Verifying Invariants of Rewriting Logic Specifications

Vlad Rusu[1] and Manuel Clavel[2,3]

[1] Inria Rennes Bretagne-Atlantique, Rennes, France
rusu@irisa.fr
[2] Universidad Complutense, Madrid, Spain
clavel@sip.ucm.es
[3] IMDEA Software Institute, Madrid, Spain
manuel.clavel@imdea.org

Abstract. We present a novel approach based on inductive theorem proving for verifying invariants of dynamic systems specified in rewriting logic, a formal specification language implemented in the Maude system. An invariant is a property that holds on all the states that are reachable from a given class of initial states. Our approach consists in encoding the semantic aspects that are relevant for our task (namely, verifying invariance properties of the specified systems) in membership equational logic, a sublogic of rewriting logic. The invariance properties are then formalized over the encoded rewrite theories and are proved using an inductive theorem prover for membership equational logic also implemented in the Maude system using its reflective capabilities. We illustrate our approach by verifying mutual exclusion in an n-process Bakery algorithm.

1 Introduction

Rewriting logic [1], abbreviated as RL in this paper, is a formal specification language, in which a system's dynamics can be expressed by means of *rewrite rules* over a system's *state* defined equationally in some version of equational logic. The adequacy of rewriting logic for specifying dynamic systems has been demonstrated by many practical applications, including programming language semantics [2], active networks, [3], and bioinformatics [4]. Currently, there are several systems which implement different variants of this logic, e.g., Maude [5], Elan [6], and CafeObj [7]. *Membership equational logic* [8], abbreviated as MEL in this paper, is the logic implemented in Maude as RL's underlying equational logic. This logic generalises order-sorted logic with so-called *membership axioms*. Our approach for inductively verifying invariants of RL theories relies on the expressivity of MEL, which allows us to inductively define the sort of those terms that are *reachable from a given class of initial terms* in a RL specification.

The Maude system [5] consists of a language for expressing RL and MEL specifications, along with a set of tools for analysing such specifications and verifying them against user-defined properties. In particular, the finite-state verification tools provided by the Maude system include a state-space searching tool and a model checker for linear temporal logic properties [9]. Infinite-state systems can also be verified in Maude by abstracting infinite state spaces to finite ones using

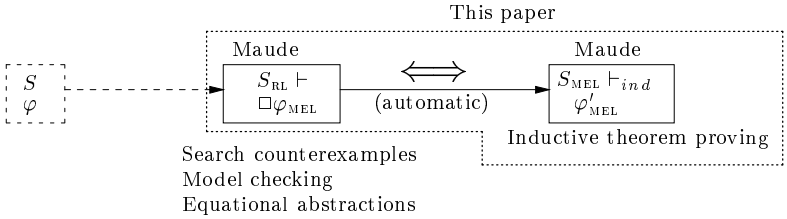

Fig. 1. Our approach in the context of Maude's verification tools.

equational abstractions [10]. However, equationally defining *sound* abstractions is not easy - equational abstractions do not preserve reachability in general [10].

Hence, in Maude, interactive theorem remains a useful verification approach, both by itself and in combination with Maude's automatic verification tools.

Our contribution in this paper is an approach based on inductive theorem proving for verifying invariants of RL specifications. Intuitively, an invariant is a predicate that holds in all states that are reachable from a given class of initial states. Our approach consists in translating invariants of RL theories into inductive properties of MEL theories. We prove that the translation is correct, i.e., it maps invariants expressed in MEL to true inductive MEL properties, which can then be proved using available inductive theorem provers like the ITP tool [11].

The proposed approach completes the set of verification tools available for Maude, and can be used in conjunction with those other tools (see Figure 1). Assume a system S and a predicate φ on the system's state. We formalise them in Maude as, respectively, a MEL sentence φ_{MEL} and RL theory S_{RL} and formalise the fact that φ is an invariant of S (starting from a given class of initial states of S) as $S_{\text{RL}} \vdash \Box\varphi_{\text{MEL}}$. Then, e.g., the user can try to falsify statement $S_{\text{RL}} \vdash \Box\varphi_{\text{MEL}}$ using Maude's model checker or search command; or she can try to prove the statement using Maude's model checker, maybe in conjuction with equational abstractions, or, *via* our translation to MEL, by inductive theorem proving.

The rest of the paper is organised as follows. In Section 2 we provide background on MEL and RL. We then define the notion of an *invariant φ of a* RL *theory $\mathcal{R}$ starting from an initial term t_0*, denoted by $\langle \mathcal{R}, t_0 \rangle \vdash \Box\varphi$, as follows: for any predicate φ expressed in MEL, $\langle \mathcal{R}, t_0 \rangle \vdash \Box\varphi$ means that $\varphi(t)$ is equationally provable in the standard (i.e., the initial) model of the underlying MEL theory of $\mathcal{R}$, for all ground terms t reachable from a term t_0 by *ground top-level rewriting* in $\mathcal{R}$. Ground top-level rewriting requires that matching occurs at the outermost level of the term to be rewritten, and only *via* ground substitutions of all the free variables occuring in the term and the rewrite rule. We demonstrate the adequacy of our notion of ground top-level rewriting for expressing the dynamics of systems expressed in RL, especially when the class of initial states is expressed by a non-ground term t_0 and when rewrite rules have supplementary variables in their right-hand sides and conditions. These features are crucial for modelling parametric systems, such as our n-proces Bakery algorithm example.

The core of the approach is presented in Section 3. First, we define an automatic translation that takes a RL theory $\mathcal{R}$ and a term t_0, and generates a MEL theory $\mathcal{M}(\mathcal{R}, t_0)$, which enriches the MEL subtheory of $\mathcal{R}$ with a new sort, called

Reachable, and with the membership axioms that inductively define this sort. Then, we show that "being of sort *Reachable* in $\mathcal{M}(\mathcal{R}, t_0)$" and "being reachable in $\mathcal{R}$ from t_0 by ground top-level rewriting" are equivalent statements. Next, we give an alternative definition of the notion of an *invariant φ of a* RL *theory $\mathcal{R}$ starting from an initial term t_0*, as follows: $\varphi(t)$ is equationally provable in the standard (i.e., the initial) model of $\mathcal{M}(\mathcal{R}, t_0)$ for all ground terms t of sort *Reachable* in $\mathcal{M}(\mathcal{R}, t_0)$. We finally prove that the two definitions of invariance are equivalent. The advantage of the second one is that allows us to prove invariants of RL specifications by induction on the sort *Reachable*, using existing inductive theorem provers for MEL such as the ITP tool.

The application of our approach to the n-process Bakery algorithm is described in Section 4. In Section 5 we conclude and discuss related and future work. A separate Appendix contains proofs for the results stated in the paper.

2 Membership Equational Logic and Rewriting Logic

We give here a brief presentation of membership equational logic and rewriting logic; for a full account the reader may consult [1,8].

A membership equational logic (MEL) *signature* is a tuple (K, Σ, S) where K is a set of *kinds*, Σ is a $K^* \times K$ indexed family of function symbols $\Sigma = \{\Sigma_{w,k}\}_{(w,k)\in K^*\times K}$, and $S = \{S_k\}_{k\in K}$ is a pairwise disjoint K-indexed family of sets of *sorts* - where S_k is the set of sorts of kind k. A signature (K, Σ, S) is often denoted simply by Σ; then, T_Σ denotes the set of ground terms over signature Σ. Given a set $X = \{x_1 : k_1, \ldots, x_n : k_n\}$ of *kinded variables*, $T_\Sigma(X)$ denotes the set of terms with free variables in the set X. Similarly, $T_{\Sigma,k}$ and $T_{\Sigma,k}(X)$ denote, respectively, the set of ground terms of kind k and the set of terms of kind k with free variables in the set X. A MEL *atomic formula* over (K, Σ, S) is either an equality $t = t'$, where t and t' are terms in $T_{\Sigma,k}(X)$, for some kind $k \in K$, or a *membership assertion* $t : s$, where t is a term in $T_{\Sigma,k}(X)$ and s is a sort in S_k, for some kind $k \in K$. A MEL *sentence* is a universally quantified Horn clause on atomic formulas:

$$(\forall X) t = t' \text{ if } C, \text{ or} \tag{1}$$

$$(\forall X) t : s \text{ if } C \tag{2}$$

where the *condition* C has the form (for some finite sets of indices I, J):

$$\bigwedge_{i\in I}(u_i = v_i) \wedge \bigwedge_{j\in J}(w_j : s_j)$$

Sentences of the form (1) are called *conditional equations*, and sentences of the form (2) are called *conditional memberships*. An equation (resp. a membership) is called *unconditional* when it does not have conditions.

