properties for RWL theories.

Definition 14 (RL Properties for RWL Theories). *An* RL property *for $\mathcal{R} = (\Sigma, E \cup A, R)$ is an* RL *formula $\varphi \Rightarrow \varphi'$, where φ and φ' are* ML *formulas defined over an* ML *extension of $\mathcal{R}$ (cf. Definition 13). 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 15 (Ground $(E \cup A)$-unifier). *Consider a rewrite theory $\mathcal{R} = (\Sigma, E \cup A, R)$. Two Σ-terms t and t' are* ground $(E \cup A)$-unifiable *if there is $\sigma : \mathit{FreeVars}(t, t') \rightarrow T_\Sigma$ such that $\sigma(t) =_{E\cup A} \sigma(t')$. The substitution σ 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 $\mathit{FreeVars}(b) \subseteq \mathit{FreeVars}(l, r)$ 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 $\mathit{FreeVars}(\phi) \subseteq \mathit{FreeVars}(\pi)$ (and hence $X \subseteq \mathit{FreeVars}(\pi)$).*

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 π ground $(E \cup A)$-unifiable, there is a set of matching substitutions $\mathit{match}(l, \pi)$ such that*

- *$\sigma_0(l) = \pi$ for each $\sigma_0 \in \mathit{match}(l, \pi)$, and*
- *for each ground $E \cup A$-unifier σ of l and π there are $\sigma_0 \in \mathit{match}(l, \pi)$ and $\sigma' : \mathit{FreeVars}(\pi) \rightarrow T_\Sigma$ satisfying $\sigma =_{E\cup A} \sigma' \circ \sigma_0$[6].*

Assumption 3 holds under reasonable constraints, cf. the *Matching Lemma* [21]. In a nutshell, the constraints distinguish a *builtin subtheory* for the equational subtheory of the rewrite theory $\mathcal{R}$, with corresponding builtin equations and axioms, assumed to be manageable by an SMT solver; there are no non-builtin equations, while non-builtin axioms may include the usual combinations of associativity, commutativity, and unity. 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 4 (Computing ML Derivatives by Matching). *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*

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

[6] $\sigma_1 =_{E\cup A} \sigma_2$ iff $\mathit{dom}(\sigma_1) = \mathit{dom}(\sigma_2)$ and $(\forall x \in \mathit{dom}(\sigma_1))\sigma_1(x) =_{E\cup A} \sigma_2(x)$.

The following theorem directly follows from Lemma 4:

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

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

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 $\Delta_R^{match}(\varphi \Rightarrow \varphi')$ be the set

$$\{(\exists X \cup \mathit{FreeVars}(r) \setminus \mathit{FreeVars}(l))\,\sigma_0(r) \wedge \sigma_0(b) \wedge \phi \Rightarrow \varphi' \\ \mid l \to r \textbf{ if } b \in R, \sigma_0 \in match(l,\pi)\} \qquad (1)$$

We have $[\![\Delta_R^{match}(\varphi \Rightarrow \varphi')]\!] = [\![\Delta_R(\varphi \Rightarrow \varphi')]\!]$ by Theorem 2, which implies that $\Delta_R^{match}(\varphi \Rightarrow \varphi')$ is a complete set of $\mathcal{R}$-derivatives. This allows us to use of Δ_R^{match} in the procedure `prove` instead of Δ_R. Next, we note that formulas $\varphi_1 \Rightarrow \varphi'$ in $\Delta_R^{match}(\varphi \Rightarrow \varphi')$ (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 ψ is a fresh variable of sort *Bool*, with a matching substitution σ_0 such that $\sigma_0(\psi) = \phi$. Moreover, φ is R-derivable iff there are a rule $l \to r$ **if** b in R and $\sigma_0 \in match(l,\pi)$ such that $(\exists X \cup \mathit{FreeVars}(r) \setminus \mathit{FreeVars}(l))\sigma_0(b) \wedge \phi$ is satisfiable, by Lemma 4. This is equivalent to saying that φ is R-derivable iff the symbolic rewrite rule $l \wedge \psi \Rightarrow r \wedge b \wedge \psi$ is applicable to φ.

Overall, R-derivatives of RL formulas $\varphi \Rightarrow \varphi'$ (where $\varphi \triangleq (\exists X)(\pi \wedge \phi)$) can be computed by transforming each rule $l \to r$ **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 send 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, respectively 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

```
--- 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).

