First-order finite satisfiability vs tree automata in safety verification

Alexei Lisitsa[1]

Department of Computer Science, The University of Liverpool
A.Lisitsa@csc.liv.ac.uk

Abstract. In this paper we deal with verification of safety properties of term-rewriting systems. The verification problem is translated to a purely logical problem of finding a finite countermodel for a first-order formula, which further resolved by a generic finite model finding procedure. A finite countermodel produced during successful verification provides with a concise description of the system invariant sufficient to demonstrate a specific safety property.
We show the relative completeness of this approach with respect to the tree automata completion technique. On a set of examples taken from the literature we demonstrate the efficiency of finite model finding approach as well as its explanatory power.

1 Introduction

The development of general automated methods for the verification of infinite-state and parameterized systems poses a major challenge. In general, such problems are undecidable, so one can not hope for the ultimate solution and the development should focus on the restricted classes of systems and properties.

In this paper we deal with a very general method for verification of *safety* properties of infinite-state systems which is based on a simple idea. If an evolution of a computational system is faithfully modelled by a derivation in a classical first-order logic then safety verification (non-reachability of unsafe states) can be reduced to the disproving of a first-order formula. The latter task can be (partially, at least) tackled by generic automated procedures searching for *finite* countermodels.

Such an approach to verification was originated in the research on formal verification of security protocols [29,28,15] and later has been extended to the wide classes of infinite-state and parameterised verification tasks. Completeness of the approach for particular classes of systems (lossy channel systems) and relative completeness with respect to general method of regular model checking has been established in [24] and [25] respectively.

Here we continue investigation of the boundaries of applicability of finite countermodels based method and are looking into verification of safety properties of term-rewriting systems (TRS). Term-rewriting systems provide with a general formalism for specification and verification of infinite-state systems. Several general automated methods for verification of safety properties of term-rewriting systems has been proposed and implemented [12,9,10] with the methods based on tree automata completion [12,9] playing the major role.

We show that verification via finite countermodels (FCM) approach provides with a viable alternative to the methods based on the tree automata completion. We show the relative completeness of FCM with respect to the tree automata completion methods (TAC).

We illustrate it on a simple example taken from [11]. Consider the TRS $\mathcal{R} = \{f(x) \to f(s(s(x)))\}$ and assume that we want to prove that $f(a) \not\to^* f(s(a))$. In [11] a simple finite-state abstraction of the set of reachable terms expressed by the equation $E = \{s(s(x) = x\}$ is explicitly added to the TRS and simple analysis of rewriting modulo E is proposed. In FCM approach, the same problem is translated into disproving of the first-order formula $\varphi_{\mathcal{R}} := (\forall x R(f(x), f(s(s(x)))) \to R(f(a), f(s(a))$. The intended meaning of the binary predicate R here is to encode the reachability relation for the TRS. The finite countermodel of $\varphi_{\mathcal{R}}$, having the size 2 (cardinality of the domain) and essentially representing the above abstraction, i.e. satisfying $s(s(x) = x$, can be found by an automated model finder, e.g. Mace4 in a fraction of a second.

On a series examples taken from the literature we demonstrate practical efficiency of FCM approach using off-the shelf and state of the art implementation of a finite model finding procedure Mace4 (W. McCune); illustrate the high degree of automation achievable as well as the explanatory power of the method.

2 Preliminaries

In this paper we use standard terminology for first-order predicate logic and term-rewriting systems, and the for detailed accounts of these areas the reader is referred to [8] and to [3], respectively. We remind here only the concepts which we are going to use in the paper.

2.1 First-order Logic

The *first-order vocabulary* is defined as a finite set $\Sigma = \mathcal{F} \cup \mathcal{P}$ where $\mathcal{F}$ and $\mathcal{P}$ are the sets of functional and predicate symbols, respectively. Each symbol in Σ has an associated arity, and we have $\mathcal{F} = \cup_{i \geq 0} \mathcal{F} i$ and $\mathcal{P} = \cup_{i \geq 1} \mathcal{P}_i$, where $\mathcal{F}_i$ and $\mathcal{P}_i$ consist of symbols of arity i. The elements of $\mathcal{F}_0$ are also called *constants.*

First-order model over vocabulary Σ, or just a *model* is a pair $\mathcal{M} = \langle D, [\Sigma]_D \rangle$ where D is a set called *domain* of $\mathcal{M}$ and $[\Sigma_D]$ denotes the *interpretations* of all symbols from Σ in D. For a domain D and a function symbol f of arity $n \geq 1$ an interpretation of f in D is a function $[f]_D : D^n \to D$. For a constant c its interpretation $[c]_D$ is an element of D. For a domain D and a predicate symbol P of arity n an interpretation of P in D is a relation of arity n on D, that is $[P]_D \subseteq D^n$. The model $\mathcal{M} = \langle D, [\Sigma]_D \rangle$ is called *finite* if D is a finite set.

We assume that the reader is familiar with the standard definitions of first-order *formula*, first-order *sentence*, satisfaction $\mathcal{M} \models \varphi$ of a formula φ in a model

$\mathcal{M}$, deducibility (derivability) $\Phi \vdash \varphi$ of a formula φ from a set of formulae Φ. We also use the existence of *complete* finite model finding procedures for the first-order predicate logic [4,26], which given a first-order sentence φ eventually produce a finite model for φ if such a model exists.

2.2 Term-rewriting systems and tree automata

To define a term-rewriting system we fix a finite set of functional symbols $\mathcal{F}$, each associated with an arity and a set of variables $\mathcal{X}$. $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and $\mathcal{T}$ denote the set of terms and ground terms, respectively, defined in the standard way using $\mathcal{F}$ and $\mathcal{X}$. The set of variables of a term t is denoted by $Var(t)$. A substitution is a function $\sigma : \mathcal{X} \to \mathcal{T}(\mathcal{F}, \mathcal{X})$, which can be extended homomorphically in a unique way (and keeping the name) to $\sigma : \mathcal{T}(\mathcal{F}, \mathcal{X}) \to \mathcal{T}(\mathcal{F}, \mathcal{X})$. Application of a substitution σ to a term t we denote by $t\sigma$.

