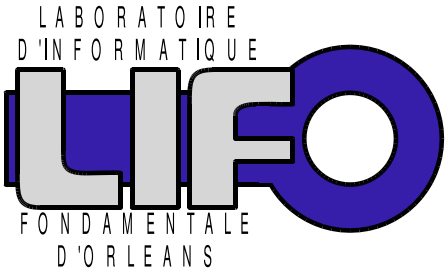


4 rue Léonard de Vinci
BP 6759
F-45067 Orléans Cedex 2
FRANCE
http://www.univ-orleans.fr/lifo

Rapport de Recherche

Characterizing Conclusive Approximations by Logical Formulae

Y. Boichut, T.-B.-H. Dao et V. Murat
LIFO, Université d'Orléans

Rapport n^o **RR-2011-04**

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. *Tree Regular Model Checking* is the name of a family of techniques for analyzing infinite-state systems in which states are represented by trees and sets of states by tree automata. We are interested in showing that a set of states *Bad* can not be reached from the set of initial states. Since the set of reachable is in general not computable, we focus on a computation of an over-approximation of the set of reachable states. 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 parameter.

1 Introduction

Infinite-state models are often used to avoid potentially artificial assumptions on data structures, e.g. artificial bound on the size of a stack or on the value of an integer variable. At the heart of most of the techniques that have been proposed for exploring infinite state spaces, is a symbolic representation that can finitely represent infinite sets of states. In this paper, we assume that states of the system are represented by trees and set of states by tree automata. The transition relation of the system is represented by a set of rewriting relations. It is known that this *Tree Regular Model Checking framework* (TRMC) [8, 1, 18] is expressive enough to describe classes of communication protocols [2], various C programs [7] with complex data structures, multi-thread programs [22, 20], as well as a wide range of cryptographic protocols [13, 15, 3] and any JAVA application [5].

In TRMC, the main objective is to decide whether a set of states can be reached from the initial states. This reduces to compute a tree automaton representing the set of states of the system. As we are dealing with infinite-state systems, the problem remains undecidable and only partial solutions can be proposed. In [6, 7], Bouajjani et al. proposed a CounterExample Guided Abstraction Refinement (CEGAR) methodology for TRMC, which they call *Abstract Tree Regular Model Checking* (ARMC). The idea behind ARMC consists of computing the automata obtained after successive applications of the rewriting relation

and then use techniques coming from the predicate abstraction area in order to over-approximate the set of reachable states. If the property holds on the abstraction, then it also holds on the concrete system. If a counter-example is found on the abstraction, then one has to check if it is indeed a counter-example to the real system. If not, this spurious counter-example must be used to refine the abstraction. The algorithm, which may not terminate, proceeds by successive abstraction/refinement until a decision can be taken.

Independently, Genet et al. [12, 11, 14] proposed *Completion* that is another technique to compute an over-approximation of the set of reachable states. The main difference with the work in [6] is that Completions use equations to compute the abstraction [14]. Equations gives a simple and formal semantics to abstractions on trees [19]. Completion has been applied to very complex case studies such as the verification of (industrial) cryptographic protocols [13, 15, 3] and Java bytecode applications [5].

For now, we are interested in showing that a system is secure. From a rewriting approximation point of view, it means that our goal is to compute an over-approximation which is a conclusive analysis, i.e. which is precise enough for showing that a set of "bad" terms is actually unreachable. Both of the techniques mentioned previously are instrumented either by equations or predicates for the computation of over-approximation. 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 characterized by a logical formula all the criteria of such a conclusive analysis performed with the technique proposed in [12, 11, 14]. 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, where 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 based on instantiations. 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. Finally, Section 6 concludes this paper with some first experiments on our approach using Mona [17] and a discussion on our future works on this topic.

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 [4, 10, 16]). 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 *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 $f(t_1, ..., t_n)\sigma = f(t_1\sigma, ..., t_n\sigma)$.

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 each variable of l occurs only once in l. 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 $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ 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 state. A transition is *normalized* when $c = f(q_1, \ldots, q_n)$, $f \in \mathcal{F}$ is of arity n, and $q_1, \ldots, q_n \in Q$.

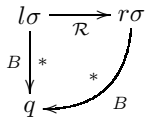
Definition 1 (Bottom-up nondeterministic finite tree automaton). *A bottom-up nondeterministic finite tree automaton (tree automaton for short) over the alphabet $\mathcal{F}$ is a tuple $A = \langle Q, \mathcal{F}, Q_F, \Delta \rangle$, where $Q_F \subseteq Q$ is the set of final states, Δ is a set 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 ([11, 21, 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.

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

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 $\mathcal{L}(B) \supseteq \mathcal{R}^*(\mathcal{L}(A))$[9].