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. They respectively denote: the index of the next record to be sent, the sender-to-receiver channel, the receiver-to-sender channel, the next expected record on the receiver side, and the list of records currently accepted and stored by the receiver.

The file to be sent is modelled 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 $(\forall j)\, 0 \leq j \leq \texttt{Max} \longrightarrow \texttt{fileToSend(j) = file[j]}$, 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: 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 at Maude's metalevel. A first transformation is applied to rewrite rules as described in the *Symbolic rewrite rules* paragraph of Section 5. This reduces the application of rules with unification to application with matching. 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. 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.
2. 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.

3. 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`.
4. 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.
5. Grigore Roşu, Andrei Ştefănescu, Ştefan Ciobâcă, 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.
6. Andrei Ştefănescu, Ştefan Ciobâcă, 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.
7. José Meseguer and Grigore Roşu. The Rewriting Logic Semantics Project. *Theoretical Computer Science*, 373(3):213–237, 2007.
8. 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.
9. 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.
10. Grigore Roşu and Traian Florin Şerbănuţă. An overview of the K semantic framework. *Journal of Logic and Algebraic Programming*, 79(6):397–434, 2010.
11. José Meseguer. Twenty years of rewriting logic. *The Journal of Logic and Algebraic Programming*, 81(7):721–781, 2012.
12. Steven Eker, José Meseguer, and Ambarish Sridharanarayanan. The Maude LTL model checker. *Electronic Notes in Theoretical Computer Science*, 71:162–187, 2004.
13. Kyungmin Bae and José Meseguer. Model checking linear temporal logic of rewriting formulas under localized fairness. *Sci. Comput. Program.*, 99:193–234, 2015.
14. Kyungmin Bae, Santiago Escobar, and José Meseguer. Abstract logical model checking of infinite-state systems using narrowing. In *24th International Conference on Rewriting Techniques and Applications, RTA 2013, June 24-26, 2013, Eindhoven, The Netherlands*, pages 81–96, 2013.
15. Camilo Rocha and José Meseguer. Proving safety properties of rewrite theories. In *Algebra and Coalgebra in Computer Science*, pages 314–328. Springer, 2011.
16. Vlad Rusu. Combining theorem proving and narrowing for rewriting-logic specifications. In *Tests and Proofs*, pages 135–150. Springer, 2010.
17. Roberto Bruni and José Meseguer. Semantic foundations for generalized rewrite theories. *Theoretical Computer Science*, 360(1):386–414, 2006.
18. José Meseguer, Miguel Palomino, and Narciso Martí-Oliet. Equational abstractions. *Theoretical Computer Science*, 403(2):239–264, 2008.
19. José Meseguer, Miguel Palomino, and Narciso Martí-Oliet. Algebraic simulations. *Journal of Logic and Algebraic Programming*, 79(2), 2009.
20. Andrei Arusoaie, 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`.

21. 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.

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.

Proofs of Technical Results

Proof of **Proposition 1**, Page 5. We use the notation convention from Def. 3.

$\Leftarrow$. If $(\gamma, \rho) \models \varphi$ then we consider ρ' 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 $\Box$ be a fresh variable ($\Box \notin \mathit{Var}$) of sort *State* and $\varphi^{\Box}$ defined in the same way like $\varphi^{=}$, but using $\Box$ instead of z. Note that $\varphi^{\Box}$ is defined on a extended signature. If $\rho : \mathit{Var} \to M$ and $\gamma \in M_{\mathit{State}}$, then let $\rho^{\gamma} : \mathit{Var} \cup \{\Box\} \to M$ denote the extension of ρ with $\rho(\Box) = \gamma$. By Proposition 1 in [3] we have $\rho^{\gamma} \models \varphi^{\Box}$ iff $(\gamma, \rho) \models \varphi$. Assume that $\rho \models \varphi^{=?}$. It follows that for any extension ρ' of ρ to $\mathit{Var} \cup \{\Box\}$ we have $\rho' \models \varphi^{=?}$. Since $\varphi^{=?}$ can be obtained from $(\exists\Box)\varphi^{\Box}$ by alpha conversion, we obtain $\rho' \models (\exists\Box)\varphi^{\Box}$. It follows that there exists $\rho'' : \mathit{Var} \cup \{\Box\} \to M$ such that $\rho''(x) = \rho'(x)$ for $x \neq \Box$ and $\rho'' \models \varphi^{\Box}$. Since ρ' extends ρ, we obtain $\rho''(x) = \rho(x)$ for $x \in \mathit{Var}$. If we take $\gamma = \rho''(\Box)$, then $\rho'' = \rho^{\gamma}$ and hence $(\gamma, \rho) \models \varphi$. □

Proof of **Proposition 2**, Page 2. The FOL formula $\phi^{=?}$ is the same as ϕ because ϕ has no basic patterns, and hence $\phi^{=?}$ is equivalent to ϕ because the existential variable z does not occur in ϕ. It follows that $(\phi \wedge \varphi)^{=?} \triangleq (\exists z)(\phi \wedge \varphi)^{=}$ is equivalent $(\exists z)(\phi^{=} \wedge \varphi^{=})$, which in turn is equivalent to $\phi \wedge (\exists z)\varphi^{=}$, which is the same as $\phi \wedge \varphi^{=?}$.

