Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 223 (2008) 61–70

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Handling Left-Quadratic Rules When Completing Tree Automata

Y. Boichut[a], R. Courbis[b], P.-C. Héam[b] and O. Kouchnarenko[b]

[a] *INRIA/PAREO*
615 rue du Jardin Botanique BP-101 F-54602 Villers-Lès Nancy Cedex
boichut@loria.fr

[b] *INRIA/CASSIS and LIFC / University of Franche-Comté 16 route de Gray F-25030 Besançon Cedex*
lastname.firstname@lifc.univ-fcomte.fr

Abstract

This paper addresses the following general problem of tree regular model-checking: decide whether $\mathcal{R}^*(\mathcal{L}) \cap \mathcal{L}_p = \emptyset$ where $\mathcal{R}^*$ is the reflexive and transitive closure of a successor relation induced by a term rewriting system $\mathcal{R}$, and $\mathcal{L}$ and $\mathcal{L}_p$ are both regular tree languages. We develop an automatic approximation-based technique to handle this – undecidable in general – problem in the case when term rewriting system rules are left-quadratic. The most common practical case is handled this way.

Keywords: Rewriting techniques, tree automata, left-linearity, security.

1 Introduction

Automatic verification of software systems is one of the most challenging research problems in computer aided verification. In this context, regular model-checking has been proposed as a general framework for analysing and verifying infinite state systems. In this framework, systems are modelled using regular representations: the systems configurations are modelled by finite words or trees (of unbounded size) and the dynamic behaviour of systems is modelled either by a transducer or a (term) rewriting system. Afterwards, a system reachability-based analysis is reduced to the regular languages closure computation under (term) rewriting systems: given a regular language $\mathcal{L}$, a relation $\mathcal{R}$ induced by a (term) rewriting system and a regular set $\mathcal{L}_P$ of *bad configurations*, the problem is to decide whether $\mathcal{R}^*(\mathcal{L}) \cap \mathcal{L}_p = \emptyset$ where $\mathcal{R}^*$ is the reflexive and transitive closure of $\mathcal{R}$. Since $\mathcal{R}^*(\mathcal{L})$ is in general neither regular nor decidable, several approaches handle restricted cases of this problem.

In this paper we address this problem for tree regular languages by automatically computing over- and under-approximations of $\mathcal{R}^*(\mathcal{L})$. Computing an over-approximation K_{over} of $\mathcal{R}^*(\mathcal{L})$ may be useful for the verification if $K_{\text{over}} \cap \mathcal{L}_p = \emptyset$, proving that $\mathcal{R}^*(\mathcal{L}) \cap \mathcal{L}_p = \emptyset$. Dually, under-approximation may be suitable to prove that $\mathcal{R}^*(\mathcal{L}) \cap \mathcal{L}_p \neq \emptyset$. This approach is relevant if the computed approximations are not too coarse. Another important point is that in general, there are some restrictions on the rewriting systems in order to ensure the soundness of the above approach. This paper follows and adapts an expert-human guided approximation technique introduced in [18] for left-linear term-rewriting systems. More precisely, the paper 1) extends this approach to term rewriting systems with left-quadratic rules, and 2) illustrates its advantages on examples.

Related Work Given a term rewriting system $\mathcal{R}$ and two ground terms s and t, deciding whether $s \to_{\mathcal{R}}^* t$ is a central question in automatic proof theory. This problem is shown decidable for term rewriting systems which are terminating but it is undecidable in general. Several syntactic classes of term rewriting systems have been pointed out to have a decidable accessibility problem, for instance by providing an algorithm to compute $\mathcal{R}^*(\mathcal{L})$ when $\mathcal{L}$ is a regular tree language [15,13,20,23,25,26]. In [18], authors focus on a general completion based human-guided technique. This technique has been successfully used (not automatically) to prove the security of cryptographic protocols [19] and recently Java Bytecode programs [5]. This framework was extended in [24] to languages accepted by AC-tree automata. Several work on tree regular model checking are proposed in [9,1,8,21].

Layout of the paper The paper is organised as follows. Section 2 introduces notations and the basic completion approach. Next, Section 3 presents the main theoretical contributions of the paper, while Section 4 describes a family of examples and gives related security issues. Finally, Section 5 concludes.

2 Preliminaries

2.1 Terms and TRSs

Comprehensive surveys can be found in [16,2] for term rewriting systems, and in [12,20] for tree automata and tree language theory.

Let $\mathcal{F}$ be a finite set of symbols, associated with an arity function $ar : \mathcal{F} \to \mathbb{N}$, and let $\mathcal{X}$ be a countable set of variables. $\mathcal{T}(\mathcal{F}, \mathcal{X})$ denotes the set of terms, and $\mathcal{T}(\mathcal{F})$ denotes the set of ground terms (terms without variables). The set of variables of a term t is denoted by $\mathcal{V}ar(t)$. A substitution is a function σ from $\mathcal{X}$ into $\mathcal{T}(\mathcal{F}, \mathcal{X})$, which can be extended uniquely to an endomorphism of $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A position p for a term t is a word over $\mathbb{N}$. The empty sequence ϵ denotes the top-most position. The set $\mathcal{P}os(t)$ of positions of a term t is inductively defined by: $\mathcal{P}os(t) = \{\epsilon\}$ if $t \in \mathcal{X}$ and $\mathcal{P}os(f(t_1, \ldots, t_n)) = \{\epsilon\} \cup \{i.p \mid 1 \leq i \leq n \text{ and } p \in \mathcal{P}os(t_i)\}$. If $p \in \mathcal{P}os(t)$, then $t|_p$ denotes the subterm of t at position p and $t[s]_p$ denotes the term obtained by replacement of the subterm $t|_p$ at position p by the term s. We also denote by $t(p)$ the symbol occurring in t at position p. Given a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, we denote $\mathcal{P}os_A(t) \subseteq \mathcal{P}os(t)$ the set of positions of t such that $\mathcal{P}os_A(t) = \{p \in \mathcal{P}os(t) \mid t(p) \in A\}$. Thus $\mathcal{P}os_{\mathcal{F}}(t)$ is the set of functional positions of t.

