Characterizing Conclusive Approximations by Logical Formulae

Yohan Boichut, Thi-Bich-Hanh Dao, Valérie Murat

Characterizing Conclusive Approximations by Logical Formulae

Y. Boichut[1], T.-B.-H. Dao[1]and V. Murat[2]

LIFO - Université Orléans, France
IRISA - Université Rennes 1, France

Abstract. Considering an initial set of terms E, a rewriting relation $\mathcal{R}$ and a goal set of terms Bad, reachability analysis in term rewriting tries to answer to the following question: does there exists at least one term of Bad that can be reached from E using the rewriting relation $\mathcal{R}$?
Some of the approaches try to show that there exists at least one term of Bad reachable from E using the rewriting relation $\mathcal{R}$ by computing the set of reachable terms. Some others tackle the unreachability problem i.e. no term of Bad is reachable by rewriting from E. For the latter, over-approximations are computed. A main obstacle is to be able to compute an over-approximation precise enough that does not intersect Bad i.e. a conclusive approximation. This notion of precision is often defined by a very technical parameter of techniques implementing this over-approximation approach. In this paper, we propose a new characterization of conclusive approximations by logical formulae generated from a new kind of automata called symbolic tree automata. Solving a such formula leads automatically to a conclusive approximation without extra technical parameters.

1 Introduction

In the rewriting theory, the reachability problem is the following: given a term rewriting system (TRS) $\mathcal{R}$ and two terms s and t, can we decide whether $s \rightarrow_{\mathcal{R}}^{*} t$ or not? This problem, which can easily be solved on strongly terminating TRSs (by rewriting s into all its possible reduced forms and compare them to t), is undecidable on non terminating TRSs. There exists several syntactic classes of TRSs for which this problem becomes decidable: some are surveyed in [9], more recent ones are [15, 20]. In general, the decision procedures for those classes compute a finite tree automaton recognising the possibly infinite set of terms reachable from a set $E \subseteq \mathcal{T}(\mathcal{F})$ of initial terms, by $\mathcal{R}$, denoted by $\mathcal{R}^{*}(E)$. Then, provided that $s \in E$, those procedures check whether $t \in \mathcal{R}^{*}(E)$ or not. On the other hand, outside of those decidable classes, one can prove $s \not\rightarrow_{\mathcal{R}}^{*} t$ using over-approximations of $\mathcal{R}^{*}(E)$ [17, 9] and proving that t does not belong to this approximation.

Recently, reachability analysis turned out to be a very efficient verification technique for proving properties on infinite systems modeled by TRSs. Some of

the most successful experiments, using proofs of $s \not\rightarrow^*_{\mathcal{R}} t$, were done on cryptographic protocols [18, 12, 4] where protocols and intruders are described using a TRS $\mathcal{R}$, E represents the set of initial configurations of the protocol and t a possible flaw. Some other have been carried out on Java byte code programs [2] and in this context, $\mathcal{R}$ encodes the byte code instructions and the evolution of the Java Virtual Machine (JVM), E specifies the set of initial configurations of the JVM and t a possible flaw.

Then reachability analysis can prove the absence of flaws (if $\forall s \in E : s \not\rightarrow^*_{\mathcal{R}} t$). In [9], given a TRS $\mathcal{R}$, a set of terms E and an abstraction function γ, a sequence of sets of terms $App^\gamma_0, App^\gamma_1, \ldots, App^\gamma_k$ is built such that $App^\gamma_0 = E$ and $\mathcal{R}(App^\gamma_i) \subseteq App^\gamma_{i+1}$. This technique is called *tree automata completion*. The role of the abstraction γ is to define equivalence classes of terms and to allot each term to an equivalence class. The computation stops when on the one hand, the number of equivalence classes introduced by the abstraction function is bounded, and on the other hand, each equivalence class is $\mathcal{R}-$closed, i.e. when there exists $N \in \mathbb{N}$ such that $\mathcal{R}(App^\gamma_N) = App^\gamma_N$. Then, App^γ_N represents an over-approximation of terms reachable by $\mathcal{R}$ from E. The abstraction function γ should be well designed in such a way that on one hand App^γ_N exists, and on the other hand $t \notin App^\gamma_N$. However, the main drawback of this technique based on tree automata, is that if $t \notin \mathcal{R}^*(E)$ then it is not trivial (when it is possible) to compute a such fix-point over-approximation App^γ_N. Indeed, a high-level expertise in this technique is required for defining a pertinent abstraction function. In a recent work [13], the approximation function is seen as a set of equations γ. Let C_1 and C_2 be two equivalence classes and t_1 and t_2 be respectively terms of classes C_1 and C_2. If $t_1 = t_2$ modulo the set of equations γ then C_1 and C_2 are merged together.

In [5], the authors propose a similar technique in which they use tree transducers instead of TRSs. The approximation functions they use can be seen as predicates. More precisely, if two different equivalence classes satisfy the same set of predicates then they are merged together. So, one has to define carefully the set of predicates. Once again, a high-level expertise in the technique itself is required for obtaining conclusive analysis. But contrary to the technique presented in [9], their technique is equipped with a refinement process acting when the approximation function leads to inconclusive analysis. Nevertheless, the refinement process may be expensive since it involves backward computations for detecting the point where the approximation has become too coarse.

So, to summarize, both of the techniques mentioned previously are instrumented either by equations or predicates for the computation of over-approximations. Both of them use tree automata to represent over-approximations. More precisely, set of terms are represented by tree automata languages. However, these parameters often require a highly specialized expertise for expecting a conclusive analysis.

In this paper, we characterize by a logical formula all the criteria of such a conclusive analysis performed with the technique proposed in [11, 9, 13]. The idea is that instead of reasoning with a tree automaton A, we generalize A to a

symbolic tree automaton (STA) A_s, whose states are represented by variables. The rewriting relations and "bad" terms are represented by boolean combinations of equalities and inequalities on these variables. An instantiation of these variables by states gives a tree automaton, and each valid instantiation of this formula ensures that, as soon as the STA is instantiated, the language of the resulting tree automaton is a conclusive over-approximation of the set of terms reachable from the language of A according to the rewriting relation. With this formulation, finding a conclusive analysis becomes solving logical formulae. Thus, different solving and search techniques, for example in artificial intelligence, can be applied.