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 $\mathcal{W}[\mathcal{X}_\mathcal{Q}]$ 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 $\mathcal{X}_Q = \{X_1, X_2, X_3\}$, then $(X_1 \neq X_2) \wedge ((X_1 = X_3) \vee (X_2 = X_3))$ 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 one of the forms $a \to X$ or $f(X_1, .., X_n) \to X$, where a is a constant, $f \in \mathcal{F}$ of arity n and $X, X_1, .., X_n \in \mathcal{X}_\mathcal{Q}$.*

Definition 6 (STA). *A 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.*

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 $\langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}_\mathcal{Q}^f, \Delta \rangle$ and ι be an instantiation $\mathcal{X}_\mathcal{Q} \to Q$. An instance of A_S by ι, denoted by A_S^ι, is a tree automaton $\langle Q^{A_S^\iota}, \mathcal{F}, Q_f^{A_S^\iota}, \Delta^{A_S^\iota} \rangle$ where:*

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

We define the relation $t \xrightarrow{\alpha}_{A_S} X$ relating that if an instantiation ι satisfies α then A_S^ι accepts the term t on the state $\iota(X)$.

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 $t \xrightarrow{\alpha_1 \wedge \cdots \wedge \alpha_n \wedge X=Y}_{A_S} X$*

The following proposition (proof in Appendix A) presents the characterization by a formula the acceptance of a term t from 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 $\texttt{Reco}(t, X) = \bigvee_{\{t \xrightarrow{\alpha}_{A_S} X\}} \alpha$. Thus, one has:*

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

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, 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 $t\sigma \xrightarrow{\alpha} X$. Suppose $\iota : \mathcal{X}_\mathcal{Q} \mapsto Q$ be an instantiation. Semantically, a solution (α, σ) means that, as soon as $\iota \models \alpha$, the substitution $\sigma \circ \iota$ is a solution the matching problem $t \trianglelefteq \iota(X)$ in the tree automaton A_S^ι.

Definition 9 (Matching Algorithm). *Let A_S be an STA $\langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta\rangle$. S^t_X is the set of the solutions of the matching problem $t \trianglelefteq X$, which is denoted $t \trianglelefteq X \vdash_{A_S} S^t_X$, if there exists a derivation of this statement using the rules:*

(Var) $\dfrac{}{x \trianglelefteq X \vdash_{A_S} \{(\top, \{x \mapsto X\})\}}$ $(x \in \mathcal{X})$

(Const) $\dfrac{}{a \trianglelefteq X \vdash_{A_S} \{(X = Y, \emptyset)\}}$ $(a \to Y \in \Delta)$

(SymbVar) $\dfrac{}{X \trianglelefteq Y \vdash_{A_S} \{(X = Y, \emptyset)\}}$ $(X, Y \in \mathcal{X}_\mathcal{Q})$

(Delta) $\dfrac{t_1 \trianglelefteq X_1 \vdash_{A_S} S_1 \quad \dots \quad t_n \trianglelefteq X_n \vdash_{A_S} S_n}{f(t_1, \dots, t_n) \trianglelefteq X \vdash_{A_S} \bigotimes^{X=Y}_{k=1\dots n}(S_k)}$ $(f(X_1, \dots, X_n) \to Y \in \Delta)$

where $\bigotimes^{\phi}_{k=1\dots n}(S_k) = \{(\phi, \emptyset) \oplus (\phi_1, \sigma_1) \oplus \dots \oplus (\phi_n, \sigma_n) \mid (\phi_i, \sigma_i) \in S_i\}$, *and* $(\phi, \sigma) \oplus (\phi', \sigma') = (\phi \wedge \phi', \sigma \cup \sigma')$.

The following proposition shows that this algorithm is sound and complete. Its proof is in Appendix B.

Proposition 2. *Let A_S be an STA $\langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta\rangle$, let $X \in \mathcal{X}_\mathcal{Q}$, let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}_Q)$ and $\sigma : \mathcal{V}ar(t) \to \mathcal{X}_\mathcal{Q}$. If $t \trianglelefteq X \vdash_{A_S} S^t_X$, then we have*

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

Example 2. 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 $\mathcal{L}(A^\star) \cap Bad = \emptyset$.

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 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 [11], 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 3. Let A_S be the STA of the example 2 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:

$$(\top \Rightarrow (X_{q0} = X_{q1} \vee X_{q0} = X_{q1})) \wedge (X_{q0} = X_{q1} \Rightarrow (X_{q0} = X_{q1} \vee X_{q0} = X_{q1}))$$

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

Proposition 3 (Proof in Appendix C). *Let A_S be an STA and $\mathcal{R}$ be left-linear a 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 satisfies also 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 (Proof in Appendix D). *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 $\mathcal{X}_\mathcal{Q} \to Q$. 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 (Proof in Appendix E). *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 that completion, by a single formula of $\mathcal{W}[\mathcal{X}_\mathcal{Q}]$.

