TAGED Approximations for Veriying Temporal Patterns*

R. Courbis[1], P.-C. Héam[1,2], and O. Kouchnarenko[1]

INRIA/CASSIS
LIFC/University of Franche-Comté
16 route de Gray
F-25030 Besançon Cedex
{rcourbis,okouchnarenko}@lifc.univ-fcomte.fr

LSV CNRS/INRIA/ENS Cachan
61 av. du Président Wilson
F-94235 Cachan Cedex
pcheam@lsv.ens-cachan.fr

Abstract. This paper investigates the use of tree automata with global equalities and disequalities (TAGED for short) in reachability analysis over term rewriting systems (TRSs). The reachability problem being in general undecidable on non terminating TRSs, we provide TAGED-based construction, and then design approximation-based semi-decision procedures to model-check useful temporal patterns on infinite state rewriting graphs. To show that the above TAGED-based construction can be effectively carried out, complexity analysis for rewriting TAGED-definable languages is given.

1 Introduction

Model-checking techniques [24,23] are commonplace in computer aided verification. Model checking refers to the following problem: given a desired property, expressed as a temporal logic formula φ, and a structure M with initial state s, decide if $M, s \models \varphi$. The use of model-checking techniques and tools is however limited to systems whose state space can be finitely represented, because of the state explosion problem.

Recently, reachability analysis turned out to be a very efficient verification technique for proving properties on infinite systems modeled by term rewriting systems (TRSs for short). In the rewriting theory, the reachability problem is the following: given a TRS $\mathcal{R}$ and two terms s and t, can we decide whether $s \to_{\mathcal{R}}^* t$ or not? This problem, which can easily be solved on strongly terminating TRSs, is undecidable on non terminating TRSs. However, on the one hand, there exist several syntactic classes of TRSs for which this problem becomes decidable [15,19,31]. On the other hand, in addition to classical proof tools of rewriting, given a set $\mathcal{E} \subseteq \mathcal{T}(\mathcal{F})$ of initial terms, provided that $s \in \mathcal{E}$, one can prove $s \not\to_{\mathcal{R}}^* t$ by using over-approximations of $\mathcal{R}^*(\mathcal{E})$ [20,15] and proving that t does not belong to these approximations. Recently, the verification of temporal properties of systems modeled by TRSs has been investigated [14,27,26]. To apply these very interesting and promising theoretical results to applications

* This work has been funded by the French ANR-06-SETI-014 RAVAJ project.

in practice, the authors look for finite abstractions to model-check temporal properties, and use proof theory methods. Unlike these works, we develop an approximation and tree automata based approach, which can provide a fully automatic verification framework.

Motivations. Recently, some of the most successful experiments using reachability analysis were done on cryptographic protocols, [17,7], and on Java byte code programs [6]. Presently, Java MIDLet applications security properties are verified through $\mathcal{R}^*(\mathcal{E})$ over-approximations [1]. To this end, following works on CEGAR [8], we developed in [5] over-approximations refinement depending on a security property to be verified. To go further, we are interested in verifying temporal properties.

Contributions. The main question is: *Is it possible to exploit rewriting approximations for model-checking temporal properties on infinite state rewriting graphs?* This paper addresses this question and offers a solution for three useful patterns of temporal properties. This solution automatically attempts to show that $M, s \models \varphi$ by exploiting TAGED approximations over M, without building M.

More precisely, the present paper makes the following contributions: Given an LTL formula (of a certain pattern) to be evaluated over M, the *first contribution* is the feasibility of a systematic translation of this formula into a language rewriting equality to be checked. Language equalities being undecidable in general, the *second contribution* is approximation-based semi-decision procedures to model-check temporal properties of three useful patterns coming from static analysis domain and having practical applications. This contribution is obtained using the recent TAGED model (Tree Automata with Global Equality and Disequality Constraints) in [16].

Structure of the paper. Section 2 introduces preliminary notions on TRSs, tree-automata, and rewriting-based linear temporal logic. Section 3 explains the interest of the proposed approach via three temporal property patterns and relates them to language rewriting equations. The main contribution in Section 4 concerns rewriting-based (semi-)decision procedures and complexity analysis for rewriting related TAGED-definable languages. Then, semi-algorithms, including approximation steps are given. Finally, Section 5 concludes and sums up related works. Appendix contains omitted proofs and examples to illustrate theoretical underpinnings.

2 Preliminaries

2.1 Terms, TRSs and Tree Automata

Comprehensive surveys can be found in [12,2] for TRSs, in [10,18] for tree automata and tree language theory, and in [16] for TAGEDs.

[1] in the framework of the French ANR Ravaj project.

Terms and TRSs. Let $\mathcal{F}$ be a finite set of symbols, associated with an arity function $ar : \mathcal{F} \to \mathbb{N}$, and let $\mathcal{X}$ be a countable set of variables. $\mathcal{T}(\mathcal{F}, \mathcal{X})$ denotes the set of terms, and $\mathcal{T}(\mathcal{F})$ denotes the set of ground terms (terms without variables). The set of variables of a term t is denoted by $\mathcal{V}ar(t)$. A substitution is a function σ from $\mathcal{X}$ into $\mathcal{T}(\mathcal{F}, \mathcal{X})$, which can be extended uniquely to an endomorphism of $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A position p for a term t is a word over $\mathbb{N}$. The empty sequence ϵ denotes the top-most position. The set $\mathcal{P}os(t)$ of positions of a term t is inductively defined by $\mathcal{P}os(t) = \{\epsilon\}$ if $t \in \mathcal{X}$ and by $\mathcal{P}os(f(t_1, \ldots, t_n)) = \{\epsilon\} \cup \{i.p \mid 1 \leq i \leq n \text{ and } p \in \mathcal{P}os(t_i)\}$ otherwise. If $p \in \mathcal{P}os(t)$, then $t|_p$ denotes the subterm of t at position p and $t[s]_p$ denotes the term obtained by replacement of the subterm $t|_p$ at position p by the term s. We also denote by $t(p)$ the symbol occurring in t at position p. Given a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, we denote $\mathcal{P}os_A(t) \subseteq \mathcal{P}os(t)$ the set of positions of t such that $\mathcal{P}os_A(t) = \{p \in \mathcal{P}os(t) \mid t(p) \in A\}$. Thus $\mathcal{P}os_{\mathcal{F}}(t)$ is the set of functional positions of t. A TRS $\mathcal{R}$ is a set of *rewrite rules* $l \to r$, where $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $l \notin \mathcal{X}$. A rewrite rule $l \to r$ is *left-linear* (resp. right-linear) if each variable of l (resp. r) occurs only once within l (resp. r). A TRS $\mathcal{R}$ is left-linear (resp. right-linear) if every rewrite rule $l \to r$ of $\mathcal{R}$ is left-linear (resp. right-linear). A TRS $\mathcal{R}$ is linear if it is right and left-linear. The TRS $\mathcal{R}$ induces a rewriting relation $\to_{\mathcal{R}}$ on terms whose reflexive transitive closure is written $\to^{\star}_{\mathcal{R}}$. The set of $\mathcal{R}$-descendants of a set of ground terms $\mathcal{E}$ is $\mathcal{R}^*(\mathcal{E}) = \{t \in \mathcal{T}(\mathcal{F}) \mid \exists s \in \mathcal{E} \text{ s.t. } s \to^{\star}_{\mathcal{R}} t\}$. Symmetrically, the set of $\mathcal{R}$-ancestors of a set of ground terms $\mathcal{E}$ is $\mathcal{R}^{-1\star}(\mathcal{E}) = \{s \in \mathcal{T}(\mathcal{F}) \mid \exists t \in \mathcal{E} \text{ s.t. } s \to^{\star}_{\mathcal{R}} t\}$.

Note that $\mathcal{R}^*(\mathcal{E})$ is possibly infinite: $\mathcal{R}$ may not terminate and/or $\mathcal{E}$ may be infinite. In general, the set $\mathcal{R}^*(\mathcal{E})$ is not computable [18]. However, it is possible to over-approximate it [15] using completion procedure over tree automata, i.e. a finite representation of infinite (but regular) sets of terms.

Tree automata. Let $\mathcal{Q}$ be a finite set of symbols, of arity 0, called *states* such that $\mathcal{Q} \cap \mathcal{F} = \emptyset$. $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is called the set of *configurations*. A *transition* is a rewrite rule $c \to q$, where $c \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is of the form $c = f(q_1, \ldots, q_n)$, $f \in \mathcal{F}$, $ar(f) = n$, and $q_1, \ldots, q_n \in \mathcal{Q}$.