A MEL *theory* is a tuple $\mathcal{M} = (\Sigma, E)$ that consists of a MEL signature Σ and a set of MEL sentences over Σ. MEL has a complete deduction system [8], in the sense that a formula φ is provable from the sentences of a theory (Σ, E), denoted as $(\Sigma, E) \vdash \varphi$ (or simply $E \vdash \varphi$), if and only φ is semantically valid, i.e., it holds in *all* the *models* of that theory.

Definition 1. *Consider a* MEL *theory* $\mathcal{M} = (\Sigma, E)$ *and an atomic formula* e *of the form* $(\forall X)t = t'$ *or* $(\forall X)t : s$. *Then,* $\mathcal{M}$ *entails* e, *denoted by* $\mathcal{M} \vdash e$ *(or just* $E \vdash e$ *when the signature is clear from the context) if* e *can be obtained from the sentences in* E *by using the following rules:*

1. *Reflexivity:* $\frac{t \in T_\Sigma(X)}{E \vdash (\forall X) t = t}$
2. *Membership:* $\frac{E \vdash (\forall X) t' = t \quad E \vdash (\forall X) t : s}{E \vdash (\forall X) t' : s}$
3. *Symmetry:* $\frac{E \vdash (\forall X) t' = t}{E \vdash (\forall X) t = t'}$
4. *Transitivity:* $\frac{E \vdash (\forall X) t_1 = t_2 \quad E \vdash (\forall X) t_2 = t_3}{E \vdash (\forall X) t_1 = t_3}$
5. *Congruence:* $\frac{f \in \Sigma_{k_1,\ldots,k_n,k} \quad t_1,\ldots t_n \in T_{\Sigma_{k_i}}(X) \quad for\ i \in [1,n] \quad t'_i \in T_{\Sigma_{k_i}}(X) \quad E \vdash (\forall X) t_i = t'_i}{E \vdash (\forall X) f(t_1,\ldots,t_i,\ldots,t_n) = f(t_1,\ldots,t'_i,\ldots,t_n)}$
6. *Replacement*$_1$ $\frac{(\forall X) t = t'\ if\ C \quad \in E \quad \sigma : X \mapsto T_\Sigma(Y) \quad E \vdash (\forall Y) C\sigma}{E \vdash (\forall Y) t\sigma = t'\sigma}$
7. *Replacement*$_2$ $\frac{(\mu)\ (\forall X) t : s\ if\ C \quad \in E \quad \sigma : X \mapsto T_\Sigma(Y) \quad E \vdash (\forall Y) C\sigma}{E \vdash (\forall Y) t\sigma : s}$

where $\sigma : X \mapsto T_\Sigma(Y)$ *are kind-preserving substitutions, and for a condition* $C : \bigwedge_{i \in I} (\forall X)(u_i = v_i) \wedge \bigwedge_{j \in J} (\forall X)(w_j : s_j)$, *the notation* $E \vdash (\forall Y) C\sigma$ *is a shortcut for the conjunction* $\bigwedge_{i \in I} E \vdash (\forall Y)(u_i\sigma = v_i\sigma) \wedge \bigwedge_{j \in J} E \vdash (\forall Y)(w_j\sigma : s_j)$.

However, semantic validity may not be a very adequate notion of *truth* when reasoning about specifications, since there are many interesting properties that do hold in their "standard" models but do not hold in others of their models. The standard model of an specification is called its *initial model* [8]. In the initial model of a MEL theory, sorts are interpreted as the smallest sets satisfying the axioms in the theory, and equality is interpreted as the smallest congruence satisfying those axioms. We write $E \vdash_{ind} (\forall X)\varphi$ to say that the sentence φ holds in the initial model of (Σ, E).

Example 1. The following MEL theory NAT is an specification of the natural numbers, with no arithmetic operations.

- $K_{NAT} = \{Nat?\}$.
- $\Sigma_{NAT_{(\lambda, Nat?)}} = \{0\}$.
- $\Sigma_{NAT_{(Nat?, Nat?)}} = \{s\}$.
- $\Sigma_{NAT_{w, Nat?}} = \emptyset$, for $w \notin \{\lambda, Nat?\}$.
- $S_{Nat?} = \{Nat\}$.
- $E_{NAT} = \{0 : Nat, (\forall N)\ s(N) : Nat\ \ if\ \ N : Nat\}$.

The initial model of the theory NAT is the (usual) set $\mathbb{N}$ of natural numbers.

A *rewriting logic* (RL) theory is a tuple $\mathcal{R} = (K, \Sigma, S, E, R)$, where (K, Σ, S, E) is a MEL theory and R is a set of *rewrite rules* of the form:

$$(\rho) \quad (\forall X)\ l \to r \quad \textit{if} \quad C \tag{3}$$

where the condition C has the form $\bigwedge_{i \in I}(u_i = v_i) \wedge \bigwedge_{j \in J}(w_j : s_j)$ for some finite sets of indices I and J; that is, like for MEL sentences, we consider that only equations and memberships are allowed in the conditions of the rules. Note that in its most general form [12], rewriting logic also allows for *rewrites in conditions* and *frozen arguments*, which we do not consider here.

Definition 2. *For a* RL *theory* $\mathcal{R} = (\Sigma, E, R)$ *and two terms* $t, t' \in T_\Sigma(X)$*, we say that* $\mathcal{R}$ entails *the sequent* $(\forall X)t \longrightarrow t'$ at top level*, denoted by* $\mathcal{R} \vdash (\forall X)t \longrightarrow t'$*, iff the sequent can be obtained by applying the following rules:*

1. *Reflexivity:* $\frac{t \in T_\Sigma(X)}{\mathcal{R} \vdash (\forall X)t \longrightarrow t}$
2. *Transitivity:* $\frac{\mathcal{R} \vdash (\forall X)t_1 \longrightarrow t_2 \quad \mathcal{R} \vdash (\forall X)t_2 \longrightarrow t_3}{\mathcal{R} \vdash (\forall X)t_1 \longrightarrow t_3}$
3. *Equality:* $\frac{E \vdash (\forall X)t = u \quad \mathcal{R} \vdash (\forall X)u \longrightarrow u' \quad E \vdash (\forall X)u' = t'}{\mathcal{R} \vdash (\forall X)t \longrightarrow t'}$
4. *Replacement:* $\frac{(\rho)\ (\forall X)\ l \longrightarrow r \text{ if } C \in R \quad \sigma : X \mapsto T_\Sigma(Y) \quad E \vdash (\forall Y)\sigma(C)}{\mathcal{R} \vdash (\forall Y)\sigma(l) \longrightarrow \sigma(r)}$

We note that in the logic's most general form [12] there is also a Congruence rule. Eliminating the Congruence rule amounts to imposing rewriting at top level.