A term-rewriting system $\mathcal{R}$ is a set of rewrite rules $l \to r$ where $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $Var(r) \subseteq Var(l)$. The notion of a *subterm* is defined in a standard way. *One-step rewriting relation* $\Rightarrow_{\mathcal{R}} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X}) \times \mathcal{T}(\mathcal{F}, \mathcal{X})$ is defined as follows: $t_1 \Rightarrow_{\mathcal{R}} t_2$ holds iff t_2 is obtained from t_1 by replacement of a subterm $l\sigma$ of t_1 with a subterm $r\sigma$ for some rewriting rule $(l \to r)$ in $\mathcal{R}$ and some substitution σ. The reflexive and transitive closure $\Rightarrow_{\mathcal{R}}$ is denoted by $\Rightarrow^*_{\mathcal{R}}$.

Definitions of tree automata we borrow largely from []. Let Q be a finite set of symbols called *states* which we formally treat as functional symbols of arity 0 (constants). We assume $Q \cap \mathcal{F} = \emptyset$. Elements of $\mathcal{T}(\mathcal{F} \cup Q)$ are called *configurations.*

Definition 1. *(Transitions) A* transition *is a rewrite rule $c \to q$, where c is a configuration, i.e. $c \in \mathcal{T}(\mathcal{F} \cup Q)$, and $q \in Q$. A* normalized *transition is a transition $c \to q$ where $c = f(q_1, \ldots, q_n)$, f is a functional symbol of arity n from $\mathcal{F}$, $q, q_1, \ldots q_n \in Q$. An* ϵ-transition *$c \to q$ is such that $c \in Q$.*

Definition 2. *(Tree automata) A (bottom-up, non-deterministic, finite) tree automaton is a quadruple $\mathcal{A} = \langle F, Q, Q_f, \Delta \rangle$, where $Q_f \subseteq Q$ is a set of final (accepting) states and Δ is a set of normalized transitions and of ϵ-transitions.*

Transitions Δ of $\mathcal{A}$ induce the *rewriting relation* on $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ which is denoted by $\Rightarrow_\Delta$ or $\Rightarrow_{\mathcal{A}}$.

Definition 3. *(Recognized language) The tree language recognized by $\mathcal{A}$ in a state q is $L(\mathcal{A}, q) = \{t \in \mathcal{T}(\mathcal{F}) \mid t \Rightarrow^*_{\mathcal{A}} q\}$. The language recognized by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \cup_{q \in Q_f} \mathcal{L}(\mathcal{A}, q)$.*

Example 1. (Tree automaton and recognized language) Let $\mathcal{F} = \{f, a, b\}$ and $\mathcal{A} = \langle \mathcal{F}, Q, Q_f, \Delta \rangle$, where $Q = \{q_1, q_2\}$, $Q_f = \{q_1\}$, and $\Delta = \{f(q_1) \to q_1, a \to q_1, b \to q_2, q_2 \to q_1\}$. Then $\mathcal{L}(\mathcal{A}, q_1) = \mathcal{T}(f, a, b)$, that is the set of all terms build on $\{f, a, b\}$, and $\mathcal{L}(\mathcal{A}, q_2) = \{b\}$.

Deterministic bottom-up tree automata have the same expressive power as non-deterministic bottom-up tree automata, that is they recognize the same classes of term languages. In what follows we assume that automata are *deterministic*, unless otherwise specified.

3 Safety via finite countermodels

3.1 Basic verification problem

The main verification problem we consider in this paper is as follows.

Problem 1.

Given: Tree automata $\mathcal{A}_I$ and $\mathcal{A}_U$, a term-rewriting system $\mathcal{R}$
Question: Does $\forall t_1 \in \mathcal{L}(\mathcal{A}_I)\ \forall t_2 \in \mathcal{L}(\mathcal{A}_U)\ t_1 \not\Rightarrow^*_{\mathcal{R}} t_2$ hold?

In applications, we assume that the states of a computational system to be verified are represented by terms, the system evolution (computation) is represented by $\mathcal{R}$; tree automata $\mathcal{A}_I$ and $\mathcal{A}_U$ provide with finitary specifications of the (infinite, in general) sets of allowed *initial* states and the sets of *unsafe* states, presented by $\mathcal{L}(\mathcal{A}_I)$ and $\mathcal{L}(\mathcal{A}_U)$, respectively. Under such assumptions, safety of the system is equivalent to the positive answer on the question of the Problem 1.

Modifications of the basic problem will be considered later.

3.2 Translation of the basic verification problem

In this subsection we show how to reduce the basic verification problem to the problem of disproving of a formula from classical first-order predicate logic. First, we define the translation Φ_R of a term-rewriting system $\mathcal{R}$ over $\mathcal{T}(\mathcal{F}, \mathcal{X})$ into a set of first-order formulae in the vocabulary $\mathcal{F} \cup \{R\}$, where R is a new binary predicate symbol. Let $\Phi_{\mathcal{R}} = \Phi^r_{\mathcal{R}} \cup \Phi_{\mathcal{F}}$, where $\Phi^r_{\mathcal{R}} = \{R(l, r) \mid (l \to r) \in \mathcal{R}\}$ and $\Phi_{\mathcal{F}}$ is the set of the following formulae, which are all assumed to be universally closed and where $x_1, \ldots x_i, \ldots x_n, x_i^{'}$ are distinct variables:

1. $R(x, y) \wedge R(y, z) \to R(x, z)$ **transitivity axiom**
2. $R(x_i, x_i^{'}) \to R(f(x_1, \ldots, x_i, \ldots x_n), f(x_1, \ldots, x_i^{'}, \ldots x_n))$ for every n-ary functional symbol f from $\mathcal{F}$ and every position i: $1 \leq i \leq n$
congruence axioms

Under such a translation first-order derivabiliy faithfully models rewriting in $\mathcal{R}$ as the following proposition shows.