A *bottom-up non-deterministic finite tree automaton* (tree automaton for short) over $\mathcal{F}$ is a 3-tuple $\mathcal{A} = (\mathcal{Q}, \mathcal{Q}_f, \Delta)$, $\mathcal{Q}_f \subseteq \mathcal{Q}$ and Δ is a finite set of transitions. The rewriting relation on $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ induced by Δ of $\mathcal{A}$ is denoted $\to_{\Delta}$ or $\to_{\mathcal{A}}$. The tree language $\{t \in \mathcal{T}(\mathcal{F}) \mid t \to^{\star}_{\mathcal{A}} q\}$ is denoted $L(\mathcal{A}, q)$ and called the *tree language recognised by $\mathcal{A}$ in q*. The language *recognised* by $\mathcal{A}$, denoted $L(\mathcal{A})$, is the language $\bigcup_{q \in \mathcal{Q}_f} L(\mathcal{A}, q)$. A tree language is *regular* if and only if it is recognised by a tree automaton. A *run* of a tree automaton $\mathcal{A} = (\mathcal{Q}, \mathcal{Q}_f, \Delta)$ on a term $t \in \mathcal{T}(\mathcal{F})$ is a function $\rho : \mathcal{P}os(t) \to Q$ such that $\rho(p) = q$ for all $p \in \mathcal{P}os(t)$, where $q \in \mathcal{Q}$ and $t|_p = f(t_1, \ldots, t_n)$, $ar(f) = n$, $f(\rho(p.1), \ldots, \rho(p.n)) \to q \in \Delta$. A run is *successful* if $\rho(\epsilon) \in \mathcal{Q}_f$.

Positive TAGEDs. A positive TAGED[16] is a 4-tuple $\mathcal{A} = (\mathcal{Q}, E, F, \Delta)$, where $(\mathcal{Q}, F, \Delta)$ is a tree automaton over $\mathcal{F}$, and $E \subseteq \mathcal{Q} \times \mathcal{Q}$ is a binary reflexive symmetric relation on a subset of $\mathcal{Q}$. The tree automaton $(\mathcal{Q}, F, \Delta)$ is denoted $ta(\mathcal{A})$. A successful run of a positive TAGED $\mathcal{A} = (\mathcal{Q}, E, F, \Delta)$ on a term $t \in \mathcal{T}(\mathcal{F})$ is a successful run ρ of $ta(\mathcal{A})$ on t satisfying: for all positions p_1, $p_2 \in \mathcal{P}os(t)$, if

$(\rho(p_1), \rho(p_2)) \in E$ then $t|_{p_1} = t|_{p_2}$. For positive TAGEDs, the emptiness problem is in EXPTIME [16, Theorem 1], and universality and inclusion problems are both undecidable [16, Proposition 5]. Following the respective definitions of runs, it is straightforward that for every positive TAGED $\mathcal{A}$, $L(\mathcal{A}) \subseteq L(ta(\mathcal{A}))$.

2.2 Linear Temporal Logic and Term Rewriting

In this section, linear temporal properties are put in a rewriting context. The approach is based on the well-known and widely used Linear Temporal Logic (LTL for short) [29]. Our goal is to express and to verify temporal constraints on the order of rewriting rules in $\rightarrow_{\mathcal{R}}^*$. The approach is very close to that in [25] when reducing the equational theory to the identity.

Let $\mathcal{R}$ be a TRS and L_0 be a set of terms. We denote by $G(L_0, \mathcal{R})$ the $\mathcal{R}$-labelled graph $(\mathcal{T}(\mathcal{F}), L_0, \Delta)$ where $\Delta = \{t_i \stackrel{l \rightarrow r}{\rightarrow} t_j \mid l \rightarrow r \in \mathcal{R} \text{ and } t_j \in \{l \rightarrow r\}(t_i)\}$. A path π in $G(L_0, \mathcal{R})$ is a (finite or infinite) sequence $(p_1, a_1, q_1) \dots (p_i, a_i, q_i) \dots$ of elements of Δ such that $p_1 \in L_0$, for every $i \geq 1$ if p_{i+1} exists, then $q_i = p_{i+1}$. The (finite or infinite) word $a_1 \dots a_i \dots$ over the alphabet $\mathcal{R}$ is called the label of π. A path π is full if it is either infinite or if there exists an integer i such that $\pi = (p_1, a_1, q_1), \dots, (p_i, a_i, q_i)$ and $\{p \mid \exists a \in \mathcal{R},\ (q_i, a, p) \in \Delta\}$ is empty.

LTL formulas over $\mathcal{R}$ are inductively defined by: $\mathcal{R}_0 \subseteq \mathcal{R}$ is an LTL formula, and if φ and ψ are LTL formulas over $\mathcal{R}$, then $\top$, $\neg\varphi$, $(\varphi \vee \psi)$, $\circ\varphi$ and $\varphi \mathcal{U} \psi$ are also LTL formulas. Following formulas are classically defined: $\Box\varphi = \neg(\top \mathcal{U} \neg\varphi)$, $(\varphi \wedge \psi) = \neg(\neg\varphi \vee \neg\psi)$ and $\varphi \Rightarrow \psi = (\neg\varphi \vee \psi)$.

Let w be a finite or infinite word over $\mathcal{R}$ (considered as an alphabet). The i-th letter of w, if it exists, is denoted $w(i)$. We inductively define the satisfaction of an LTL formula φ by w at position i, denoted $(w, i) \models \varphi$ by:

$(w, i) \models \top$	iff $w(i)$ exists,
$(w, i) \models \mathcal{R}_0$, with $\mathcal{R}_0 \subseteq \mathcal{R}$	iff $w(i)$ exists and $w(i) \in \mathcal{R}_0$,
$(w, i) \models \neg\varphi$	iff $(w, i) \not\models \varphi$,
$(w, i) \models (\varphi_1 \vee \varphi_2)$	iff $(w, i) \models \varphi_1$ or $(w, i) \models \varphi_2$,
$(w, i) \models \circ\varphi$	iff $(w, i+1) \models \varphi$,
$(w, i) \models (\varphi_1 \mathcal{U} \varphi_2)$	iff there exists $j \geq i$ such that $(w, i) \models \varphi_2$ and for every $i \leq k < j$, $(w, k) \models \varphi_1$.

We say that w is a model of φ if $(w, 1) \models \varphi$. A graph $G(L_0, \mathcal{R})$ satisfies an LTL formula φ, denoted $G \models \varphi$, if and only if the label of each full path in $G(L_0, \mathcal{R})$ satisfies φ. Illustrated examples are given in Section 3.

3 Three LTL Patterns and Related Language Equalities

In this section, we study three LTL formula patterns which are useful to express security requirements when performing Java MIDLet applications static analysis.

- Formula $\Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$ intuitively means that if an accessible term is rewritten using a rule in $\mathcal{R}_1$, then the obtained term can be rewritten using a rule in $\mathcal{R}_2$ and only by a rule in $\mathcal{R}_2$, as illustrated on an abstract graph in Fig. 1. In our application domain, this temporal pattern is used to express that if

a method m_1 is invoked, then a method m_2 must be invoked just after. For instance, if the method asks the user to authentify using his PINCODE, then the next invoked method is either the authentication or the cancellation of the authentication.

- Formula $\neg\mathcal{R}_2 \wedge \Box(\circ\mathcal{R}_2 \Rightarrow \mathcal{R}_1)$ is the dual of the above temporal pattern: if an accessible term is rewritten using a rule in $\mathcal{R}_2$, then just before it was rewritten using a rule in $\mathcal{R}_1$, as illustrated on an abstract graph in Fig. 2. For instance, this temporal formula pattern expresses that if a SMS is sent, then the user has just before provided his agreement.
- Formula $\Box(\mathcal{R}_1 \Rightarrow \Box\neg\mathcal{R}_2)$ encodes that if a rule in $\mathcal{R}_1$ is used in a rewriting derivation, then no rule of $\mathcal{R}_2$ can be used in the future, as shown in Fig. 3. Thanks to this temporal formula pattern, one can express that if a particular application accesses to the user's private data, like his address book, no message can be sent by this application in the future. So, the user's private data cannot be exploited unbeknown to him. Notice that, according to [13], this formula pattern appears to be commonly used for system specification.

3.1 Formula $\Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$

We explore in this section how the model-checking of the formula $\Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$ can be translated into language equations. A $\mathcal{R}$-labelled graph satisfying this formula is depicted in Fig. 1.