We now introduce *ground top-level rewriting*. Intuitively, when rewriting a term $(\forall X)t$ with a rule $(\forall Y)l \to r \ if \ C$ by ground top-level rewriting, the term $(\forall X)t$ is first transformed into a ground instance $\sigma(t)$ of itself, and then $\sigma(t)$ is rewritten at top level, such that the matching of $\sigma(t)$ by l is performed by means of a ground substitution of *all* variables occuring in l, r, and C.

Definition 3. *For a* RL *theory* $\mathcal{R} = (\Sigma, E, R)$ *and two terms* $t \in T_\Sigma(X)$ *and* $t \in T_\Sigma$*, we say that* $\mathcal{R}$ entails *the sequent* $(\forall X)t \longrightarrow t'$ at ground top level*, denoted by* $\mathcal{R} \vdash\!\downarrow (\forall X)t \longrightarrow t'$*, if there exists a ground substitution* $\sigma : X \mapsto T_\Sigma$ *and a derivation* $\mathcal{R} \vdash \sigma(t) \longrightarrow t'$ *at top level (cf. Definition 2) in which all applications of the* Replacement *rule use ground substitutions.*

Example 2. To illustrate ground top-level rewriting, consider the term $s(W)$ in the theory NAT and the rule $s(X) \to s(s(Y))$. Then, $s(W)$ can be ground top-level rewritten into $s(s(0))$ in one step: $s(W)$ is transformed into the ground term $s(0)$ using the ground substitution $W \leftarrow 0$, and $s(0)$ is rewritten to $s(s(0))$ by the rule, using the ground substitution $X \leftarrow 0, Y \leftarrow 0$. Note that "standard" rewriting of the given term with the given rule generates sequences of non-ground terms, e.g., $s(W), s(s(W)), \ldots, s^n(W), \ldots$, none of which denote states of a system modelled in RL, whereas ground rewriting only generates ground terms, which do denote states. This is the main reason why ground rewriting is the adequate notion for expressing the dynamics of systems specified in rewriting logic, especially when the set of initial states is given by a non-ground term and rules have supplementary variables in right-hand sides and conditions. Parametric systems, such as the n-process Bakery algorithm discussed in Section 4, naturally use such supplementary variables for expressing their dynamics.

The next example is relatively simple (its initial state is denoted by a ground initial term, and its rewrite rules have no supplementary variables in right-hand sides/conditions). We use it as a running example, and to illustrate the adequacy of *top-level* rewriting for describing the dynamics of systems specified in RL.

Example 3. Suppose that a file can be accessed in reading and in writing, but that it can not be read and written at the same time and that no more than one writer can access it at any time. A specification of this system in RL (borrowed

from [5]), called *READERS-WRITERS*, includes the declaration of a sort *State* of a kind [*State*] for the states of the system, together with a constructor $\langle ., . \rangle$ that takes two natural numbers and returns a *State*.

The evolution of the system is then specified by the following rewrite rules:

$$R_{READERS\text{-}WRITERS} = \begin{cases} \langle 0, 0 \rangle \to \langle 0, s(0) \rangle, \\ \langle R, s(W) \rangle \to \langle R, W \rangle, \\ \langle R, W \rangle \to \langle s(R), W \rangle \text{ if } W = 0, \\ \langle s(R), W \rangle \to \langle R, W \rangle \end{cases}$$

The first rule specifies that, from an initial state $\langle 0, 0 \rangle$ without any readers or writers, the system can accept one writer; the second rule specifies that the number of writers can decrease; the third rule specifies that the number of readers can increase if there are no writers active on the file; and the fourth rule specifies that the number of readers can decrease as well. We note that the system is infinite-state, as the number of readers can grow beyond any bound by the third rule. We also note that the rewriting in *READERS-WRITERS* necessarily occurs at *top-level*: it transforms terms of kind [*State*] to terms of the same kind, and no subterms of terms of kind [*State*] have kind [*State*] - hence, rewriting cannot occur below the top level. This is often the case in practice, e.g., [13] suggests that top-level rewriting is enough for specifying a broad class of systems in RL.

Invariants of Rewrite Theories

We continue this section by defining the notion of *invariant* for a rewrite theory. Intuitively, a predicate over the states of a dynamic system is an invariant if the predicate holds in all the states of the system that are reachable from a given class of initial states. To formalize this notion in the case of a system which is specified in a RL theory (K, Σ, S, E, R) we have to settle the following issues:

1. how are the *states* and the *dynamics* to be specified?
2. how are the state *predicates* to be formalized?
3. when are the predicates to be considered as *holding* in a state?

We propose that states of a system be represented by ground terms of a certain kind [*State*], and the dynamics of the system be defined by ground top-level rewriting, starting from an initial term t_0 (not necessarily ground) of kind [*State*].

With regards to state predicates, they shall be formalized by Horn sentences $(\forall x : [State])(\forall Y)\varphi$ having a free variable x of the kind [*State*] (and possibly other free variables, denoted here by Y, with $x \notin Y$). Finally, a state predicate φ shall be considered to hold in a state t when the predicate $\varphi(t/x)$, obtained from φ by substitution the variable x with the term t, holds in the initial model of the MEL subtheory of the RL specification of the system: $E \vdash_{ind} (\forall Y)\varphi(t/x)$.

Definition 4. *Let $\mathcal{R} = \langle K, \Sigma, S, E, R \rangle$ be a* RL *theory with a kind $[State] \in K$, $(\forall X)t_0 \in T_{\Sigma,[State]}(X)$ a term, and $(\forall x : [State], \forall Y)\varphi$ a state predicate. Then, φ is an invariant of $\mathcal{R}$ for the initial states t_0, denoted by $\langle \mathcal{R}, t_0 \rangle \vdash \Box\varphi$, if for all ground terms $t \in T_{\Sigma,[State]}$, $\mathcal{R} \vdash\downarrow (\forall X)t_0 \longrightarrow t$ implies $E \vdash_{ind} (\forall Y)\varphi(t/x)$.*

Example 4. Consider the RL specification *READERS-WRITERS*. Assuming a $>$ predicate defined in the MEL subtheory *NAT* and assuming the standard accessors *fst* and *snd* to pairs, we can describe the mutual exclusion between readers and writers as the state predicate *mutex*, defined by the following equations:

$$(\forall\, x : [State])\, mutex(x) = true \ \textit{if}\ \ fst(x) = 0$$
$$(\forall\, x : [State])\, mutex(x) = true \ \textit{if}\ \ snd(x) = 0$$
$$(\forall\, x : [State])\, mutex(x) = false \ \textit{if}\ \ fst(x) > 0 \wedge\ \ snd(x) > 0$$

The invariance of *mutex* on the *READERS-WRITERS* system starting from the (ground) initial term $\langle 0, 0\rangle$, is denoted by $\langle READERS\text{-}WRITERS, \langle 0, 0\rangle\rangle \vdash \Box\, mutex$ and is defined by the fact that, for all terms t of the kind $[State]$,

$READERS\text{-}WRITERS \vdash\!\downarrow \langle 0, 0\rangle \longrightarrow^{*} t$ *implies* $NAT \vdash_{ind} mutex(t/x)$.

Automatic falsification of invariants

Before describing the interactive proof of invariants in the next section we discuss here the automatic falsification of invariants. Proving and falsifying are complementary techniques, and automatically trying to falsify an invariant is useful before starting an interactive proof, which, in case the invariant does not hold, would not succeed anyway. We shall consider invariance statements $\langle \mathcal{R}, t_0\rangle \vdash \Box\varphi$ in which the theory $\mathcal{R}$ is *executable*[4] and such that the initial term t_0 is ground and the state predicate φ has only one free variable, which has kind $[State]$.

We show that, under these constraints, Maude's `search` command [5] provides us with a sound and complete procedure for falsifying invariants. The constraints on the initial term and state predicate can be dropped; then, the falsification procedure becomes incomplete but remains sound.