Theorem 1 (Proof in Appendix F). *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 (Proof in Appendix G). *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 Experiments & Conclusion

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 $\mathcal{R}$ are such that k is even. The given inputs are: $\mathcal{R} = \{f(x) \to f(s(s(x))), even(f(s(s(x)))) \to even(f(x)), even(f(0)) \to true, even(f(s(0))) \to false\}$,$Bad = \{false\}$, $A = \langle Q, \mathcal{F}, Q^f, \delta\rangle$ and $A_S = \langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta\rangle$ with $Q = \{q_0, q_1, q_2\}$, $\mathcal{F} = \{f : 1, s : 1, 0 : 0, even : 1, true : 0, false : 0\}$, $Q^f = \{q_2\}$, $\delta = \{even(q_1) \to q_2, f(q_0) \to q_1, 0 \to q_0\}$, $\mathcal{X}_\mathcal{Q} = \{X_{q_0}, \ldots, X_{q_{11}}\}$, $\mathcal{X}^f_\mathcal{Q} = \{X_{q_2}\}$ and $\Delta = \{false \to X_{q_{11}}, 0 \to X_{q_0}, s(X_{q_5}) \to X_{q_6}, s(X_{q_4}) \to X_{q_5}, even(X_{q_9}) \to X_{q_7}, even(X_{q_1}) \to X_{q_2}, true \to X_{q_{10}}, f(X_{q_8}) \to X_{q_9}, f(X_{q_6}) \to X_{q_3}, f(X_{q_0}) \to X_{q_1}\}$. Note that A_S is A-compatible. So, one has to find a $\mathcal{R}-$closed automaton which does not contain $false$. Following Definition 11, one obtains $\phi^{Bad}_{A_S} = X_{q_2} \neq X_{q_{11}}$.

We have used Mona [17] for solving this formula. Mona is a tool handling monadic second-order logic. Given a formula, Mona computes an automaton recognizing all of its solutions. We have implemented an automatic generator of Mona programs from a specification containing an STA, a TRS and a set of bad terms.

For the Mona program corresponding to our example [2], Mona returns the following output:

```
MONA v1.4-13 for WS1S/WS2S
Copyright (C) 1997-2008 BRICS

PARSING
Time: 00:00:00.00

CODE GENERATION
DAG hits: 5549, nodes: 447
Time: 00:00:00.01

REDUCTION
Projections removed: 2 (of 3)
Products removed: 60 (of 433)
Other nodes removed: 1 (of 7)
DAG nodes after reduction: 380
Time: 00:00:00.00
```

[2] The Mona program and thus, 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`

```
AUTOMATON CONSTRUCTION
100% completed
Time: 00:00:00.22

Automaton has 764 states and 27075
BDD-nodes
...
Xq11 = {1}, Xq10 = {0}, Xq9 = {1}, Xq8 = {1}
Xq7 = {1}, Xq6 = {1}, Xq5 = {0}, Xq4 = {1}
Xq3 = {1}, Xq2 = {0}, Xq1 = {0}, Xq0 = {0}

Total time: 00:00:00.25
```

Let us construct the instantiation ι from the solution returned by Mona. We obtain: $\iota = \{X_{q0} \mapsto q_0, X_{q1} \mapsto q_0, X_{q2} \mapsto q_0, X_{q3} \mapsto q_1, X_{q4} \mapsto q_1, X_{q5} \mapsto q_0, X_{q6} \mapsto q_1, X_{q7} \mapsto q_1, X_{q8} \mapsto q_1, X_{q9} \mapsto q_1, X_{q10} \mapsto q_0, X_{q11} \mapsto q_1\}$. Applying ι on A_S, the resulting TA is: $A^\iota_S = \langle\{q_0, q_1\}, \mathcal{F}, \{q_0\}, \Delta^\iota\rangle$ 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 to 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 $even(f(s(s(q_0)))) \to^*_{A^\iota_S} q_0$ and $even(q_0) \to^*_{A_S} q_0$. Similarly, one has $even(f(s(s(q_1)))) \to^*_{A^\iota_S} q_1$ and $even(q_1) \to^*_{A^\iota_S} q_1$. Finally, for the rule $even(f(0)) \to true$ one has $even(f(0)) \to^*_{A^\iota_S} q_0$ and $true \to^*_{A^\iota_S} q_0$. Moreover, the term $false$ is not in $\mathcal{L}(A^\iota_S)$. Thus, A^ι_s is a conclusive fix-point automaton.

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 term 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 [11]. 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 [14], 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 to terminate. 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 a STA of a given size 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.

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* and first results are very promising. Coming next.