The paper is organized as follows: Section 2 recalls background on terms, rewriting and tree automata as well as the connection between rewriting and tree automata. In this section we also describe the kind of formulae we manipulate and notion of instantiations. Section 3 introduces symbolic tree automata. In this section, we point out the connection between an STA and traditional tree automata. Section 4 describes the cornerstone of our contribution: the matching algorithm for STA. In other words, given a term t, we characterize each solution of this pattern as well as its existence condition by a formula. Section 5 presents our main contribution: the characterization of a conclusive over-approximation by a formula. Before concluding, in Section 6, given a TRS $\mathcal{R}$, a tree automaton A and a set of goal terms Bad, we describe a semi-algorithm for computing automatically a conclusive approximation. For a lack of space, the proofs of this paper are available at `http://www.univ-orleans.fr/lifo/prodsci/rapports/RR/RR2011/RR-2011-04.pdf`.

2 Background and Notations

In this section, we introduce some definitions and concepts that will be used throughout the rest of the paper (see also [1, 8, 14]). Let $\mathcal{F}$ be a finite set of symbols, each one is associated with an arity, 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 term t is said *linear* if there is no variable appearing more than once in t. A *substitution* is a function σ from $\mathcal{X}$ into $\mathcal{T}(\mathcal{F}, \mathcal{X})$, which can be uniquely extended to an endomorphism of $\mathcal{T}(\mathcal{F}, \mathcal{X})$. The substitution σ applied to the term t (denoted $t\sigma$) is constructed such that $x\sigma = \sigma(x)$, where $x \in \mathcal{X}$, and $f(t_1, ..., t_n)\sigma = f(t_1\sigma, ..., t_n\sigma)$. Let A, B and C be three sets of elements. Let σ and μ be two substitutions such that $\sigma : A \mapsto B$ and $\mu : B \mapsto C$. We denote by $\sigma \circ \mu$ the substitution such that $\sigma \circ \mu(x) = \mu(\sigma(x))$ where $x \in A$.

A *term rewriting system* (TRS) $\mathcal{R}$ is a set of *rewrite rules* $l \to r$, where $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l, r \notin \mathcal{X}$[1], and $\mathcal{V}ar(l) \supseteq \mathcal{V}ar(r)$. A rewrite rule $l \to r$ is *left-linear* if l is linear. A TRS $\mathcal{R}$ is left-linear if every rewrite rule $l \to r$ of $\mathcal{R}$ is left-linear. The TRS $\mathcal{R}$ induces a rewriting relation $\to_{\mathcal{R}}$ on terms as follows. Let

[1] A more general definition is that only l must not be a variable.

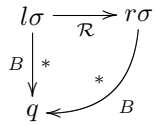
$s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $l \to r \in \mathcal{R}$, $s \to_{\mathcal{R}} t$ denotes that there exists a subterm u of s and a substitution σ such that $u = l\sigma$ and t is obtained by substituting u by $r\sigma$ in s. The reflexive transitive closure of $\to_{\mathcal{R}}$ is denoted by $\to^*_{\mathcal{R}}$. The set of $\mathcal{R}$-descendants of a set of ground terms I is $\mathcal{R}^*(I) = \{t \in \mathcal{T}(\mathcal{F}) \mid \exists s \in I \text{ s.t. } s \to^*_{\mathcal{R}} t\}$. We now define tree automata that are used to recognize possibly infinite sets of terms. Let Q be a finite set of symbols with arity 0, called *states*, such that $Q \cap \mathcal{F} = \emptyset$. $\mathcal{T}(\mathcal{F} \cup Q)$ is called the set of *configurations*. A *transition* is a rewrite rule $c \to q$, where c is a configuration and q is a state. A transition is *normalized* when $c = f(q_1, \ldots, q_n)$, $f \in \mathcal{F}$ is of arity n, and $q_1, \ldots, q_n \in Q$.

Definition 1 (Bottom-up nondeterministic finite tree automaton). *A bottom-up nondeterministic finite tree automaton (tree automaton for short) over the alphabet $\mathcal{F}$ is a tuple $A = \langle Q, \mathcal{F}, Q_F, \Delta \rangle$, where $Q_F \subseteq Q$ is the set of final states, Δ is a set of normalized transitions.*

The transitive and reflexive *rewriting relation* on $\mathcal{T}(\mathcal{F} \cup Q)$ induced by all the transitions of A is denoted by $\to^*_A$. The tree language recognized by A in a state q is $\mathcal{L}(A, q) = \{t \in \mathcal{T}(\mathcal{F}) \mid t \to^*_A q\}$. We define $\mathcal{L}(A) = \bigcup_{q \in Q_F} \mathcal{L}(A, q)$.

Some of the techniques marry ([9, 19, 6]) tree automata and rewriting for computing the set of reachable terms from a given tree automata A i.e. $\mathcal{R}^*(\mathcal{L}(A))$. Unfortunately, enumerating reachable terms may never terminate. There is thus a need to "accelerate" the search through the term space in order to reach, in a finite amount of time, terms at unbounded depths.

Definition 2. *A tree automaton B is $\mathcal{R}$-closed if for each rule $l \to r \in \mathcal{R}$, for any substitution $\sigma : \mathcal{X} \mapsto Q$, $l\sigma$ is recognized by B into state q then so is $r\sigma$. The situation is represented with the following graph:*



It is easy to see that if B is $\mathcal{R}$-closed and $\mathcal{L}(B) \supseteq \mathcal{L}(A)$, then $\mathcal{L}(B) \supseteq \mathcal{R}^*(\mathcal{L}(A))$[7].

In the following definitions, we introduce the logical formulae that we manipulate as well as notions of instantiation and satisfaction of a formula.

Definition 3 ($\mathcal{W}[\mathcal{X}_\mathcal{Q}]$). *Let $\mathcal{X}_\mathcal{Q}$ be a set of variables. We define $\mathcal{W}[\mathcal{X}_\mathcal{Q}]$ to be the set of logical formulae on $\mathcal{X}_\mathcal{Q}$ as following:*

- $\top, \bot \in \mathcal{W}[\mathcal{X}_\mathcal{Q}]$;
- $X = Y$, $X \neq Y \in \mathcal{W}[\mathcal{X}_\mathcal{Q}]$ *with* $X, Y \in \mathcal{X}_\mathcal{Q}$;
- *if* $\alpha, \beta \in \mathcal{W}[\mathcal{X}_\mathcal{Q}]$ *then* $\neg\alpha$, $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \Rightarrow \beta$ *are in* $\mathcal{W}[\mathcal{X}_\mathcal{Q}]$.

