Finite Models vs Tree Automata in Safety Verification

Alexei Lisitsa[1]

1 **Department of Computer Science**
University of Liverpool, UK
`a.lisitsa@liverpool.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 is 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.

1998 ACM Subject Classification I.2.3 Deduction and Theorem Proving, F.4.2 Grammars and Other Rewriting Systems, D.2.4 Software/Program Verification

Keywords and phrases term-rewriting systems, safety verification, first-order logic, finite model finding

Digital Object Identifier 10.4230/LIPIcs.RTA.2012.225

Category Regular Research Paper

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 [28, 27, 13] 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 [20] and [21] 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, the high degree of automation and the explanatory power of FCM approach using off-the shelf and state of the art implementation of a finite model finding procedure Mace4 (W. McCune).

The preliminary version of this paper has appeared as the report [22].

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 [2], 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, first-order *theory*, *equational* theory.

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 $\mathcal{T}(\mathcal{F})$ 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 $\Rightarrow^*_{\mathcal{R}}$.

Given a set E of $\mathcal{F}$-equations, $\mathcal{T}_E$ denotes a set of equivalent classes of ground $\mathcal{F}$-terms modulo the equations E. E-equivalence class of a term t is denoted by $[t]_E$. Given a term rewriting system $\mathcal{R}$ and an equational theory given by a set of equations E we denote by $\Rightarrow_{\mathcal{R},E}$ one step rewriting relation modulo E, that is $t \Rightarrow_{\mathcal{R},E} t'$ iff $\exists \tau \in [t]_E \; \exists \tau' \in [t']_E \; \tau \Rightarrow_{\mathcal{R}} \tau'$.

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 2.1.** (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 2.2.** (Tree automata) A (bottom-up, non-deterministic, finite) tree automaton is a quadruple $\mathcal{A} = \langle \mathcal{F}, Q, Q_f, \Delta \rangle$, where $Q_f \subseteq Q$ is a set of final (accepting) states and Δ is a set of normalized transitions and of ϵ-transitions. A tree automaton is deterministic if there are no two transitions in Δ with the same left-hand side.

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 2.3.** (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 2.4.** (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.

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 $x_1, \dots x_i, \dots x_n, x_i^{'}$ are distinct variables:

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

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

▶ **Proposition 3.1.** *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 $\tau_0 = l\sigma, \tau_1, \dots, \tau_k = t_1$ and $\rho_0 = r\sigma, \rho_1, \dots, \rho_k = t_2$ 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, $i = 0,\dots,k$. Then we show by easy induction on i that $\Phi_{\mathcal{R}} \vdash R(\tau_i,\rho_i)$ for $i = 0,\dots,k$. 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 $\tau_{i+1} = f(\dots,\tau_i,\dots)$ and $\rho_{i+1} = f(\dots,\rho_i,\dots)$. Then we have $R(\tau_i,\rho_i) \to R(\tau_{i+1},\rho_{i+1})$ is a ground instance of one of the formulae in $\Phi^r_{\mathcal{R}}$. So we have $\Phi_{\mathcal{R}} \vdash R(\tau_i,\rho_i) \to R(\tau_{i+1},\rho_{i+1})$ 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 $\Phi_{\Delta} = \{R(c, q) \mid (c \to q) \in \Delta\}$ 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 3.2.** *(Adequacy of automata translation)*
If $t \in \mathcal{L}_{\mathcal{A}}$ then $\Phi_{\mathcal{A}} \vdash \vee_{q \in Q_f} R(t, q)$

Proof. The statement of the proposition follows immediately from Definitions 2.2 and 2.3 and Proposition 3.1. ◀

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 3.3.** *(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 2.2 and 2.3 and Propositions 3.1 and 3.2. ◀

By contraposition we have the following

▶ **Corollary 3.4.** *If $\Phi_P \nvdash \psi_P$ 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 is 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 of the set of all reachable terms in a term-rewriting system by a tree automaton is not always possible. The main direction in the development of TAC methods is a development of more efficient and more precise approximations methods.

