Verifying Temporal Regular properties of Abstractions of Term Rewriting Systems

Benoît Boyer
Université Rennes 1, France
Benoit.Boyer@irisa.fr

Thomas Genet
Université Rennes 1, France
Thomas.Genet@irisa.fr

The tree automaton completion is an algorithm used for proving safety properties of systems that can be modeled by a term rewriting system. This representation and verification technique works well for proving properties of infinite systems like cryptographic protocols or more recently on Java Bytecode programs. This algorithm computes a tree automaton which represents a (regular) over approximation of the set of reachable terms by rewriting initial terms. This approach is limited by the lack of information about rewriting relation between terms. Actually, terms in relation by rewriting are in the same equivalence class: there are recognized by the same state in the tree automaton.

Our objective is to produce an automaton embedding an abstraction of the rewriting relation sufficient to prove temporal properties of the term rewriting system.

We propose to extend the algorithm to produce an automaton having more equivalence classes to distinguish a term or a subterm from its successors w.r.t. rewriting. While ground transitions are used to recognize equivalence classes of terms, ε-transitions represent the rewriting relation between terms. From the completed automaton, it is possible to automatically build a Kripke structure abstracting the rewriting sequence. States of the Kripke structure are states of the tree automaton and the transition relation is given by the set of ε-transitions. States of the Kripke structure are labelled by the set of terms recognized using ground transitions. On this Kripke structure, we define the Regular Linear Temporal Logic (R-LTL) for expressing properties. Such properties can then be checked using standard model checking algorithms. The only difference between LTL and R-LTL is that predicates are replaced by regular sets of acceptable terms.

1 Introduction

Our main objective is to formally verify programs or systems modeled using Term Rewriting Systems. In a previous work [2], we have shown that it is possible to translate a Java bytecode program into a Term Rewriting System (TRS). In this case, terms model Java Virtual Machine (JVM) states and the execution of bytecode instructions is represented by rewriting, according to the small-step semantics of Java. An interesting point of this approach is the possibility to classify rewriting rules. More precisely, there is a strong relation between the position of rewriting in a term and the semantics of the executed transition on the corresponding state. For the case of Java bytecode, since a term represents a JVM state, rewriting at the top-most position corresponds to manipulations of the call stack, i.e. it simulates a method call or method return. On the other hand, since the left-most subterm represents the execution context of the current method (so called frame), rewriting at this position simulates the execution of the code of *this* method. Hence, by focusing on rewriting at a particular position, it is possible to analyse a Java program at the method call level (inter procedural control flow) or at the instruction level (local control flow). The contribution of this paper is dual. First, we propose an abstract rewriting relation to characterize the rewriting paths at a particular depth in terms. Second, we propose an algorithm which builds a tree automaton recognizing this relation between terms. Thus, it is possible for instance to build a tree automaton recognizing the graph of method calls by abstracting the rewriting relation for the top-most position of JVM terms.

The verification technique used in [2], called Tree Automata Completion [5], is able to finitely over-approximate the set of reachable terms, i.e. the set of all reachable states of the JVM. However, this technique lacks precision in the sense that it makes no difference between all those reachable terms. Due to the approximation algorithm, all reachable terms are considered as equivalent and the execution ordering is lost. In particular, this prevents to prove temporal properties of such models. However, using approximations makes it possible to prove unreachability properties of infinite state systems.

In this preliminary work, we propose to improve the Tree Automata Completion method so as to prove temporal properties of a TRS representing a finite state system. The first step is to refine the algorithm so as to produce a tree automaton keeping an approximation of the rewriting relation between terms. Then, in a second step, we propose a way to check LTL-like formulas on this tree automaton.

2 Preliminaries

Comprehensive surveys can be found in [1] for rewriting, and in [4, 7] for tree automata and tree language theory.