Proposition 1. *For ground terms $t_1,\ t_2 \in \mathcal{T}(\mathcal{F})$ if $t_1 \Rightarrow^*_{\mathcal{R}} t_2$ then $\Phi_{\mathcal{R}} \vdash R(t_1, t_2)$.*

Proof. Due to the transitivity of R specified in $\Phi_{\mathcal{R}}$ it is sufficient to show that if $t_1 \Rightarrow_{\mathcal{R}} t_2$ then $\Phi_{\mathcal{R}} \vdash R(t_1, t_2)$. Assume $t_1 \Rightarrow t_2$ then t_2 is obtained from t_1 by the replacement of some subterm $l\sigma$ of t_1 with a subterm $r\sigma$ for some $(l \to r) \in \mathcal{R}$ and some substitution σ. Consider two sequences of subterms $\tau_0 = l\sigma, \tau_1, \ldots, \tau_k = t_1$ and $\rho_0 = r\sigma, \rho_1, \ldots, \rho_k = t_2$ with the property that τ_i is an immediate subterm of τ_{i+1} within t_1 and ρ_i is an immediate subterm of

ρ_{i+1} within t_2, $i = 0, \ldots, k$. Then we show by easy induction on i that $\Phi_{\mathcal{R}} \vdash R(\tau_i, \rho_i)$ for $i = 0, \ldots, k$. Indeed, for $i = 0$ we have $R(\tau_0, \rho_0) \equiv R(l\sigma, r\sigma)$ is a ground instance of $R(l, r) \in \Phi^r_{\mathcal{R}}$ and therefore $\Phi_{\mathcal{R}} \vdash R(\tau_0, \rho_0)$. For the step of induction, assume $\Phi_{\mathcal{R}} \vdash R(\tau_i, \rho_i)$. Notice that by construction of sequences of τ's and ρ's τ_{i+1} and ρ_{i+1} should have the same outermost functional symbol f and coincide everywhere apart of subterms τ_i and ρ_i. Let $\tau_{i+1} = f(\ldots, \tau_i, \ldots)$ and $\rho_{i+1} = f(\ldots, \rho_i, \ldots)$. Then we have $R(\tau_i, \rho_i) \rightarrow R(\tau_{i+1}, \rho_{i+1})$ is a ground instance of one of the formulae in $\Phi^r_{\mathcal{R}}$. So we have $\Phi_{\mathcal{R}} \vdash R(\tau_i, \rho_i) \rightarrow R(\tau_{i+1}, \rho_{i+1})$ and by inductive assumption $\Phi_{\mathcal{R}} \vdash R(\tau_i, \rho_i)$. It follows $\Phi_{\mathcal{R}} \vdash R(\tau_{i+1}, \rho_{i+1})$. The induction step is completed. We have $\Phi_{\mathcal{R}} \vdash R(\tau_k, \rho_k)$, which is $\Phi_{\mathcal{R}} \vdash R(t_1, t_2)$

Now we define a first-order translation of a tree automaton.

Let $\mathcal{A} = \langle \mathcal{F}, Q_I, Q_f, \Delta \rangle$ be a tree automaton. let $\Sigma_{\mathcal{A}}$ be the following first-order vocabulary:

- constants for all elements of Q;
- all functional symbols from $\mathcal{F}$;
- a binary predicate symbol R;

Let $\Phi_{\mathcal{A}}$ to be the set of first-order formulae in vocabulary $\Sigma_{\mathcal{A}}$ defined as $\Phi_{\mathcal{A}} = \Phi_{\Delta} \cup \Phi_{\mathcal{F}}$, where $\Phi_{\Delta} = \{R(c, q) \mid (c \rightarrow q) \in \Delta\}$ and $\Phi_{\mathcal{F}}$ is as defined above.

As the following proposition shows first-order logic derivations from $\Phi_{\mathcal{A}}$ faithfully simulate the work of the automaton $\mathcal{A}$

Proposition 2. *(Adequacy of automata translation)*
If $t \in \mathcal{L}$ then $\Phi_{\mathcal{A}} \vdash \vee_{q \in Q_f} R(t, q)$

Proof. The statement of the proposition follows immediately from Definitions 2 and 3 and Proposition 1.

Now we are ready to define the translation of the basic verification problem. Assume we are given an instance $P = \langle \mathcal{A}_I, \mathcal{R}, \mathcal{A}_U \rangle$ of Problem 1, with a term-rewriting system $\mathcal{R}$ over $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and tree automata $\mathcal{A}_I = \langle \mathcal{F}, Q_I, Q_{f_I}, \Delta_I \rangle$, $\mathcal{A}_U = \langle \mathcal{F}, Q_U, Q_{f_U}, \Delta_U \rangle$. Assume also (without loss of generality) that sets $\mathcal{F}$, Q_I and Q_U are disjoint.

We define translation of P as $\Phi_P = \Phi_{\mathcal{A}_I} \cup \Phi_{\mathcal{A}_U} \cup \Phi_{\mathcal{R}}$. By the above definitions we then also have $\Phi_P = \Phi_{\mathcal{F}} \cup \Phi_{\Delta_I} \cup \Phi_{\Delta_U} \cup \Phi^r_{\mathcal{R}}$. We further define the translation of (negation of) correctness condition from P as a formula $\psi_P = \exists x \exists y \vee_{q_i \in Q_I, q_u \in Q_U} R(x, q_i) \wedge R(x, y) \wedge R(y, q_u)$.

The following proposition and corollary serves as a formal underpinning of the proposed verification method.

Proposition 3. *(Correctness of the translation)*
Let P be an instance of the basic verification problem as detailed above. Then if P has a negative answer then $\Phi_P \vdash \psi_P$

Proof. The statement of the proposition immediately follows from Definitions 2 and 3 and Propositions 1 and 2.