A term rewriting system (TRS) $\mathcal{R}$ is a set of *rewrite rules* $l \to r$, where $l, r \in$

$\mathcal{T}(\mathcal{F},\mathcal{X})$ and $l \notin \mathcal{X}$. A rewrite rule $l \to r$ is *left-linear* (resp. right-linear) if each variable of l (resp. r) occurs only once within l (resp. r). A TRS $\mathcal{R}$ is left-linear (resp. right-linear) if every rewrite rule $l \to r$ of $\mathcal{R}$ is left-linear (resp. right-linear). A TRS $\mathcal{R}$ is linear if it is right and left-linear. The TRS $\mathcal{R}$ induces a rewriting relation $\to_{\mathcal{R}}$ on terms whose reflexive transitive closure is written $\to^{\star}_{\mathcal{R}}$. The set of $\mathcal{R}$-descendants of a set of ground terms E is $\mathcal{R}^*(E) = \{t \in \mathcal{T}(\mathcal{F}) \mid \exists s \in E \text{ s.t. } s \to^{\star}_{\mathcal{R}} t\}$.

2.2 Tree Automata Completion

Note that $\mathcal{R}^*(E)$ is possibly infinite: $\mathcal{R}$ may not terminate and/or E may be infinite. The set $\mathcal{R}^*(E)$ is generally not computable [20]. However, it is possible to over-approximate it [18] using tree automata, i.e. a finite representation of infinite (regular) sets of terms. We next define tree automata.

Let $\mathcal{Q}$ be a finite set of symbols, of arity 0, called *states* such that $\mathcal{Q} \cap \mathcal{F} = \emptyset$. $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is called the set of *configurations* A *transition* is a rewrite rule $c \to q$, where $c \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is a configuration and $q \in \mathcal{Q}$. A *normalised transition* is a transition $c \to q$ where $c = f(q_1, \ldots, q_n)$, $f \in \mathcal{F}$, $ar(f) = n$, and $q_1, \ldots, q_n \in \mathcal{Q}$. A bottom-up non-deterministic finite *tree automaton* (tree automaton for short) is a quadruple $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, $\mathcal{Q}_f \subseteq \mathcal{Q}$ and Δ is a finite set of normalised transitions. The *rewriting relation* on $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ induced by the transition set Δ of $\mathcal{A}$ is denoted $\to_{\Delta}$. When Δ is clear from the context, $\to_{\Delta}$ is also written $\to_{\mathcal{A}}$. The *tree language recognised* by $\mathcal{A}$ in a state q is $\mathcal{L}(\mathcal{A}, q) = \{t \in \mathcal{T}(\mathcal{F}) \mid t \to^{\star}_{\mathcal{A}} q\}$. The language recognised by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \bigcup_{q \in \mathcal{Q}_f} \mathcal{L}(\mathcal{A}, q)$. A tree language is regular if and only if it is recognised by a tree automaton.

Let us now recall how tree automata and TRSs can be used for term reachability analysis. Given a tree automaton $\mathcal{A}$ and a TRS $\mathcal{R}$, the tree automata completion algorithm proposed in [18] computes a tree automaton $\mathcal{A}^k_{\mathcal{R}}$ such that $\mathcal{L}(\mathcal{A}^k_{\mathcal{R}}) = \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$ when it is possible (for the classes of TRSs where an exact computation is possible, see [18]), and such that $\mathcal{L}(\mathcal{A}^k_{\mathcal{R}}) \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$ otherwise.

The tree automata completion works as follows. From $\mathcal{A} = \mathcal{A}^0_{\mathcal{R}}$ completion builds a sequence $\mathcal{A}^0_{\mathcal{R}}, \mathcal{A}^1_{\mathcal{R}} \ldots \mathcal{A}^k_{\mathcal{R}}$ of automata such that if $s \in \mathcal{L}(\mathcal{A}^i_{\mathcal{R}})$ and $s \to_{\mathcal{R}} t$ then $t \in \mathcal{L}(\mathcal{A}^{i+1}_{\mathcal{R}})$. If there is a fix-point automaton $\mathcal{A}^k_{\mathcal{R}}$ such that $\mathcal{R}^*(\mathcal{L}(\mathcal{A}^k_{\mathcal{R}})) = \mathcal{L}(\mathcal{A}^k_{\mathcal{R}})$, then $\mathcal{L}(\mathcal{A}^k_{\mathcal{R}}) = \mathcal{R}^*(\mathcal{L}(\mathcal{A}^0_{\mathcal{R}}))$ (or $\mathcal{L}(\mathcal{A}^k_{\mathcal{R}}) \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$ if $\mathcal{R}$ is in no class of [18]). To build $\mathcal{A}^{i+1}_{\mathcal{R}}$ from $\mathcal{A}^i_{\mathcal{R}}$, a *completion step* is achieved. It consists of finding *critical pairs* between $\to_{\mathcal{R}}$ and $\to_{\mathcal{A}^i_{\mathcal{R}}}$. To define the notion of critical pair, the substitution definition is extended to terms in $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$. For a substitution $\sigma : \mathcal{X} \mapsto \mathcal{Q}$ and a rule $l \to r \in \mathcal{R}$ such that $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$, if there exists $q \in \mathcal{Q}$ satisfying $l\sigma \to^*_{\mathcal{A}^i_{\mathcal{R}}} q$ then $l\sigma \to^*_{\mathcal{A}^i_{\mathcal{R}}} q$ and $l\sigma \to_{\mathcal{R}} r\sigma$ is a critical pair. Note that since $\mathcal{R}$ and $\mathcal{A}^i_{\mathcal{R}}$ is finite, there is only a finite number of critical pairs. Thus, for every critical pair detected between $\mathcal{R}$ and $\mathcal{A}^i_{\mathcal{R}}$ such that $r\sigma \not\to^*_{\mathcal{A}^i_{\mathcal{R}}} q$, the tree automaton $\mathcal{A}^{i+1}_{\mathcal{R}}$ is constructed by adding a new transition $r\sigma \to q$ to $\mathcal{A}^i_{\mathcal{R}}$. Consequently, $\mathcal{A}^{i+1}_{\mathcal{R}}$ recognises $r\sigma$ in q, i.e. $r\sigma \to_{\mathcal{A}^{i+1}_{\mathcal{R}}} q$.