References

1. P. A. Abdulla, B. Jonsson, P. Mahata, and J. d'Orso. Regular tree model checking. In *CAV*, volume 2404 of *LNCS*, pages 555–568. Springer, 2002.
2. P. A. Abdulla, A. Legay, A. Rezine, and J. d'Orso. Simulation-based iteration of tree transducers. In *TACAS*, volume 3440 of *LNCS*, pages 30–40. Springer, 2005.
3. Avispa – a tool for Automated Validation of Internet Security Protocols. `http://www.avispa-project.org`.
4. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
5. 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.
6. A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract regular tree model checking. *ENTCS*, 149(1):37–48, 2006.
7. A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract regular tree model checking of complex dynamic data structures. In *SAS*, volume 4134 of *LNCS*, pages 52–70. Springer, 2006.
8. A. Bouajjani and T. Touili. Extrapolating tree transformations. In *CAV*, volume 2404 of *LNCS*, pages 539–554. Springer, 2002.
9. B. Boyer, T. Genet, and T. Jensen. Certifying a Tree Automata Completion Checker. In *IJCAR'08*, volume 5195 of *LNCS*. Springer, 2008.
10. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, and M. Tommasi. Tree automata techniques and applications. 2008.
11. 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.
12. 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.
13. T. Genet and F. Klay. Rewriting for Cryptographic Protocol Verification. In *Proc. 17th CADE*, volume 1831 of *LNAI*. Springer-Verlag, 2000.
14. T. Genet and R. Rusu. Equational tree automata completion. *JSC*, 45:574–597, 2010.
15. T. Genet, Y.-M. Tang-Talpin, and V. Viet Triem Tong. Verification of Copy Protection Cryptographic Protocol using Approximations of Term Rewriting Systems. In *WITS'2003*, 2003.
16. R. Gilleron and S. Tison. Regular tree languages and rewrite systems. *Fundamenta Informaticae*, 24:157–175, 1995.
17. 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.
18. Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In *CAV*, volume 1254 of *LNCS*, pages 424–435. Springer, 1997.
19. J. Meseguer, M. Palomino, and N. Martí-Oliet. Equational abstractions. *TCS*, 403:239–264, 2008.
20. G. Patin, M. Sighireanu, and T. Touili. Spade: Verification of multithreaded dynamic and recursive programs. In *CAV*, volume 4590 of *LNCS*, pages 254–257. Springer, 2007.
21. T. Takai. A Verification Technique Using Term Rewriting Systems and Abstract Interpretation. In *RTA*, volume 3091 of *LNCS*, pages 119–133. Springer, 2004.
22. T. Touili. *Analyse Symbolique de Systèmes infinis basée sur les automates: Application à la vérification de systèmes paramétrés*. PhD thesis, Paris 7, 2003.

Appendix

A Proof of Proposition 1

Let us recall Proposition 1.

Let $A_S = \langle \mathcal{X}_\mathcal{Q}, \mathcal{F}, \mathcal{X}^f_\mathcal{Q}, \Delta \rangle$ be a STA and ι be an instantiation. Let t be a term of $\mathcal{T}(\mathcal{F}, \mathcal{X}_Q)$ and X be a symbolic state of $\mathcal{X}_\mathcal{Q}$. Let $\texttt{Reco}(t, X)$ be a formula of $\mathcal{W}[\mathcal{X}_\mathcal{Q}]$ such that $\texttt{Reco} = \bigvee_{\{t \xrightarrow{\alpha}_{A_S} X\}} \alpha$. Thus, one has:

$$\iota \models \texttt{Reco}(t, X) \quad \textit{iff} \quad t\iota \rightarrow^*_{A^\iota_S} \iota(X).$$

Proof. First, let us show that $\iota \models \texttt{Reco}(t, X) \Rightarrow t\iota \rightarrow^*_{A^\iota_S} \iota(X)$. Note that $\texttt{Reco}(t, X)$ is a formula in DNF. Indeed, a formula α involved in a reduction such that $t \xrightarrow{\alpha}_{A_S} X$ is a conjunction of atoms $X = Y$ or $X \neq Y$ according to Definition 8. Since $\iota \models \texttt{Reco}(t, X)$, according to Definition 4, one can guess that there exists a subformula α_i such that $\texttt{Reco}(t, X) = \alpha_1 \vee \ldots \vee \alpha_i \vee \ldots \vee \alpha_n$, $t \xrightarrow{\alpha_i}_{A_S} X$ and $\iota \models \alpha_i$. Let us proceed by induction on the term t.

Base case:

- Suppose t is a constant: Since $t \xrightarrow{\alpha_i}_{A_S} X$, then there exists a transition of the form $t \rightarrow X' \in \Delta$ according to Definition 8. Consequently, $\alpha_i = (X = X')$. Since $\iota \mapsto \alpha_i$ and $\alpha_i = (X = X')$, $\iota(X) = \iota(X')$, so $t\iota \rightarrow_{A^\iota_S} \iota(X)$.
- Suppose t is a variable of $\mathcal{X}_\mathcal{Q}$: Since we have $t \xrightarrow{\alpha_i}_{A_S} X$, so $t = X$ and $t \xrightarrow{\tau}_{A_S} X$ according to Definition 8. Consequently, $\alpha_i = \tau$, and since $\iota \models \alpha_i$, we can deduce that $t\iota \rightarrow_{A^\iota_S} \iota(X)$.

Induction:

Suppose t be a term of the form $f(t_1, \ldots, t_n)$. By hypothesis, $t \xrightarrow{\alpha_i}_{A_S} X$. From Definition 8, one can deduce that there exist $f(Y_1, \ldots, Y_n) \rightarrow X' \in \Delta$ and $\gamma_1, \ldots \gamma_n$ such that:

1. for each t_j, one has $t_j \xrightarrow{\gamma_j} Y_j$ and
2. $\alpha_i = \gamma_1 \wedge \ldots \wedge \gamma_n$.