Let $\mathscr{F}$ be a finite set of symbols, each associated with an arity function, and let $\mathscr{X}$ be a countable set of variables. $\mathscr{T}(\mathscr{F},\mathscr{X})$ denotes the set of terms, and $\mathscr{T}(\mathscr{F})$ denotes the set of ground terms (terms without variables). The set of variables of a term t is denoted by $\mathscr{V}ar(t)$. A substitution is a function σ from $\mathscr{X}$ into $\mathscr{T}(\mathscr{F},\mathscr{X})$, which can be uniquely extended to an endomorphism of $\mathscr{T}(\mathscr{F},\mathscr{X})$. A position p for a term t is a word over $\mathbb{N}$. The empty sequence λ denotes the top-most position. The set $\mathscr{P}os(t)$ of positions of a term t is inductively defined by:

- $\mathscr{P}os(t) = \{\lambda\}$ if $t \in \mathscr{X}$
- $\mathscr{P}os(f(t_1,\dots,t_n)) = \{\lambda\} \cup \{i.p \mid 1 \leq i \leq n$ and $p \in \mathscr{P}os(t_i)\}$

If $p \in \mathscr{P}os(t)$, 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. A term rewriting system (TRS) $\mathscr{R}$ is a set of *rewrite rules* $l \to r$, where $l,r \in \mathscr{T}(\mathscr{F},\mathscr{X})$, $l \notin \mathscr{X}$, and $\mathscr{V}ar(l) \supseteq \mathscr{V}ar(r)$. The TRS $\mathscr{R}$ induces a rewriting relation $\to_{\mathscr{R}}$ on terms as follows. Let $s,t \in \mathscr{T}(\mathscr{F},\mathscr{X})$ and $l \to r \in \mathscr{R}$, $s \to^{p}_{\mathscr{R}} t$ denotes that there exists a position $p \in \mathscr{P}os(t)$ and a substitution σ such that $s|_p = l\sigma$ and $r = s[r\sigma]_p$. Note that the rewriting position p can generally be omitted, i.e. we generally write $s \to_{\mathscr{R}} t$. The reflexive transitive closure of $\to_{\mathscr{R}}$ is denoted by $\to^{*}_{\mathscr{R}}$. The set of $\mathscr{R}$-descendants of a set of ground terms E is $\mathscr{R}^*(E) = \{t \in \mathscr{T}(\mathscr{F}) \mid \exists s \in E \text{ s.t. } s \to^{*}_{\mathscr{R}} t\}$.

The *verification technique* defined in [6, 5] is based on the approximation of $\mathscr{R}^*(E)$. Note that $\mathscr{R}^*(E)$ is possibly infinite: $\mathscr{R}$ may not terminate and/or E may be infinite. The set $\mathscr{R}^*(E)$ is generally not computable [7]. However, it is possible to over-approximate it [6, 5, 9] using tree automata, i.e. a finite representation of infinite (regular) sets of terms. In this verification setting, the TRS $\mathscr{R}$ represents the system to verify, sets of terms E and Bad respectively represent the set of initial configurations and the set of "bad" configurations that should not be reached. Using tree automata completion, we construct a tree automaton B whose language $\mathscr{L}(B)$ is such that $\mathscr{L}(B) \supseteq \mathscr{R}^*(E)$. If $\mathscr{L}(B) \cap Bad = \emptyset$ then this proves that $\mathscr{R}^*(E) \cap Bad = \emptyset$, and thus that none of the "bad" configurations is reachable. We now define tree automata.

Let Q be a finite set of symbols, with arity 0, called *states* such that $Q \cap \mathscr{F} = \emptyset$. $\mathscr{T}(\mathscr{F} \cup Q)$ is called the set of *configurations*.

Definition 1 (Transition, normalized transition, ε-transition)**.** *A* transition *is a rewrite rule $c \to q$, where c is a configuration i.e. $c \in \mathcal{T}(\mathcal{F} \cup Q)$ and $q \in Q$. A* normalized transition *is a transition $c \to q$ where $c = f(q_1, \ldots, q_n)$, $f \in \mathcal{F}$ whose arity is n, and $q_1, \ldots, q_n \in Q$. An ε-*transition *is a transition of the form $q \to q'$ where q and q' are states.*

Definition 2 (Bottom-up nondeterministic finite tree automaton)**.** *A bottom-up nondeterministic finite tree automaton (tree automaton for short) is a quadruple $A = \langle \mathcal{F}, Q, Q_F, \Delta \cup \Delta_\varepsilon \rangle$, where $Q_F \subseteq Q$, Δ is a set of normalized transitions and Δ_ε is a set of ε-transitions.*

The *rewriting relation* on $\mathcal{T}(\mathcal{F} \cup Q)$ induced by the transitions of A (the set $\Delta \cup \Delta_\varepsilon$) is denoted by $\to_{\Delta \cup \Delta_\varepsilon}$. When Δ is clear from the context, $\to_{\Delta \cup \Delta_\varepsilon}$ will also be denoted by $\to_A$. We also introduce $\to_A^{\not\varepsilon}$ the *transitive relation* which is induced by the set Δ alone.