Definition 4 (Instantiation/satisfaction). *Let D be a domain which is a non-empty set. An instantiation ι of variables of $\mathcal{X}_\mathcal{Q}$ is a function $\iota : \mathcal{X}_\mathcal{Q} \to D$. The instantiation ι satisfies a formula $\alpha \in \mathcal{W}[\mathcal{X}_\mathcal{Q}]$, denoted by $\iota \models \alpha$, iff:*

– $\iota \models \top$;
– $\iota \models X = Y$ *iff* $\iota(X) = \iota(Y)$; $\iota \models X \neq Y$ *iff* $\iota(X) \neq \iota(Y)$;
– $\iota \models \neg\alpha$ *iff* $\iota \not\models \alpha$; $\iota \models \alpha \wedge \beta$ *iff* $\iota \models \alpha$ *and* $\iota \models \beta$;
$\iota \models \alpha \vee \beta$ *iff* $\iota \models \alpha$ *or* $\iota \models \beta$; $\iota \models \alpha \Rightarrow \beta$ *iff* $\iota \not\models \alpha$ *or* $\iota \models \alpha \wedge \beta$.

Example 1. Let $\mathcal{X}_Q$ be the set of variables such that $\mathcal{X}_Q = \{X_1, X_2, X_3\}$. Thus, $(X_1 \neq X_2) \wedge ((X_1 = X_3) \vee (X_2 = X_3))$ is a formula in $\mathcal{W}[\mathcal{X}_\mathcal{Q}]$. Let $D = \{1, 2\}$ and ι be the instantiation such that $\iota(X_1) = 2$, $\iota(X_2) = \iota(X_3) = 1$. We have $\iota \not\models X_1 = X_2$ and $\iota \models (X_1 = X_2) \vee (X_2 = X_3)$.

Note that instantiations will be also considered as substitutions in the remainder of the paper.

3 Symbolic Tree Automata

Let $\mathcal{X}_\mathcal{Q}$ be a set of variables that we call symbolic states. Symbolic tree automata (STA) are tree automata where states are variables. An STA is composed of normalized symbolic transitions as defined below.

Definition 5 (Normalized symbolic transition). *Let $\mathcal{X}_\mathcal{Q}$ be a set of symbolic states. A normalized symbolic transition is of the form $f(X_1, .., X_n) \to X$ where $f \in \mathcal{F}$ of arity n and $X, X_1, .., X_n \in \mathcal{X}_\mathcal{Q}$.*

Definition 6 (STA). *An STA is a tuple $\langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta \rangle$ where $\mathcal{X}_\mathcal{Q}$ is a set of symbolic states, $\mathcal{F}$ a set of functional symbols, $\mathcal{X}^f_\mathcal{Q} \subseteq \mathcal{X}_\mathcal{Q}$ a set of final symbolic states and Δ a set of normalized symbolic transitions.*

Example 2. Let $\mathcal{F}$ be a set of functional symbols such that $\mathcal{F} = \{a : 0, s : 1\}$. Let $\mathcal{X}_\mathcal{Q}$ and $\mathcal{X}^f_\mathcal{Q}$ be two sets of symbolic states such that $\mathcal{X}_\mathcal{Q} = \{X_{q_0}, X_{q_1}\}$ and $\mathcal{X}^f_\mathcal{Q} = \{X_{q_1}\}$. Let Δ be a set of symbolic transitions such that $\Delta = \{a \to X_{q_0}, a \to X_{q_1}, s(X_{q_0}) \to X_{q_1}\}$. Thus, considering $A_S = \langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta \rangle$, A_S is an STA.

The following definition gives details on how a tree automaton can be obtained from a STA and a given instantiation from $\mathcal{X}_\mathcal{Q}$ to a domain Q.

Definition 7 (Instance of a STA). *Let Q be a non-empty set of states. Let A_S be an STA $\langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta \rangle$ and ι be an instantiation $\mathcal{X}_\mathcal{Q} \to Q$. An instance of A_S by ι, denoted by A^ι_S, is a tree automaton $\langle Q^{A^\iota_S}, \mathcal{F}, Q_f^{A^\iota_S}, \Delta^{A^\iota_S} \rangle$ where:*

– $Q^{A^\iota_S} = \{\iota(X) \mid X \in \mathcal{X}_\mathcal{Q}\}$; $Q_f^{A^\iota_S} = \{\iota(X) \mid X \in \mathcal{X}^f_\mathcal{Q}\}$;
– $\Delta^{A^\iota_S} = \{f(\iota(X_1), \ldots, \iota(X_n)) \to \iota(X) \mid f(X_1, \ldots, X_n) \to X \in \Delta\}$.

Example 3. Let A_S be the STA defined in Example 2. Let ι_1 and ι_2 be two instantiations such that $\iota_1 = \{X_{q_0} \mapsto q, X_{q_1} \mapsto q\}$ and $\iota_2 = \{X_{q_0} \mapsto q', X_{q_1} \mapsto q\}$. Thus, $A^{\iota_1}_S = \langle \{q\}, \mathcal{F}, \{q\}, \{a \to q, s(q) \to q\} \rangle$ and $A^{\iota_2}_S = \langle \{q', q\}, \mathcal{F}, \{q\}, \{a \to q', a \to q, s(q') \to q\} \rangle$. Note that $\mathcal{L}(A^{\iota_1}_S) = \{s^n(a) \mid n \geq 0\}$ and $\mathcal{L}(A^{\iota_2}_S) = \{a, s(a)\}$.

For a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}_Q)$, a formula $\alpha \in \mathcal{W}[\mathcal{X}_\mathcal{Q}]$ and a symbolic state X, we define the relation $t \xrightarrow{\alpha}_{A_S} X$. In a couple of words, if an instantiation ι satisfies α then the relation ensures that A_S^ι accepts the term t in the state $\iota(X)$. Note that if $t \xrightarrow{\alpha}_{A_S} X$ then α is a conjunction of equalities between symbolic states. This is involved by a straightforward reduction of the term t using transitions of A_S.