▶ **Theorem 3.5.** *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: $D = Q^\perp_I \times Q^\perp_* \times Q^\perp_U$, where $Q^\perp_I = Q_I \cup \{\perp\}$.

Define interpretations of constants $[\![c]\!] = \langle a_I, a_*, a_U\rangle$, where $a_x = q$ if $(c, q) \in \Delta_x$, or $a_x = \perp$ otherwise, $x \in \{I, *, U\}$. For a functional symbol f of arity $n \geq 1$ define its interpretation $[\![f]\!] : D^n \to D$ as $[\![f]\!](\langle a^1_I, a^1_*, a^1_U\rangle, \dots, \langle a^n_I, a^n_*, a^n_U\rangle) = \langle a_I, a_*, a_U\rangle$, where for all $x \in \{I, *, U\}$, either $(f(a^1_x, a^2_x, \dots a^n_x) \to a_x) \in \Delta_x$, 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 $t \Rightarrow^*_x a_x = \perp$.

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 $\Phi_P = \Phi_\mathcal{F} \cup \Phi_{\Delta_I} \cup \Phi_{\Delta_U} \cup \Phi^r_\mathcal{R}$.

We have
- $\mathcal{M} \models \Phi_\mathcal{F}$ (by definition of rewriting and definition of $[R]$)
- $\mathcal{M} \models \Phi_{\Delta_I} \cup \Phi_{\Delta_U} \cup \Phi^r_\mathcal{R}$ (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 Finite models as invariants

Assume that for a basic verification problem $P = \langle \mathcal{A}_I, \mathcal{A}_U, \mathcal{R}\rangle$, a model $\mathcal{M}$ such that $\mathcal{M} \models \Phi_P \wedge \neg\psi_P$ is found. Then not only the safety is established, the model itself provides with a finite description of an invariant of a TRS sufficient to prove the safety. Define the following subsets of a domain D of $\mathcal{M}$.
- $I = \{[\![t]\!] \mid t \in \mathcal{L}(\mathcal{A}_\mathcal{I})\}$
- $U = \{[\![t]\!] \mid t \in \mathcal{L}(\mathcal{A}_\mathcal{U})\}$
- $Reach = \{[\![t]\!] \mid \exists t_0 \in \mathcal{L}(\mathcal{A}_\mathcal{I})\ \ t_0 \to^*_\mathcal{R} t\}$
- $I \bullet [\![R]\!] = \{y \mid \exists x \in I \wedge (x, y) \in [\![R]\!]\}$ (where $[\![R]\!]$ is an interpretation of R in $\mathcal{M}$)

Then it is easy consequence of definitions of Φ_P and ψ_P that $Reach \subseteq I \bullet [\![R]\!]$ and $I \bullet [\![R]\!] \cap U = \emptyset$. Thus $I \bullet [\![R]\!]$ is a finite invariant set which subsumes the interpretations of all reachable terms and at the same time is disjoint with the set of interpretations of all unsafe terms. For a finite model $\mathcal{M}$ both conditions can be verified by a straightforward induction, providing thereby the translation of FCM verification into an inductive proof of a safety property.

3.6 Variations on a theme

Theorem 3.5 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.

3.6.1 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 $\forall t_1 \in g(B_I)\ \forall t_2 \in g(B_U)\ t_1 \not\to^*_{\mathcal{R}} t_2$ 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 $\psi_P = \exists \bar{x} \bigvee_{t_1 \in g(B_I), t_2 \in g(B_U)} R(t_1, t_2)$.

Now we have the following analogue of Proposition 3.3

▶ **Proposition 3.6.** *(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 $\Phi_{\mathcal{R}} \vdash \psi_P$

3.6.2 Outermost rewriting

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 rewriting)". The translation of a term rewriting system $\mathcal{R}$ becomes especially simple in this case: $\Phi_{\mathcal{R}} = \{R(t) \to R(t') \mid (t \to t') \in \mathcal{R}\}$.

We omit the obvious further details of translation (e.g. on specification of the sets of initial and unsafe terms) as well as the statement on its correctness and rather refer the reader to the Example 6.2.