By induction hypothesis, for each t_j, Y_j and $\texttt{Reco}(t_j, X)$, one has

$$\iota \models \texttt{Reco}(t_j, Y_j) \Leftrightarrow t_j\iota \rightarrow^*_{A^\iota_S} \iota(Y_j), \tag{1}$$

with $\texttt{Reco}(t_j, Y_j) = \bot \vee \bigvee_{t_j \xrightarrow{\phi} Y_j} \phi$. Since $t_j \xrightarrow{gamma_i} Y_j$ and $\iota \models \gamma_j$, one can trivially deduce that $\iota \models \texttt{Reco}(t_j, Y_j)$. So, applying (1), one has that for each t_j, $t_j\iota \rightarrow^*_{A^\iota_S} \iota(Y_j)$. According to Definition 7, one can deduce that

$$f(\iota(Y_1), \ldots, \iota(Y_n)) \rightarrow \iota(X') \text{ is a transition of } A^\iota_S. \tag{2}$$

Consequently, one can deduce that $t\iota = f(t_1\iota, \ldots, t_1\iota) \rightarrow^*_{A^\iota_S} f(\iota(Y_1), \ldots, \iota(Y_n))$ from $t_j\iota \rightarrow^*_{A^\iota_S} \iota(Y_j)$. Moreover, using transition (2), one has that $t\iota \rightarrow^*_{A^\iota_S} \iota(X'_i)$. Since $\iota(X'_i) = \iota(X)$, one trivially deduce that $t\iota \rightarrow^*_{A^\iota_S} \iota(X)$.

So, let us show that $\iota \models \texttt{Reco}(t, X) \Leftarrow t\iota \rightarrow^*_{A^\iota_S} \iota(X)$. Let us proceed by induction on the structure of t.

Base case:

- Suppose t is a constant: By hypothesis, $t\iota \rightarrow^*_{A^\iota_S} \iota(X)$. Note that, in this case, $t\iota = t$. So according to Definition 7, there exists a transition $t \rightarrow X' \in \Delta$ such that $\iota(X) = \iota(X')$. Consequently, from Definition 4, one can deduce that $t \xrightarrow{X=X'}_{A_S} X$ and $\iota \models X = X'$. Thus, one has $\iota \models \texttt{Reco}(t, X)$.
- $t\iota$ is a state: Since there is no epsilon transition $q \rightarrow q'$ in the considered tree automata, one has necessarily $t\iota = \iota(X)$. According to Definition 8 $X' \xrightarrow{\alpha}_{A_S} X$ if $X = X$ and $\alpha = \top$. Consequently, one can deduce that $\iota \models \texttt{Reco}(t, X)$.

Induction:

Suppose t be a term of the form $f(t_1, \ldots, t_n)$: By hypothesis, $f(t_1, \ldots, t_n)\iota \rightarrow^*_{A^\iota_S} \iota(X)$. So, there exists $X_1, \ldots, X_n, X' \in \mathcal{X}_\mathcal{Q}$ such that

$$t_i\iota \rightarrow^*_{A^\iota_S} \iota(X_i),\ \iota(X') = \iota(X) \textit{ and } f(X1, \ldots, X_n) \rightarrow X' \in \Delta. \tag{3}$$

For each t_i, one can apply the induction hypothesis. Thus, one obtains that $\iota \models \texttt{Reco}(t_k, X_k)$, for $k = 1, \ldots, n$. Let us focus on the induction hypothesis: for each t_k: $\iota \models \texttt{Reco}(t_k, X_k)$ iff $t_k\iota \rightarrow^*_{A^\iota_S} \iota(X_k)$ with $\texttt{Reco}(t_k, X_k) = \vee \bigvee_{\{t \xrightarrow{\gamma}_{A_S} X_k\}} \gamma$. Since $\iota \models \texttt{Reco}(t_k, X_k)$, thus there exists a subformula of $\texttt{Reco}(t_k, X_k)$, i.e. γ_k, such that $\iota \models \gamma_k$. Consequently, $f(t_1, \ldots, t_n) \xrightarrow{\gamma_1}_{A_S} f(X_1, t_2, \ldots, t_n)$. By iterating this process, one obtains that $f(t_1, \ldots, t_n) \xrightarrow{\gamma_1 \wedge \ldots \wedge \gamma_n}_{A_S} f(X_1, \ldots, X_n)$. Since $f(X1, \ldots, X_n) \rightarrow X' \in \Delta$, one obtains $t \xrightarrow{\alpha}_{A_S} X$ with $\alpha = \gamma_1 \wedge \ldots \wedge \gamma_n \wedge X' = X$. According to (3) and Definition 4, one obtains $\iota \models \alpha$. Concluding the proof.

B Proof of Proposition 2

Let us recall Proposition 2.

Let A_S be a STA, X one of its symbolic states, $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ a term and σ a $\mathcal{X}_\mathcal{Q}$-substitution with a domain range-restricted to $\mathcal{V}(t)$. If the set S^t_X is solution of the matching problem $t \trianglelefteq X$, then we have

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

Let us recall that S^t_X represents the set of solutions of the matching problem $t \trianglelefteq X$.

Proof. Let $\alpha \in \mathcal{W}[\mathcal{X}_\mathcal{Q}]$ and $\sigma : \mathcal{X} \mapsto \mathcal{X}_\mathcal{Q}$ such that $t\sigma \xrightarrow{\alpha}_{A_S} X$. Let us proceed by case analysis.