We only present here the features of the command that are useful for our purposes. Consider a state predicate $(\forall x : [State])\varphi$ with $\varphi \triangleq \varphi_0$ *if* $\varphi_1 \ldots \varphi_n$, such that all φ_i, for $i = 0, \ldots, n (n \in \mathbb{N})$ are atomic equations or memberships. We shall employ `search` commands of the form

$$\texttt{search}\ t_0\ \texttt{=>*}\ x\ \texttt{such that}\ \varphi_1(x) \wedge \ldots \wedge \varphi_n(x) \wedge \neg\varphi_0(x) \qquad (4)$$

where t_0 is a ground term, and x is a variable of the same kind as t_0.

The `search` command (4) performs a breadth-first exploration of the set of terms x that are reachable by rewriting from the initial term t_0[5]. The command *terminates sucessfully* if it returns at least one term x, reachable from t_0, and satisfying all the conditions listed in the `such that` clause of the command (4).

Observation 1 *Assume an invariance statement of the form* $\langle \mathcal{R}, t_0\rangle \vdash \Box\varphi$ *such that: the* RL *theory* $\mathcal{R}$ *is executable; the initial term* t_0 *is ground; and the state predicate* φ *has only one free variable, which has kind* $[State]$. *Then, the* `search` *command (4) terminates sucessfully iff* $\langle \mathcal{R}, t_0\rangle \not\vdash \Box\varphi$.

[4] In particular, executability requires that there are no supplementary variables in right-hand sides and conditions of rules; see [5] for details.

[5] we assume that the rewriting occurs at top level only; if this is not the case, rewriting at top-level can always be forced by *encapsulating* the rewrite rules of $\mathcal{R}$ [13].

The statement holds for the following reasons: executabality ensures that, on the one hand, the `such that` clause is decidable, and, on the other hand, that the `search` command is complete (it finds any reachable state in finite time); and, moreover, when the initial term is ground and the specification is executable (in particular, it has no supplementary variables in the right-hand sides of rules and in the conditions), rewriting at top-level *is* ground by construction.

Example 5. The RL specification *READERS-WRITERS* and the predicate *mutex* from Example 4 satisfy all the conditions in Observation 1. We can try to falsify the invariance $\langle \mathit{READERS\text{-}WRITERS}, \langle 0,0 \rangle \rangle \vdash \Box \mathit{mutex}$ as follows:

$$\texttt{search } \langle 0,0 \rangle \texttt{ =>* } x \texttt{ such that } \neg \mathit{mutex}(x) \tag{5}$$

The command does not terminate because $\langle \mathit{READERS\text{-}WRITERS}, \langle 0,0 \rangle \rangle \vdash \Box \mathit{mutex}$ actually holds, as we shall see in the following section. However, e.g., if the condition $W = 0$ was dropped from the third rule of *READERS-WRITERS*, then the command would terminate sucessufully with $x = \langle s(0), s(0) \rangle$. We can then ask Maude for a counterexample trace, and use the trace for debuging.

The `search` command does require an initial ground term and no supplementary variables in the conditions, but if t_0 and φ have (more) free variables, the user can instantiate them before the search, without compromising soundness.

3 Theorem Proving for Invariance Properties

In this section we present our theorem-proving based approach for verifying invariants of rewrite theories. The section is organised as follows. We first define an automatic translation that takes a RL theory $\mathcal{R}$ and a term t_0 and generates a MEL theory $\mathcal{M}(\mathcal{R}, t_0)$, which enriches $\mathcal{R}$ with a sort called *Reachable* and with memberships defining it. We show that "being reachable in $\mathcal{R}$ from t_0 by ground top-level rewriting" and "having sort *Reachable* in $\mathcal{M}(\mathcal{R}, t_0)$" are equivalent statements. Then, as a corollary to the above result, we prove that, for any state predicate $(\forall x : [\mathit{State}])(\forall X)\varphi$, $\langle \mathcal{R}, t_0 \rangle \vdash \Box\varphi$ is equivalent to the fact that the following implication: $(\forall x : [\mathit{State}])(\forall X)(x : \mathit{Reachable} \Rightarrow \varphi)$ holds in the initial model of the theory $\mathcal{M}(\mathcal{R}, t_0)$. This equivalence gives us the formal basis for proving invariants using inductive theorem provers, such as the ITP tool.

In the following definition, we "encode" reachability in a RL theory $\mathcal{R}$ (starting from a term t_0) in a MEL theory $\mathcal{M}(\mathcal{R}, t_0)$ using a membership axiom for t_0 and a membership axiom $\mu(\rho)$ for each rule ρ in $\mathcal{R}$. The axiom for t_0 says that t_0 is reachable, and the axiom $\mu(\rho)$ "reverses" the rule ρ, encoding the fact that the right-hand side of ρ is reachable whenever its left-hand side is.

Definition 5. *Consider a* RL *theory* $\mathcal{R} = (K, \Sigma, S, E, R)$*, a sort* $\mathit{State} \in S$ *with kind* $[\mathit{State}] \in K$*, and a term* $t_0 \in T_{\Sigma,[\mathit{State}]}(X)$*. We denote by* $\mathcal{M}(\mathcal{R}, t_0)$ *the* MEL *theory* $(K_{\mathcal{M}(\mathcal{R},t_0)}, \Sigma_{\mathcal{M}(\mathcal{R},t_0)}, S_{\mathcal{M}(\mathcal{R},t_0)}, E_{\mathcal{M}(\mathcal{R},t_0)})$ *built as follows:*

- $K_{\mathcal{M}(\mathcal{R},t_0)} = K$
- $\Sigma_{\mathcal{M}(\mathcal{R},t_0)} = \Sigma$
- $S_{\mathcal{M}(\mathcal{R},t_0)} = S \cup S'_{[\mathit{State}]}$ *with* $S'_{[\mathit{State}]} = S_{[\mathit{State}]} \cup \{\mathit{Reachable}\}$ *and* $\mathit{Reachable} \notin S$

- $E_{\mathcal{M}(\mathcal{R},t_0)} = E \cup \{(\forall X) t_0 : \mathit{Reachable}\} \cup \{\mu(\rho) | (\rho) \in R\}$, *with* $\mu((\rho)(\forall X)\ l \rightarrow r\ \mathit{if}\ C))$ *being the membership* $(\forall X)\ r : \mathit{Reachable}\ \mathit{if}\ \ l : \mathit{Reachable} \wedge C$.

Example 6. Consider the readers-writers system given in Example 3. The MEL theory $\mathcal{M}(\mathit{READERS\text{-}WRITERS}, \langle 0,0 \rangle)$ consists of the sort declaration and memberships shown below.

- $K_{\mathcal{M}(\mathit{READERS\text{-}WRITERS}\langle 0,0 \rangle)} = K_{\mathit{READERS\text{-}WRITERS}}$
- $\Sigma_{\mathcal{M}(\mathit{READERS\text{-}WRITERS},\langle 0,0 \rangle)} = \Sigma_{\mathit{READERS\text{-}WRITERS}}$
- $S_{\mathcal{M}(\mathit{READERS\text{-}WRITERS},\langle 0,0 \rangle)} = S_{\mathit{READERS\text{-}WRITERS}} \cup S'_{[\mathit{State}]}$, with $S'_{[\mathit{State}]} = S_{\mathit{READERS\text{-}WRITERS}_{[\mathit{State}]}} \cup \{\mathit{Reachable}\}$
- $E_{\mathcal{M}(\mathit{READERS\text{-}WRITERS},\langle 0,0 \rangle)} = E_{\mathit{READERS\text{-}WRITERS}} \cup E'$, where

$$E' = \begin{cases} \langle 0,0 \rangle : \mathit{Reachable} \\ \langle 0, s\,(0) \rangle : \mathit{Reachable}\ \mathit{if}\ \ \langle 0,0 \rangle : \mathit{Reachable} \\ \langle R, W \rangle : \mathit{Reachable}\ \mathit{if}\ \ \langle R, s(W) \rangle : \mathit{Reachable} \\ \langle s(R), W \rangle : \mathit{Reachable}\ \mathit{if}\ \langle R, W \rangle : \mathit{Reachable} \wedge\ \ W = 0 \\ \langle R, W \rangle : \mathit{Reachable}\ \mathit{if}\ \langle s(R), W \rangle : \mathit{Reachable}\ . \end{cases}$$