However, the transition $r\sigma \to q$ is not necessarily a normalised transition of the form $f(q_1, \ldots, q_n) \to q'$. Then, we use abstraction functions whose goal is to define a set of normalised transitions $Norm$ such that $r\sigma \rightarrow^*_{Norm} q$. Thus, instead of adding the transition $r\sigma \rightarrow q$ which is not normalised, the set of transitions $Norm$ is added to Δ, i.e., the transition set of the current automaton $\mathcal{A}^i_{\mathcal{R}}$. For example, to normalize a transition of the form $f(g(a), h(q')) \rightarrow q$, we need to find some states q_1, q_2, q_3 and replace the previous transition by a set of normalized transitions: $\{a \rightarrow q_1, g(q_1) \rightarrow q_2, h(q') \rightarrow q_3, f(q_2, q_3) \rightarrow q\}$.

Assume that q_1, q_2, q_3 are new states, then adding the transition itself or its normalised form does not make any difference. Now, assume that $q_1 = q_2$, the normalised form becomes $\{a \rightarrow q_1, g(q_1) \rightarrow q_1, h(q') \rightarrow q_3, f(q_1, q_3) \rightarrow q\}$. This set of normalised transitions represents the regular set of non normalised transitions of the form $f(g^\star(a), h(q')) \rightarrow q$; which contains the transition initially we wanted to add amongst many others. Hence, this is an over-approximation. We could have made an even more drastic approximation by identifying q_1, q_2, q_3 with q, for instance.

We give below a very general definition of abstraction functions which allot to each functional position of $r\sigma$ a state of $\mathcal{Q}$. The role of an abstraction function remains to define equivalence classes of terms where one class corresponds to one state of $\mathcal{Q}$. An *abstraction function* γ is a function $\gamma : ((\mathcal{R} \times (\mathcal{X} \rightarrow \mathcal{Q}) \times \mathcal{Q}) \mapsto \mathbb{N}^*) \mapsto \mathcal{Q}$ such that $\gamma(l \rightarrow r, \sigma, q)(\epsilon) = q$. Thus, given an abstraction function γ, the normalisation of a transition $r\sigma \rightarrow q$ is defined as follows. Let γ be an abstraction function, Δ be a transition set, $l \rightarrow r \in R$ with $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$ and $\sigma : \mathcal{X} \rightarrow \mathcal{Q}$ such that $l\sigma \rightarrow^*_\Delta q$. The γ*−normalisation* of the transition $r\sigma \rightarrow q$, written $Norm_\gamma(l \rightarrow r, \sigma, q)$, is defined by:

$$\begin{aligned} Norm_\gamma(l \rightarrow r, \sigma, q) = \{ & r(p)(\beta_{p.1}, \ldots, \beta_{p.n}) \rightarrow \beta \mid \\ & p \in \mathcal{P}os_{\mathcal{F}}(r), \\ & \beta = \begin{cases} q \ if \ p = \epsilon \\ \gamma(l \rightarrow r, \sigma, q)(p) \ otherwise. \end{cases} \\ & \beta_{p.i} = \begin{cases} \sigma(r(p.i)) \ if \ r(p.i) \in \mathcal{X} \\ \gamma(l \rightarrow r, \sigma, q)(p.i) \ otherwise. \end{cases} \end{aligned}$$

Example 2.1 Let $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ be the tree automaton such that $\mathcal{F} = \{a, b, c, d, e, f, \omega\}$ with $ar(s) = 1$ with $s \in \{a, b, c, d, e, f\}$ and $ar(\omega) = 0$, $\mathcal{Q} = \{q_b, q_f, q_\omega\}$, $\mathcal{Q}_f = \{q_f\}$ and $\Delta = \{\omega \rightarrow q_\omega, b(q_\omega) \rightarrow q_b, a(q_b) \rightarrow q_f\}$. Thus, $\mathcal{L}(\mathcal{A}) = \{a(b(\omega))\}$. Given the TRS $\mathcal{R} = \{a(x) \rightarrow c(d(x)),\ b(x) \rightarrow e(f(x))\}$, two critical pairs are computed: $a(q_b) \rightarrow^*_{\mathcal{A}} q_f$, $a(q_b) \rightarrow_{\mathcal{R}} c(d(q_b))$ and $b(q_\omega) \rightarrow^*_{\mathcal{A}} b(q_\omega) \rightarrow_{\mathcal{R}} e(f(q_\omega))$. Let γ be the abstraction function such that $\gamma(a(x) \rightarrow c(d(x)), \{x \rightarrow q_b\}, q_f)(\epsilon) = q_f$, $\gamma(a(x) \rightarrow c(d(x)), \{x \rightarrow q_b\}, q_f)(1) = q_f$, $\gamma(b(x) \rightarrow e(f(x)), \{x \rightarrow q_\omega\}, q_b)(\epsilon) = q_b$ and $\gamma(b(x) \rightarrow e(f(x)), \{x \rightarrow q_\omega\}, q_b)(1) = q_b$. So, $Norm_\gamma(a(x) \rightarrow c(d(x)), \{x \rightarrow q_b\}, q_f) = \{d(q_b) \rightarrow q_f, c(q_f) \rightarrow q_f\}$ and $Norm_\gamma(b(x) \rightarrow e(f(x)), \{x \rightarrow q_\omega\}, q_b) = \{f(q_\omega) \rightarrow q_b, e(q_b) \rightarrow q_b\}$.