By contraposition we have the following

Corollary 1. *If $\Phi_P \not\vdash \psi_P$ the instance P has a positive answer and the safety property holds.*

3.3 FCM method

By FCM (finite countermodels) verification method we understand the following. Given an instance $P = \langle \mathcal{A}_I, \mathcal{R}, \mathcal{A}_U \rangle$ of the basic verification problem, translate it into a set of first-order formulae Φ_P and a formula ψ_P as described above. Then apply a generic finite model finding procedure to find a countermodel for $\Phi_P \to \psi_P$. If a countermodel found the safety property is established and the instance P has got a positive answer.

3.4 Relative completeness

In this section we show the *relative completeness* of FCM with respect to verification methods based on tree automata completion techniques (TAC). More precisely, we show that if safety of TRS can be demonstrated by TAC, it can be demonstrated by FCM too.

Given an instance P of basic verification problem (Problem 1) verification by TAC approach would proceed as follows. Starting from $\mathcal{A}_\mathcal{I}$ and $\mathcal{R}$ completion procedure yields an automaton $\mathcal{A}^*$ wich describes, in general, an *overapproximation* of the set of terms reachable in $\mathcal{R}$ from $\mathcal{L}(\mathcal{A}_\mathcal{I})$), that is $\mathcal{L}(\mathcal{A}^*) \supseteq \{t \mid \exists t_0 \in \mathcal{L}(\mathcal{A}_\mathcal{I}) \;\; t_0 \to^*_\mathcal{R} t\}$. Further, the check of whether $\mathcal{L}(\mathcal{A}^*) \cap \mathcal{L}(\mathcal{A}_\mathcal{U}) = \emptyset$ is performed and, if it holds, the safety is established.

Exact description by of the set of all reachable terms in a term-rewriting system by a tree automaton not always possible. The main direction in the development of TAC methods is a development of more efficient and more precise approximations methods.

Theorem 1. *Let $P = \langle \mathcal{A}_I, \mathcal{A}_U, \mathcal{R} \rangle$ be a basic verification problem and there exists a tree automaton $\mathcal{A}^* = \langle \mathcal{F}, Q^*, Q^*_f, \Delta^* \rangle$ such that $\mathcal{L}(\mathcal{A}^*) \supseteq \{t \mid \exists t_0 \in \mathcal{L}(\mathcal{A}_\mathcal{I}) \;\; t_0 \to^*_\mathcal{R} t\}$ and $\mathcal{L}(\mathcal{A}^*) \cap \mathcal{L}(\mathcal{A}_\mathcal{U}) = \emptyset$. Then there exists a finite model $\mathcal{M}$ such that $\mathcal{M} \models \Phi_P \wedge \neg\psi_P$ (i.e $\mathcal{M}$ is a countermodel for $\Phi_P \to \psi_P$).*

Proof. Assume the conditions of the theorem hold. Define the domain D of the required model: $D = Q_I^\perp \times Q_*^\perp \times Q_U^\perp$, where $Q_I^\perp = Q_I \cup \{\perp\}$.

Define interpretations of contants $[c] = \langle a_I, a_*, a_U \rangle$, where
$a_x = q$ if $(c, q) \in \Delta_x$, or $a_x = \perp$ otherwise, $x \in \{I, *, U\}$.

For a functional symbols f of arity $n \geq 1$ define its interpretation $[f] : D^n \to D$ as follows

$[f](\langle a_I^1, a_*^1, a_U^1 \rangle, \ldots, \langle a_I^n, a_*^n, a_U^n \rangle) = \langle a_I, a_*, a_U \rangle$, where for all $x \in \{I, *, U\}$, either $(f(a_x^1, a_x^2, \ldots a_x^n) \to a_x) \in \Delta_x$, or $a_x = \perp$, otherwise.

Once we defined the interpretations of all functional symbols (including constants) any ground term t gets its interpretation $[t] \in D$ in a standard way. Then it is an easy consequence of definitions that $[t]$ is a triple of states of automata $\mathcal{A}_I$, $\mathcal{A}_*$, $\mathcal{A}_U$, respectively, into which they get working on the input t. More formally, if $[t] = \langle a_I, a_*, a_U \rangle$, then for all $x \in \{I, *, U\}$ either $t \Rightarrow_x^* a_x \in Q_x$, or there is no such $q \in Q_x$ that $t \Rightarrow_x^* q$, and then $t \Rightarrow_x^* a_x = \bot$.

Define the interpretation $[R] \subseteq D \times D$ of R as follows.

$$[R] = \{\langle [t_1], [t_2] \rangle \mid t_1, t_2 \text{ are ground in } D, t_1 \Rightarrow^* t_2\}$$

where $\Rightarrow$ denotes $\Rightarrow_{\mathcal{R}} \cup \Rightarrow_{\Delta_I} \cup \Rightarrow_{\Delta_U}$

Now we are going to show that in a such defined model $\mathcal{M}$ we have $\Phi_P \wedge \neg\psi_P$ satisfied. Recall $\Phi_P = \Phi_{\mathcal{F}} \cup \Phi_{\Delta_I} \cup \Phi_{\Delta_U} \cup \Phi_{\mathcal{R}}^r$.