Definition 8 ($t \xrightarrow{\alpha}_{A_S} X$). *Let A_S be an STA $\langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}_\mathcal{Q}^f, \Delta \rangle$. Let t be a term of $\mathcal{T}(\mathcal{F}, \mathcal{X}_Q)$ and X a symbolic state of $\mathcal{X}_\mathcal{Q}$. One has:*

- $X \xrightarrow{\top}_{A_S} X$
- *If $t \to Y \in \Delta$ then $t \xrightarrow{X=Y}_{A_S} X$*
- *If $t = f(t_1, ..., t_n)$ and $t_1 \xrightarrow{\alpha_1}_{A_S} X_1$, ..., $t_n \xrightarrow{\alpha_n}_{A_S} X_n$ and $f(X_1, ..., X_n) \to Y \in \Delta$ then $t \xrightarrow{\alpha_1 \wedge \cdots \wedge \alpha_n \wedge X=Y}_{A_S} X$*

Example 4. Let A_S be the STA defined in Example 2. Let t be a term of $\mathcal{T}(\mathcal{F})$ such that $t = s(s(s(a)))$. According to Definition 8, one has $t \xrightarrow{\alpha}_{A_S} X_{q_1}$ with $\alpha = X_{q_0} = X_{q_1}$. Let ι_1 be the instantiation defined in Example 3. Note that, according to Definition 4, $\iota_1 \models \alpha$. Note also that $s(s(s(a))) \to^*_{A_S^{\iota_1}} \iota_1(X_{q_1})$.

Usually, given an STA A_S, a term t, a formula α, a symbolic state X and an instantiation ι, one cannot deduce that $t \not\to^*_{A_S^\iota} \iota(X)$ if $\iota \not\models \alpha$. Nevertheless, if for any formula α such that $t \xrightarrow{\alpha}_{A_S} X$ one has $\iota \not\models \alpha$ then one can conclude that $t \not\to^*_{A_S^\iota} \iota(X)$.

The following proposition presents the characterization by a formula of the acceptance of a term t by a given STA. Consequently, each instantiation satisfying this formula leads to an automaton recognizing t.

Proposition 1. *Let $A_S = \langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}_\mathcal{Q}^f, \Delta \rangle$ be an STA and ι be an instantiation. Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}_Q)$ and $X \in \mathcal{X}_\mathcal{Q}$. Let $\mathtt{Reco}(t, X) = \bigvee_{\{t \xrightarrow{\alpha}_{A_S} X\}} \alpha$. Then, one has:*

$$\iota \models \mathtt{Reco}(t, X) \quad \textit{iff} \quad t\iota \to^*_{A_S^\iota} \iota(X).$$

4 Solutions for Patterns in STA

Let t be a term of $\mathcal{T}(\mathcal{F}, \mathcal{X})$. For a classical tree automaton A and a state q, the matching problem $t \trianglelefteq q$ has a solution if there exists a substitution $\sigma : \mathcal{X} \mapsto Q$ such that $t\sigma \to^*_A q$. Let us recall that this point is essential for testing whether an automaton is $\mathcal{R}-$closed or not (see Definition 2).

In this section, we propose to solve this problem in the context of STA. Thus, the matching problem is formalized on symbolic states instead of classical states i.e. $t \trianglelefteq X$ with $X \in \mathcal{X}_\mathcal{Q}$. Actually, in this context and considering an STA A_S, solutions are represented as a set of pairs (α, σ) where σ is a substitution from $\mathcal{X}$ to $\mathcal{X}_\mathcal{Q}$ and α a formula such that $t\sigma \xrightarrow{\alpha}_{A_S} X$. Suppose $\iota : \mathcal{X}_\mathcal{Q} \mapsto Q$ is an instantiation. Semantically, a solution (α, σ) means that, as soon as $\iota \models \alpha$, the substitution $\sigma \circ \iota$ is a solution of the matching problem $t \trianglelefteq \iota(X)$ in the tree automaton A_S^ι.

Definition 9 (Matching Algorithm – S^t_X). *Let A_S be an STA $\langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta\rangle$. We denote by S^t_X the solution set of the matching problem $t \trianglelefteq X$ where t is a linear term. S^t_X is built recursively as follows:*

$$S^t_X = \begin{cases} \{(\top, \{t \mapsto X\})\} & \text{if } t \in \mathcal{X} \qquad (Var) \\ \{(X = Y, \emptyset)\} & \text{if } t = Y \in \mathcal{X}_\mathcal{Q} \text{ or } t \to Y \in \Delta \quad (SymbVar) \\ \bigotimes^{X=Y}_{k=1\ldots n}(S^{t_k}_{X_k}) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } \quad (Delta) \\ & f(X_1, \ldots, X_n) \to Y \in \Delta \end{cases}$$

where $\bigotimes^{\phi}_{k=1\ldots n}(S^{t_k}_{X_k}) = \{(\phi, \emptyset) \oplus (\phi_1, \sigma_1) \oplus \cdots \oplus (\phi_n, \sigma_n) \mid (\phi_i, \sigma_i) \in S^{t_i}_{X_i}\}$, and $(\phi, \sigma) \oplus (\phi', \sigma') = (\phi \wedge \phi', \sigma \cup \sigma')$.

The following proposition shows that this algorithm is sound and complete.

Proposition 2. *Let A_S be an STA $\langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta\rangle$, X be a symbolic state in $\mathcal{X}_\mathcal{Q}$, t be a linear term in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and σ be a substitution from $\mathcal{V}ar(t)$ into $\mathcal{X}_\mathcal{Q}$. One has*

$$\forall(\alpha, \sigma),\ t\sigma \xrightarrow{\alpha}_{A_S} X \text{ iff } (\alpha, \sigma) \in S^t_X.$$

Example 5. Let A_S be an STA whose symbolic transition set $\Delta = \{a \to X_{q_0}, a \to X_{q_1}, s(X_{q_0}) \to X_{q_1}\}$. Using the rules we can find that $S^a_{X_{q0}} = \{(\top, \emptyset), (X_{q1} = X_{q0}, \emptyset)\}$, $S^{s(a)}_{X_{q1}} = \{(\top, \emptyset), (X_{q1} = X_{q0}, \emptyset)\}$ and $S^{s(s(a))}_{X_{q1}} = \{(X_{q0} = X_{q1}, \emptyset), (X_{q0} = X_{q1}, \emptyset)\}$.