We now prove the second part of the proposition. By Proposition 1 and $(\gamma, \rho) \models \varphi$ we obtain $\rho \models \varphi^{=?}$. By applying the definition of the FOL satisfaction relation to $\rho \models \phi$ and $\rho \models \varphi^{=?}$ we obtain $\rho \models \phi \wedge \varphi^{=?}$, which, by the first part of this proposition, is equivalent to $\rho \models (\phi \wedge \varphi)^{=?}$. Then, using Proposition 1 and its proof we obtain that $(\gamma, \rho) \models \phi \wedge \varphi$, which concludes the proof. □

Proof of **Lemma 1**, Page 8. Assume that φ_1 is $(\exists \mathit{FreeVars}(\varphi_l, \varphi_r))(\varphi_l \wedge \varphi)^{=?} \wedge \varphi_r$ for some $\varphi_l \Rightarrow \varphi_r \in \mathcal{S}$. If $[\![\varphi_1]\!] = \emptyset$ then $\varphi_1 \Rightarrow \varphi'$ is an $\mathcal{S}$-derivative of $\varphi \Rightarrow \varphi'$ by Definition 8. Assume $[\![\varphi_1]\!] \neq \emptyset$, i.e. there exists (τ_1, ρ_1) starting from φ_1, with

$\tau_1 \triangleq \gamma_1 \Rightarrow_{\mathcal{S}} \cdots$. Then

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

where, by Assumption 1, we may w.l.o.g. choose ρ such that $\rho(x) = \rho_1(x)$ for all $x \notin \mathit{FreeVars}(\varphi_l, \varphi_r)$. Hence there is $\tau = \gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1 \Rightarrow_{\mathcal{S}} \cdots$ such that $\tau|_{1..} = \tau_1$ and $(\gamma_0, \rho_1) \models \varphi$. If τ_1 is infinite then τ 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 γ_1 and ρ_1 have been chosen arbitrarily we conclude that $\varphi_1 \Rightarrow \varphi'$ is an $\mathcal{S}$-derivative of $\varphi \Rightarrow \varphi'$. □

Proof of **Lemma 2**, Page 8. Suppose that $(\tau_1, \rho) \models \varphi_1 \Rightarrow \varphi'$ and $\tau_1 \triangleq \gamma_1 \Rightarrow_{\mathcal{S}} \cdots$. Then (τ_1, ρ) starts from φ and one of the following two claims holds: a) there exists $i \geq 1$ such that $(\gamma_i, \rho) \models \varphi'$ or b) τ is infinite. So, to prove that $(\tau_1, \rho) \models \varphi_1' \Rightarrow \varphi'$ it is enough to prove that (τ_1, ρ) starts from some $\varphi_1' \in \Delta_{\mathcal{S}}(\varphi)$. The pair (τ_1, ρ) can be extended to (τ, ρ) 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 γ_0 such that $\tau \triangleq \gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1 \Rightarrow_{\mathcal{S}} \cdots$, $(\gamma_0, \rho) \models \varphi$, and $(\gamma_1, \rho) \models \varphi_1$. There is $\varphi_l \Rightarrow \varphi_r \in \mathcal{S}$ and ρ' 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 ρ' such that $\rho'(x) = \rho(x)$ for all $x \notin \mathit{FreeVars}(\varphi_l, \varphi_r)$. Hence $(\gamma_0, \rho') \models \varphi \wedge \varphi_l$. We take $\varphi_1' \triangleq (\exists \mathit{FreeVars}(\varphi_l, \varphi_r))(\varphi \wedge \varphi_l)^{=?} \wedge \varphi_r$. We obviously have $\varphi_1' \in \Delta_{\mathcal{S}}(\varphi)$ and $(\gamma_1, \rho) \models \varphi_1'$ because there exists ρ' (defined above) such that $(\gamma_1, \rho') \models (\varphi \wedge \varphi_l)^{=?} \wedge \varphi_r$. Since $\tau_1 = \gamma_1 \Rightarrow_{\mathcal{S}} \cdots$, it follows that (τ_1, ρ) starts from $\varphi_1' \in \Delta_{\mathcal{S}}(\varphi)$, which implies $(\tau_1, \rho) \models \varphi_1' \Rightarrow \varphi'$. Since (τ_1, ρ) has been chosen arbitrary, the conclusion of the lemma follows. □