Theorem 1. *Consider a* RL *theory* $\mathcal{R} = (K, \Sigma, S, E, R)$ *with a sort* $\mathit{State} \in S$ *with kind* $[\mathit{State}] \in K$, *and a term* $t \in T_{\Sigma,[\mathit{State}]}(X)$. *Then, for all ground terms* $t' \in T_{\Sigma,[\mathit{State}]}$, $\mathcal{R} \vdash\!\downarrow (\forall X) t \longrightarrow t'$ *if and only if* $\mathcal{M}(\mathcal{R}, t) \vdash (\forall X) t' : \mathit{Reachable}$.

The idea of the proof (cf. the Appendix) is that rewriting steps that use a given rule ρ in the ground top-level deduction system of the rewrite theory $\mathcal{R}$, can be emulated by equivalent proof steps in the deduction system of the MEL theory $\mathcal{M}(\mathcal{R}, t_0)$, using the corresponding membership $\mu(\rho)$ from Definition 5. We also use the fact that the *new* sort *Reachable* cannot influence deduction in (Σ, E).

Observation 2 *We emphasise the importance of* ground *rewriting for Theorem 1 to hold. Consider the* RL *theory consisting only of a kind Foo, of a constant a of kind Foo and of the function* $f : \mathit{Foo} \mapsto \mathit{Foo}$. *Let* $t_0 = f(x)$. *Then, by usual (non-ground) rewriting, only* $f(x)$ *is reachable, as there are no equations or rewrite rules. Yet, in the translation to* MEL, $f(a) : \mathit{Reachable}$ *is provable (using* Replacement_2 *in Definition 1, with the membership* $f(x) : \mathit{Reachable}$ *and substitution* $\sigma : x \leftarrow a$*) in contradiction with Theorem 1. By contrast, with ground rewriting,* $f(a)$ *is reachable from* t_0 *(with substitution* $\sigma :\ x \leftarrow a$ *in Definition 3).*

Next, remember that $\vdash_{ind}$ denotes truth in an initial model.

Corollary 1. *Consider a* RL *theory* $\mathcal{R} = (K, \Sigma, S, E, R)$, *a sort* $\mathit{State} \in S$ *with kind* $[\mathit{State}] \in K$ *and a state predicate* $(\forall x : [\mathit{State}], \forall X)\varphi$. *Then,* $\langle \mathcal{R}, t_0 \rangle \vdash \Box\varphi$ *if and only if* $\mathcal{M}(R, t_0) \vdash_{ind} (\forall x : [\mathit{State}])(\forall X)(x : \mathit{Reachable} \Rightarrow \varphi)$.

The proof (cf. the Appendix) uses Theorem 1, together with some standard properties about initial models and the fact that the sort *Reachable* in $\mathcal{M}(\mathcal{R}, t)$ is *new* and therefore does not influence truth in the initial model of (Σ, E).

Hence, proving invariants $\langle \mathcal{R}, t_0 \rangle \vdash \Box\varphi$ is equivalent to proving inductive theorems of the form $\mathcal{M}(R, t_0) \vdash_{ind} (\forall x : [\mathit{State}])(\forall X)(x : \mathit{Reachable} \Rightarrow \varphi)$. The latter are statements about the initial model of a MEL theory, hence, they can be proved by induction - here, induction is performed on the sort *Reachable*.

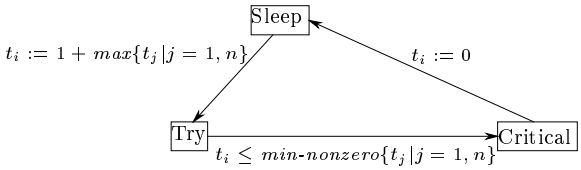

Fig. 2. The i-th process in the n-process Bakery algorithm

Example 7. Consider again the specification of a readers-writers system in Example 3, and the *mutex* predicate defined in Example 5. To prove the invariant

$$\langle READERS\text{-}WRITERS, \langle 0, 0\rangle\rangle \vdash \Box\, mutex$$

one proves that $(\forall\, x)(x : Reachable \Longrightarrow mutex(x) = true)$ holds in the initial model of $\mathcal{M}(READERS\text{-}WRITERS, \langle 0, 0\rangle)$. Using the ITP tool, induction on the sort *Reachable* generates, as expected, five subgoals: one for the membership defining the initial state, and four for the four other memberships defining the sort *Reachable*. Here, all subgoals are automatically proved using the ITP command `auto`, which invokes automatic rewriting and decision procedures. Of course, proofs are not always so automatic. In Section 4 we shall see an example (an n-processes Bakery Algorithm) where auxiliary lemmas (some of which are also invariance properties) are required for proving the main invariant.

4 Example: verifying an n-process Bakery Algorithm

The n-process Bakery algorithm considered here is an example of a parametric system, i.e., it consists of a parametric number n of identical processes. Each of these processes alternates between *Sleep*, *Try*, and *Critical* locations, and has a ticket denoted by t_i for the i-th process (cf. Fig. 2). When going from *Sleep* to *Try*, process number i sets its ticket t_i to the maximum over all tickets plus one.

The condition to enter the *Critical* locations is that the current ticket is less or equal to the *minimum non-zero* of the multiset of tickets. The minimum non-zero of a nonempty multiset S of natural numbers is defined by *min-nonzero(S)* $= 0$ if S only contains copies of 0; and *min-nonzero*(S) $=$ *min*$(S \setminus \{0\})$ otherwise. Here, *min* denotes the smallest element of a (nonempty) multiset, and $S \setminus \{0\}$ denotes the multiset obtained by removing all copies of 0 from S.

Finally, when going back to *Sleep* each process resets its ticket to zero.

The mutual exclusion property means that two different processes can not be in *Critical* state at any time. We need the three following auxiliary lemmas:

- the tickets of processes in locations *Try* or *Critical* are all *non-zero*;
- the tickets of processes in locations *Try* or *Critical* are all *distinct*;
- the tickets of processes in *Critical* equal the minimum-non-zero of all tickets.

The first lemma holds, because entering *Try* sets a ticket to a non-zero value, and entering *Critical* does not modify the ticket. The second lemma holds because, on the transition from *Sleep* to *Try*, the tickets are set to *new* values - the maximum over all tickets plus one, and, again, entering *Critical* does not modify the tickets. The third lemma holds because: (a) the condition to enter *Critical* is that the ticket is less or equal to the minimum non-zero of all tickets, (b) the minimum

nonzero of a multi-set of non-zero values (guaranteed the first lemma) equals the (usual) minimum; and (c) there are no smaller tickets than the minimum. Finally, mutual exclusion holds because, moreover, the minimum of a multi-set in which all elements are distinct (guaranteed by the second lemma), is unique; hence, a unique process, holding that unique value, may be in the critical section.