5 Finding a Conclusive Fix-Point Automaton

Let us recall that the *Graal* of the tree automata completion is to detect a conclusive fix-point automaton. Given a set of terms Bad, a TRS $\mathcal{R}$ and an initial tree automaton A, a conclusive fix-point automaton is a tree automaton $A^\star$ such that $A^\star$ is $\mathcal{R}$-closed with regard to A and $\mathcal{L}(A^\star) \cap Bad = \emptyset$. Note also that the tree automata completion is only sound for left linear TRSs. So, we only consider left linear TRSs .

In this section, given an STA A_S, a TA A, a TRS $\mathcal{R}$ and a set of bad terms Bad, we propose two formulae $\phi^{FP}_{\mathcal{R},A_S}$ and $\phi^{Bad}_{A_S}$ such that any instantiation ι of A_S satisfying both formulae leads to a conclusive automaton. Moreover, we define a notion of compatibility between A and A_S ensuring that the automaton A^ι_S is a conclusive automaton with regard to A.

The constraint presented below depicts a condition, built from A_S, to satisfy for any instantiation ι in order to ensure that A^ι_S is $\mathcal{R}$-closed. In [9], a TA A is $\mathcal{R}$-closed (fix-point automaton) if $\forall l \to r \in \mathcal{R}$, $\forall \sigma : \mathcal{X} \mapsto Q$ and $\forall q$, if $l\sigma \to^*_A q$ then $r\sigma \to^*_A q$.

Definition 10 ($\phi^{FP}_{\mathcal{R},A_S}$). *Let A_S be an STA $\langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta\rangle$ and let $\mathcal{R}$ be a left-linear TRS. We denote by $\phi^{FP}_{\mathcal{R},A_S}$ the formula defined as follows:*

$$\phi^{FP}_{\mathcal{R},A_S} \stackrel{def}{=} \bigwedge_{l\to r\in\mathcal{R}} \bigwedge_{X\in\mathcal{X}_\mathcal{Q}} \bigwedge_{(\alpha,\sigma)\in S^l_X} (\alpha \Rightarrow \bigvee_{(\beta,\_)\in S^{r\sigma}_X} \beta)$$

Example 6. Let A_S be the STA of the example 5 and let $\mathcal{R}$ be a TRS such that $\mathcal{R} = \{s(a) \to s(s(a))\}$. The formula $\phi^{FP}_{\mathcal{R},A_S}$ is then:

$$(\top \Rightarrow (X_{q_0} = X_{q_1} \vee X_{q_0} = X_{q_1})) \wedge (X_{q_0} = X_{q_1} \Rightarrow (X_{q_0} = X_{q_1} \vee X_{q_0} = X_{q_1}))$$

We state in the following proposition the use of $\phi^{FP}_{\mathcal{R},A_S}$.

Proposition 3. *Let A_S be an STA and $\mathcal{R}$ be a left-linear TRS. Let Q be a set of states and ι be an instantiation $\mathcal{X}_\mathcal{Q} \to Q$. Thus, $\iota \models \phi^{FP}_{\mathcal{R},A_S}$ iff A^ι_S is $\mathcal{R}$-closed.*

At this point, for a given STA A_S, we are able to formalize a fix-point condition. However, a particular fix-point is needed. Suppose that there exists an instantiation ι such that $\iota \models \phi^{FP}_{\mathcal{R},A_S}$. We recall that our goal is to find a fix-point automaton $A^\star$ such that $\mathcal{L}(A^\star) \cap Bad = \emptyset$. The following Definition proposes a formula characterizing the no-recognition of the whole set Bad by any instance of A^ι_S as soon as ι also satisfies this formula.

Definition 11 ($\phi^{Bad}_{A_S}$). *Let A_S be an STA $\langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta\rangle$ and Bad be a finite set of ground terms. We denote by $\phi^{Bad}_{A_S}$ the formula defined as follows:*

$$\phi^{Bad}_{A_S} \stackrel{def}{=} \bigwedge_{t \in Bad} \bigwedge_{X \in \mathcal{X}^f_\mathcal{Q}} \bigwedge_{(\alpha,\_) \in S^t_X} \neg\alpha.$$

Proposition 4. *Let A_S be a STA $\langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta\rangle$. Let Bad be a finite set of ground terms. Let Q be a set of states and ι be an instantiation $\mathcal{X}_\mathcal{Q} \mapsto Q$. Thus, $\iota \models \phi^{Bad}_{A_S}$ iff $\mathcal{L}(A^\iota_S) \cap Bad = \emptyset$.*

We are close to the claimed goal. Indeed, given a STA A_S, a TRS $\mathcal{R}$ and a set of terms Bad, we can deduce that for any instantiation ι satisfying $\phi^{Bad}_{A_S} \wedge \phi^{FP}_{\mathcal{R},A_S}$, $\mathcal{R}(\mathcal{L}(A^\iota_S)) \subseteq \mathcal{L}(A^\iota_S)$ and $\mathcal{L}(A^\iota_S) \cap Bad = \emptyset$. Is it sufficient to ensure that this fix-point is interesting for our input data i.e. A, $\mathcal{R}$ and Bad? In other words, can we deduce that $\mathcal{R}^*(\mathcal{L}(A)) \cap Bad = \emptyset$ from $\iota \models \phi^{Bad}_{A_S} \wedge \phi^{FP}_{\mathcal{R},A_S}$? Trivially the answer is no since no relation is specified between A_S and A. So, we define a compatibility notion between A_S and A leading to our expected result.

Definition 12 (A-compatibility). *Let A_S be an STA $\langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta_S\rangle$ and A be a TA $\langle Q, \mathcal{F}, Q_F, \Delta\rangle$. The STA A_S is said to be A-compatible iff these three criteria are satisfied: (1) $\{X_q | q \in Q\} \subseteq \mathcal{X}_\mathcal{Q}$; (2) $\{X_q | q \in Q_F\} \subseteq \mathcal{X}^f_\mathcal{Q}$; and (3) $\{f(X_{q_1}, \ldots, X_{q_n}) \to X_q | f(q_1, \ldots, q_n) \to q \in \Delta\} \subseteq \Delta_S$.*

The notion of A-compatibility presented above ensures that each instantiation of a STA A_S contains the language $\mathcal{L}(A)$.