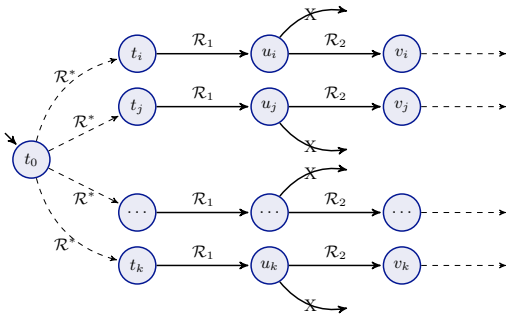

Fig. 1. A graph satisfying $\Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$

Proposition 1. *Let $\mathcal{R}$ be a TRS, $\mathcal{R}_1, \mathcal{R}_2 \subseteq \mathcal{R}$ and L_0 be a tree language. One has $G(L_0, \mathcal{R}) \models \Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$ iff $(\mathcal{R} \setminus \mathcal{R}_2)(\mathcal{R}_1(\mathcal{R}^*(L_0))) = \emptyset$ and $\mathcal{R}_1(\mathcal{R}^*(L_0)) \cap \mathcal{R}_2^{-1}(\mathcal{T}(\mathcal{F})) = \mathcal{R}_1(\mathcal{R}^*(L_0))$.*

Example 1. Let $\mathcal{F} = \{\bot, a, b, c, f, g\}$ where $ar(\bot) = 0$, $ar(a) = ar(b) = ar(c) = 1$, and $ar(f) = ar(g) = 2$. Let consider the TRS $\mathcal{R} = \{r_1, \ldots, r_5\}$ with $r_1 = f(b(x), b(x)) \to g(x, x)$, $r_2 = a(x) \to a(a(x))$, $r_3 = a(\bot) \to b(\bot)$, $r_4 = a(b(x)) \to b(b(x))$ and $r_5 = g(x, y) \to c(g(x, y))$. Finally, let $L_0 = \{f(a(u(\bot)), v(a(\bot))) \mid u \in \{a, b\}^* \text{ and } v \in a^*\}$. One has $\{r_1\}(\mathcal{R}^*(L_0)) \subseteq g(b^*(\bot), b^*(\bot))$. Thus $(\mathcal{R} \setminus$

$\{r_5\})(\{r_1\}(\mathcal{R}^*(L_0))) = \emptyset$. Moreover, $\{r_5\}^{-1}(\mathcal{T}(\mathcal{F}))$ is the set of terms where g occurs once at least. Consequently, $\{r_1\}(\mathcal{R}^*(L_0)) \cap \{r_5\}^{-1}(\mathcal{T}(\mathcal{F})) = \{r_1\}(\mathcal{R}^*(L_0))$. It follows that $G(L_0, \mathcal{R}) \models \Box(\{r_1\} \Rightarrow \circ\{r_5\})$.

3.2 Formula $\neg\mathcal{R}_2 \wedge \Box(\circ\mathcal{R}_2 \Rightarrow \mathcal{R}_1)$

In this section the formula $\neg\mathcal{R}_2 \wedge \Box(\circ\mathcal{R}_2 \Rightarrow \mathcal{R}_1)$ is compiled to into a language equation to be checked. A $\mathcal{R}$-labelled graph satisfying this formula is depicted in Fig. 2.

Proposition 2. *Let $\mathcal{R}$ be a TRS, $\mathcal{R}_1, \mathcal{R}_2 \subseteq \mathcal{R}$ and L_0 be a tree language. One has $G(L_0, \mathcal{R}) \models \neg\mathcal{R}_2 \wedge \Box(\circ\mathcal{R}_2 \Rightarrow \mathcal{R}_1)$ iff $\mathcal{R}_2((\mathcal{R} \setminus \mathcal{R}_1)(\mathcal{R}^*(L_0))) = \emptyset$ and $\mathcal{R}_2(L_0) = \emptyset$.*

Example 2. In the setting of Example 1, one has $\{r_5\}(L_0) = \emptyset$. Moreover, one can check that g doesn't occur in terms of $\mathcal{R} \setminus \{r_1, r_5\}(\mathcal{R}^*(L_0))$, proving that $\{r_5\}(\mathcal{R} \setminus \{r_1, r_5\}(\mathcal{R}^*(L_0))) = \emptyset$. Consequently, $G(L_0, \mathcal{R}) \models \neg\{r_5\} \wedge \Box(\circ\{r_5\} \Rightarrow \{r_1, r_5\})$.

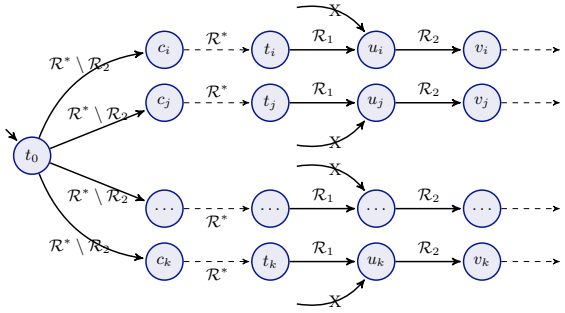

Fig. 2. A graph satisfying $\neg\mathcal{R}_2 \wedge \Box(\circ\mathcal{R}_2 \Rightarrow \mathcal{R}_1)$

3.3 Formula $\Box(\mathcal{R}_1 \Rightarrow \Box\neg\mathcal{R}_2)$

This section shows how the model-checking of the formula $\Box(\mathcal{R}_1 \Rightarrow \Box\neg\mathcal{R}_2)$ can be done thanks to language equations. A $\mathcal{R}$-labelled graph satisfying this formula is depicted in Fig. 3.

Proposition 3. *Let $\mathcal{R}$ be a TRS, $\mathcal{R}_1, \mathcal{R}_2 \subseteq \mathcal{R}$ and L_0 be a tree language. One has $G(L_0, \mathcal{R}) \models \Box(\mathcal{R}_1 \Rightarrow \Box\neg\mathcal{R}_2)$ if and only if $\mathcal{R}_2(\mathcal{R}^*(\mathcal{R}_1(\mathcal{R}^*(L_0)))) = \emptyset$.*

Example 3. In Example 1 setting, one has $\{r_1\}(\mathcal{R}^*(L_0)) \subseteq g(b^*(\bot), b^*(\bot))$. It follows that a never occurs in terms of $\mathcal{R}^*(\{r_1\}(R^*(L_0)))$. Consequently, $\{r_2\}(\mathcal{R}^*(\{r_1\}(R^*(L_0)))) = \emptyset$, proving that $G(L_0, \mathcal{R}) \models \Box(\{r_1\} \Rightarrow \Box\neg\{r_2\})$.

4 Semi-decision Procedures

In Section 4.1, we first show that for the above properties, model-checking is undecidable; That is not surprising. To obtain semi-decision procedures for model-checking these properties, we then provide TAGED-based construction presented in this section. As explained in Sect. 1, given a set $\mathcal{E} \subseteq \mathcal{T}(\mathcal{F})$ of initial terms, over-approximations of the set of reachable terms $\mathcal{R}^*(\mathcal{E})$ can be computed [20,15]. In Sect. 4.2, we explain how to exploit these over-approximations and use constructions of Sect. 4.1 to verify three rewriting temporal properties introduced in Sect. 3.

4.1 Language Equalities and Positive TAGEDs

First we claim that the model-checking of the three pointed out formulas is undecidable.

Proposition 4. *Given a TRS $\mathcal{R}$, $\mathcal{R}_1, \mathcal{R}_2 \subseteq \mathcal{R}$ and a term t_0, one cannot decide whether $G(\{t_0\}, \mathcal{R})) \models \Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$ (resp. whether $G(\{t_0\}, \mathcal{R}) \models \Box(\circ\mathcal{R}_2 \Rightarrow \mathcal{R}_1)$) (resp. whether $G(\{t_0\}, \mathcal{R})) \models \Box(\mathcal{R}_1 \Rightarrow \Box\neg\mathcal{R}_2)$).*

Now we provide several positive TAGED-based constructions in order to cope with the language equalities involved in Sect. 3.