Specifying the protocol in RL. To specify the n-process Bakery algorithm in RL we first define the sort of *local states*: pairs consisting of a location and a natural number (the ticket). We also define accessors to local states:

$$(\forall L, N)(\langle L, N\rangle : \mathit{LocalState}\ \mathit{if}\ \ L : \mathit{Loc}\ \wedge\ N : \mathit{Nat})$$
$$(\forall L, N)\langle L, N\rangle.\mathit{loc} = L$$
$$(\forall L, N)\langle L, N\rangle.\mathit{tic} = N$$

Next, we define the sort of *states*: they are either the local states or the composition (denoted by the infix operation ";") of a local state and a state:

$$(\forall\, \mathit{LS})(\mathit{LS} : \mathit{State}\ \mathit{if}\ \mathit{LS} : \mathit{LocalState})$$
$$(\forall\, \mathit{LS}, \mathit{ST})((\mathit{LS}; \mathit{ST}) : \mathit{State}\ \mathit{if}\ \ \mathit{LS} : \mathit{LocalState} \wedge \mathit{ST} : \mathit{State})$$

We then equationally define, in the expected way, the *size* of a state as its number of processes, denoted by $\mathit{dim}()$, and the accessors and modifiers for states, namely, $ST[i]$ that returns the i-th local state of the state ST, and $ST\ \mathit{with}[i] := LS$, which returns the state ST whose i-th local state has been replaced by LS. We also define the maximum $\mathit{max}(ST)$ and minimum non-zero $\mathit{min\text{-}nonzero}(ST)$ over the tickets of all the local states in a state ST.

Finally, the three transitions of the system, parameterised by the component that performs them, are specified by rewrite rules:

$$(\forall\, i, ST)ST \to (ST\ \mathit{with}[i] := \langle \mathit{Try}, 1 + \mathit{max}(ST)\rangle) \quad \mathit{if}\ \ i : \mathit{Int}\ \wedge i \geq 0 \wedge\ i < \mathit{dim}(ST)\ \wedge\ ST[i].\mathit{loc} = \mathit{Sleep} \tag{6}$$

$$(\forall\, i, ST)ST \to (ST\ \mathit{with}[i] := \langle \mathit{Critical}, ST[i].\mathit{tic}\rangle) \quad \mathit{if}\ \ i : \mathit{Int}\ \wedge i \geq 0 \wedge\ i < \mathit{dim}(ST)\ \wedge\ ST[i].\mathit{loc} = \mathit{Try} \tag{7}$$

$$(\forall\, i, ST)ST \to (ST\ \mathit{with}[i] := \langle \mathit{Sleep}, 0\rangle) \quad \mathit{if}\ \ i : \mathit{Int}\ \wedge i \geq 0 \wedge\ i < \mathit{dim}(ST)\ \wedge\ ST[i].\mathit{loc} = \mathit{Critical} \tag{8}$$

Notice that the rules have the supplementary variable i in their right-hand sides. Without it, it would be very difficult to express which process makes a move. Such supplementary variables are crucial for modelling parametric systems.

Verifying the protocol in MEL To verify the algorithm following our approach, we first translate its specification in RL into MEL. The reachability relation defined by the rewrite rules (6),(7), and (8) are encoded by the following memberships:

$$(\forall\, i, ST)(ST\ \mathit{with}[i] := \langle \mathit{Try}, 1 + \mathit{max}(ST)\rangle) : \mathit{Reachable}\ \mathit{if}\ \ ST : \mathit{Reachable} \wedge\ i : \mathit{Int} \wedge i \geq 0 \wedge i < \mathit{dim}(ST)\ \wedge\ ST[i].\mathit{loc} = \mathit{Sleep}$$

$$(\forall\, i, ST)(ST\ \mathit{with}[i] := \langle \mathit{Critical}, ST[i].\mathit{tic}\rangle) : \mathit{Reachable}\ \mathit{if}\ \ ST : \mathit{Reachable} \wedge\ i : \mathit{Int} \wedge i \geq 0 \wedge i < \mathit{dim}(ST)\ \wedge\ ST[i].\mathit{loc} = \mathit{Try}$$

$$(\forall\, i, ST)(ST\, with[i] := \langle Sleep, 0\rangle) : Reachable\ \ if\ \ ST : Reachable$$
$$\wedge\ i : Int \wedge i \geq 0 \wedge i < dim(ST)\ \ \wedge\ \ ST[i].loc = Critical$$

Next we specify the (arbitrarily many) initial states of the n-process Bakery algorithm. For this, we declare that the composition of any number of local states with location *Sleep* and ticket 0 is an initial state.

$$Init(0) = \langle Sleep, 0\rangle$$
$$(\forall n) Init(s(n)) = \langle Sleep, 0\rangle; Init(n)$$
$$(\forall n) Init(n) : Reachable\ \ if\ \ n : Nat$$

Finally, we formalize the *mutual exclusion property* over the resulting MEL specification as the following theorem:

$$(\forall\, ST, i, j)\, ST : Reachable \wedge i : Int \wedge i \geq 0 \wedge i < dim(ST) \wedge j : Int$$
$$\wedge\ j \geq 0 \wedge j < dim(ST) \wedge\ \ ST[i] = Critical \wedge ST[j] = Critical$$
$$\Longrightarrow i = j$$

This formula says that in all reachable (global) states, if two local states at (valid) positions i and j are both in the *Critical* location, then necessarily $i = j$.

The ITP proof of mutual exclusion goes like the informal proof sketched at the beginning of this section. The user states and proves three auxiliary lemmas. The first one says that in all locations except *Sleep*, tickets are strictly positive:

$$(\forall\, ST, i)\, ST : Reachable\ \ \wedge\ \ i : Int\ \ \wedge i \geq 0 \wedge\ \ i < dim(ST) \wedge\ \ \neg isSleep(ST[i].loc)$$
$$\Longrightarrow ST[i].tic > 0$$

The second one says that distinct processes outside *Sleep* hold distinct tickets:

$$(\forall\, ST, i, j)\, ST : Reachable \wedge i : Int \wedge i \geq 0 \wedge i < dim(ST) \wedge j : Int \wedge j \geq 0$$
$$\wedge j < dim(ST) \wedge\ \ i < j \wedge \neg isSleep(ST[i].loc) \wedge \neg isSleep(ST[j].loc)$$
$$\Longrightarrow ST[i].tic \neq ST[j].tic$$

The third one says that processes in *Critical* hold the minimum-nonzero ticket:

$$(\forall\, ST, i)\, ST : Reachable\ \ \wedge\ \ i : Int \wedge i \geq 0 \wedge i < dim(ST) \wedge ST[i].loc = Critical$$
$$\Longrightarrow ST[i].tic = min\text{-}nonzero(ST)$$

The ITP proofs of the above theorem and lemmas follow the same pattern. First, induction over the *Reachable* sort generates four subgoals: one for the initial state, and three for the three transitions. The base case (for the initial state) is solved by the ITP command `auto`, which invokes for automatic rewriting and decision procedure for Presburger arithmetic. The inductive steps are solved by combinations of `auto`, case analysis, and auxiliary invariants. We also had to prove several lemmas about *all* states. They formalize properties of the operators used in the specification: *max*, *min-nonzero* (etc) and the relations between them. These properties hold over all states, and not only over *reachable* states and, therefore, they do not amount to invariants of the system. The ITP sources for the example are available at `www.irisa.fr/vertecs/Equipe/Rusu/itp/mutex-n`.

5 Conclusion, Related Work, and Future Work

Automatic state-space exploration, model checking for finite-state systems, abstraction for reducing infinite-state systems to finite ones, and interactive theorem proving for infinite-state systems are well-known verification techniques. For rewriting logic specifications, all but the last are currently supported in the Maude environment. Our present contribution adds the missing part.

The approach is based on an automatic translation of invariance properties of a fragment of rewriting logic (without frozen arguments, with no rewrites in conditions, and with ground top-level rewritings) into inductive properties of membership equational logic. Thanks to this translation, we can use existing theorem provers for membership equational logic, like the ITP tool, to inductively prove invariants of infinite-state systems specified in rewriting logic.