We have

- $\mathcal{M} \models \Phi_{\mathcal{F}}$ (by definition of rewriting and definition of $[R]$)
- $\mathcal{M} \models \Phi_{\Delta_I} \cup \Phi_{\Delta_U} \cup \Phi_{\mathcal{R}}^r$ (by definition of $[R]$)

To show $\mathcal{M} \models \neg\psi_P$ assume the opposite i.e $\mathcal{M} \models \psi_P$ that is $\mathcal{M} \models \exists x \exists y \vee_{q_i \in Q_I, q_u \in Q_U} R(x, q_i) \wedge R(x, y) \wedge R(y, q_u)$. That means there are $a, b \in D$ such that $(a, [q_i]) \in [R]$, $(a, b) \in [R]$, $(b, [q_u]) \in [R]$. Consider the ground terms τ_1 and τ_2 such that $[\tau_1] = a$ and $[\tau_2] = b$. We have $\tau_1 \in \mathcal{L}(\mathcal{A}_I)$, $\tau_1 \Rightarrow^* \tau_2$, $\tau_2 \in \mathcal{L}(\mathcal{A}_U)$. It follows that $\tau_2 \in \mathcal{L}(\mathcal{A}^*) \cap \mathcal{L}(\mathcal{A}_{\mathcal{U}})$ which contradicts to the assumption of the theorem on emptiness of $\mathcal{L}(\mathcal{A}^*) \cap \mathcal{L}(\mathcal{A}_{\mathcal{U}})$.

Note 1. The above model construction serves only the purpose of proof and it is not efficient in practical use of the method. Instead we assume that the task of model construction is delegated to a generic finite model building procedure.

3.5 Variations on a theme

Theorem 1 provides with a lower bound for the verifying power of FCM method applied to a basic verification problem. In this section we consider practically important variations of the basic verification problem which allow simplified translations and more efficient verification.

Finitely based sets of terms In many cases of safety verification tasks for TRS the sets of initial and/or unsafe terms are given not by tree automata, but rather described as the sets of ground instances of terms from a given finite set of terms. More precisely, let B be a finite set of terms in a vocabulary $\mathcal{F}$ and $g(B) = \{\tau \mid \exists t \in B \wedge \tau = t\theta; \theta \text{ is ground }\}$. It is easy to see that for the finite B $g(B)$ is a regular set.

Consider the following modification of the basic verification problem.

Problem 2.

Given: Finite sets of terms B_I and B_U, a term-rewriting system $\mathcal{R}$
Question: Does $\forall t_1 \in g(B_I)\ \forall t_2 \in g(B_U)\ t_1 \not\Rightarrow_{\mathcal{R}}^* t_2$ hold?

Let $P = \langle B_I, \mathcal{R}, B_U \rangle$ be an instance of the Problem 2.

The translation $\Phi_{\mathcal{R}}$ of the term rewriting system $\mathcal{R}$ is defined in 3.2.

The translation of (negation of) correctness condition from P is defined as $\psi_P = \exists \bar{x} \vee_{t_1 \in g(B_I), t_2 \in g(B_U)} R(t_1, t_2)$.

Now we have the following analogue of Proposition 3

Proposition 4. *(Correctness of the translation)*

Let P be an instance of the basic verification problem as detailed above. Then if P has a negative answer then $\Phi_{\mathcal{R}} \vdash \psi_P$

Rewriting strategies Another simplification of the translation may come from the restrictions on the rewriting strategies in TRSs. If rewriting can only be applied at the outer level, i.e. redex can be only the whole term, not its proper subterm, then the first-order translation of an TRS can be simplified by using unary reachability predicate $R(-)$ instead of binary $R(-,-)$. The intended meaning of $R(t)$ is "term t is reachable from some of the initial terms (using outermost strategy)". We omit the obvious details of translation (axiomatization of R) and rather refer to an Example 3. Notice, that congruence axioms are not needed in this case and it was observed empirically that their absence makes the countermodel search more efficient.

4 Experiments

In this section we present three examples of application of FCM method for safety verification and compare the results with the results of alternative methods reported in the literature.

4.1 Parity of n^2

Example 2.

The following verification task is taken from [12,10].

Let $P_{n^2} = \langle \mathcal{A}_I, \mathcal{R}, \mathcal{A}_U \rangle$ be an instance of basic verification task. Term rewriting system $\mathcal{R}$ consists of the following rewriting rules

- $plus(0, x) \to x$
- $plus(s(x), y) \to s(plus(x, y))$
- $times(0, x) \to 0$
- $times(s(x), y) \to plus(y, times(x, y))$
- $square(x) \to times(x, x)$
- $even(0) \to true$
- $even(s(0)) \to false$
- $even(s(x)) \to odd(x))$
- $odd(0) \to false$
- $odd(s(0)) \to true$
- $odd(s(x)) \to even(x)$

– $even(square(x)) \to odd(square(s(x)))$
– $odd(square(x)) \to even(square(s(x)))$

The tree automaton $\mathcal{A}_I$ recognizes the set of initial terms. It has the set of states $Q_I = \{s0, s1, s2\}$, the set of the final states $Q_{I_f} = \{s0\}$ and the set of rewriting rules $\Delta_I = \{even(s1) \to s0, square(s2) \to s1, 0 \to s2\}$ It is easy to see that $\mathcal{L}(\mathcal{A}_I) = \{even(square(0))\}$

The tree automaton $\mathcal{A}_U$ recognizes the set of unsafe terms. It has the set of states $Q_U = Q_{U_f} = \{q0\}$ and the set of rewriting rules $\Delta_U = \{false \to q0\}$.

So the question of the verification problem P_{n^2} is whether $false$ is reachable from $even(square(0))$.

First-order translation Φ_P of P_{n^n} consists of the following formulae:

– $R(plus(0, x), x)$
– $R(plus(s(x), y), s(plus(x, y)))$
– $R(times(0, x), 0)$
– $R(times(s(x), y), plus(y, times(x, y)))$
– $R(square(x), times(x, x))$
– $R(even(0), t)$
– $R(even(s(0)), f)$
– $R(even(s(x)), odd(x))$
– $R(odd(0), f)$
– $R(odd(s(0)), t)$
– $R(odd(s(x)), even(x))$
– $R(even(square(x)), odd(square(s(x))))$
– $R(odd(square(x)), even(square(s(x))))$
– $R(x, y) \wedge R(y, z) \to R(x, z)$
– $R(x, y) \to R(even(x), even(y))$
– $R(x, y) \to R(odd(x), odd(y))$
– $R(x, y) \to R(plus(x, z), plus(y, z))$
– $R(x, y) \to R(plus(z, x), plus(z, y))$
– $R(x, y) \to R(times(x, z), times(y, z))$
– $R(x, y) \to R(times(z, x), times(z, y))$
– $R(x, y) \to R(square(x), square(y))$
– $R(0, s2)$
– $R(even(s1), s0)$
– $R(square(s2), s1)$
– $R(f, q0)$

The formula $\psi_P : \exists x \exists y (R(x, s0) \wedge R(x, y) \wedge R(y, q0)$ expresses the negation of correctness condition.