Proposition 5. *Let $\mathcal{R}$ be a TRS. One can compute in polynomial time a positive TAGED accepting $\mathcal{R}^{-1}(\mathcal{T}(\mathcal{F}))$.*

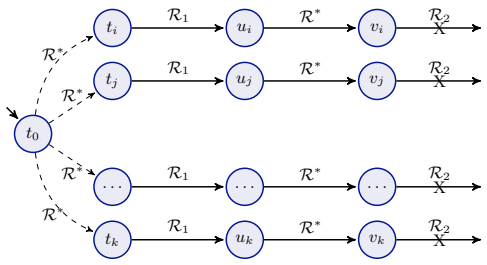

Fig. 3. A graph satisfying $\Box(\mathcal{R}_1 \Rightarrow \Box\neg\mathcal{R}_2)$

An example of the construction described in the proof of Proposition 5 can be found in Appendix, Sect. 6.6. Notice that if $\mathcal{R}$ is left-linear, the obtained TAGED is a tree automaton as for any variable x, the state q_x occurs at most once in runs; This is a well-known result.

Proposition 6. *Let $\mathcal{A}$ be a positive TAGED automaton and $\mathcal{R}$ be a TRS. Deciding whether $\mathcal{R}(L(\mathcal{A}))$ is empty is in EXPTIME.*

Proposition 7. *Let $\mathcal{A}$ be a tree automaton and $\mathcal{R}$ be a TRS. The language $\mathcal{R}(L(\mathcal{A}))$ is accepted by a positive TAGED.*

A constructive example is given in Appendix, Section 6.9

4.2 Algorithms

In order to semi-decide whether the temporal properties are satisfied or not, we introduce the following procedures.

- $\mathsf{Approx}(\mathcal{A},\mathcal{R})$, where $\mathcal{A}$ is a tree automaton and $\mathcal{R}$ is a TRS, returns a tree automaton $\mathcal{B}$ such that $\mathcal{R}^*(L(\mathcal{A})) \subseteq L(\mathcal{B})$. This can be done using the procedure defined in [7].
- $\mathsf{ta}(\mathcal{A})$, where $\mathcal{A}$ is a positive TAGED, returns the tree automaton $ta(\mathcal{A})$.
- $\mathsf{OneStep}(\mathcal{A},\mathcal{R})$, where $\mathcal{A}$ is a tree automaton and $\mathcal{R}$ is a TRS, returns the positive TAGED $\mathcal{B}$ accepting $\mathcal{R}(L(\mathcal{A}))$ built as in Proposition 7.
- $\mathsf{Backward}(\mathcal{R})$, where $\mathcal{R}$ is a TRS, returns the positive TAGED $\mathcal{B}$ accepting $\mathcal{R}^{-1}(\mathcal{T}(\mathcal{F}))$ built as in Proposition 5.
- $\mathsf{IsEmpty}(\mathcal{A},\mathcal{R})$, where $\mathcal{A}$ is a positive TAGED and $\mathcal{R}$ is a TRS, returns true if $\mathcal{R}(L(\mathcal{A}))$ is empty and false, otherwise.

The above procedures and the results in Section 3 allow to deduce the following result.

Proposition 8. *Let $\mathcal{R}$ be a TRS, $\mathcal{R}_1, \mathcal{R}_2 \subseteq \mathcal{R}$ and $\mathcal{A}$ be a tree automaton. The following properties hold:*

(1) If $\mathcal{R}_2$ is left-linear and if $\mathsf{IsEmpty}(\mathsf{OneStep}(\mathsf{Approx}(\mathcal{A},\mathcal{R}),\mathcal{R}_1),\mathcal{R} \setminus \mathcal{R}_2) = \mathsf{true}$ *and if* $\mathsf{OneStep}(\mathsf{Approx}(\mathcal{A},\mathcal{R}),\mathcal{R}_1) \subseteq \mathsf{Backward}(\mathcal{R}_2)$*, then* $G(L(\mathcal{A}),\mathcal{R}) \models \Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$.

(2) If $\mathsf{IsEmpty}(\mathcal{A},\mathcal{R}_2)$ *and if* $\mathsf{IsEmpty}(\mathsf{OneStep}(\mathsf{Approx}(\mathcal{A},\mathcal{R}),\mathcal{R} \setminus \mathcal{R}_1),\mathcal{R}_2) = \mathsf{true}$*, then* $G(L(\mathcal{A}),\mathcal{R}) \models \Box(\circ\mathcal{R}_2 \Rightarrow \mathcal{R}_1)$.

(3) If $\mathsf{IsEmpty}(\mathsf{Approx}(\mathsf{ta}(\mathsf{OneStep}(\mathsf{Approx}(\mathcal{A},\mathcal{R}),\mathcal{R}_1)),\mathcal{R}),\mathcal{R}_2) = \mathsf{true}$*, then* $G(L(\mathcal{A}),\mathcal{R}) \models \Box(\mathcal{R}_1 \Rightarrow \Box\neg\mathcal{R}_2)$.

Notice that in *(1)* $\mathcal{R}_2$ is required to be left-linear in order to make the inclusion test decidable.

5 Conclusion and Related Work

We proposed to exploit abstraction-based rewriting approximations to model-check some LTL temporal properties on infinite state systems, and to combat a combinatorial state-space blow up faced by model-checking tools. Our approach is based on the reachability analysis through rewriting approximations as well as tree automata with global equality constraints. We address static analysis problems. Approximation techniques were already implemented in [3]. In the future we plan to integrate TAGED-based algorithms into this tool in order to treat practical applications.

Related work.

Temporal properties and rewriting. Hundreds of works exist using LTL [29] in order to model and to verify systems properties. We refer the interested reader to the Spin Model-Checker home page[2].

[2] http://spinroot.com/spin/whatispin.html

Rewriting logics [25] is a very general theoretical framework allowing one to model various systems. In this context, rewriting graphs are considered: nodes of these graphs are labeled by equivalence classes of an equational theory. There is an edge between two nodes if an element of the first node can be rewritten into an element of the second node, using a rule of TRS $\mathcal{R}$. When the considered equational theory is the identity, these rewriting graphs are exactly the graphs underlying our labeled transition systems. In this framework, works [14,27,26] focuses on LTL approaches. In [1] authors proposes proposes a general model for security protocols based on the set-rewriting formalism in a decidable context (considered underlying graphs are finite).

Tree automata with constraints. Tree automata were intensively studied in the literature, in particular for program verification, where tree automata provide abstraction-based approximations of program configurations. In this direction, several classes of extended automata were defined in order to provide finer approximations [4,11,9,16,30,22,28,21].

References

1. A. Armando, R. Carbone, and L. Compagna. Ltl model checking for security protocols. In *CSF*, pages 385–396, 2007.
2. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
3. E. Balland, Y. Boichut, T. Genet, and P.-E. Moreau. Towards an efficient implementation of tree automata completion. In *AMAST*, pages 67–82, 2008.
4. B. Bogaert and S. Tison. Equality and disequality constraints on direct subterms in tree automata. In *STACS*, pages 161–171, 1992.
5. Y. Boichut, R. Courbis, P.-C. Héam, and O. Kouchnarenko. Finer is better: Abstraction refinement for rewriting approximations. In *Rewriting Techniques and Application, RTA'08*, volume 5117 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2008.
6. Y. Boichut, Th. Genet, Th. Jensen, and L. Le Roux. Rewriting approximations for fast prototyping of static analyzers. In *Rewriting Techniques and Applications, RTA'07*, Lecture Notes in Computer Science 4533, pages 48–62. Springer, 2007.
7. Y. Boichut, P.-C. Héam, and O. Kouchnarenko. Approximation-based tree regular model-checking. *Nordic Journal of Computing*, 2009. To appear.
8. E. M. Clarke. Counterexample-guided abstraction refinement. In *TIME-ICTL*, page 7. IEEE Computer Society, 2003.
9. H. Comon and V. Cortier. Tree automata with one memory set constraints and cryptographic protocols. *Theoretical Computer Science (TCS'05)*, 331, 2005.
10. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications.* 2002. Available at *http://www.grappa.univ-lille3.fr/tata/*.
11. M. Dauchet, A.-C. Caron, and J.-L. Coquidé. Automata for reduction properties solving. *J. Symb. Comput.*, 20(2):215–233, 1995.
12. N. Dershowitz and J.-P. Jouannaud. *Handbook of Theoretical Computer Science*, volume B, chapter 6: Rewrite Systems, pages 244–320. Elsevier Science Publishers B. V, 1990.