Proof of **Lemma 3**, Page 9. The following equivalences hold:

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

Then, we have:

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

where, by the definition of $\models$, ρ satisfies $\rho(x) = \rho_1(x)$ for all $x \notin \mathit{FreeVars}(\varphi_l, \varphi_r)$. We obtained that $\bigvee_{\varphi_1 \in \Delta_{\mathcal{S}}(\varphi)} \varphi_1^{=?}$ is satisfiable iff there exists (γ, ρ) such that $(\gamma, \rho) \models \varphi$ and there exists γ_1 such that $\gamma \Rightarrow_{\mathcal{S}} \gamma_1$, i.e., iff φ is $\mathcal{S}$-derivable. $\square$

Proof of **Proposition 4**, Page 9. We use the notation convention in Definition 3.

$$\begin{array}{lll}
M \models \varphi^{=?} \longrightarrow \bigvee_{\varphi_1 \in \Delta_{\mathcal{S}}(\varphi)} \varphi_1^{=?} & \longleftrightarrow & \\
(\forall \rho)\rho \models \varphi^{=?} \longrightarrow \rho \models \bigvee_{\varphi_1 \in \Delta_{\mathcal{S}}(\varphi)} \varphi_1^{=?} & \longleftrightarrow & \\
(\forall \rho)(\exists \gamma)(\gamma, \rho) \models \varphi \longrightarrow & & \\
\quad (\exists \varphi_l \Rightarrow \varphi_r \in \mathcal{S})\rho \models (\exists \mathit{FreeVars}(\varphi_l, \varphi_r))((\varphi_l \wedge \varphi)^{=?} \wedge \varphi_r)^{=?} & \longleftrightarrow & \\
(\forall \rho)(\exists \gamma)(\gamma, \rho) \models \varphi \longrightarrow & & \\
\quad (\exists \varphi_l \Rightarrow \varphi_r \in \mathcal{S})\rho \models (\exists \mathit{FreeVars}(\varphi_l, \varphi_r))(\varphi_l \wedge \varphi)^{=?} \wedge \varphi_r^{=?} & \longleftrightarrow & (2) \\
(\forall \rho)(\exists \gamma)(\gamma, \rho) \models \varphi \longrightarrow & & \\
\quad (\exists \varphi_l \Rightarrow \varphi_r \in \mathcal{S})(\exists \rho')\rho' \models (\varphi_l \wedge \varphi)^{=?} \wedge (\exists \gamma_1)(\gamma_1, \rho') \models \varphi_r & \longleftrightarrow & (3) \\
(\forall \rho)(\exists \gamma)(\gamma, \rho) \models \varphi \longrightarrow & & \\
\quad (\exists \varphi_l \Rightarrow \varphi_r \in \mathcal{S})(\exists \rho')(\gamma, \rho') \models (\varphi_l \wedge \varphi) \wedge (\exists \gamma_1)(\gamma_1, \rho') \models \varphi_r & \longleftrightarrow & (4) \\
(\forall \rho)(\exists \gamma)(\gamma, \rho) \models \varphi \longrightarrow (\exists \gamma_1)\gamma \Rightarrow_{\mathcal{S}} \gamma_1 & &
\end{array}$$

where, by Assumption 1, we may w.l.o.g. choose ρ' such that $\rho'(x) = \rho(x)$ for all $x \notin \mathit{FreeVars}(\varphi_l, \varphi_r)$, which implies $(\gamma', \rho') \models \varphi$ iff $\gamma' = \gamma$ and $(\gamma, \rho) \models \varphi$. Therefore in the equivalence (3) $\longleftrightarrow$ (4) we could take $(\gamma, \rho') \models (\varphi_l \wedge \varphi)$. The equivalence (2) follows by Proposition 2. $\square$

Proof of **Theorem 1**, Page 10. The following lemmas are needed in the proof.