The finite model finder Mace4 has found a finite countermodel for $\Phi_P \to \psi_P$ (i.e a finite model for $\Phi_P \wedge \neg\psi_P$) in 0.03s (see further details in 4.4). The domain D of the model is a two element set $\{0, 1\}$. Interpretations of constants: $[f] = [q0] = [s1] = [s2] = 0$; $[s0] = [t] = 1$. Interpretations of functions: $[even](0) = 1$, $[even](1) = 0$; $[odd](0) = 0$, $[odd](1) = 1$; $[s](0) = 1$, $[s](1) = 0$; $[square](0) = 0$;

$[square](1) = 1$; $[plus]$(x,y) $= (x + y)mod2$; $[times]$(x,y) $= x \times y$. Interpretation of reachability relation: $[R] = \{(0, 0), (1, 1)\}$.

Notice that verification is done here automatically. This can be contrasted with the verification of the same system by a tree completion algorithm implemented in Timbuk system [9], where an user interaction was required to add an approximation equation $s(s(x)) = x$ manually. In [10] an automated verification of the same system was reported using Horn Clause approximation technique. The system was specified as a Horn Clause program and the verification followed by producing a model for the program which contained 53 elements. The above model produced by Mace4 within FCM approach provides with much more concise explanation of why the safety holds: interpretation of any ground term (0 or 1) is an invariant for reachability in TRS, $[even(square(0))] = 1$ and $[f] = 0$.

4.2 Readers-writers system verification

In this subsection we consider the example of a readers-writers system verification taken from [5,11].

Example 3.

In the TRS specifying the system the only *outermost* rewriting is possible, so for the translation we use *monadic* reachability predicate. Furthermore, both the set of initial terms and the set of unsafe terms are finitely based. The vocabulary consists the constant 0, unary functional symbol s (for successor) and binary functional symbol *state*.

The rules are as follows

– $state(0, 0) \to state(0, s(0))$
– $state(x, 0) \to state(s(x), 0)$
– $state(x, s(y)) \to state(x, y)$
– $state(s(x), y) \to state(x, y)$

The set of initial terms is $I = \{state(0, 0)\}$.

The set of unsafe terms U is finitely based with the base
$B = \{state(s(x), s(y)), state(x, s(s(y)))\}$.

The first-order translation Φ consists the conjunction of the following formulae

– $R(state(0, 0))$
– $R(state(0, 0)) \to R(state(0, s(0)))$
– $R(state(x, 0)) \to R(state(s(x), 0))$
– $R(state(x, s(y))) \to R(state(x, y))$
– $R(state(s(x), y)) \to R(state(x, y))$

The formula $\psi \equiv \exists x \exists y R(s(x), s(y)) \lor R(x, s(s(y)))$ expresses the negation of the correctness condition.

The system can be then successfully verified by an FCM method. The search for the countermodel for $\Phi \to \psi$ took 0.01s and the model found is as follows.

The domain D of the model is a three element set $\{0, 1, 2\}$; $[s](0) = 1$, $[s](1) = 2$, $[s](2) = 2$; $[R] = \{(0, 0), (0, 1), (1, 0), (2, 0)\}$.

Notice that no additional information is needed for FCM method to automatically verify the reader-writer system. That may be contrasted with the verification using tree automata completion approach (Timbuk 3.0 system), reported in [11] where an equational abstraction rule $s(s(x)) = s(s(0))$ should be manually added to the TRS for the successful verification.

4.3 Reverse function

In this section we consider a verification problem from [12]. The problem here is to show that list reverse function satisfies the following property: if in a list all symbols 'a' are before all symbols 'b' then after reversing there are no 'a' before 'b'.

Example 4.

Vocabulary $\mathcal{F}$ consists of one 0-ary functional (constant) sumbol 0 and three binary symbols *app*, *cons*, *rev*.

The automaton recognizing is initial terms is defined as $\mathcal{A}_I = \langle \mathcal{F}, Q_I, Q_{f_I}, \Delta_I \rangle$, where $\mathcal{F}$ is as defined above; $Q_I = \{qrev, qlab, qlb, qa, qb\}$; $Q_{f_I} = \{qrev\}$; Δ_I contains

- $rev(qlab) \to qrev$
- $cons(qa, qlab) \to qlab$
- $0 \to qlb$
- $a \to qa$
- $0 \to qlab$
- $cons(qa, qlb) \to qlab$
- $cons(qb, qlb) \to qlb$
- $b \to qb$

The automaton recognizing *unsafe* terms is defined as $\mathcal{A}_U = \langle \mathcal{F}, Q_U, Q_{f_U}, \Delta_U \rangle$, where $\mathcal{F}$ is as above; $Q_U = \{qlab1, qlb1, q1, qa, qb\}$, $Q_{f_U} = \{qlab1\}$; Δ_U contains

- $cons(qa, qlab1) \to qlab1$
- $cons(qa, qlb1) \to qlab1$
- $cons(qa, q1) \to q1$
- $a \to qa$
- $0 \to q1$
- $cons(qb, qlab1) \to qlab1$
- $cons(qb, q1) \to qlb1$
- $cons(qb, q1) \to q1$
- $b \to qb$

The term-rewriting system $\mathcal{R}$ consists of the following rules

- $app(0, x) \to x$

- $app(cons(x, y), z) \rightarrow cons(x, app(y, z))$
- $rev(0) \rightarrow 0$
- $rev(cons(x, y)) \rightarrow app(rev(y), cons(x, 0))$

First-order translation Φ_P consists of the following formulae.