13. M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Property specification patterns for finite-state verification. In *Proceedings of the Second Workshop on Formal Methods in Software Practice*, pages 7–15. ACM Press, 1998.
14. S. Escobar and J. Meseguer. Symbolic model checking of infinite-state systems using narrowing. In *Rewriting Techniques and Applications, RTA'07*, pages 153–168, 2007.
15. G. Feuillade, Th. Genet, and V. VietTriemTong. Reachability analysis over term rewriting systems. *Journal on Automated Reasoning*, 33 (3-4), 2004.
16. E. Filiot, J.-M. Talbot, and S. Tison. Tree automata with global constraints. In *Developments in Language Theory*, pages 314–326, 2008.
17. Th. Genet and F. Klay. Rewriting for Cryptographic Protocol Verification. In *Conference on Automated Deduction, CADE'00*, volume 1831 of *Lecture Notes in Computer Science*, pages 271–290. Springer-Verlag, 2000.
18. R. Gilleron and S. Tison. Regular tree languages and rewrite systems. *Fundamenta Informatica*, 24(1/2):157–174, 1995.
19. P. Gyenizse and S. Vágvölgyi. Linear Generalized Semi-Monadic Rewrite Systems Effectively Preserve Recognizability. *Theoretical Computer Science*, 194(1-2):87–122, 1998.
20. F. Jacquemard. Decidable approximations of term rewriting systems. In *Rewriting Techniques and Applications, RTA'96*, volume 1103, pages 362–376. Springer Verlag, 1996.
21. F. Jacquemard, M. Rusinowitch, and L. Vigneron. Tree automata with equality constraints modulo equational theories. In *IJCAR*, pages 557–571, 2006.
22. W. Karianto and Ch. Löding. Unranked tree automata with sibling equalities and disequalities. In *ICALP*, pages 875–887, 2007.
23. L. Lamport. A temporal logic of actions. *ACM Transactions On Programming Languages And Systems, TOPLAS*, 16(3):872–923, May 1994.
24. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. SV, 1992.
25. J. Meseguer. Conditioned rewriting logic as a united model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.
26. J. Meseguer. The temporal logic of rewriting. Technical Report UIDCS-R-2007-2815, Dept of Computer Science, University of Illinois at Urbana-Champaign, September 2007.
27. J. Meseguer. The temporal logic of rewriting: A gentle introduction. *Concurrency, Graphs and Models: Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday*, pages 354–382, 2008.
28. H. Ohsaki and T. Takai. ACTAS: A system design for associative and commutative tree automata theory. *Electronic Notes in Theoretical Computer Science*, 124(1):97–111, 2005.
29. A. Pnueli. The temporal logic of programs. In *FOCS'77*, pages 46–57, 1977.
30. H. Seidl, Th. Schwentick, A. Muscholl, and P. Habermehl. Counting in trees for free. In *ICALP*, pages 1136–1149, 2004.
31. T. Takai, Y. Kaji, and H. Seki. Right-linear finite-path overlapping term rewriting systems effectively preserve recognizability. In *proceedings of RTA*, volume 1833 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

6 Appendix

6.1 Proof of Proposition 1

Assume that $G(L_0, \mathcal{R}) \models \Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$. Let t be a term in $(\mathcal{R}\backslash\mathcal{R}_2)(\mathcal{R}_1(\mathcal{R}^*(L_0)))$. There exist terms t_1 and t_2 such that $t_1 \in \mathcal{R}^*(L_0)$ and

$$t_1 \rightarrow_{\mathcal{R}_1} t_2 \rightarrow_{(\mathcal{R}\backslash\mathcal{R}_2)} t.$$

Since $t_1 \in \mathcal{R}^*(L_0)$, there exist terms $s_0, \ldots, s_k$ such that $s_0 \in L_0$, $s_k = t_1$ and $s_{i+1} \in \mathcal{R}(s_i)$ for every $i < k$. Therefore there is a path

$$(s_0, a_0, s_1) \ldots (s_{k-1}, a_k, t_1)(t_1, a, t_2), (t_2, b, t)$$

in $G(L_0, \mathcal{R})$ such that $s_0 \in L_0$, $a \in \mathcal{R}_1$ and $b \in \mathcal{R} \setminus \mathcal{R}_2$. This path may be extended to a full path whose label is not a model of $\Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$, a contradiction.

Now since $\mathcal{R}_1(\mathcal{R}^*(L_0)) \cap \mathcal{R}_2^{-1}(\mathcal{T}(\mathcal{F})) \subseteq \mathcal{R}_1(\mathcal{R}^*(L_0))$, if $\mathcal{R}_1(\mathcal{R}^*(L_0)) \cap \mathcal{R}_2^{-1}(\mathcal{R}^*(L_0)) \neq \mathcal{R}_1(\mathcal{R}^*(L_0))$, then there exists $t \in \mathcal{R}_1(\mathcal{R}^*(L_0))$ such that $t \notin \mathcal{R}_2^{-1}(\mathcal{T}(\mathcal{F}))$. It follows there exists a term $t_1 \in \mathcal{R}^*(L_0)$ such that $t \in \mathcal{R}_1(t_1)$. Therefore there exist terms $s_0, \ldots, s_k$ such that $s_0 \in L_0$, $s_k = t_1$ and $s_{i+1} \in \mathcal{R}(s_i)$ for every $i < k$. Consequently, there is a path

$$\pi = (s_0, a_0, s_1) \ldots (s_{k-1}, a_k, t_1)(t_1, b, t)$$

in $G(L_0, \mathcal{R})$ such that $s_0 \in L_0$, $b \in \mathcal{R}_1$. Since $t \notin \mathcal{R}_2^{-1}(\mathcal{T}(\mathcal{F}))$, there is no term t_2 such that $t_2 \in \mathcal{R}_2(t)$. Consequently, π cannot be extended using a transition whose label is in $\mathcal{R}_2$. It follows that either π is maximal and its label is not a model of $\Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$, or π may be extended to a full path which is not a model of $\Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$, a contradiction.

Conversely, assume that $(\mathcal{R} \setminus \mathcal{R}_2)(\mathcal{R}_1(\mathcal{R}^*(L_0))) = \emptyset$ and $\mathcal{R}_1(\mathcal{R}^*(L_0)) \cap \mathcal{R}_2^{-1}(\mathcal{T}(\mathcal{F})) = \mathcal{R}_1(\mathcal{R}^*(L_0))$. Let $\pi = (t_0, a_0, t_1) \ldots (t_k, a_k, t_{k+1}) \ldots$ be a maximal path in $G(L_0, \mathcal{R})$ whose label is not a model of $\Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$. It follows there exists i such that $a_i \in R_1$ and either $a_{i+1} \notin \mathcal{R}_2$ or a_{i+1} doesn't exist (the trace is finite). If $a_{i+1} \notin \mathcal{R}_2$ then $t_{i+1} \in (\mathcal{R} \setminus \mathcal{R}_2)(\mathcal{R}_1(\mathcal{R}^*(L_0)))$, a contradiction. If a_{i+1} doesn't exist, then $t_{i+1} \in \mathcal{R}_1(\mathcal{R}^*(L_0))$ but, by maximality of π, $\mathcal{R}_2(t_{i+1}) = \emptyset$, proving that $t_{i+1} \notin \mathcal{R}_2^{-1}(\mathcal{T}(\mathcal{F}))$, a contradiction.

6.2 Proof of Proposition 2

Proof. It is straightforward that $G(L_0, \mathcal{R}) \models \neg\mathcal{R}_2$ iff $\mathcal{R}_2(L_0) = \emptyset$. Now we will prove that $G(L_0, \mathcal{R}) \models \Box(\circ\mathcal{R}_2 \Rightarrow \mathcal{R}_1)$ iff $\mathcal{R}_2((\mathcal{R} \setminus \mathcal{R}_1)(\mathcal{R}^*(L_0))) = \emptyset$.

Assume first that $G(L_0, \mathcal{R}) \models \Box(\circ\mathcal{R}_2 \Rightarrow \mathcal{R}_1)$. Let $t \in \mathcal{R}_2((\mathcal{R}\setminus\mathcal{R}_1)(\mathcal{R}^*(L_0)))$. There exist terms t_1 and t_2 such that $t_1 \in \mathcal{R}^*(L_0)$ a rule $a \in \mathcal{R} \setminus \mathcal{R}_1$ such that $t_2 \in \{a\}(t_1)$ and $t \in \mathcal{R}_2(t_2)$. Since $t_1 \in \mathcal{R}^*(L_0)$, there exist terms $s_0, \ldots, s_k$ such that $s_0 \in L_0$, $s_k = t_1$ and $s_{i+1} \in \mathcal{R}(s_i)$ for every $i < k$. Therefore there is a path

$$(s_0, a_0, s_1) \ldots (s_{k-1}, a_k, t_1)(t_1, a, t_2), (t_2, b, t)$$

in $G(L_0, \mathcal{R})$ such that $s_0 \in L_0$, $a \in \mathcal{R} \setminus \mathcal{R}_1$, and $b \in \mathcal{R}_2$. Since $a \in \mathcal{R} \setminus \mathcal{R}_1$, this path may be extended to a full path whose label is not a model of $\Box(\circ\mathcal{R}_2 \Rightarrow \mathcal{R}_1)$, a contradiction.