Base case:

- $t\sigma$ is a constant: By hypothesis, $t\sigma \xrightarrow{\alpha}_{A_S} X$. Since t is a constant, $\mathcal{V}ar(t) = \emptyset$ and so is σ. So, there exists a transition $t \to X' \in \Delta$ and $\alpha = X' = X$. Applying Rule *constant* with the transition $t \to X'$, one has $(X' = X, \emptyset) \in S^t_X$.
- $t\sigma$ is a symbolic state: According to Definition 8, $t\sigma \xrightarrow{\alpha}_{A_S} X$ only if $t\sigma = X$ and $\alpha = \top$. Moreover, if $t\sigma \in \mathcal{X}_\mathcal{Q}$ then $t \in \mathcal{X}$. Consequently, one can define the substitution $\sigma : \{t \mapsto X\}$. Let us compute the set of solutions S^t_X. We recall that $t \in \mathcal{X}$. According to Rule *Var*, one obtains that $(\top, \{(t, X)\}) \in S^t_X$.

Induction case:

Suppose t be a term of the form $f(t_1, \ldots, t_n)$. Since t is linear, one can decompose σ in n substitutions $\sigma_1 : \mathcal{V}ar(t_1) \mapsto \mathcal{X}_\mathcal{Q}$, $\ldots$, $\sigma_n : \mathcal{V}ar(t_n) \mapsto \mathcal{X}_\mathcal{Q}$. Thus, $\sigma(x) = \sigma_i(x)$ if $x \in \mathcal{V}ar(t_i)$. In other words, $\sigma = \sigma_1 \cup \ldots \cup \sigma_n$. So, $t\sigma = f(t_1\sigma_1, \ldots, t_n\sigma_n)$. By hypothesis, $t\sigma \xrightarrow{\alpha}_{A_S} X$. So, using Definition 8, one can deduce that there exists $\alpha_1, \ldots, \alpha_n \in \mathcal{W}[\mathcal{X}_\mathcal{Q}]$ and a transition $f(Y_1, \ldots, Y_n) \to Y \in \Delta$ such that $\alpha = \alpha_1 \wedge \ldots \alpha_n \wedge X = Y$ and $t_i \xrightarrow{\alpha_i}_{A_S} Y_i$. So applying the induction hypothesis on each t_i, one obtains that $(\alpha_i, \sigma_i) \in S^{t_i}_{Y_i}$. The rule *Delta* is applied to all the premises $t_i \trianglelefteq Y_i \vdash_{A_s} S^{t_i}_{Y_i}$ for the transition $f(Y_1, \ldots, Y_n) \to Y$. So, we obtain a set $S = \bigotimes_{k=1\ldots n}^{X=Y} (S^{t_k}_{Y_k})$. Unfolding the definition of $\bigotimes$ and since $(\alpha_i, \sigma_i) \in S^{t_i}_{Y_i}$, S^t_X contains $((X = Y, \emptyset) \oplus (\alpha_1, \sigma_1) \oplus \ldots \oplus (\alpha_n, \sigma_n))$. By definition of $\oplus$, one obtains that $(X = Y \wedge \alpha_1 \wedge \ldots \wedge \alpha_n, \sigma_1 \cup \ldots \cup \sigma_n)$. Consequently, $(\alpha, \sigma) \in S^t_X$. Concluding the proof.

C Proof of Proposition 3

Let us recall Proposition 3.

Let A_S and $\mathcal{R}$ be respectively a STA and a TRS. Let ι be an instantiation from $\mathcal{X}_\mathcal{Q}$ to Q. Thus,

$$\iota \models \phi^{FP}_{\mathcal{R},A_S} \text{ iff } A^\iota_S \text{ is } \mathcal{R} - \text{closed}.$$

For showing it, we need of the intermediary Lemma given right below.

Lemma 1. *Let t be a term of $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and A_S be a STA. Let X be a symbol state of A_S. Let S^t_X be the set of solution of the matching problem $t \trianglelefteq X$. Let σ be a substitution such that $\sigma : \mathcal{V}ar(t) \mapsto \mathcal{X}_\mathcal{Q}$ and $S^{t\sigma}_X$ be the solution set the matching problem $t\sigma \trianglelefteq X$. Thus, one has: For any $(\beta, \emptyset) \in S^{t\sigma}_X$, there exists $(\alpha, \mu) \in S^t_X$ such that*

$$\beta = \alpha \wedge \bigwedge_{x \in \mathcal{V}ar(t)} (\mu(x) = \sigma(x)).$$

Proof. A direct consequence of Rule *SymbVar* of Definition 9.

Let us finally show Proposition 3.