Lemma 5 (Coverage Step). *Let γ, γ', ρ, φ, and $\alpha \triangleq \varphi_l \Rightarrow \varphi_r \in \mathcal{S}$ such that $\gamma \Rightarrow_{\{\alpha\}} \gamma'$ and $(\gamma, \rho) \models \varphi$. Then, $(\gamma', \rho) \models \Delta_{\{\alpha\}}(\varphi)$.*

Proof. From $\gamma \Rightarrow_{\{\alpha\}} \gamma'$ we obtain a valuation ρ' such that $(\gamma, \rho') \models \varphi_l$ and $(\gamma', \rho') \models \varphi_r$. By Assumption 1, $\mathit{FreeVars}(\varphi_l, \varphi_r) \cap \mathit{FreeVars}(\varphi) = \emptyset$. Hence we can choose ρ' such that $\rho'(x) = \rho(x)$ for all $x \in \mathit{FreeVars}(\varphi)$. Thus, $(\gamma, \rho') \models \varphi$. From the latter and $(\gamma, \rho') \models \varphi_l$ we obtain $(\gamma, \rho') \models \varphi \wedge \varphi_l$, and using Proposition 1 we have $\rho' \models (\varphi \wedge \varphi_l)^{=?}$. Using Proposition 2 we obtain $(\gamma', \rho') \models (\varphi \wedge \varphi_l)^{=?} \wedge \varphi_r$ which implies $(\gamma', \rho) \models (\exists \mathit{FreeVars}(\varphi_l, \varphi_r))(\varphi \wedge \varphi_l)^{=?} \wedge \varphi_r$ (using Assumption 1). By Definition 10 $(\exists \mathit{FreeVars}(\varphi_l, \varphi_r))(\varphi \wedge \varphi_l)^{=?} \wedge \varphi_r$ is just $\Delta_{\{\alpha\}}(\varphi)$, which ends the proof. □

Lemma 6 (Coverage by Derivatives). *Any computation $\tau \triangleq \gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1 \Rightarrow_{\mathcal{S}} \cdots$ with (τ, ρ) starting from φ is "covered" by derivatives, i.e., there exists a sequence $\varphi_0, \varphi_1, \ldots$ of* ML *formulas such that*

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

Proof. By induction on i using Lemma 5 in the induction step. □

A successful execution of `prove`$(\mathcal{S}, G_0, \Delta_{\mathcal{S}}(G_0))$ consists of a sequence of calls

$$\texttt{prove}(\mathcal{S}, G_0, \mathcal{G}_1),\ \ldots,\ \texttt{prove}(\mathcal{S}, G_0, \mathcal{G}_n)$$

such that

- $\mathcal{G}_0 = G_0$,
- $\mathcal{G}_1 = \Delta_{\mathcal{S}}(G_0)$,
- $\mathcal{G}_n = \emptyset$,
- for all $i \in 0 \ldots n-1$, $\mathcal{G}_{i+1} = \mathcal{G}_i \setminus \{\varphi \Rightarrow \varphi'\} \cup \mathcal{G}_{\varphi \Rightarrow \varphi'}$, for some $\varphi \Rightarrow \varphi' \in \mathcal{G}_i$, where

$$\mathcal{G}_{\varphi \Rightarrow \varphi'} = \begin{cases} \emptyset & \text{, } \mathit{if}\ M \models \varphi \longrightarrow \varphi' \\ \Delta_{\varphi_c \Rightarrow \varphi'_c}(\varphi \Rightarrow \varphi') & \text{, } \mathit{if\ there\ is}\ \varphi_c \Rightarrow \varphi'_c \in G_0, s.t. M \models \varphi \longrightarrow \overline{\varphi}_c, \\ \Delta_{\mathcal{S}}(\varphi \Rightarrow \varphi'_c) & \text{, } \mathit{if}\ \varphi\ \mathcal{S}\text{-derivable} \end{cases}$$

In the following we let $\mathcal{F} = \bigcup_{i=0}^{n} \mathcal{G}_i$.