Now we formally define what a completion step is. Let $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ be a tree automaton, γ an abstraction function and $\mathcal{R}$ a left-linear TRS. We define a tree automaton $C_\gamma^{\mathcal{R}}(\mathcal{A}) = \langle \mathcal{F}, \mathcal{Q}', \mathcal{Q}'_f, \Delta' \rangle$ with:

- $\Delta' = \Delta \ \cup \bigcup_{l \to r \in R,\ \sigma:\mathcal{X} \mapsto \mathcal{Q},\ l\sigma \to_{\mathcal{A}}^* q, r\sigma \not\to_{\mathcal{A}}^* q} \ Norm_\gamma(l \to r, \sigma, q)$,
- $Q' = \{q \mid c \to q \in \Delta'\}$ and
- $Q'_f = \mathcal{Q}_f$.

Example 2.2 Given $\mathcal{A}$, $\mathcal{R}$ and γ of Example 2.1, performing one completion step on $\mathcal{A}$ gives the automaton $C_\gamma^{\mathcal{R}}(\mathcal{A})$ such that $C_\gamma^{\mathcal{R}}(\mathcal{A}) = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta' \rangle$ where $\Delta' = \Delta \cup Norm_\gamma(a(x) \to c(d(x)), \{x \to q_b\}, q_f) \cup Norm_\gamma(b(x) \to e(f(x)), \{x \to q_\omega\}, q_b) = \{\omega \to q_\omega, b(q_\omega) \to q_b, a(q_b) \to q_f, d(q_b) \to q_f, c(q_f) \to q_f, f(q_\omega) \to q_b, e(q_b) \to q_b\}$. Notice that $C_\gamma^{\mathcal{R}}(\mathcal{A})$ is $\mathcal{R}$-close, and in fact an over-approximation of $\mathcal{R}^*(\mathcal{L}(\mathcal{A}))$ is computed. Indeed, the tree automaton $C_\gamma^{\mathcal{R}}(\mathcal{A})$ recognises the term $a(e(e(f(\omega))))$ when

$$\mathcal{R}^*(\mathcal{L}(\mathcal{A})) = \{a(b(\omega)), a(e(f(\omega))), c(d(b(\omega))), c(d(e(f(\omega))))\}.$$

Proposition 2.3 ([18, Theorem 1]) *Let $\mathcal{A}$ be a tree automaton and $\mathcal{R}$ be a TRS such that $\mathcal{A}$ is deterministic or $\mathcal{R}$ is left-linear, and for every $l \to r \in \mathcal{R}$, $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. For any abstraction function γ, one has:*

$$\mathcal{L}(\mathcal{A}) \cup \mathcal{R}(\mathcal{L}(\mathcal{A})) \subseteq C_\gamma^{\mathcal{R}}(\mathcal{A}).$$

In addition, an abstraction functions can be defined in such a way only terms, actually reachable, will be computed. This class of abstraction functions is called $(\mathcal{A}, \mathcal{R})-$exact abstraction functions in [3].

Let $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ be a tree automaton and $\mathcal{R}$ be a TRS. Let $Im(\gamma) = \{q \mid \forall l \to r \in \mathcal{R},\ \forall p \in \mathcal{P}os_{\mathcal{F}}(r)\ s.t.\ \gamma(l \to r, \sigma, q)(p) = q\}$. An abstraction function γ is $(\mathcal{A}, \mathcal{R})-$*exact* if γ is injective and $Im(\gamma) \cap \mathcal{Q} = \emptyset$.

By adapting the proof of Theorem 2 in [18] to the new class of abstractions, we show that with such abstraction functions, only reachable terms are computed.

Theorem 2.4 ([18, Theorem 2]) *Let $\mathcal{A}$ be a tree automaton and $\mathcal{R}$ be a TRS such that $\mathcal{A}$ is deterministic or $\mathcal{R}$ is right-linear. Let α be an $(\mathcal{A}, \mathcal{R})-$exact abstraction function. One has: $C_\alpha^{\mathcal{R}}(\mathcal{A}) \subseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$.*

We now give the general result in [18] saying that, if there exists a fix-point automaton, then its language contains all the terms actually reachable by rewriting, at least. $(\mathcal{A}, \mathcal{R})-$exact abstraction functions.

Theorem 2.5 ([18, Theorem 1]) *Let $\mathcal{A}$, $\mathcal{R}$ and γ be respectively a tree automaton, a TRS. For any abstraction function, if there exists $N \in \mathbb{N}$ and $N \geq 0$ such that $(C_\gamma^{\mathcal{R}})^{(N)}(\mathcal{A}) = (C_\gamma^{\mathcal{R}})^{(N+1)}(\mathcal{A})$, then $\mathcal{R}^*(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}((C_\gamma^{\mathcal{R}})^{(N)}(\mathcal{A}))$.*

The above method does not work for all TRSs. For instance, consider a constant A and the tree automaton $\mathcal{A} = (\{q_1, q_2, q_f\}, \{A \to q_1, A \to q_2, f(q_1, q_2) \to q_f\}, \{q_f\})$ and the TRS $\mathcal{R} = \{f(x, x) \to g(x)\}$. There is no substitution σ such that $l\sigma \to_{\mathcal{A}}^* q$, for a q in $\{q_1, q_2, q_f\}$. Thus, following the procedure, there is no transition to add.