Proof. Let us first show that if $\iota \models \phi^{FP}_{\mathcal{R},A_S}$ then A^{ι}_S is $\mathcal{R}-$closed. According to Definition 10, $\phi^{FP}_{\mathcal{R},A_S} = \bigwedge_{l\to r\in\mathcal{R}}(\bigwedge_{\forall X\in\mathcal{X}_\mathcal{Q},(\alpha,\sigma)\in S^l_X}(\alpha \Rightarrow \bigvee_{\forall(\beta,\_)\in S^{r\sigma}_X}(\beta)))$, where S^l_X and $S^{r\sigma}_X$ are respectively the solution sets of the matching problems $l \trianglelefteq X$ and $r\sigma \trianglelefteq X$. According to Definition 4, $\iota \models \bigwedge_{l\to r\in\mathcal{R}}(\bigwedge_{\forall X\in\mathcal{X}_\mathcal{Q},(\alpha,\sigma)\in S^l_X}(\alpha \Rightarrow \bigvee_{\forall(\beta,\_)\in S^{r\sigma}_X}(\beta)))$ can be rewritten in $\forall l \to r \in \mathcal{R}$, $\forall X \in \mathcal{X}_\mathcal{Q}$ and $\forall(\alpha,\sigma) \in S^l_X$ $\iota \models (\alpha \Rightarrow \bigvee_{\forall(beta,\_)\in S^{r\sigma}_X}(\beta)))$. According to the definition of $\models$, one can deduce that $\iota \models (\alpha \Rightarrow \bigvee_{\forall(\beta,\_)\in S^{r\sigma}_X}(\beta)))$ iff $\iota \models \neg\alpha$ or $\iota \models \bigvee_{\forall(\beta,\_)\in S^{r\sigma}_X}(\beta)))$. Let us proceed by case analysis:

- $\iota \models \alpha$ and $\iota \models \bigvee_{\forall(\beta,\_)\in S^{r\sigma}_X}(\beta)))$: Recall that $(\alpha,\sigma) \in S^l_X$. According to Proposition 2, one deduce that $l\sigma \xrightarrow{\alpha}_{A_S} X$. Consequently, if $\iota \models \alpha$ then $\iota \models \bigvee_{\{t\xrightarrow{\alpha'}_{A_S}X\}} \alpha' = \texttt{Reco}(t, X)$. Applying Proposition 1, one can deduce that

$$l\sigma \circ \iota \to^*_{A^{\iota}_S} \iota(X). \tag{4}$$

  By hypothesis, $\iota \models \bigvee_{\forall(\beta,\_)\in S^{r\sigma}_X}(\beta)))$. Consequently, there exists $(\gamma,\_) \in S^{r\sigma}_X$ such that $\iota \models \gamma$. According to Lemma 1, one can deduce that there exists $(\alpha',\sigma') \in S^r_X$ such that $\gamma = \alpha' \wedge \bigwedge_{x\in\mathcal{V}ar(r)}(\sigma(x) = \sigma'(x))$. Moreover, one can deduce that $\iota \models \alpha' \wedge \bigwedge_{x\in\mathcal{V}ar(r)}(\sigma(x) = \sigma'(x))$. According to Proposition 2, $r\sigma' \circ \iota \to^*_{A^{\iota}_S} \iota(X)$. Since $\iota \models \alpha' \wedge \bigwedge_{x\in\mathcal{V}ar(r)}(\sigma(x) = \sigma'(x))$, one can deduce that $r\sigma \circ \iota = r\sigma' \circ \iota$. So,

$$r\sigma \circ \iota \to^*_{A^{\iota}_S} \iota(X). \tag{5}$$

  Thus, if one has (4) then one has also (5) which characterizes a $\mathcal{R}-$closed automaton.
- $\iota \not\models \alpha$: Since α represents a particular reduction of l to the symbolic state X, $\iota \models \neg\alpha$ means that the reduction involved is not possible in the tree automaton A^{ι}_S. Since this reduction is not possible, we do not have to check whether $r\sigma \circ \iota \to^*_{A^{\iota}_S} \iota(X)$ or not in order to ensure that A^{ι}_S is $\mathcal{R}-$closed.

Now, let us show that if A^{ι}_S is $\mathcal{R}-$closed then $\iota \models \phi^{FP}_{\mathcal{R},A_S}$. According to Definition 2, A^{ι}_S is $\mathcal{R}-$closed if for each rule $l \to r \in \mathcal{R}$, for any substitution $\mu : \mathcal{X} \mapsto Q$, $l\mu$ is recognized by A^{ι}_S into state $\iota(X)$ then so is $r\mu$. According to Proposition 1, one can deduce that there exists $\sigma : \mathcal{X} \mapsto \mathcal{X}_\mathcal{Q}$ such that $\iota \models \texttt{Reco}(l\sigma, X)$ and $\mu = \sigma \circ \iota$. Since $\iota \models \texttt{Reco}(l\sigma, X)$, one can deduce that there exists α such that $l\sigma \xrightarrow{\alpha}_{A_S} X$ and $\iota \models \alpha$. According to Proposition 1, one can deduce that there exists $\sigma' : \mathcal{X} \mapsto \mathcal{X}_\mathcal{Q}$ such that $\iota \models \texttt{Reco}(r\sigma', X)$. Since $\iota \models \texttt{Reco}(r\sigma', X)$, one can deduce that there exists β such that $r\sigma' \xrightarrow{\beta}_{A_S} X$ and $\iota \models \beta$. Note that $(\beta,\sigma') \in S^r_X$ where S^r_X denotes the set of solutions of the

matching problem $r \trianglelefteq X$. Now, let us consider $r\sigma$. One can construct γ such that $\gamma = \beta \wedge \bigwedge_{x \in \mathcal{V}ar(r)}(\sigma(x) = \sigma'(x))$ and $(\gamma, \emptyset) \in S_X^{r\sigma}$ (considering $S_X^{r\sigma}$ as the set of solutions of the matching problem $r\sigma \trianglelefteq X$). By construction, $\sigma \circ \iota(x) = \sigma' \circ \iota(x)$, for any $x \in \mathcal{V}ar(r)$. Consequently, one can deduce that $\iota \models \gamma$. Thus, ι satisfies also the formula $\bigvee_{\forall(\beta,\_) \in S_X^{r\sigma}}(\beta)))$. Iterating this process for any substitution $\sigma : \mathcal{X} \mapsto Q$, for any rule $l \to r$ and for any state of A_S^ι, we obtain that $\iota \models \phi^{FP}_{\mathcal{R},A_S}$. Concluding the proof.