Lemma 7. *Let $\varphi \Rightarrow \varphi' \in \mathcal{F}$. Then $M \models \varphi \longrightarrow \varphi'$ or φ is $\mathcal{S}$-derivabile.*

Proof. Let $\varphi \Rightarrow \varphi' \in \mathcal{F}$. If $\varphi \Rightarrow \varphi' \in \mathcal{G}_0 = G_0$ then φ is $\mathcal{S}$-derivabile (any formula in G_0 has the lhs $\mathcal{S}$-derivable). Otherwise, there is i such that $\varphi \Rightarrow \varphi' \in \mathcal{G}_i \setminus \mathcal{G}_{i+1}$. By the definition of $\mathcal{G}_{i+1}$ the formula $\varphi \Rightarrow \varphi'$ was eliminated from $\mathcal{G}_i$ in one of the three situations:

1. $M \models \varphi \longrightarrow \varphi'$
2. $M \models \varphi \longrightarrow \overline{\varphi}_c$ for some $\varphi_c \Rightarrow \varphi'_c \in G_0$
3. φ is $\mathcal{S}$-derivable.

In the first and the third cases we obtain directly the conclusion of our lemma. The only case we have to discuss is the second one. Note that there are γ and ρ such that $(\gamma, \rho) \models \varphi$. Otherwise, we have $M \models \varphi \longrightarrow \varphi'$ which corresponds to the first case. From $(\gamma, \rho) \models \varphi$ and $M \models \varphi \longrightarrow \overline{\varphi}_c$ we have $(\gamma, \rho) \models \overline{\varphi}_c$. By Definition 2, there is ρ' such that $(\gamma, \rho') \models \varphi_c$. Since φ_c is derivable (because $\varphi_c \Rightarrow \varphi'_c \in G_0$ and $\mathcal{S}$ is total, there exists a transition $\gamma \Rightarrow_{\mathcal{S}} \gamma_1$, which, by Definition 11, implies that φ is $\mathcal{S}$-derivable. □

Lemma 8. *For all τ, for all ρ, for all $\varphi \Rightarrow \varphi' \in \mathcal{F}$, if τ is finite and complete, and (τ, ρ) starts from φ then $(\tau, \rho) \models \varphi \Rightarrow \varphi'$.*

Proof. We proceed by induction on the length of τ. We an consider arbitrary $\varphi \Rightarrow \varphi' \in \mathcal{F}$ and ρ satisfying the hypotheses of the lemma.

Base case. Assume that $\tau = \gamma_0$ and that $(\gamma_0, \rho) \models \varphi$. Since τ is complete then there is no γ_1 such that $\gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1$. Therefore, any φ' such that $(\gamma_0, \rho) \models \varphi'$, is not $\mathcal{S}$-derivable (otherwise, it contradicts Definition 11). Thus, φ is not $\mathcal{S}$-derivable.

By Lemma 7 we have $M \models \varphi \longrightarrow \varphi'$, and using the fact that $(\gamma_0, \rho) \models \varphi$ we obtain $(\gamma_0, \rho) \models \varphi'$, i.e. $(\tau, \rho) \models \varphi \Rightarrow \varphi'$.

Induction step. Assume that $\tau = \gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1 \cdots$, and $(\gamma_0, \rho) \models \varphi$. In this case φ is $\mathcal{S}$-derivable (by Definition 11). Since $\varphi \Rightarrow \varphi' \in \mathcal{F}$ then $\varphi \Rightarrow \varphi'$ has been eliminated at some point, so there is i such that $\varphi \Rightarrow \varphi' \in \mathcal{G}_i \setminus \mathcal{G}_{i+1}$.

Again, by the definition of $\mathcal{G}_{i+1}$, we have three possible cases:

1. $M \models \varphi \longrightarrow \varphi'$. Since $(\gamma_0, \rho) \models \varphi$ we obtain $(\gamma_0, \rho) \models \varphi'$, which implies $(\tau, \rho) \models \varphi \Rightarrow \varphi'$.