But $f(A, A) \in \mathcal{L}(\mathcal{A})$. Thus $g(A) \in \mathcal{R}(\mathcal{L}(\mathcal{A}))$. Since $g(A) \notin \mathcal{L}(\mathcal{A})$, the procedure stops (in fact does not begin) before providing an over-approximation of $\mathcal{R}^*(\mathcal{L}(\mathcal{A}))$.

3 Contributions

This section extends an approximation-based technique introduced in [18] for left-linear term-rewriting systems, to TRSs with left-quadratic rules.

Let $\mathcal{A} = (\mathcal{Q}, \Delta, \mathcal{Q}_f)$ be a finite bottom-up tree automaton. The automaton $\mathcal{A}^\square = (\mathcal{Q}^\square, \Delta^\square, \mathcal{Q}_f^\square)$ is defined by:

- $\mathcal{Q}^\square = \{\{q\} \mid q \in \mathcal{Q}\} \cup \{\{q_1, q_2\} \mid q_1, q_2 \in \mathcal{Q}\}$ (states of $\mathcal{Q}^\square$ are denoted with a $\square$ exponent),
- $\mathcal{Q}_f^\square = \{\{q\} \mid q \in \mathcal{Q}_f\}$,
- $\Delta^\square = \{f(q_1^\square, \dots, q_n^\square) \to q^\square \mid \forall q \in q^\square,\ \exists q_1, \dots, q_n \in \mathcal{Q},\ \forall 1 \leq i \leq n, q_i \in q_i^\square$ and $f(q_1, \dots, q_n) \to q \in \Delta\}$.

To illustrate the definition above, let's consider the automaton $\mathcal{A}$ whose final state is q_f and whose transitions are $A \to q_1$, $A \to q_2$ and $f(q_1, q_2) \to q_f$. The states of $\mathcal{A}^\square$ are all pairs of states and singletons over $\{q_1, q_2, q_f\}$, and the transitions are $A \to \{q_1\}$, $A \to \{q_2\}$, $A \to \{q_1, q_2\}$, $f(\{q_1\}, \{q_2\}) \to \{q_f\}$, $f(\{q_1, q_i\}, \{q_2, q_j\}) \to \{q_f\}$ for all $i, j \in \{1, 2, f\}$. When considering only the accessible states, among all the transitions above we just have the transition $f(\{q_1, q_2\}, \{q_2, q_1\}) \to \{q_f\}$ ($i = 2$ and $j = 1$).

Proposition 3.1 *One has $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}^\square)$.*

Proof. By definition of $\mathcal{A}^\square$, if $f(q_1, \dots, q_n) \to q \in \Delta$, then $f(\{q_1\}, \dots, \{q_n\}) \to \{q\} \in \Delta^\square$. Consequently, for every term t such that $t \to_{\mathcal{A}}^* q$, one also has $t \to_{\mathcal{A}^\square}^* \{q\}$. Since for every $q_f \in \mathcal{Q}_f$, $\{q_f\} \in \mathcal{Q}_f^\square$, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}^\square)$.

It remains to prove that $\mathcal{L}(\mathcal{A}^\square) \subseteq \mathcal{L}(\mathcal{A})$. We will prove by induction on k that for every $k \geq 1$, for every term t, every state $q^\square$ of $\mathcal{A}^\square$, if $t \to_{\mathcal{A}^\square}^k q^\square$, then for all $q \in q^\square$, $t \to_{\mathcal{A}}^k q$.

- If $t \to_{\mathcal{A}^\square} q^\square$, then, by definition of $\Delta^\square$, t is a constant and for all $q \in q^\square$, there exists a transition $t \to q$ of $\mathcal{A}$.
- Assume now that the claim is true for a fixed positive integer k. Let t be a term and $q^\square \in \mathcal{A}^\square$ such that $t \to_{\mathcal{A}^\square}^{k+1} q^\square$. Consequently, there exists $f \in \mathcal{F}_n$ such that $t \to_{\mathcal{A}^\square}^k f(q_1^\square, \dots\dots, q_n^\square) \to_{\mathcal{A}^\square} q^\square$. It follows that $t = f(t_1, \dots, t_k)$ and for all $1 \leq i \leq k$, $t_i \to_{\mathcal{A}^\square}^k q_i^\square$. Using the induction hypothesis, $t_i \to_{\mathcal{A}}^k q_i$, for all $q_i \in q_i^\square$. Consequently, for all $q \in q^\square$, $f(q_1, \dots, q_n) \to q \in \Delta$, proving the induction.

So, $\mathcal{L}(\mathcal{A}^\square) \subseteq \mathcal{L}(\mathcal{A})$. $\square$

Lemma 3.2 *If $C[q_1, \dots, q_n] \to_{\mathcal{A}}^* q$ and if $q_1^\square, \dots q_n^\square$ are states of $\mathcal{A}^\square$ satisfying $q_i \in q_i^\square$ for all $1 \leq i \leq n$, then $C[q_1^\square, \dots, q_n^\square] \to_{\mathcal{A}^\square}^* \{q\}$.*

Proof. We prove by induction on k that for every $k \geq 1$, if $C[q_1, \dots, q_n] \to_{\mathcal{A}}^k q$ and if $q_1^\square, \dots q_n^\square$ are states of $\mathcal{A}^\square$ satisfying $q_i \in q_i^\square$ for all $1 \leq i \leq n$, then

$C[q_1^\square, \ldots, q_n^\square] \rightarrow^k_{\mathcal{A}^\square} \{q\}$.