4 Rewriting modulo theories

Proposed FCM method is very flexible and can be naturally extended to the rewriting modulo equational theories. We consider here only the case of basic verification problem. The extensions of other variants (such as Problem 2, or as in subsection 3.6.2) can be dealt with in a similar way.

▶ **Problem 3.**
Given: *Tree automata $\mathcal{A}_I$ and $\mathcal{A}_U$, a term-rewriting system $\mathcal{R}$, and a set of equations E*
Question: *Does $\forall t_1 \in \mathcal{L}(\mathcal{A}_I)\ \forall t_2 \in \mathcal{L}(\mathcal{A}_U)\ [t_1] \not\Rightarrow^*_{\mathcal{R},E} [t_2]$ hold?*

Using notation introduced in subsection 3.2 we now have

▶ **Proposition 4.1.** *For ground terms $t_1, t_2 \in \mathcal{T}(\mathcal{F})$ if $[t_1] \Rightarrow^*_{\mathcal{R},E} [t_2]$ then $\Phi_{\mathcal{R}} \cup E \vdash R(t_1, t_2)$.*

Proof. (Hint) The proof follows the proof of Proposition 3.1 using the following straightforward statement. If for some ground terms t_1, t_2 $\Phi_{\mathcal{R}} \cup E \vdash R(t_1, t_2)$ then for any $t_1' \in [t_1]$ and $t_2' \in [t_2]$ $\Phi_{\mathcal{R}} \cup E \vdash R(t_1', t_2')$. ◀

Based on that we have an equational analogue of Proposition 3.3 supporting applicability of FCM for proving the safety of TRS modulo equational theories:

▶ **Proposition 4.2.** *For the instance P of the above verification problem if P has a negative answer then $\Phi_{\mathcal{A}_I} \cup \Phi_{\mathcal{A}_U} \cup \Phi_{\mathcal{R}} \cup E \vdash \psi_P$*

Example 6.3 illustrates the use of FCM for equational rewriting. Since the outermost rewriting strategy is assumed in this Example the simplified translation with unary reachability predicate R is used.

5 Approximation by symmetric closure

In theory the application of FCM method is simple - apply finite model builder to the first-order encoding of the problem and wait. If there is a finite countermodel it will be eventually found if a complete model builder is used. In practice, however, it may take a long time to find a countermodel. To accelerate the search one may use the approximation of reachability relation $\Rightarrow_{\mathcal{R}}$ ($\Rightarrow_{\mathcal{R},E}$) by its symmetric closure $\Leftrightarrow_{\mathcal{R}}$ ($\Leftrightarrow_{\mathcal{R},E}$). It is obvious that if the safety holds for the original problem, it may be lost after the symmetric closure is applied. If however the safety is shown for the problem with the symmetric closure it holds for the original problem too. To take an account of the symmetric closure it is sufficient to add to the first-order translation of the basic verification problem just one formula $R(x, y) \to R(y, x)$. Notice that after such a modification an interpretation of R in any model is an equivalence relation which is a congruence. It follows that if a countermodel searched in FCM method exists then there exists a countermodel where R is interpreted by the *equality* relation. This observation allows to simplify the translation of such modified problem further. Assuming that a sound model finder procedure for the first-order logic with equality is used, in the first-order translation all occurrences of R can be replaced with equality and reflexivity, transitivity and congruence axioms can be dropped. Example 6.4 demonstrates the use of such a technique.

6 Experiments

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

In the experiments we used the finite model finder Mace4[24] within the package Prover9-Mace4, Version 0.5, December 2007. 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.

6.1 Parity of n^2

▶ **Example 6.1.**

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 $even(square(0))$.

Denote by Φ_P the first-order translation of P_{n^n} as defined in Section 3.2. 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. 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) $= (x + y) mod 2$; $[\![times]\!]$(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 $[\![f]\!] = 0$.

6.2 Readers-writers system verification

In this subsection we consider the example of a readers-writers system verification taken from [6, 11].

▶ **Example 6.2.**

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 $\psi \equiv \exists x \exists y R(s(x), s(y)) \vee R(x, s(s(y)))$ 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.