The approach is illustrated on an n-process Bakery algorithm. The results are encouraging, despite the fact that the ITP tool is still a prototype: for example, it does not provide appropriate support for dealing with existential quantifiers. Finally, a distinctive feature of our approach is that it is seamleasly integrated with Maude, and thus it offers access to all of Maude's analysis and verification tools, including state-space exploration, model checking, and equational abstractions.

The proposed approach also has some limitations. Like all approaches based on theorem proving, it requires user input and expertise with the prover. In our experience, the user gains such expertise during the verification process and benefits from feedback that the prover provides when it fails to prove a given subgoal due to insufficient information. In such cases the user typically "sees" from the pending subgoal and from its proof context what the missing information is, and is able to provide it under the form of a lemma that, when proved, allows to settle the pending subgoal and to progress in the proof.

Related Work. The CafeOBJ group from the Japan Advanced Institute of Science and Technology (JAIST) also propose an approach for proving invariants of dynamic systems. Their approach consists in encoding the system under verification as an Observational Transition System (OTS), and invariants as state predicates over the states of OTSs. Then, the OTS and invariant can represented into several formalisms (Figure 3): CafeOBJ and Coq, for theorem proving [14,15]; Maude, for invariant falsification [16]; and SMV, for model checking [17].

Closest to our work is the theorem-proving approach in CafeOBJ. We now briefly describe it and and compare it to ours. The base and inductive steps of an invariance proof are encoded as predicates over (pairs of) states of the CafeOBJ representation of the OTS describing the system. Then, the equational reduction of CafeOBJ is used to show that all those predicates evaluate to *true*. Operations that are typically the role of a theorem prover (case analysis, fresh constant generation,...) have to be done by the user. This may be a source of unsoundess in large proofs. By contrast, in our approach, the ITP theorem prover generates the base and induction step as proof obligations, and manages the constant generation and case-splitting so that unsoundness is not introduced; and proofs are typically more automatic thanks to ITP's decision procedures.

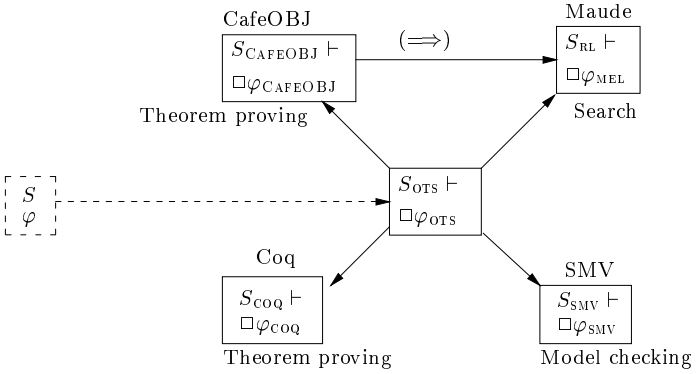

Fig. 3. Observational Transition Systems: representation in CafeOBJ and other tools.

Another difference between our approach and that of the CafeOBJ group lies in the fact that we remain within one single, integrated environment and formalism (that of Maude and of rewriting logic/membership equational logic), which allows us to rely on a common semantics for the various verification activities (Figure 1). By contrast, the CafeOBJ group use several tools, with different formalisms and different underlying semantics (Figure 3). This naturally raises the question of whether, for a given verification problem, one gets consistent answers with the different tools. A partial answer is given in [18], where an implication between invariant verification in CafeOBJ and Maude (Figure 3, top) is proved.

In [12], Bruni and Meseguer present an exhaustive encoding of RL into MEL, including the features that we did not consider here (rewrites in conditions of rules, frozen arguments, and general rewriting, not just ground top-level). Their goal is to define the semantics and proof theory of RL in terms of those of MEL. Our goal is different: to encode just the *invariance of* RL *specifications in* MEL, with an encoding that is simple enough to be used in a theorem-proving approach. A simple encoding is crucial in a theorem-proving approach for users to recognise, through the encoding, the theorems that they are trying to prove.

The verification of parameterised systems is undecidable in general. Hence, automatic approaches are limited to subclasses of such systems. Among the (many) existing automatic approaches we can mention *regular model checking* [19], in which the system's states are encoded as regular languages, and the transition relation is expressed by transducers; and *counting abstractions* [20], in which only the number of processes whose control reside in a given location is recorded, and verification is performed on the resulting abstraction. Theorem proving approaches such as the one proposed here are more general, but they require, in general, user intervention in building proofs.

In the future we are planning to experiment a combination of our theorem-proving approach and of state-space exploration with abstractions. The idea is that, although the main goal may not be provable automatically using abstractions, some of its auxiliary subgoals might indeed be provable automatically.

References

1. N. Martí-Oliet and J. Meseguer. Rewriting logic: roadmap and bibliography. *TCS*, 285(2):121–154, 2002.
2. J. Meseguer and G. Rosu. The rewriting logic semantics project. *TCS*, 373(3):213–237, 2007.
3. J. Meseguer, P. C. Ölveczky, M. O. Stehr, and C. L. Talcott. Maude as a wide-spectrum framework for formal modeling and analysis of active networks. In *DANCE*, pages 494–510. IEEE Comp. Soc., 2002.
4. S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer, and M. K. Sönmez. Pathway logic: Symbolic analysis of biological signaling. In *Pacific Symposium on Biocomputing*, pages 400–412, 2002.
5. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott. *All About Maude, A High-Performance Logical Framework*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.
6. P. Borovanský, C. Kirchner, H. Kirchner, P. E. Moreau, and C. Ringeissen. An overview of ELAN. *Electr. Notes Theor. Comput. Sci.*, 15, 1998.
7. R. Diaconescu and K. Futatsugi. Logical foundations of CafeOBJ. *TCS*, 285(2):289–318, 2002.
8. J. Meseguer. Membership algebra as a logical framework for equational specification. In F. Parisi-Presicce, editor, *WADT*, volume 1376 of *Lecture Notes in Computer Science*, pages 18–61. Springer, 1997.
9. S. Eker, J. Meseguer, and A. Sridharanarayanan. The Maude LTL model checker. *Electr. Notes Theor. Comput. Sci.*, 71, 2002.
10. J. Meseguer, M. Palomino, and N. Martí-Oliet. Equational abstractions. In *CADE*, pages 2–16, 2003.
11. M. Clavel, M. Palomino, and A. Riesco. Introducing the ITP tool: a tutorial. *J. Universal Computer Science*, 12(11):1618–1650, 2006.
12. R. Bruni and J. Meseguer. Semantic foundations for generalized rewrite theories. *TCS*, 360(1-3):386–414, 2006.
13. J. Meseguer and P. Thati. Symbolic reachability analysis using narrowing and its application to the verification of cryptographic protocols. In *WRLA*, volume 117 of *Electronic Notes in Theoretical Computer Science*, 2004.
14. K. Futatsugi. Verifying specifications with proof scores in CafeOBJ. In *ASE*, pages 3–10. IEEE Comp. Soc., 2006.
15. Kazuhiro Ogata and Kokichi Futatsugi. State machines as inductive types. *IEICE Transactions*, 90-A(12):2985–2988, 2007.
16. W. Kong, T. Seino, K. Futatsugi, and K. Ogata. A lightweight integration of theorem proving and model checking for system verification. In *APSEC*, pages 59–66. IEEE Comp. Soc., 2005.
17. K. Ogata, M. Nakano, M. Nakamura, and K. Futatsugi. Chocolat/SMV: a translator from CafeOBJ to SMV. In *PDCAT*, pages 416–420. IEEE Comp. Soc., 2005.
18. M. Nakamura, W. Kong, K. Ogata, and K. Futatsugi. A specification translation from behavioral specifications to rewrite specifications. *IEICE Transactions*, 91-D(5):1492–1503, 2008.
19. Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. *TCS*, 256(1-2):93–112, 2001.
20. A. Pnueli, J. Xu, and L. Zuck. Liveness with (0, 1, infty)-counter abstraction. In Ed Brinksma and Kim Guldstrand Larsen, editors, *CAV*, volume 2404 of *Lecture Notes in Computer Science*, pages 107–122. Springer, 2002.