2. $M \models \varphi \longrightarrow \overline{\varphi}_c$. From $(\gamma_0, \rho) \models \varphi$ we obtain $(\gamma_0, \rho) \models \overline{\varphi}_c$, and, by Definition 2, there is ρ' with $\rho'(x) = \rho(x)$ for all $x \notin \mathit{FreeVars}(\varphi_c)$ such that $(\gamma_0, \rho') \models \varphi_c$. If $\gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1$ then there is a rule $\alpha \triangleq \varphi_l \Rightarrow \varphi_r \in \mathcal{S}$ such that $\gamma_0 \Rightarrow_{\{\alpha\}} \gamma_1$ (Definition 5). From $(\gamma_0, \rho') \models \varphi_c$ and Lemma 5 we obtain $\varphi_1 \in \Delta_{\{\alpha\}}(\varphi_c) \subseteq \Delta_{\mathcal{S}}(\varphi_c)$ such that $(\gamma_1, \rho') \models \varphi_1$. Since $\varphi_1 \in \Delta_{\mathcal{S}}(\varphi_c)$ and $\varphi_c \Rightarrow \varphi'_c \in G_0 = \mathcal{G}_0$ then $\varphi_1 \Rightarrow \varphi'_c \in \Delta_S(\varphi_c \Rightarrow \varphi'_c) \subseteq \mathcal{G}_1 \subseteq \mathcal{F}$. Now, the inductive hypothesis holds for $\varphi_1 \Rightarrow \varphi'_c$, and we have $(\tau|_{1..}, \rho') \models \varphi_1 \Rightarrow \varphi'_c$. Since τ is finite, there exists $j \geq 1$ such that $(\gamma_j, \rho') \models \varphi'_c$.
Next, we want show that $(\gamma_j, \rho) \models (\exists \mathit{FreeVars}(\varphi_c, \varphi'_c))((\varphi_c \wedge \varphi)^{=?} \wedge \varphi'_c)$. This is equivalent (by Definition 2) to showing that there is a valuation ρ'' with $\rho''(x) = \rho(x)$ for all $x \notin \mathit{FreeVars}(\varphi_{c'})$ such that $(\gamma_j, \rho'') \models (\varphi_c \wedge \varphi)^{=?} \wedge \varphi'_c$. Let us consider $\rho'' = \rho'$. From the hypothesis of Theorem 1 we have $\mathit{FreeVars}(\varphi'_c) \subseteq \mathit{FreeVars}(\varphi_c)$, which implies that $\mathit{FreeVars}(\varphi_c, \varphi'_c) \subseteq \mathit{FreeVars}(\varphi_c)$. Also, note that $\rho'(x) = \rho(x)$, for all $x \notin \mathit{FreeVars}(\varphi_c)$. Using Assumption 1, i.e. $\mathit{FreeVars}(\varphi) \cap \mathit{FreeVars}(\varphi_c) = \emptyset$, and $(\gamma_0, \rho) \models \varphi$ we obtain $(\gamma_0, \rho') \models \varphi$. Given the fact that $(\gamma_0, \rho') \models \varphi_c$, by Definition 2, $(\gamma_0, \rho') \models \varphi_c \wedge \varphi$. By Proposition 1, from $(\gamma_0, \rho') \models \varphi_c \wedge \varphi$ we obtain $\rho' \models (\varphi_c \wedge \varphi)^{=?}$. Moreover, by Proposition 2 and the fact that $(\gamma_j, \rho') \models \varphi'_c$ we obtain $(\gamma_j, \rho') \models (\varphi_c \wedge \varphi)^{=?} \wedge \varphi'_c$. Therefore, there is $\rho'' = \rho'$, such that $(\gamma_j, \rho'') \models (\varphi_c \wedge \varphi)^{=?} \wedge \varphi'_c$, and we can conclude that $(\gamma_j, \rho) \models (\exists \mathit{FreeVars}(\varphi_c, \varphi'_c))((\varphi_c \wedge \varphi)^{=?} \wedge \varphi'_c)$.
Note that the set $\mathcal{G}_{i+1}$ includes $\Delta_{\varphi_c \Rightarrow \varphi'_c}(\varphi \Rightarrow \varphi'_c)$, and we can apply again the inductive hypothesis: $(\tau|_{j..}, \rho) \models (\exists \mathit{FreeVars}(\varphi_c, \varphi'_c))((\varphi_c \wedge \varphi)^{=?} \wedge \varphi'_c) \Rightarrow \varphi'$, i.e. there is $k \geq j$ such that $(\gamma_k, \rho) \models \varphi'$, which implies $(\tau, \rho) \models \varphi \Rightarrow \varphi'$.