6.3 Glass replacement puzzle

▶ **Example 6.3.**

The following verification example is taken from [17], where it has been formalized using higher-order rewriting by simply-typed term rewriting systems (STRS) and solved by using higher-order Knuth-Bendix completion procedure. We use first-order term rewriting formalization of the same problem .

Assume a sequence of sake, whisky and beer glasses. The sequence can be modified by application of the following glass-replacement rules.

- A sake glass may be inserted to the left of a beer glass. A sake glass having a beer glass to its right may be removed.
- A sake glass and whisky glass may be added to the left and right, respectively, of an arbitrary contiguou s subsequence of glasses. The reverse operation may also be performed.

The verification problem is to show that starting from *sake-whisky-whisky-whisky* sequence and applying the replacement rules above one can not reach the sequence *sake-whisky-whisky-beer*.

We formalize the problem by the term-rewriting system $\mathcal{R}$ consisting the following rewriting rule

- $x * (sake * (beer * y)) \to x * (beer * y)$
- $x * (beer * y) \to x * (sake * (beer * y))$
- $v * (sake * (x * (whisky * y))) \to v * (x * y)$
- $v * (x * y) \to (v * (sake * (x * (whisky * y)))$

where $sake$, $beer$, $whisky$ are constants, and $*$ is a binary associative term constructor (we use it in infix notation). Associativity of $*$ means we consider rewriting modulo equational theory $E = \{(x * y) * z = x * (y * z)\}$.

First-order translation Φ consists the following formulae:

- $(x * y) * z = x * (y * z)$
- $P(x * (sake * (beer * y))) \to P(x * (beer * y))$
- $P(x * (beer * y)) \to P(x * (sake * (beer * y)))$
- $P((v * (sake * (x * (whisky * y)))) \to P(v * (x * y))$
- $P(v * (x * y)) \to P(v * (sake * (x * (whisky * y))))$

Now, to resolve the puzzle it is sufficient to show $\Phi \wedge P(sake * (whisky * (whisky * whisky))) \not\vdash P(sake * (whisky * (whisky * beer)))$. A countermodel of size 2 is found by finite model finder MACE4 in 0.01sec.

6.4 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 6.4.**

Vocabulary $\mathcal{F}$ consists of one 0-ary functional (constant) symbol 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

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

First-order translation Φ_P consists of the following formulae.

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

- $R(x,y) \to R(cons(z,x), cons(z,y))$
- $R(x,y) \to R(cons(x,z), cons(y,z))$
- $R(x,y) \to R(app(z,x), app(z,y))$
- $R(x,y) \to R(app(x,z), app(y,z))$

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 $\Phi_P \to \psi_P$ within 250000s. However after removing the congruence axiom $R(x,y) \to R(rev(x), rev(y))$ Mace4 has found the model of size 3 (cardinality of the domain) in 0.06s. (see further details in [18]. 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. Alternatively one can apply an approximation by symmetric closure, in which case Mace4 finds the model of size 3 in 0.1s.

Notice that the verification of the same system in [12] has been done using tree automata completion technique, which required interactive approximation.

7 Related work and further directions

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 [28, 27, 13, 16, 14].

In [27], 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 [13] has shown how a countermodel produced during the verification of cryptographic protocols can be converted into a formal induction proof.

Later FCM approach has been applied to the various problems of infinite-state and parameterized system [18, 19, 20, 21, 23]. In [20] it was shown that FCM provides a decision procedure for the verification of safety properties of lossy channel systems and applications to the verification of parameterized cache coherence protocols are presented . In [21, 23] the relative completeness of FCM with respect to regular model checking and regular tree model checking, respectively, has been established. Despite its simplicity, in many cases FCM has turned out to be very efficient and comparable with or even outperforming (e.g. in [23]) the best reported alternative methods.

From the viewpoint of this paper the verification problems considered in the above papers can be seen as safety problems for the specific classes of (conditional) TRS modulo equational theories.