Proposition 5. *Let A_S be a STA and A be a TA such that A_S is A-compatible. For any $\iota : \mathcal{X}_\mathcal{Q} \mapsto Q$, one has $\mathcal{L}(A) \subseteq \mathcal{L}(A^\iota_S)$.*

Consequently, our main result is that we are able to characterize a conclusive fix-point automaton, that can be found using a technique such as completion, by a single formula of $\mathcal{W}[\mathcal{X}_\mathcal{Q}]$.

Theorem 1. *Let A_S be a STA and A be a TA such that A_S is A-compatible. Let $\mathcal{R}$ be left-linear TRS and Bad be a finite set of ground terms. Let ι be an instantiation from $\mathcal{X}_\mathcal{Q}$ to Q. Thus,*

$$\iota \models \phi^{Bad}_{A_S} \wedge \phi^{FP}_{\mathcal{R},A_S} \text{ iff } A^\iota_S \text{ is } \mathcal{R}\text{-closed, } \mathcal{L}(A) \subseteq \mathcal{L}(A^\iota_S) \text{ and } \mathcal{L}(A^\iota_S) \cap Bad = \emptyset.$$

Another way to interpret this result is the following:

Theorem 2. *Let A_S be a STA and A be a TA such that A_S is A-compatible. Let $\mathcal{R}$ be left-linear TRS and Bad be a finite set of ground terms. Let ι be an instantiation from $\mathcal{X}_\mathcal{Q}$ to Q. Thus,*

$$\iota \models \phi^{Bad}_{A_S} \wedge \phi^{FP}_{\mathcal{R},A_S} \text{ implies that } \mathcal{R}^*(\mathcal{L}(A)) \subseteq \mathcal{L}(A^\iota_S) \text{ and } \mathcal{R}^*(\mathcal{L}(A)) \cap Bad = \emptyset.$$

6 Reachability Analysis via Logical Formula Solving

In this section we synthesize our contribution in the semi-algorithm Algorithm 6.1. Given a TRS $\mathcal{R}$, a tree automaton A and a set of goal terms Bad, Algorithm 6.1 searches an STA for which there exists an instantiation leading to a conclusive fix-point. It is indeed a semi-algorithm since a such conclusive fix-point may not exist (see [3]). In this case, the computation will not terminate. In a couple of words, the algorithm starts with the STA immediately obtained from A. If the whole formula has no solution then the current STA is improved by adding new symbolic transitions (using `Norm` defined in Algorithm 6.1). The whole formula is computed for the new STA and its satisfiability is checked using $hasNoValidSolution$. The process is iterated until finding a solution. We have used Mona [16] for solving formulae ($hasNoValidSolution$). Mona is a tool handling monadic second-order logic. Given a formula, Mona computes an automaton recognizing all of its solutions.

Algorithm 6.1 *Given a left-linear TRS $\mathcal{R}$, a tree automaton $A = \langle Q, \mathcal{F}, Q_F, \Delta\rangle$ and a set of goal terms Bad,* `areTermsUnreachable`$?(A, \mathcal{R}, Bad)$ *is defined as follows*

Variables
(Starting STA *)*
$A_S := \langle \{X_q \mid q \in Q\}, \mathcal{F}, \{X_q \mid q \in Q_F\}, \{f(X_{q_1}, \ldots, X_{q_n}) \to X_q | f(q_1, \ldots, q_n) \in \Delta\}\rangle$;
(Starting Formula *)*
$\phi := \phi^{FP}_{\mathcal{R},A_S} \wedge \phi^{Bad}_{A_S}$;
00 ***Begin***
01 ***While*** *(hasNoValidSolution(ϕ))* ***do***
02 ***Foreach*** $l \to r \in \mathcal{R}$ ***do***
03 $\sigma := \{x_1 \mapsto X_1, \ldots, x_n \mapsto X_n\}$ *where $X_1, \ldots, \mathcal{X}_n$ are new symbolic states*
04 $(\Delta', \mathcal{X}'_\mathcal{Q}) :=$ `Norm`$(r\sigma, X_{n+1})$ *where X_{n+1} is new symbolic state*
05 $A_S := \langle \mathcal{X}_\mathcal{Q} \cup \mathcal{X}'_\mathcal{Q} \cup \{X_{q_1}, \ldots, X_{q_n}\}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta' \cup \Delta\rangle$;
06 ***done;***
07 $\phi := \phi^{FP}_{\mathcal{R},A_S} \wedge \phi^{Bad}_{A_S}$;
08 ***EndWhile***

09 return true;
10 ***End***

The function `Norm` *used at Line 04 is defined as follows:*

$$\texttt{Norm}(t, X) = \begin{cases} (\emptyset, \emptyset), & \text{if } t \in \mathcal{X}_{\mathcal{Q}} \\ (\Delta', \mathcal{X}'_{\mathcal{Q}}) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

where X_i *is either new symbolic states or* t_i *if* $t_i \in \mathcal{X}_{\mathcal{Q}}$, $\Delta' = \{f(X_1, \ldots, X_n) \rightarrow X\} \cup \bigcup_{i=1}^{n}(\Delta^i)$, $\mathcal{X}'_{\mathcal{Q}} = \{X\} \bigcup_{i=1}^{n}(\mathcal{X}^i_{\mathcal{Q}})$ *and* $\texttt{Norm}(t_i, X_i) = (\Delta^i, \mathcal{X}^i_{\mathcal{Q}})$.

We present now a complete example. The idea is to show that all terms of the form $f(s^{(k)}(0)))$ reachable from $f(0)$ using the following TRS $\mathcal{R}_f = \{f(x) \rightarrow f(s(s(x)))\}$ are such that k is even. So, we define the parity test using three rules: $\mathcal{R}_{parity} = \{even(f(s(s(x)))) \rightarrow even(f(x)), even(f(0)) \rightarrow true, even(f(s(0))) \rightarrow false\}$. Thus, the given inputs are: $\mathcal{R} = \mathcal{R}_f \cup \mathcal{R}_{parity}$, $Bad = \{false\}$ and $A = \langle Q, \mathcal{F}, Q^f, \delta\rangle$ with $Q = \{q_0, q_1, q_2\}$, $\mathcal{F} = \{f : 1, s : 1, 0 : 0, even : 1, true : 0, false : 0\}$, $Q^f = \{q_2\}$ and $\delta = \{even(q_1) \rightarrow q_2, f(q_0) \rightarrow q_1, 0 \rightarrow q_0\}$. So, if a conclusive over-approximation can be found then it ensures that the set of terms reachable from $f(0)$ using the rule $f(x) \rightarrow f(s(s(x)))$ is necessarily of the form $f(s^k(0))$ with k an even integer.