3. φ is $\mathcal{S}$-derivable. Then $\Delta_{\mathcal{S}}(\varphi \Rightarrow \varphi') \subseteq \mathcal{G}_{i+1} \subseteq \mathcal{F}$. If $\gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1$ then there is a rule $\alpha \triangleq \varphi_l \Rightarrow \varphi_r \in \mathcal{S}$ s. t. $\gamma_0 \Rightarrow_{\{\alpha\}} \gamma_1$ (Definition 5). Since $(\gamma_1, \rho) \models \varphi_1$, then, by Lemma 5, there is $\varphi_1 \in \Delta_{\{\alpha\}}(\varphi) \subseteq \Delta_{\mathcal{S}}(\varphi)$ such that $(\gamma_1, \rho) \models \varphi_1$. We obtain $(\tau|_{1..}, \rho) \models \varphi_1 \Rightarrow \varphi'$ by the inductive hypothesis, which implies that there is $j \geq 1$ s. t. $(\gamma_j, \rho) \models \varphi'$. Hence $(\tau, \rho) \models \varphi \Rightarrow \varphi'$. $\square$

Proof (of Theorem 1). Let $\tau \triangleq \gamma_0 \Rightarrow_{\mathcal{S}} \gamma_1 \Rightarrow_{\mathcal{S}} \cdots$ be a complete execution path, and let the valuation ρ be such that (τ, ρ) starts from φ_0 with $\varphi_0 \Rightarrow \varphi'_0 \in G_0$. If τ is finite then $(\tau, \rho) \models \varphi \Rightarrow \varphi'_c$ by Lemma 8. If τ is infinite then $(\tau, \rho) \models \varphi \Rightarrow \varphi'_c$ by Definition 7. $\square$

Proof of **Lemma 4**, Page 13. By definition of $\Delta_R(\varphi)$, φ_1 is $(\exists \mathit{FreeVars}(l,r))(l \wedge b \wedge (\exists X)(\pi \wedge \phi))^{=?} \wedge r$ for some rewrite rule $l \to r$ **if** $b \in R$. By Assumption 1, $\mathit{FreeVars}(l,r) \cap X = \emptyset$ and hence φ_1 is equivalent to $(\exists X \cup \mathit{FreeVars}(l,r))(l = \pi) \wedge b \wedge \phi \wedge r$. We have:

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

where

- $\gamma_1 \in T_{\Sigma, E \cup A}$ of sort *State*, i.e., γ_1 is an $(E \cup A)$-equivalence class $[t]$ with $t \in T_{\Sigma, \mathit{State}}$;
- by Assumption 1, we may assume w.l.o.g. that $\rho(x) = \rho_1(x)$ for all $x \notin X \cup \mathit{FreeVars}(l,r)$;
- $\sigma : \mathit{FreeVars}(l, r, \phi) \to T_\Sigma$ with $[\sigma(x)] = \rho(x)$;
- the substitutions $\sigma_0 : \mathit{FreeVars}(l) \to \mathit{FreeVars}(\pi)$ and $\sigma' : \mathit{FreeVars}(\pi) \to T_\Sigma$ are given by Assumption 3, i.e., $\sigma|_{\mathit{FreeVars}(l,\pi)} = \sigma' \circ \sigma_0$; note that σ' is uniquely determined by σ and σ_0;
- $\sigma'' = \sigma|_{\mathit{FreeVars}(r) \setminus \mathit{FreeVars}(l)}$;
- $\sigma'' \uplus \sigma'(x) = \sigma''(x)$ if $x \in \mathit{FreeVars}(r) \setminus \mathit{FreeVars}(l)$, and $\sigma'' \uplus \sigma'(x) = \sigma'(x)$ if $x \in \mathit{FreeVars}(\sigma_0(l))$;
- $\rho_0(x) = [\sigma'(x)]$, for $x \in \mathit{FreeVars}(\pi)$, $\rho_0(x) = [\sigma''(x)]$, for $x \in \mathit{FreeVars}(r) \setminus \mathit{FreeVars}(l)$, and $\rho_0(x) = \rho(x)$ in the rest (hence $\rho_0(x) = \rho_1(x)$ for $x \notin X \cup \mathit{FreeVars}(r) \setminus \mathit{FreeVars}(l)$); and
- $\rho \models b$ iff $\sigma''(\sigma_0(b)) =_{E \cup A} \mathit{true}$ iff $\rho_0 \models \sigma_0(b)$. $\square$