D Proof of Proposition 4

Let us recall Proposition 4.

Let A_S and $\mathcal{R}$ be respectively a STA whose set of final symbolic states is $\mathcal{X}^f_\mathcal{Q}$ and a TRS. Let Bad be a set of ground terms. Let ι be an instantiation from $\mathcal{X}_\mathcal{Q}$ to Q. Thus,

$$\iota \models \phi^{Bad}_{A_S} \text{ iff } \mathcal{L}(A_S^\iota) \cap Bad = \emptyset.$$

Proof. According to Definition 11, $\phi^{Bad}_{A_S} = \bigwedge_{\forall t \in Bad}(\bigwedge_{\forall X \in \mathcal{X}^f_\mathcal{Q},(\alpha,\_) \in S^t_X} \neg\alpha)$.
So $\iota \models \phi^{Bad}_{A_S}$
$\Leftrightarrow \iota \models \bigwedge_{\forall t \in Bad}(\bigwedge_{\forall X \in \mathcal{X}^f_\mathcal{Q},(\alpha,\_) \in S^t_X} \neg\alpha)$
$\Leftrightarrow \forall t \in Bad, \iota \models \bigwedge_{\forall X \in \mathcal{X}^f_\mathcal{Q},(\alpha,\_) \in S^t_X} \neg\alpha$
$\Leftrightarrow \forall t \in Bad, \iota \models \neg \bigvee_{\forall X \in \mathcal{X}^f_\mathcal{Q},(\alpha,\_) \in S^t_X} \alpha$.
According to Proposition 2, $(\alpha, \_) \in S^t_X \Leftrightarrow t \xrightarrow{\alpha}_{A_S} X$,
so $\forall t \in Bad, \iota \models \neg \bigvee_{\forall X \in \mathcal{X}^f_\mathcal{Q},(\alpha,\_) \in S^t_X} \alpha \Leftrightarrow \forall t \in Bad, \iota \models \neg \bigvee_{\forall X \in \mathcal{X}^f_\mathcal{Q}, t \xrightarrow{\alpha}_{A_S} X} \alpha$.
And according to Proposition 1, this is equivalent to $t\iota \twoheadrightarrow^*_{A_S^\iota} \iota(X)$, for all $t \in Bad$ and for all $X \in \mathcal{X}^f_\mathcal{Q}$. So, for all X, $\iota(X)$ is a final state of A_S^ι. So none of the bad terms is recognized by the instantiated automaton A_S^ι. Thus we can deduce that it is equivalent to $\mathcal{L}(A_S^\iota) \cap Bad = \emptyset$.

E Proof of Proposition 5

Let us recall 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 $\mathcal{L}(A) \subseteq \mathcal{L}(A_S^\iota)$.

Proof. According to Definition 12, each state q is transformed into a symbolic state X_q. So, for any $t \in \mathcal{L}(A, q_f)$ with q_f a final state of A, one has $t \xrightarrow{top} X_{q_f}$. Thus, for any instantiation ι, $\iota \models \texttt{Reco}(t, X_{q_f})$. Consequently, applying Proposition 1, $t \in \mathcal{L}(A_S^i)$.

F Proof of Theorem 1

Let us recall Theorem 1.

Let A_S be a STA and A be a TA such that A_S is $A-$compatible. Let ι be an instantiation from $\mathcal{X}_\mathcal{Q}$ to Q. Thus,

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

Proof. As A_S is $A-$compatible and $\iota \models \phi^{Bad}_{A_S}(Q\mathcal{X}) \wedge \phi^{FP}_{\mathcal{R},A_S}$ iff $\iota \models \phi^{Bad}_{A_S} \wedge \iota \models \phi^{FP}_{\mathcal{R},A_S}$, Theorem 1 is a direct consequence of propositions 3, 4 and 5.

G Proof of Theorem 2

Let us recall Theorem 2.

Let A_S be a STA and A be a TA such that A_S is $A-$compatible. Let ι be an instantiation from $\mathcal{X}_\mathcal{Q}$ to Q.Thus,

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

Proof. According to Theorem 1,

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

So, if A^ι_S is $\mathcal{R}-$ closed and $\mathcal{L}(A^\iota_S) \supseteq \mathcal{L}(A)$, then $\mathcal{L}(A^\iota_S) \supseteq \mathcal{R}^*(\mathcal{L}(A))$ and $\mathcal{R}^*(\mathcal{L}(A)) \cap Bad = \emptyset$.