- If $k = 1$, then $C[q_1, \ldots, q_n] \rightarrow q$ is a transition of $\mathcal{A}$. Therefore, by definition of $\Delta^\square$, $C[q_1^\square, \ldots, q_n^\square] \rightarrow \{q\}$ is a transition of $\mathcal{A}^\square$.
- Assume now that the proposition is true for all $j \leq k$ and that $C[q_1, \ldots, q_n] \rightarrow^{k+1}_{\mathcal{A}} q$. There exist $q'_1, \ldots, q'_\ell$ states of $\mathcal{A}$ and $f \in \mathcal{F}_\ell$ such that $C[q_1, \ldots, q_n] \rightarrow^k_{\mathcal{A}} f(q'_1, \ldots, q'_\ell) \rightarrow_{\mathcal{A}} q$. Consequently, $C[q_1, \ldots, q_n]$ is of the form $C[q_1, \ldots, q_n] = f(t_1, \ldots, t_\ell)$ where the t_i's are terms over $\mathcal{F} \cup \{q_1, \ldots, q_n\}$. Moreover, for all i, there exists $k_i \leq k$ such that $t_i \rightarrow^{k_i}_{\mathcal{A}} \{q'_i\}$ and $\sum_i k_i = k$. Therefore, by induction hypothesis, $t_i^\square \rightarrow^{k_i}_{\mathcal{A}^\square} \{q'_i\}$ where $t_i^\square$ is the term obtained from t_i by substituting q_i by $q_i^\square$. Now, since $f(q'_1, \ldots, q'_\ell) \rightarrow q$ is a transition of $\mathcal{A}$, $f(\{q'_1\}, \ldots, \{q'_\ell\}) \rightarrow \{q\}$ is a transition of $\mathcal{A}^\square$. It follows that $C[q_1^\square, \ldots, q_n^\square] \rightarrow^{k+1}_{\mathcal{A}^\square} \{q\}$, proving the lemma. $\square$

Lemma 3.3 *If $t \rightarrow^*_{\mathcal{A}} q_1$ and $t \rightarrow^*_{\mathcal{A}} q_2$, then $t \rightarrow^*_{\mathcal{A}^\square} \{q_1, q_2\}$.*

Proof. If $t \rightarrow^*_{\mathcal{A}} q_1$ and $t \rightarrow^*_{\mathcal{A}} q_2$, then there exists a function π_1 (reps. π_2) from positions of t into $\mathcal{Q}$ such that $\pi_1(\varepsilon) = q_1$ (resp. $\pi_2(\varepsilon) = q_2$) and for every position p of t, if $t_p \in \mathcal{F}_n$, then $t(p)(\pi_1(p.1), \ldots, \pi_1(p.n)) \rightarrow \pi_1(p)$ (resp. $t(p)(\pi_2(p.1), \ldots, \pi_2(p.n)) \rightarrow \pi_2(p)$) is a transition of $\mathcal{A}$. Therefore, by definition of $\Delta^\square$, $t(p)(\{\pi_1(p.1), \pi_2(p.1)\}, \ldots, \{\pi_1(p.n), \pi_2(p.n)\}) \rightarrow \{\pi_1(p), \pi_2(p)\}$ is in $\Delta^\square$. It follows that $t \rightarrow^*_{\mathcal{A}^\square} \{q_1, q_2\}$. $\square$

Proposition 3.4 *If $\mathcal{R}$ is left-quadratic, then $\mathcal{R}(\mathcal{L}(\mathcal{A})) \cup \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{C}_\gamma(\mathcal{A}^\square))$.*

Proof. Since $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}^\square)$ and since $\mathcal{L}(\mathcal{A}^\square) \subseteq \mathcal{L}(\mathcal{C}_\gamma(\mathcal{A}^\square))$, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{C}_\gamma(\mathcal{A}^\square))$.

Let $t \in \mathcal{R}(\mathcal{L}(\mathcal{A}))$. By definition there exists a rule $l \rightarrow r \in \mathcal{R}$, a position p of t and a substitution μ from $\mathcal{X}$ into $\mathcal{T}(\mathcal{F})$ such that

$$t = t[r\mu]_p \quad \text{and} \quad t[l\mu]_p \in \mathcal{L}(\mathcal{A}) \tag{1}$$

It follows there exist states q, q_f of $\mathcal{A}$ such that q_f is final,

$$l\mu \rightarrow^*_{\mathcal{A}} q \quad \text{and} \quad t[q]_p \rightarrow^*_{\mathcal{A}} q_f. \tag{2}$$

Consequently,

$$l\mu \rightarrow^*_{\mathcal{A}^\square} \{q\} \quad \text{and} \quad t[\{q\}]_p \rightarrow^*_{\mathcal{A}^\square} \{q_f\}. \tag{3}$$

If $r\mu \rightarrow^*_{\mathcal{A}^\square} \{q\}$, then (3) implies that $t[r\mu]_p \rightarrow^*_{\mathcal{A}^\square} \{q_f\}$. In this case, since $t = t[r\mu]_p$ and since $\{q_f\}$ is by construction a final state of $\mathcal{A}^\square$, t is in $\mathcal{L}(\mathcal{A}^\square)$, which is a subset of $\mathcal{L}(\mathcal{C}_\gamma(\mathcal{A}^\square))$.

Now we may assume that $r\mu \not\rightarrow^*_{\mathcal{A}^\square} \{q\}$. Let P_l be the set of variable positions of l; i.e. $P_l = \{p \mid l(p) \in \mathcal{X})\}$. Set $P_l = \{p_1, \ldots, p_\ell\}$. Since $l\mu \rightarrow^*_{\mathcal{A}} q$, by (2) there exist states $q_1, \ldots, q_\ell$ of $\mathcal{A}$ such that

$$\mu(l(p_i)) \rightarrow^*_{\mathcal{A}} q_i \quad \text{and} \quad l[q_1]_{p_1} \ldots [q_\ell]_{p_\ell} \rightarrow^*_{\mathcal{A}} q. \tag{4}$$

We define the substitution σ from variables occurring in l into $2^{\mathcal{Q}}$ by: $\sigma(x_i) = \{q_i \mid l(p_i) = x_i\}$. Since l is left-quadratic, for each x_i, $\sigma(x_i)$ contains at most two states.