- $R(rev(qlab), qrev)$
- $R(cons(qa, qlab), qlab)$
- $R(0, qlb)$
- $R(a, qa)$
- $R(0, qlab)$
- $R(cons(qa, qlb), qlab)$
- $R(cons(qb, qlb), qlb)$
- $R(b, qb)$
- $R(cons(qa, qlab1), qlab1)$
- $R(cons(qa, qlb1), qlab1)$
- $R(cons(qa, q1), q1)$
- $R(0, q1)$
- $R(cons(qb, qlab1), qlab1)$
- $R(cons(qb, q1), qlb1)$
- $R(cons(qb, q1), q1)$
- $R(b, qb)$
- $R(app(0, x), x)$
- $R(app(cons(x, y), z), cons(x, app(y, z)))$
- $R(rev(0), 0)$
- $R(rev(cons(x, y)), app(rev(y), cons(x, 0)))$
- $(R(x, y) \wedge R(y, z)) \rightarrow R(x, z)$
- $R(x, x)$
- $R(x, y) \rightarrow R(rev(x), rev(y))$
- $R(x, y) \rightarrow R(cons(z, x), cons(z, y))$
- $R(x, y) \rightarrow R(cons(x, z), cons(y, z))$
- $R(x, y) \rightarrow R(app(z, x), app(z, y))$
- $R(x, y) \rightarrow R(app(x, z), app(y, z))$

The formula $\psi_P : \exists x \exists y((R(rev(x), qrev) \wedge R(y, qlab1)) \wedge R(rev(x), y)$ expresses the negation of the correctness condition.

For this standard encoding Mace4 has failed to find a countermodel for $\Phi_P \rightarrow \psi_P$ within 40000s. However after removing the congruence axiom $R(x, y) \rightarrow R(rev(x), rev(y))$ Mace4 has found the model of size 3 (cardinality of the domain) in 0.06s. (see further details in [21]. The absence of such a congruence axiom means that no rewriting of proper subterms of $rev(\ldots)$ is allowed. One can either easily argue that in TRS given above no such rewriting possible anyway, or, remaining in a pure automated verification scenario, just accept verification modulo restrictions on the rewriting strategy. This can be contrasted with the verification of the same system in [12] using tree automata completion technique, which required interactive approximation.

4.4 Experimental results

In the experiments we used the finite model finder Mace4[26] within the package Prover9-Mace4, Version 0.5, December 2007. It is not the latest available version, but it provides with convenient GUI for both the theorem prover and the finite model finder. The system configuration used in the experiments: Microsoft Windows XP Professional, Version 2002, Intel(R) Core(TM)2 Duo CPU, T7100 @ 1.8Ghz 1.79Ghz, 1.00 GB of RAM. The time measurements are done by Mace4 itself, upon completion of the model search it communicates the CPU time used. The table below lists the parameterised/infinite state protocols together with the references and shows the time it took Mace4 to find a countermodel and verify a safety property. The time shown is an average of 10 attempts. ∞ means not return in 40000s.

Problem	Reference	Time
Parity of n^2	[12]	0.3s
Readers-Writers	[11]	0.01s
Reverse	[12]	∞
Reverse (no congruence for rev) II	Example 4	0.06s

5 Related work

5.1 Discussion and Related work

The verification of safety properties for term-rewriting systems using tree automata completion techniques has been addressed in [12,9,11]. The paper [10] presents a method based on encoding both term-rewriting system and tree automata into Horn logic and application of the static analysis techniques to compute a tree automaton accepting an approximation of the set of reachable terms. The main conceptual difference between these approaches and FCM presented in this paper, is that in [12,9,11,10] the safety verification is performed in two stages: first, a tree automaton approximating all reachable terms is obtained and it depends only on TRS but *not on the safety property*, and, second, an intersection of the language of this automaton with the language of unsafe states is computed. FCM method we presented here operates in one stage and computing *regular approximations* (in terms of finite countermodels) is done for concrete safety properties. It has its disadvantage that the results of the verification of a TRS can not be re-used for the verification of different safety properties for the same TRS. On the other hand this disadvantage is compensated by a higher degree of automation and higher explanatory power of FCM methods as our experimental results suggest. Another advantage of FCM is its flexibility. Rewriting modulo theory can be easily incorporated into a general FCM framework and previous work on FCM illustrates this point. In [24] dealing with the verification of lossy automata and cache coherence protocols, rewriting modulo first-order specifications of automata and modulo simple arithmetics, was used. In [25] the translation of regular model checking into FCM framework, the associativity of a monoid multiplication was explicitly specified.

As mentioned Section 1 the approach to verification using the modeling of protocol executions by first-order derivations and together with countermodel finding for disproving was introduced within the research on the formal analysis of cryptographic protocols. It can be traced back to the early papers by Weidenbach [29] and by Selinger [28]. In [29] a decidable fragment of Horn clause logic has been identified for which resolution-based decision procedure has been proposed (disproving by the procedure amounts to the termination of saturation without producing a proof). It was also shown that the fragment is expressive enough to encode cryptographic protocols and the approach has been illustrated by the automated verification of some protocols using the SPASS theorem prover. In [28], apparently for the first time, explicit building of finite countermodels has been proposed as a tool to establish correctness of cryptographic protocols. It has been illustrated by an example, where a countermodel was produced manually, and the automation of the process has not been discussed. The later work by Goubault-Larrecq [15] has shown how a countermodel produced during the verification of cryptographic protocols can be converted into a formal induction proof. Also, in [15] different approaches to model building have been discussed and it was argued that an implicit model building procedure using alternating tree automata is more efficient in the situations when no small countermodels exist. Very recently, in the paper [19] by J. Jurgens and T. Weber, an extension of Horn clause logic was proposed and the soundness of a countermodel finding procedure for this fragment has been shown, again in the context of cryptographic protocol verification.