So, the starting STA is such that $\Delta = \{even(X_{q_1}) \rightarrow X_{q_2}, 0 \rightarrow X_{q_0}, f(X_{q_0}) \rightarrow X_{q_1}\}$ and $\mathcal{X}_{\mathcal{Q}} = \{X_{q_0}, X_{q_1}, X_{q_2}\}$. Applying Definition 10, one obtains the following formula:

$\phi^{FP}_{\mathcal{R},A_S} =$	Rule involved
$\bigwedge_{Y \in \{X_{q_1}, X_{q_2}, X_{q_3}\}}(\top \wedge X_{q_1} = Y \Rightarrow \bot) \wedge$	$f(x) \rightarrow f(s(s(x)))$
$\bigwedge_{X,Y \in \{X_{q_1}, X_{q_2}, X_{q_3}\}}(\bot \Rightarrow \top \wedge X = Y) \wedge$	$even(f(s(s(x)))) \rightarrow even(f(x))$
$\bigwedge_{Y \in \{X_{q_1}, X_{q_2}, X_{q_3}\}}(\top \wedge X_{q_2} = Y \Rightarrow \bot) \wedge$	$even(f(0)) \rightarrow true$
$\bot \Rightarrow \bot$	$even(f(s(0))) \rightarrow false$

Clearly, $\phi^{FP}_{\mathcal{R},A_S}$ is unsatisfiable. Indeed, $\phi^{FP}_{\mathcal{R},A_S} = \phi_1 \wedge (\top \wedge X_{q_1} = X_{q_1} \Rightarrow \bot) \wedge \phi_2$ with $\phi_1, \phi_2 \in \mathcal{W}[\mathcal{X}_{\mathcal{Q}}]$. By simplifying the formula, one obtains that $\phi^{FP}_{\mathcal{R},A_S} = \phi_1 \wedge (\top \Rightarrow \bot) \wedge \phi_2 = \bot$. So, ϕ (at line 01 in Algorithm 6.1) is also unsatisfiable. Consequently, the STA A_S needs to be extended. In this example, four substitutions (one per rule following the order of the table above) $\sigma_1, \sigma_2, \sigma_3$ and σ_4 are created such that $\sigma_1 = \{x \mapsto X_{q_4}\}$, $\sigma_2 = \{x \mapsto X_{q_8}\}$ and $\sigma_3 = \sigma_4 = \emptyset$ where X_{q_4} and X_{q_8} are two new symbolic states.

Consequently, applying the substitutions $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ respectively on terms $f(s(s(x)))$, $even(f(x))$, $true$ and $false$, one has to normalize the terms $f(s(s(X_{q_4})))$, $even(f(X_{q_8}))$, $true$ and $false$. Let X_{q_3}, X_{q_7}, $X_{q_{10}}$ and $X_{q_{11}}$ be four new symbolic states, the normalization steps at Line 04 – $\texttt{Norm}(f(s(s(X_{q_4}))), X_{q_3})$, $\texttt{Norm}(even(\ f(X_{q_8})), X_{q_7})$, $\texttt{Norm}(true, X_{q_{10}})$ and $\texttt{Norm}(false, X_{q_{11}})$ – may produce the following STA: $A_S = \langle \mathcal{X}_{\mathcal{Q}}, \mathcal{F}, \mathcal{X}^f_{\mathcal{Q}}, \Delta\rangle$ $\mathcal{X}_{\mathcal{Q}} = \{X_{q_0}, \ldots, X_{q_{11}}\}$, $\mathcal{X}^f_{\mathcal{Q}} = \{X_{q_2}\}$ and $\Delta = \{true \rightarrow X_{q_{10}}, false \rightarrow X_{q_{11}}, s(X_{q_5}) \rightarrow X_{q_6}, s(X_{q_4}) \rightarrow X_{q_5}, 0 \rightarrow X_{q_0}, even(X_{q_9}) \rightarrow X_{q_7}, even(X_{q_1}) \rightarrow X_{q_2}, f(X_{q_8}) \rightarrow X_{q_9}, f(X_{q_6}) \rightarrow X_{q_3}, f(X_{q_0}) \rightarrow X_{q_1}\}$. Note that A_S is still A-compatible.

Following Definition 11, one obtains $\phi^{Bad}_{A_S} = X_{q_2} \neq X_{q_{11}}$.

Let us construct the instantiation ι from the solution returned by Mona[2]. We obtain: $\iota = \{X_{q_0} \mapsto q_0, X_{q_1} \mapsto q_0, X_{q_2} \mapsto q_0, X_{q_3} \mapsto q_1, X_{q_4} \mapsto q_1, X_{q_5} \mapsto q_0, X_{q_6} \mapsto q_1, X_{q_7} \mapsto q_1, X_{q_8} \mapsto q_1, X_{q_9} \mapsto q_1, X_{q_{10}} \mapsto q_0, X_{q_{11}} \mapsto q_1\}$. Applying ι on A_S, the resulting TA is: $A_S^\iota = \langle \{q_0, q_1\}, \mathcal{F}, \{q_0\}, \Delta^\iota \rangle$ with $\Delta^\iota = \{false \to q_1, 0 \to q_0, s(q_0) \to q_1, s(q_1) \to q_0, even(q_1) \to q_1, even(q_0) \to q_0, true \to q_0, f(q_1) \to q_1, f(q_0) \to q_0\}$.