Conversely, assume that $\mathcal{R}_2((\mathcal{R} \setminus \mathcal{R}_1)(\mathcal{R}^*(L_0))) = \emptyset$. Let

$$\pi = (t_0, a_0, t_1) \dots (t_k, a_k, t_{k+1}) \dots$$

be a maximal path in $G(L_0, \mathcal{R})$ whose label is not a model of $\Box(\circ\mathcal{R}_2 \Rightarrow \mathcal{R}_1)$. It follows there exists i such that $a_i \notin \mathcal{R}_1$ and $a_{i+1} \in \mathcal{R}_2$. Therefore $t_{i+1} \in \mathcal{R}_2((\mathcal{R} \setminus \mathcal{R}_1)(\mathcal{R}^*(L_0))) \neq \emptyset$, a contradiction.

6.3 Proof of Proposition 3

Proof. Assume first that $\mathcal{R}_2(\mathcal{R}^*(\mathcal{R}_1(\mathcal{R}^*(L_0)))) \neq \emptyset$. Let t be in $\mathcal{R}_2(\mathcal{R}^*(\mathcal{R}_1(\mathcal{R}^*(L_0))))$. There exist terms t_0, t_1, t_2 such that $t_0 \in \mathcal{R}^*(L_0)$, and $t_0 \rightarrow_{\mathcal{R}_1} t_1 \rightarrow^*_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}_2} t$. It implies that there is a full path w in $G(L_0, \mathcal{R})$ such that $(w, 1) \not\models \Box(\mathcal{R}_1 \Rightarrow \Box\neg\mathcal{R}_2)$.

Assume now that $\mathcal{R}_2(\mathcal{R}^*(\mathcal{R}_1(\mathcal{R}^*(L_0)))) = \emptyset$. Let

$$\pi = (t_0, a_0, t_1) \dots (t_k, a_k, t_{k+1}) \dots$$

be a maximal path in $G(L_0, \mathcal{R})$ whose label is not a model of $\Box(\mathcal{R}_1 \Rightarrow \Box\neg\mathcal{R}_2)$. It follows there exist i such that $a_i \in \mathcal{R}_1$ and $j > i$ such that $a_j \in \mathcal{R}_2$. Therefore $t_{j+1} \in \mathcal{R}_2(\mathcal{R}^*(\mathcal{R}_1(\mathcal{R}^*(L_0))))$, a contradiction.

6.4 Proof of Proposition 4

It is well known that the following problem, called **Reachability**$(\mathcal{R}, s, t, \mathcal{F})$ is undecidable.
Input: A TRS $\mathcal{R}$ on $\mathcal{T}(\mathcal{F})$, two terms s and t of $\mathcal{T}(\mathcal{F})$.
Question: Does $s \rightarrow^*_{\mathcal{R}} t$?

Assume there exists an algorithm $P_1(\mathcal{R}, \mathcal{R}_1, \mathcal{R}_2, L_0, \mathcal{F})$ that , given a TRS $\mathcal{R}$ and a set of terms L_0 of $\mathcal{T}(\mathcal{F})$, decides whether $G(L_0, \mathcal{R}) \models \Box(\mathcal{R}_1 \Rightarrow \circ\mathcal{R}_2)$. Let $\mathcal{R}_0, s_0, t_0, \mathcal{F}_0$ be an instance of the Reachability problem. Let $\#, \$ \notin \mathcal{F}_0$ and $\mathcal{F}_1 = \mathcal{F} \cup \{\#, \$\}$, with $ar(\#) = ar(\$) = 0$. We claim that $P_1(\mathcal{R}_0 \cup \{t_0 \rightarrow \#, \$ \rightarrow \#\}, \{t_0 \rightarrow \#\}, \{\$ \rightarrow \#\}, \{s_0\}, \mathcal{F}_1) = \mathsf{false}$ if and only if Reachability$(\mathcal{R}_0, s_0, t_0, \mathcal{F}_0)$=true. Indeed, if Reachability$(\mathcal{R}_0, s_0, t_0, \mathcal{F}_0)$=true, then there exists in $G(\{s_0\}, \mathcal{R}_0)$ a path π from s_0 to t_0. By construction, π also is a path in $G(\{s_0\}, \mathcal{R} \cup \{t_0 \rightarrow \#, \$ \rightarrow \#\})$. But π,$(t_0, \{t_0 \rightarrow \#\}, \#)$ is a full path in $G(\{s_0\}, \mathcal{R} \cup \{t_0 \rightarrow \#, \$ \rightarrow \#\})$ whose label doesn't model $\{t_0 \rightarrow \#\} \Rightarrow \circ\{\$ \rightarrow \#\}$. Consequently $P_1(\mathcal{R}_0 \cup \{t_0 \rightarrow \#, \$ \rightarrow \#\}, \{t_0 \rightarrow \#\}, \{\$ \rightarrow \#\}, \{s_0\}, \mathcal{F}_1) = \mathsf{false}$.
Conversely, if $P_1(\mathcal{R}_0 \cup \{t_0 \rightarrow \#, \$ \rightarrow \#\}, \{t_0 \rightarrow \#\}, \{\$ \rightarrow \#\}, \{s_0\}, \mathcal{F}_1) = \mathsf{false}$, then there exists a full path π' in $G(\{s_0\}, \mathcal{R} \cup \{t_0 \rightarrow \#, \$ \rightarrow \#\})$ whose label doesn't model $\{t_0 \rightarrow \#\} \Rightarrow \circ\{\$ \rightarrow \#\}$. Therefore, the transition $\{t_0 \rightarrow \#\}$ is used in π'. It follows that t_0 is reachable in $G(\{s_0\}, \mathcal{R} \cup \{t_0 \rightarrow \#, \$ \rightarrow \#\})$ from

s_0. It is straightforward that Reachability$(\mathcal{R}_0, s_0, t_0, \mathcal{F}_0)$=true, which concludes the proof.

The undecidability proofs for the two other formulas can be done with similar reductions.

6.5 Proof of Proposition 5

Let $l \to r \in \mathcal{R}$. Let $\mathcal{A}_l = (\mathcal{Q}_l, E_l, F_l, \Delta_l)$ be the positive TAGED defined by:

- $\mathcal{Q}_l = \{q_i \mid i \in \mathcal{P}os_{\mathcal{F}}(l)\} \cup \{q_x, q_x^a \mid x \in \mathcal{V}ar(l)\} \cup \{q^a\}$,
- $E_l = \{(q_x, q_x) \mid x \in \mathcal{V}ar(l)\}$,
- $\Delta_l = \Delta_1 \cup \Delta_2$ with $\Delta_1 = \{l(p)(q_{\alpha_1}, \ldots, q_{\alpha_n}) \to q_p \mid p \in \mathcal{P}os(l)$ and $\alpha_i = p.i$ if $l(p.i) \in \mathcal{F}$ and $\alpha_i = x$ otherwise$\} \cup \{f(q_x^a, \ldots, q_x^a) \to q_x^a \mid f \in \mathcal{F},\ x \in \mathcal{V}ar(l)\} \cup \{f(q_x^a, \ldots, q_x^a) \to q_x \mid f \in \mathcal{F},\ x \in \mathcal{V}ar(l)\}$ and $\Delta_2 = \{f(q^a, \ldots, q^a) \to q^a \mid f \in \mathcal{F}\} \cup \{f(q^a, \ldots, q^a, q_\varepsilon, q^a, \ldots, q^a) \to q_\varepsilon \mid f \in \mathcal{F}\}$,
- $F_l = \{q_\varepsilon\}$.