We claim that $l\sigma \rightarrow^*_{\mathcal{A}^\square} q$. Indeed by (4) and by Lemma 3.3 for each x_i occurring in l, $\mu(x_i) \rightarrow^*_{\mathcal{A}^\square} \sigma(x_i)$. It follows that $l\mu \rightarrow^*_{\mathcal{A}^\square} l\sigma$. By (4) and using Lemma 3.2, $l\sigma \rightarrow^*_{\mathcal{A}^\square} \{q\}$, proving the claim. By construction of $\mathcal{C}_\gamma(\mathcal{A}^\square)$, $r\sigma \rightarrow^*_{\mathcal{C}_\gamma(\mathcal{A}^\square)} \{q\}$. Moreover, by definition of σ, $r\mu \rightarrow^*_{\mathcal{A}^\square} r\sigma$. It follows that

$$t = t[r\mu]_p \rightarrow^*_{\mathcal{A}^\square} t[r\sigma]_p \rightarrow^*_{\mathcal{C}_\gamma(\mathcal{A}^\square)} t[\{q\}]_p \rightarrow^*_{\mathcal{A}^\square} \{q_f\},$$

which completes the proof. □

Proposition 3.5 *If $\mathcal{R}$ is right-linear and if α is $(\mathcal{A}, \mathcal{R})$-exact, then $\mathcal{L}(\mathcal{C}_\gamma(\mathcal{A}^\square)) \subseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$.*

Proof. This is a direct consequence of Theorem 2.4 and Proposition 3.1. □

4 Example and Application Domains

4.1 Example

We have tested our approach on the following family of examples. We first consider a family of tree automata $(\mathcal{A}_n)$ defined as follows: the set of states of $\mathcal{A}_n$ is $\{q_1, \ldots, q_{2n+2}, q_f\}$, the set of final state is $\{q_f\}$, and the set of transitions is $\{\omega \rightarrow q_1, \omega \rightarrow q_2, a(q_1) \rightarrow q_1, a(q_2) \rightarrow q_2, b(q_1) \rightarrow q_1, b(q_2) \rightarrow q_2, a(q_1) \rightarrow q_3, a(q_2) \rightarrow q_4, a(q_i) \rightarrow q_{i+2}, b(q_i) \rightarrow q_{i+2}, f(q_{2n+1}, q_{2n+2}) \rightarrow q_f\}$, for $i \geq 3$. The automaton $\mathcal{A}_n$ accepts the set of terms of the form $f(t_1, t_2)$ where t_1 and t_2 are terms over $\{a, b, \omega\}$ such that $t_1|_{1^{n-1}}$ and $t_2|_{1^{n-1}}$ exist and are in $\{a\}.\{a, b\}^*$. Roughly speaking, when using word automata, $a(b(\omega))$ denotes ab, and each pair (t_1, t_2) can be viewed as words of $\mathcal{L} = \{a, b\}^{n-1}.\{a\}.\{a, b\}^*$ satisfying the condition above. We second consider the term rewriting system $\mathcal{R}$ containing the single rule $f(x, x) \rightarrow x$, and we want to prove that $b^{n-1}a(\omega) \in \mathcal{R}^*(\mathcal{L}(\mathcal{A}_n))$. Using finitely many times Theorem 2.4 directly on $\mathcal{A}_n$ may not prove the results. However, to prove the results, one can determinise $\mathcal{A}_n$ before using Theorem 2.4. But, the minimal automaton of $\mathcal{L}(\mathcal{A}_n)$ has 2^n states at least [22], [Exercise 3.20, p. 73]. Then, the completion should be applied to this automaton. Consequently, this automatic proof requires an exponential time step. Using our approach, one can compute $\mathcal{A}^\square$ and apply Proposition 3.5, that provides the proof requiring a polynomial time step.

4.2 Left-linearity and Security Issues

4.2.1 Security Protocol Analysis

The TRSs used in the security protocol verification context are often non left-linear. Indeed, there is a lot of protocols that cannot be modeled by left-linear TRSs. Unfortunately, to be sound, the approximation-based analysis described in [19] requires the use of left-linear TRSs. Nevertheless, this method can still be applied to some non left-linear TRSs, which satisfy some weaker conditions. In [17] the authors propose new linearity conditions. However, these new conditions are not well-adapted to be automatically checked.

In our previous work [6] we explain how to define a criterion on $\mathcal{R}$ and $\mathcal{A}$ to make the procedure automatically work for industrial protocols analysis. This criterion ensures the soundness of the method described in [19,17]. However, to handle protocols the approach in [6] is based on a kind of constant typing. In [7] we go further and propose a procedure supporting a fully automatic analysis and handling – without typing – algebraic properties like XOR.
Let us first remark that the criterion defined in [17] does not allow managing the XOR non-left linear rule. Second, in [6] we have restricted XOR operations to typed terms to deal with the XOR non-left linear rule. However, some protocols are known to be flawed by type confusing attacks [14,10,11]. Notice that our approach in [7] can be applied to any kinds of TRSs. Moreover, it can cope with exponentiation algebraic properties and this way analyse Diffie-Hellman based protocols.

4.2.2 Backward Analysis of Java Bytecode

A recent work [4], dedicated to the static analysis of Java bytecode programs using term-rewriting systems, provides an automatic procedure to translate a Java bytecode into a term rewriting system modeling the code execution on the Java Virtual Machine. In this context, generated TRSs are left-linear but right-quadratic. In order to compute approximation refinements as in [3] or to manage backward analyses that are – in general and in practice – more efficient that forward analyses – term rewriting systems have to be turned left-right, i.e. left- and right-hand sides of rules have to be permuted. By this permutation right-quadratic TRSs become left-quadratic ones.