The work we reported in this paper differs from all the approaches mentioned previously in two important aspects. Firstly, to the best of our knowledge, none of the previous work addressed verification via countermodel finding applied outside of the area of cryptographic protocols (that includes the most recent work [17] we are aware of). Secondly, the (relative) completeness for the classes of verification tasks has not been addressed in previous work.

References

1. Abdulla, P.A., Jonsson, B., (1996) Verifying programs with unreliable channels. *Information and Computation*, 127(2):91-101, 1996.
2. Abdulla, P.A., Jonsson,B., Nilsson, M., & Saksena, M., (2004) A Survey of Regular Model Checking, In Proc. of CONCUR'04, volume 3170 of LNCS, pp 35–58, 2004.
3. Baader F., Nipkow, T., Term Rewriting and All That, Cambridge University Press, 1998.
4. Caferra, R., Leitsch, A., & Peltier, M., (2004) *Automated Model Building*, Applied Logic Series, 31, Kluwer, 2004.
5. Clavel, M., Duran, F., Eker, S., Lincoln, P., Marti-Oliet, N., Meseguer, J., Talcott., C.L., 2007. All About Maude, A High-Performance Logical Framework. Vol. 4350 of Lecture Notes in Computer Science. Springer.

6. Comon, H., (1994), Inductionless induction. In R. David, ed. *2nd Int. Conf. in Logic for Computer Science: Automated Deduction. Lecture Notes*, Chambery, Uni de Savoie, 1994.
7. Delzanno, G., (2003) Constraint-based Verification of Parametrized Cache Coherence Protocols. *Formal Methods in System Design*, 23(3):257–301, 2003.
8. Enderton, H.B., A mathematical introduction to logic. Academic Press, 1972.
9. Feuillade, G., Genet, T., Tong, V.V.T.: Reachability Analysis over Term Rewriting Systems. J. Autom. Reasoning 33(3-4), 341–383 (2004).
10. Gallagher, John P., & Rosendahl, M.,(2008) Approximating Term Rewriting Systems: A Horn Clause Specification and Its Implementation. I.Cervesato, H. Veith, and A. Voronkov (Eds.): LPAR 2008, LNCS 5330, pp. 682–696, 2008.
11. Genet, T., Rusu, V., Equational Approximations for Tree Automata Completion, Journal of Symbolic Computation Volume 45, Issue 5, May 2010, Pages 574-597
12. Genet, T., Tong, V.V.T.: Reachability Analysis of Term Rewriting Systems with *timbuk*. In: Nieuwenhuis, R., Voronkov, A. (eds.) LPAR 2001, LNCS, vol. 2250, pp. 695–706. Springer, Heidelberg, 2001.
13. Ghilardi, S., & Ranise, S. (2010) MCMT: A Model Checker Modulo Theories. Lecture Notes in Computer Science, 2010, Volume 6173, 22–29.
14. Ghilardi, S., Nikolini, E., Ranise, S., & Zucchelli, D., (2008) Towards SMT Model-Checking of Array-based Systems. In IJCAR, LNCS, 2008
15. Goubault-Larrecq, J., (2008), Towards producing formally checkable security proofs, automatically. In: Computer Security Foundations (CSF), pp. 224–238 (2008)
16. Goubault-Larrecq, J., (2009) "Logic Wins!". In ASIAN'09, LNCS 5913, pages 1-16. Springer, 2009.
17. Guttman, J., (2009) Security Theorems via Model Theory, Proceedings 16th International Workshop on Expressiveness in Concurrency, EXPRESS, EPTCS, vol. 8 (2009)
18. Habermehl, P., & Vojnar, T., (2005), Regular Model Checking Using Inference of Regular Languages, Electronic Notes in Theoretical Computer Science (ENTCS), Volume 138, Issue 3 (December 2005), pp 21–36, 2005
19. Jurjens, J., & Weber, T., (2009), Finite Models in FOL-Based Crypto-Protocol Verification, P. Degano and L. Vigan'o (Eds.): ARSPA-WITS 2009, LNCS 5511, pp. 155–172, 2009.
20. Kapur, D., & Musser, D.R., (1987), Proof by consistency. *Artificial Intelligence*, 31:125–157, 1987.
21. Lisitsa, A., (2009a), Verfication via countermodel finding `http://www.csc.liv.ac.uk/~alexei/countermodel/`
22. Lisitsa, A., (2009b), Reachability as deducibility, finite countermodels and verification. In preProceedings of AVOCS 2009, Technical Report of Computer Science, Swansea University, CSR-2-2009, pp 241-243.
23. Lisitsa, A., (2010a), Finite countermodels as invariants. A case study in verification of parameterized mutual exclusion protocol. In Proceedings of WING 2010, 1pp
24. Lisitsa, A., (2010b), Reachability as deducibility, finite countermodels and verification. In Proceedings of ATVA 2010, LNCS 6252, 233–244
25. Lisitsa, A., (2010c), Finite model finding for parameterized verification, CoRR abs/1011.0447: (2010)
26. McCune, W., Prover9 and Mace4 `http://www.cs.unm.edu/~mccune/mace4/`
27. Nilsson, M., (2005) Regular Model Checking. Acta Universitatis Upsaliensis. Uppsala Dissertations from the Faculty of Science and Technology 60. 149 pp. Uppsala. ISBN 91-554-6137-9, 2005.

28. Selinger, P., (2001), Models for an adversary-centric protocol logic. Electr. Notes Theor. Comput. Sci. 55(1) (2001)
29. Weidenbach, C., (1999), Towards an Automatic Analysis of Security Protocols in First-Order Logic, in H. Ganzinger (Ed.): CADE-16, LNAI 1632, pp. 314–328, 1999.