Appendix (for reviewers only)

Theorem 1 *Consider a* RL *theory* $\mathcal{R} = (K, \Sigma, S, E, R)$ *with a sort* $State \in S$ *with kind* $[State] \in K$*, and a term* $t \in T_{\Sigma,[State]}(X)$*. Then, for all ground terms* $t' \in T_{\Sigma,[State]}$*,* $\mathcal{R} \vdash\downarrow (\forall X) t \longrightarrow t'$ *if and only if* $\mathcal{M}(\mathcal{R}, t) \vdash t' : Reachable$*.*

Proof. ($\Rightarrow$) By induction on the length of the top-level derivation $\mathcal{R} \vdash \sigma(t) \longrightarrow t'$ corresponding to $\mathcal{R} \vdash\downarrow (\forall X) t \longrightarrow t'$. Here, the length is the number of applications of the *Replacement* rule. If the length is 0 then $\sigma(t)$ and t' are equal modulo E (that is, $E \vdash \sigma(t) = t'$). The membership $(\forall X) t : Reachable$ from $\mathcal{M}(\mathcal{R}, t)$ implies $\mathcal{M}(\mathcal{R}, t) \vdash \sigma(t) : Reachable$ which implies $\mathcal{M}(\mathcal{R}, t) \vdash t' : Reachable$.

We assume the implication is true for a derivation of length n, and prove it for length $n+1$. Up to equality modulo the equations E, the derivation of length $n+1$ can always be decomposed as $\mathcal{R} \vdash \sigma(t) \longrightarrow t'' \longrightarrow t'$, for some ground term t'' such that the last step $\mathcal{R} \vdash t'' \longrightarrow t'$ is an application of the *Replacement* rule with a given rule (ρ) $(\forall X) l \to r$ *if* C of $\mathcal{R}$ and ground substitution $\sigma' : X \mapsto T_\Sigma$.

1. then, $t'' \equiv \sigma'(l)$, $t' \equiv \sigma'(r)$, and $E \vdash \sigma'(C) = true$, which implies *a fortiori* $\mathcal{M}(\mathcal{R}, t) \vdash \sigma'(C) = true$.
2. by induction hypothesis, $\mathcal{M}(\mathcal{R}, t) \vdash t'' : Reachable$;
3. hence, using the $Replacement_2$ rule in Definition 1 with the membership $(\mu(\rho))$ $r : Reachable$ *if* $l : Reachable \wedge C$ (cf. Definition 5) and substitution σ' we obtain $\mathcal{M}(\mathcal{R}, t) \vdash t' : Reachable$, which concludes the ($\Rightarrow$) direction.

($\Leftarrow$) By induction on the length n of the derivation $\mathcal{M}(\mathcal{R}, t) \vdash t' : Reachable$. Here, the length is the number of applications of the $Replacement_2$ rule from Definition 1, such that the sort s in the rule is *Reachable*. Note that the length n is at least 1, because, in order to infer a membership to *Reachable*, the rule $Replacement_2$ in which the sort s is *Reachable* must be used at least once.

The base case $n = 1$ implies that the unique application of $Replacement_2$ uses the membership $(\forall X) t : Reachable$ corresponding to the initial term. Indeed, all the other memberships defining *Reachable* are recursive, i.e., they require that some other membership to *Reachable* was proved before. Hence, there exists a ground substitution σ such that $t' = \sigma(t)$, and then $\mathcal{R} \vdash \sigma(t) \longrightarrow t'$ by *Reflexivity*.

We now assume the statement true for $n \geq 1$ and prove it for $n + 1$. Up to equality modulo the equations E, a proof of length $n + 1$ can always be decomposed into a proof $\mathcal{M}(R, t) \vdash t'' : Reachable$ of length n, for some term $t'' \in T_\Sigma$, followed by an application of the $Replacement_2$ rule with a membership of the form $(\mu(\rho))$ $r : Reachable$ *if* $l : Reachable \wedge C$, obtained from a rule (ρ) $(\forall X) l \to r$ *if* C of $\mathcal{R}$. Then, $t' \equiv \sigma'(r)$, $t'' \equiv \sigma'(l)$, and $\mathcal{M}(R, t) \vdash \sigma'(C) = true$ for some ground substitution σ'. Since the *Reachable* sort in $\mathcal{M}(R, t)$ is "new", it does not occur in the conditions C of rewrites rules, and then $E \vdash \sigma'(C) = true$, and, by *Replacement*, $\mathcal{R} \vdash t'' \longrightarrow t'$. By induction hypothesis, from the proof of $\mathcal{M}(R, t) \vdash t'' : Reachable$ we obtain $\mathcal{R} \vdash\downarrow (\forall X) t \longrightarrow t''$ i.e., there exists a ground substitution $\sigma : X \mapsto T_\Sigma$ such that we have a derivation $\mathcal{R} \vdash \sigma(t) \longrightarrow t''$ in which all applications of *Replacement* are used with ground substitutions. Since the derivation $\mathcal{R} \vdash t'' \longrightarrow t'$ from the previous item was also obtained with

a ground substitution (σ') we obtain by *Transitivity* and Definition 3, that $\mathcal{R} \vdash \downarrow (\forall X) t \longrightarrow t'$, which concludes the proof. □

Corollary 1 *Consider a* RL *theory* $\mathcal{R} = (K, \Sigma, S, E, R)$ *with a sort* $State \in S$ *with kind* $[State] \in K$ *and a state predicate* $(\forall x : [State], \forall X)\varphi$. *Then,* $\langle \mathcal{R}, t_0 \rangle \vdash \Box\varphi$ *if and only if* $\mathcal{M}(R, t_0) \vdash_{ind} (\forall x : [State])(\forall X)(x : Reachable \Rightarrow \varphi)$. □

Proof. From

$$\mathcal{M}(R, t_0) \vdash_{ind} (\forall x : [State])(x : Reachable \Rightarrow (\forall X)\varphi)$$

and knowing that a universal property holds in an initial model *if and only if* all its ground instances hold in the initial model, we obtain equivalentl

$$\forall t \in T_{\Sigma,[State]}.\, \mathcal{M}(R, t_0) \vdash_{ind} (t : Reachable \Rightarrow (\forall X)\varphi(t/x))$$

Then, using basic logical reasoning: $A \vdash (B \Rightarrow C)$ if and only of ($A \vdash B$ implies $A \vdash C$), we obtain equivalently

$$\forall t \in T_{\Sigma,[State]}.\, ([\mathcal{M}(R, t_0) \vdash t : Reachable] \text{ implies } [\mathcal{M}(R, t_0) \vdash_{ind} (\forall X)\varphi(t/x)])$$

Now, using Theorem 1, $\mathcal{M}(\mathcal{R}, t_0) \vdash t : Reachable$ is equivalent to $\mathcal{R} \vdash\downarrow (\forall X) t_0 \longrightarrow t$; and, since φ does not refer to the sort *Reachable*, $\mathcal{M}(\mathcal{R}, t_0) \vdash_{ind} (\forall X)\varphi(t/x)$ if and only if $E \vdash_{ind} (\forall X)\varphi(t/x)$. Hence, the last implication yields equivalently

$$\forall t \in T_{\Sigma,[State]}.\, \mathcal{R} \vdash\downarrow (\forall X) t_0 \longrightarrow t \text{ implies } E \vdash_{ind} (\forall X)\varphi(t/x) \tag{9}$$

which, by definition of invariance, (cf. Section 2) is exactly $\langle \mathcal{R}, t_0 \rangle \vdash \Box\varphi$. □