The verification of safety properties for general 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. 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 or invariants (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 method as our experimental results suggest. Furthermore, as the Example 6.1 has demonstrated the invariants for specific safety properties might be much simpler than the approximations of all reachable terms. That raises hope that the scalability, which is a potential issue for FCM applications, can be dealt with. Tree automata completion has shown that it scales well on very large TRS and large tree automata (with thousands of rewrite rules and automata states) [3]. The applicability of FCM for the verification of specific safety properties of the large systems requires further investigation.

Another feature of FCM approach which makes it different from tree automata completion techniques is that it does not depend on linearity of TRS.

Perhaps, conceptually closest work to that reported in this paper has appeared very recently in [4]. In the context of safety verification the computation of the overapproximation of the set of reachable terms is reduced in [4] to the problem of instantiation of a symbolic tree automaton, which in turn is reduced to the satisfiability testing of a boolean combination of equalities and inequalities between variables. The latter task is then delegated to the second-order satisfiability checker Mona. Similar to FCM and unlike to the methods reported in [12, 9, 11, 10] approximations of reachable terms are computed there with respect to a particular safety property. Theoretical and practical comparisons of FCM and the method of [4] is a very interesting topic for future work.

Finally, we would like to notice that the translations we have defined in this paper can be seen as the axiomatizations of rewriting theories in a sense of rewriting logic [25] within the standard first-order logic, where binary predicate R represents the entailment $\vdash$ for rewriting theory. Rewriting logic formalizing conditional and modulo theory rewriting may serve as an unifying formalism for representing all applications of FCM. Development of such an approach is another direction for future work.

References

1 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.

2 Baader, F., Nipkow, T., (1998) *Term Rewriting and All That*, Cambridge University Press, 1998

3 Boichut, Y., Genet, T., Jensen T.P., Le Roux, L., (2007) Rewriting Approximations for Fast Prototyping of Static Analyzers, in Proc. of RTA'2007, 48–62, 2007.

4 Boichut, Y., Dao, T.-B.-H., Murat, V., (2011) Characterizing Conclusive Approximations by Logical Formulae, in Proc. of Reachability Problems 2011, pp72-84, 2011

5 Caferra, R., Leitsch, A., & Peltier, M., (2004) *Automated Model Building*, Applied Logic Series, 31, Kluwer, 2004.

6 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.

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., (1972) *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 Goubault-Larrecq, J., (2008), Towards producing formally checkable security proofs, automatically. In: Computer Security Foundations (CSF), pp. 224–238 (2008)

14 Guttman, J., (2009) Security Theorems via Model Theory, Proceedings 16th International Workshop on Expressiveness in Concurrency, EXPRESS, EPTCS, vol. 8 (2009)

15 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

16 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.

17 Kusakari, K., & Chiba, Y., (2007), Higher-Order Knuth-Bendix Procedure and Its Applications, IEICE Transactions on Information and Systems, Vol.E90-D, No.4, pp.707-715, Apr 2007.

18 Lisitsa, A., (2009a), Verfication via countermodel finding `http://www.csc.liv.ac.uk/~alexei/countermodel/`

19 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.

20 Lisitsa, A., (2010b), Reachability as deducibility, finite countermodels and verification. In Proceedings of ATVA 2010, LNCS 6252, 233–244

21 Lisitsa, A., (2010c), Finite model finding for parameterized verification, CoRR abs/1011.0447: (2010)

22 Lisitsa, A., (2011a), First-order finite satisfiability vs tree automata in safety verification Corr abs/1107.0349:(2011)

23 Lisitsa, A.,(2011), Finite countermodels for safety verification of parameterized tree systems, CoRR abs/1107.5142:(2011)

24 McCune, W., Prover9 and Mace4 `http://www.cs.unm.edu/~mccune/mace4/`

25 Meseguer, J., (1992), Conditional Rewriting Logic as a unified model of concurrency, *Theoretical Computer Science*, 96:73–155, 1992.

26 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.

27 Selinger, P., (2001), Models for an adversary-centric protocol logic. Electr. Notes Theor. Comput. Sci. 55(1) (2001)

28 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.