Notice first that $\{t \mid t \to_{\mathcal{A}_l}^* q_x\} = \mathcal{T}(\mathcal{F})$. Second, $\{t \mid t \to_{\Delta_l}^* q_\varepsilon\} = \{t \mid \exists p \in \mathcal{P}os(t), \mu : \mathcal{X} \mapsto \mathcal{T}(\mathcal{F})\ s.t.\ t|_p = l\mu\}$. It follows that $L(\mathcal{A}_l) = \{l \to r\}^{-1}(\mathcal{T}(\mathcal{F}))$. The construction is clearly polynomial ($\mathcal{F}$ is considered as fixed and is not a parameter of the problem). Polynomial time complexity results directly from [16, Proposition 2]. However complexity is exponential relatively to the maximal arity of a symbol in $\mathcal{F}$.

6.6 Example for Proposition 5

Example 4. Let $\mathcal{F} = \{\bot, h, f\}$ where $ar(\bot) = 0$, $ar(h) = 1$ and $ar(f) = 2$. The language $\{f(x, x) \to h(x)\}^{-1}(\mathcal{T}(\mathcal{F}))$ is accepted by the positive TAGED $\mathcal{A}_l = (\mathcal{Q}_l, E_l, F_l, \Delta_l)$ with

- $\mathcal{Q}_l = \{q_\varepsilon, q_1, q_2\} \cup \{q_x, q_x^a\} \cup \{q^a\}$,
- $E_l = \{(q_x, q_x)\}$,
- $\Delta_l = \Delta_1 \cup \Delta_2$ with $\Delta_1 = \{f(q_x, q_x) \to q_\varepsilon\} \cup \{f(q_x^a, q_x^a) \to q_x^a, \bot \to q_x^a, h(q_x^a) \to q_x^a\} \cup \{f(q_x^a, q_x^a) \to q_x, \bot \to q_x, h(q_x^a) \to q_x\}$ and $\Delta_2 = \{f(q^a, q^a) \to q^a, \bot \to q_a, h(q_a) \to q_a\} \cup \{f(q^a, q_\varepsilon) \to q_\varepsilon, f(q_\varepsilon, q^a) \to q_\varepsilon, h(q_\varepsilon) \to q_\varepsilon\}$
- $F_l = \{q_\varepsilon\}$.

6.7 Proof of Proposition 6

It suffices to note that $\mathcal{R}(L(\mathcal{A}))$ is empty if and only if $L(\mathcal{A}) \cap \mathcal{R}^{-1}(\mathcal{T}(\mathcal{F})) = \emptyset$. The proposition is then a direct consequence of Proposition 5 and [16, Proposition 2 and Theorem 1].

6.8 Proof of Proposition 7

Notice that the proof is constrcutive and that an example is is given in Section 6.9

Since $\mathcal{R}(L(\mathcal{A})) = \cup_{l\to r\in\mathcal{R}}\{l \to r\}(L(\mathcal{A}))$ and since positive TAGED languages are closed by union, it suffices to prove the proposition for a single rule $l \to r$.

The proof is composed of three parts: first, in (**Point 1**), a construction of some useful positive TAGEDs $\mathcal{A}_{r,\sigma,q}$ is proposed. Second, in (**Point 2**), we prove that $\{l \to r\}(L(\mathcal{A}))$ is accepted by the (finite) union of the $\mathcal{A}_{r,\sigma,q}$'s by showing that $L(\mathcal{A}_{r,\sigma,q}) \supseteq \{l \to r\}(L(\mathcal{A}))$ and that $L(\mathcal{A}_{r,\sigma,q}) \subseteq \{l \to r\}L(\mathcal{A})$ (**Point 3**). Since the class of languages accepted by positive TAGEDs is closed under finite union, the proof is then complete.

Point 1

Let $l \to r \in \mathcal{R}$. An $(l \to r)$-substitution is an application from $\mathcal{P}os_{\mathcal{X}}(l)$ into $\mathcal{Q}$. Let σ be a $(l \to r)$-substitution. We denote by $l\sigma$ the term of $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ defined as follows: $\mathcal{P}os(l\sigma) = \mathcal{P}os(l)$, and for each $p \in \mathcal{P}os(l)$, if $p \in \mathcal{P}os_{\mathcal{X}}(l)$ then $l\sigma(p) = \sigma(l(p))$, otherwise $l\sigma(p) = l(p)$.

Set $\mathcal{A} = (\mathcal{Q}, F, \Delta)$. Since the class of regular tree languages is closed by intersection, for each variable x occurring in l and for each $(l \to r)$-substitution σ, there exists a finite tree automaton $\mathcal{A}^\sigma_x = (\mathcal{Q}^\sigma_x, F^\sigma_x, \Delta^\sigma_x)$ such that

$$L(\mathcal{A}^\sigma_x) = \bigcap_{p\in\mathcal{P}os_{\{x\}}(l)} L(\mathcal{A}, \sigma(p)).$$

We may assume, w.l.o.g., that states of F^σ_x do not occur in left hand sides of transitions of Δ^σ_x.

Let $\mathcal{A}_{r,\sigma,q} = (\mathcal{Q}_{r,\sigma,q}, E_{r,\sigma,q}, F_{r,\sigma,q}, \Delta_{r,\sigma,q})$ be the positive TAGED defined by:

- $\mathcal{Q}_{r,\sigma,q} = \mathcal{Q} \cup \{q_i \mid i \in \mathcal{P}os_{\mathcal{F}}(r)\} \cup \{q^+ \mid q \in \mathcal{Q}\} \cup \bigcup_{x\in\mathcal{V}ar(r)} \mathcal{Q}^\sigma_x$,
- $E_{r,\sigma,q} = \{(q^1, q^2) \mid \exists x \in \mathcal{V}ar(r) \text{ s.t. } q^1, q^2 \in F^\sigma_x\}$,
- $F_{r,\sigma,q} = \{q^+_f \mid q_f \in F\}$,
- $\Delta_{r,\sigma,q} = \Delta \cup \Delta_1 \cup \Delta_2$ with
  $\Delta_1 = \{r(p)(q_{\alpha_1}, \ldots, q_{\alpha_n}) \to q_p \mid p \in \mathcal{P}os(r)$ and $\alpha_i = p.i$ if $r(p.i) \in \mathcal{F}$ and $q_{\alpha_i} \in F^\sigma_{r(p.i)}$ otherwise$\} \cup \bigcup_{x\in\mathcal{V}ar(r)} \Delta^\sigma_x$
  $\Delta_2 = \{f(s_1, \ldots, s_j, q_\varepsilon, s_{j+1}, \ldots, s_n) \to s^+_{n+1} \mid s_i \in \mathcal{Q}$ and
  $f(s_1, \ldots, s_{j-1}, q, s_{j+1}, \ldots, s_n) \to s_{n+1} \in \Delta\}$
  $\cup \{f(s_1, \ldots, s_{j-1}, s^+_j, s_{j+1}, \ldots, s_n) \to s^+_{n+1} \mid s_i \in \mathcal{Q}$ and
  $f(s_1, \ldots, s_{j-1}, s_j, s_{j+1}, \ldots, s_n) \to s_{n+1} \in \Delta,\ ar(f) \geq 1\}$.

We claim that

$$\mathcal{R}(L(\mathcal{A})) = \bigcup_{l\sigma\to^*_{\mathcal{A}} q} L(\mathcal{A}_{r,\sigma,q}),$$

where the union is taken for every state $q \in Q$, every $(l \to r)$-substitution σ such that $l\sigma \to^*_{\mathcal{A}} q$ and $L(\mathcal{A}^\sigma_x) \neq \emptyset$ for every $x \in \mathcal{V}ar(l)$.