This tree automaton is actually $\mathcal{R}$-closed. Indeed, concerning the rule $f(x) \to f(s(s(x)))$, note that $f(q_0)$ and $f(s(s(q_0)))$ can both be reduced to q_0. Similarly, $f(q_1)$ and $f(s(s(q_1)))$ can be reduced on q_1. For the rule $even(f(s(s(x)))) \to even(f(x))$, one has $even(f(s(s(q_0)))) \to^*_{A_S^\iota} q_0$ and $even(q_0) \to^*_{A_S^\iota} q_0$. Similarly, one has $even(f(s(s(q_1)))) \to^*_{A_S^\iota} q_1$ and $even(q_1) \to^*_{A_S^\iota} q_1$. Finally, for the rule $even(f(0)) \to true$ one has $even(f(0)) \to^*_{A_S^\iota} q_0$ and $true \to^*_{A_S^\iota} q_0$. Moreover, the term $false$ is not in $\mathcal{L}(A_S^\iota)$. Thus, A_s^ι is a conclusive fix-point automaton.

7 Conclusion

To summarize, given an STA A_S, a set of forbidden terms Bad, a TA A and a TRS $\mathcal{R}$, we have characterized by a logical formula what a conclusive fix-point in terms of reachability analysis is. Each solution of such a formula is an instantiation that can be applied on A_S. The automatically obtained automaton is an automaton that could have been obtained using a technique as in [9]. Such a technique requires a technical parameter (a set of equations or an approximation function) influential on the quality of the approximation computed. This parameter requires a certain expertise of the technique itself. For instance in [13], one has to define a set of equations whose goal is to define a finite number of equivalence classes of terms. A finite number of equivalence classes ensures the computation terminates. But, the crucial point remains in finding a conclusive approximation. Thus, the set of equation has to be defined very carefully. In [6], they used a set of predicates for defining a finite set of equivalence classes of terms. Once again, a highly specialized expertise in the technique itself is needed. Concerning ours, we generate an STA from the initial TA A and we are looking for solutions. If no solution is found then we are sure that there is no conclusive $\mathcal{R}$-closed automaton for the given A_S. So, we increase the size of the starting STA and so on.

In [10], the authors encode the tree automata completion by logic programs (Horn clauses). Consequently, they use several results obtained on static analysis of logic programs in order to compute precise approximations in the sense of static analysis. In the field of the tree automata completion, an approximation is precised enough as soon as this latter allows us to show the unreachability of a term or a set of terms. In this context, our proposition allows us to find only conclusive approximations, contrary to the ones obtained in [10]. Indeed, our approach is in some sense goal oriented while the one proposed in [10] check the reachability of a term only after having computed an approximation.

[2] The Mona program and thus the formula $\phi^{FP}_{\mathcal{R},A_S}$, can be downloaded at `http://www.univ-orleans.fr/lifo/Members/Yohan.Boichut/research/exampleMona.txt`

This work is a first step towards a verification technique based on formula solving. In the verification framework, it allows us to prove safety property. We claim that it is only a first step since specifications involving STA containing more than 20 variables or a bigger TRS are out of the Mona scope. Even if the formulas involved by such a specification present a certain regularity in their form, their size may be huge (in particular for $\phi^{FP}_{\mathcal{R},A_S}$ see Definition 10). We are also aware that the solving problem is not elementary, but we are working on dedicated solving techniques and search heuristics for handling huge formulae. We are also studying a symbolic technique *à la Mona*. First results of both techniques are very promising.

References

1. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
2. Y. Boichut, T. Genet, T. Jensen, and L. Leroux. Rewriting Approximations for Fast Prototyping of Static Analyzers. In *RTA*, LNCS 4533, pages 48–62, 2007.
3. Y. Boichut and P.-C. Héam. A theoretical limit for safety verification techniques with regular fix-point computations. *Inf. Process. Lett.*, 108(1):1–2, 2008.
4. Y. Boichut, P.-C. Héam, and O. Kouchnarenko. Approximation-based tree regular model-checking. *Nord. J. Comput.*, 14(3):216–241, 2008.
5. A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract regular tree model checking. *ENTCS*, 149(1):37–48, 2006.
6. A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract regular tree model checking. *ENTCS*, 149(1):37–48, 2006.
7. B. Boyer, T. Genet, and T. Jensen. Certifying a Tree Automata Completion Checker. In *IJCAR'08*, volume 5195 of *LNCS*. Springer, 2008.
8. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, and M. Tommasi. Tree automata techniques and applications. 2008.
9. G. Feuillade, T. Genet, and V. Viet Triem Tong. Reachability Analysis over Term Rewriting Systems. *Journal of Automated Reasonning*, 33 (3-4):341–383, 2004.
10. J. Gallagher and M. Rosendahl. Approximating term rewriting systems: a horn clause specification and its implementation. In *LPAR'08*, volume 5330. Springer, 2008.
11. T. Genet. Decidable approximations of sets of descendants and sets of normal forms. In *RTA*, volume 1379 of *LNCS*, pages 151–165. Springer-Verlag, 1998.
12. T. Genet and F. Klay. Rewriting for Cryptographic Protocol Verification. In *Proc. 17th CADE*, volume 1831 of *LNAI*. Springer-Verlag, 2000.
13. T. Genet and R. Rusu. Equational tree automata completion. *JSC*, 45:574–597, 2010.
14. R. Gilleron and S. Tison. Regular tree languages and rewrite systems. *Fundamenta Informaticae*, 24:157–175, 1995.
15. P. Gyenizse and S. Vágvölgyi. Linear Generalized Semi-Monadic Rewrite Systems Effectively Preserve Recognizability. *TCS*, 194(1-2):87–122, 1998.
16. J. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, B. Paige, T. Rauhe, and A. Sandholm. Mona: Monadic second-order logic in practice. In *TACAS '95*, volume 1019 of *LNCS*, 1995.
17. F. Jacquemard. Decidable approximations of term rewriting systems. In H. Ganzinger, editor, *Proc. 7th RTA Conf., New Brunswick (New Jersey, USA)*, pages 362–376. Springer-Verlag, 1996.

18. D. Monniaux. Abstracting Cryptographic Protocols with Tree Automata. In *Proc. 6th SAS, Venezia (Italy)*, 1999.
19. T. Takai. A Verification Technique Using Term Rewriting Systems and Abstract Interpretation. In *RTA*, volume 3091 of *LNCS*, pages 119–133. Springer, 2004.
20. T. Takai, Y. Kaji, and H. Seki. Right-linear finite-path overlapping term rewriting systems effectively preserve recognizability. In *Proc. 11th RTA Conf., Norwich (UK)*, volume 1833 of *LNCS*. Springer-Verlag, 2000.