5 Conclusion

Regular approximation techniques have been successfully used in the context of security protocol analysis. In order to apply them to other applications, this paper proposed an extension of the completion procedure for handling left-quadratic rules. Our contributions allow analysing some reachability problems using polynomial steps computing $\mathcal{A}^{\Box}$, rather than automata determinisation steps that are exponential, even in practical cases. Notice that the approach presented only for quadratic rules can be extended to more complex TRSs. We intend to optimise this technique: polynomial is better than exponential but may also lead to huge automata in few steps. We have been implementing the techniques in an efficient rewriting tool in order to investigate complex systems backward analyses.

References

[1] P. A. Abdulla, B. Jonsson, P. Mahata, and J. d'Orso. Regular tree model checking. In E. Brinksma and K. G. Larsen, editors, *Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 555–568. Springer-Verlag, July 27–31 2002.

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

[3] Y. Boichut, R. Courbis, P.-C. Ham, and O. Kouchnarenko. Finer is better: Abstraction refinement for rewriting approximations. In *RTA'08*, LNCS, 2008. To appear.

[4] Y. Boichut, Th. Genet, Th. Jensen, and L. Le Roux. Rewriting approximations for fast prototyping of static analyzers. In *proceedings of RTA*, LNCS 4533, pages 48–62. Springer, 2007.

[5] Y. Boichut, Th. Genet, Th. P. Jensen, and L. Le Roux. Rewriting approximations for fast prototyping of static analyzers. In Franz Baader, editor, *Term Rewriting and Applications, 18th International Conference, RTA 2007, Paris, France, June 26-28, 2007, Proceedings*, volume 4533 of *LNCS*, pages 48–62. Springer, 2007.

[6] Y. Boichut, P.-C. Héam, and O. Kouchnarenko. Automatic Verification of Security Protocols Using Approximations. Technical Report RR-5727, INRIA, 2005.

[7] Y. Boichut, P.-C. Héam, and O. Kouchnarenko. Handling algebraic properties in automatic analysis of security protocols. In Kamel Barkaoui, Ana Cavalcanti, and Antonio Cerone, editors, *ICTAC'06*, volume 4281 of *LNCS*, pages 153–167. Springer, 2006.

[8] A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract regular tree model checking. In *Infinity'05*, volume 149 of *Electronic Notes in Theoretical Computer Science*, pages 37–48, 2006.

[9] A. Bouajjani and T. Touili. Extrapolating tree transformations. In Ed Brinksma and Kim Guldstrand Larsen, editors, *Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 539–554. Springer-Verlag, July 27–31 2002.

[10] L. Bozga, Y. Lakhnech, and M. Perin. Pattern-based abstraction for verifying secrecy in protocols. In *9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems,TACAS 2003, Proceedings*, volume 2619 of *LNCS*. Springer-Verlag, 2003.

[11] I. Cibrario, L. Durante, R. Sisto, and A. Valenzano. Automatic detection of attacks on cryptographic protocols: A case study. In Christopher Kruegel Klaus Julisch, editor, *Intrusion and Malware Detection and Vulnerability Assessment: Second International Conference*, volume 3548 of *LNCS*, Vienna, 2005.

[12] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree Automata Techniques and Applications, 2002.

[13] J.-L. Coquidé, M. Dauchet, R. Gilleron, and Vágvölgyi S. Bottom-up tree pushdown automata and rewrite systems. In Ronald V. Book, editor, *Rewriting Techniques and Applications, 4th International Conference, RTA-91*, LNCS 488, pages 287–298, Como, Italy, April 10–12, 1991. Springer-Verlag.

[14] V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14:1–43, 2006.

[15] Dauchet and Tison. The theory of ground rewrite systems is decidable. In *LICS: IEEE Symposium on Logic in Computer Science*, 1990.

[16] N. Dershowitz and J.-P. Jouannaud. *Handbook of Theoretical Computer Science*, volume B, chapter 6: Rewrite Systems, pages 244–320. Elsevier Science Publishers B. V, 1990.

[17] G. Feuillade, Th. Genet, and V. Viet Triem Tong. Reachability analysis of term rewriting systems. Technical Report RR-4970, INRIA, 2003. To be published in Journal of Automated Reasoning, 2004.

[18] G. Feuillade, Th. Genet, and V. VietTriemTong. Reachability analysis over term rewriting systems. *Journal of Automated Reasonning*, 33 (3-4), 2004.

[19] Th. Genet and F. Klay. Rewriting for Cryptographic Protocol Verification. In *proceedings of CADE*, volume 1831 of *LNCS*, pages 271–290. Springer-Verlag, 2000.

[20] R. Gilleron and S. Tison. Regular tree languages and rewrite systems. *Fundamenta Informatica*, 24(1/2):157–174, 1995.

[21] P. Habermehl, R. Iosif, A. Rogalewicz, and T. Vojnar. Abstract regular tree model checking of complex dynamic data structures. In *SAS'06, 13th International Static Analysis Symposium*, volume 4134 of *LNCS*, pages 52–70, 2006.

[22] J. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[23] F. Jacquemard. Decidable approximations of term rewriting systems. In *proceedings of RTA*, volume 1103, pages 362–376. Springer Verlag, 1996.

[24] H. Ohsaki and T. Takai. ACTAS: A system design for associative and commutative tree automata theory. *Electr. Notes Theor. Comput. Sci*, 124(1):97–111, 2005.

[25] P. Réty and J. Vuotto. Regular sets of descendants by leftmost strategy. *Electr. Notes Theor. Comput. Sci*, 70(6), 2002.

[26] K. Salomaa. Deterministic tree pushdown automata and monadic tree rewriting systems. *JCSS: Journal of Computer and System Sciences*, 37, 1988.