Point 2
Assume that $t \in \mathcal{R}(L(\mathcal{A}))$. There exist a term $t_0 \in L(\mathcal{A})$, a substitution μ from $\mathcal{X}$ into $\mathcal{T}(\mathcal{F})$ and a position p of t_0 such that

$$t_0 = t_0[l\mu]_p \quad \text{and} \quad t = t_0[r\mu]_p. \tag{1}$$

Let $\{p_1, \ldots, p_k\} = \mathcal{P}os_{\mathcal{X}}(l)$. Since $t_0 \in L(\mathcal{A})$ there exist $q, q_1, \ldots, q_k \in \mathcal{Q}$ such that

$$l\mu \rightarrow_{\mathcal{A}}^* l[q_1]_{p_1} \ldots [q_k]_{p_k} \rightarrow_{\mathcal{A}}^* q \quad \text{and} \quad t_0[q]_p \rightarrow_{\mathcal{A}}^* q_f \in F. \tag{2}$$

Let σ be the $(l \rightarrow r)$ substitution defined by $\sigma(p_i) = q_i$. By construction one has for every $x \in \mathcal{V}ar(l)$,

$$\mu(x) \in \bigcap_{p \in \mathcal{P}os_{\{x\}}(l)} L(\mathcal{A}, \sigma(p)). \tag{3}$$

By definition of $\mathcal{A}_x^\sigma$ one then has

$$\mu(x) \in L(\mathcal{A}_x^\sigma). \tag{4}$$

It follows that for every $x \in \mathcal{V}ar(r)$,

$$\mu(x) \rightarrow_{\Delta_1}^* q_x \in F_x^\sigma \tag{5}$$

Therefore,

$$r\mu \rightarrow_{\Delta_1}^* q_\varepsilon \tag{6}$$

Using (1) and (2) it follows that

$$t \rightarrow_{\Delta_1}^* t_0[q]_p \rightarrow_{\Delta_2}^* q_f^+, \tag{7}$$

proving that $t \in L(\mathcal{A}_{r,\sigma,q})$. Notice that the constraints defined by $E_{r,\sigma,q}$ are satisfied: if during the reduction $t \rightarrow_{\Delta_1}^* t_0[q]_p$, two states $q^1, q^2 \in F_x^\sigma$ are used in position p_1' and p_1', then $t_{|p_1'} = t_{|p_2'} = \mu(x)$.

Point 3
Assume now that $t \in L(\mathcal{A}_{r,\sigma,q})$ for a state $q \in Q$ and an $(l \rightarrow r)$-substitution σ such that $l\sigma \rightarrow_{\mathcal{A}}^* q$ and $L(\mathcal{A}_x^\sigma) \neq \emptyset$ for every $x \in \mathcal{V}ar(l)$. Let ρ be a successful run of $\mathcal{A}_{r,\sigma,q}$ on t. It is straightforward that there exists a unique position p of t such that $\rho(p) = q_\varepsilon$. Let $\{p_1, \ldots, p_k\} = \mathcal{P}os_{\mathcal{X}}(r)$. By definition of $E_{r,\sigma,q}$, if $\rho(p_i), \rho(p_j) \in F_x^\sigma$ for a variable x occurring in r, then $r_{|p_i} = r_{|p_j}$. Therefore one can define the substitution μ from $\mathcal{V}ar(r)$ into $\mathcal{T}(\mathcal{F})$ by: if $\rho(p_i) \in F_x^\sigma$, then $\mu(x) = r_{|p_i}$. This construction provides

$$\mu(x) \in L(\mathcal{A}_x^\sigma) \tag{8}$$

and

$$t = t[r\mu]_p. \tag{9}$$

Remind that $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$, μ is extended to $\mathcal{V}ar(l)$ by: if $z \in \mathcal{V}ar(l)$ and $z \notin \mathcal{V}ar(r)$, let $\mu(z)$ be an element arbitrarily chosen in $L(\mathcal{A}_x^\sigma)$ (which is by

hypotheses non empty). Consequently, for every $x \in \mathcal{V}ar(l)$ and every position p_x of l such that $l(p_x) = x$,

$$\mu(x) \rightarrow_{\mathcal{A}}^{*} \sigma(p_x) \tag{10}$$

Thus

$$l\mu(x) \rightarrow_{\mathcal{A}}^{*} l\sigma \tag{11}$$

Since $t \in L(\mathcal{A}_{r,\sigma,q})$ one has

$$t \rightarrow_{\Delta_1}^{*} t[q_\varepsilon]_p \rightarrow_{\Delta_2}^{*} q_f^+ \quad \text{with } q_f^+ \in F_{r,\sigma,q}. \tag{12}$$

Since $t[q_\varepsilon]_p \rightarrow_{\Delta_2}^{*} q_f^+$,

$$t[q]_p \rightarrow_{\mathcal{A}}^{*} q_f. \tag{13}$$

Using (11) and (13) one has

$$t[l\mu]_p \rightarrow_{\mathcal{A}}^{*} t[l\sigma]_p \rightarrow_{\mathcal{A}}^{*} t[q]_p \rightarrow_{\mathcal{A}}^{*} q_f. \tag{14}$$

Therefore $t \in \mathcal{R}(L(\mathcal{A}))$, proving the claim.

6.9 Example for Proposition 7

Let $\mathcal{F} = \{\bot, a, b, f\}$ where $ar(\bot) = 0$, $ar(a) = ar(b) = 1$ and $ar(f) = 2$. We consider the tree automaton $\mathcal{A}$ whose set of state is $\{s_0, s_1, s_2, s_4, s_f\}$, whose final state is s_f and whose set of transition Δ is

$\bot \rightarrow s_0$	$\bot \rightarrow s_2$	$f(s_1, s_3) \rightarrow s_4$
$a(s_0) \rightarrow s_0$	$b(s_2) \rightarrow s_3$	$f(s_3, s_4) \rightarrow s_f$
$b(s_0) \rightarrow s_0$	$a(s_3) \rightarrow s_3$	
$a(s_0) \rightarrow s_1$	$b(s_3) \rightarrow s_3$	

The terms which can be reduced to s_1, are those of $L_1 = a(\{a, b\}^*(\bot))$. The terms which can be reduced to s_3 are those of $L_2 = \{a, b\}^*(b(\bot))$. The language accepted by $\mathcal{A}$ is $f(L_2, f(L_1, L_2))$.

Let $\mathcal{R} = \{f(x, x) \rightarrow a(f(x, b(x))\}$. We will construct a TAGED accepting $\mathcal{R}(L(\mathcal{A}))$ using the method developed in the proof of Proposition 7.

The only variable occurring in $f(x, x)$ is x. So we are looking for substitutions such that $L(\mathcal{A}, \sigma(1)) \cap L(\mathcal{A}, \sigma(2)) \neq \emptyset$ and $f(\sigma(1), \sigma(2)) \rightarrow_{\mathcal{A}}^{*} q$, where q is a state of $\mathcal{A}$ The second condition implies that only substitutions σ_0 and σ_1 defined by $\sigma_0(1) = s_1$, $\sigma_0(2) = s_3$ and $\sigma_1(1) = s_3$, $\sigma(2) = s_4$ have to be considered. Now σ_1 doesn't satisfy the first condition on languages intersection. It follows that $\mathcal{R}(L(\mathcal{A})) = L(\mathcal{A}_{a(f(x,b(x))),\sigma_0,s_4})$.

Since $L(\mathcal{A}, \sigma_0(1)) \cap L(\mathcal{A}, \sigma_0(2)) = L_1 \cap L_2 = a(\{a, b\}^*(b(\bot)))$, one can choose for $\mathcal{A}_x^{\sigma_0}$ the automaton whose set of states is $\{s_5, s_6, s_7\}$, whose final state is $\{s_7\}$ and whose transitions are $\bot \rightarrow s_5$, $b(s_5) \rightarrow s_6$, $a(s_6) \rightarrow s_6$, $b(s_6) \rightarrow s_6$ and $a(s_6) \rightarrow s_7$.

The automaton $\mathcal{A}_{a(f(x,b(x))),\sigma_0,s_4}$ is defined by:

- Its set of states is
  $\{s_0, s_1, s_2, s_3, s_4, s_f\} \cup \{q_1, q_{1.2}, q_\varepsilon\} \cup \{s_0^+, s_1^+, s_2^+, s_3^+, s_4^+, s_f^+\} \cup \{s_5, s_6, s_7\}$,
- $E_{a(f(x,b(x))),\sigma_0,s_4} = (s_7, s_7)$,
- Its set of final states is $\{s_f^+\}$,
- Its set of transition is $\Delta \cup \Delta_1 \cup \Delta_2$, with $\Delta_1 = \{b(s_7) \to q_{1.2}, f(s_7, q_{1.2}) \to q_1, a(q_1) \to q_\varepsilon\} \cup \{\bot \to s_5, b(s_5) \to s_6, a(s_6) \to s_6, b(s_6) \to s_6, a(s_6) \to s_7\}$ and Δ_2 is the union of $\{f(s_3, q_\varepsilon) \to s_f^+\}$ and of the following set:

$$\begin{array}{lll} a(s_0^+) \to s_0^+ & b(s_2^+) \to s_3 & f(s_3^+, s_4) \to s_f^+ \\ b(s_0^+) \to s_0^+ & a(s_3^+) \to s_3 & f(s_1, s_3^+) \to s_4^+ \\ a(s_0^+) \to s_1^+ & b(s_3^+) \to s_3 & f(s_3, s_4^+) \to s_f^+ \\ & & f(s_1^+, s_3) \to s_4^+ \end{array}$$