Definition 3 (Recognized language, canonical term)**.** *The tree language recognized by A in a state q is $\mathcal{L}(A,q) = \{t \in \mathcal{T}(\mathcal{F}) \mid t \to_A^* q\}$. The language recognized by A is $\mathcal{L}(A) = \bigcup_{q \in Q_F} \mathcal{L}(A,q)$. A tree language is regular if and only if it can be recognized by a tree automaton. A term t is a* canonical term *of the state q, if $t \to_A^{\not\varepsilon} q$.*

Example 1. *Let A be the tree automaton $\langle \mathcal{F}, Q, Q_F, \Delta \rangle$ such that $\mathcal{F} = \{f, g, a\}$, $Q = \{q_0, q_1, q_2\}$, $Q_F = \{q_0\}$, $\Delta = \{f(q_0) \to q_0, g(q_1) \to q_0, a \to q_1, b \to q_2\}$ and $\Delta_\varepsilon = \{q_2 \to q_1\}$. In Δ, transitions are* normalized. *A transition of the form $f(g(q_1)) \to q_0$ is not normalized. The term $g(a)$ is a term of $\mathcal{T}(\mathcal{F} \cup Q)$ (and of $\mathcal{T}(\mathcal{F})$) and can be rewritten by Δ in the following way: $g(a) \to_A^{\not\varepsilon} g(q_1) \to_A^{\not\varepsilon} q_0$. Hence $g(a)$ is a canonical term of q_1. Note also that $b \to_A q_2 \to_A q_1$. Hence, $\mathcal{L}(A, q_1) = \{a, b\}$ and $\mathcal{L}(A) = \mathcal{L}(A, q_0) = \{g(a), g(b), f(g(a)), f(f(g(b))), \ldots\} = \{f^*(g([a|b]))\}$.*

3 The Tree Automata Completion with ε-transitions

Given a tree automaton A and a TRS $\mathcal{R}$, the tree automata completion algorithm, proposed in [6, 5], computes a *tree complete automaton* $A_\mathcal{R}^*$ such that $\mathcal{L}(A_\mathcal{R}^*) = \mathcal{R}^*(\mathcal{L}(A))$ when it is possible (for some of the classes of TRSs where an exact computation is possible, see [5]), and such that $\mathcal{L}(A_\mathcal{R}^*) \supseteq \mathcal{R}^*(\mathcal{L}(A))$ otherwise. In this paper, we only consider the exact case.

The tree automata completion with ε-transtions works as follow. From $A = A_\mathcal{R}^0$ completion builds a sequence $A_\mathcal{R}^0.A_\mathcal{R}^1 \ldots A_\mathcal{R}^k$ of automata such that if $s \in \mathcal{L}(A_\mathcal{R}^i)$ and $s \to_\mathcal{R} t$ then $t \in \mathcal{L}(A_\mathcal{R}^{i+1})$. Transitions of $A_\mathcal{R}^i$ are denoted by the set $\Delta^i \cup \Delta_\varepsilon^i$. Since for every tree automaton, there exists a deterministic tree automaton recognizing the same language, we can assume that initially A has the following properties:

Property 1 ($\to^{\not\varepsilon}$ deterministic)**.** *If Δ contains two normalized transitions of the form $f(q_1, \ldots, q_n) \to q$ and $f(q_1, \ldots, q_n) \to q'$, it means $q = q'$. This ensures that the rewriting relation $\to^{\not\varepsilon}$ is deterministic.*

Property 2. *For all state q there is at most one normalized transition $f(q_1, \ldots, q_n) \to q$ in Δ. This ensures that if we have $t \to^{\not\varepsilon} q$ and $t' \to^{\not\varepsilon} q$ then $t = t'$.*

If we find a fixpoint automaton $A_\mathcal{R}^k$ such that $\mathcal{R}^*(\mathcal{L}(A_\mathcal{R}^k)) = \mathcal{L}(A_\mathcal{R}^k)$, then we note $A_\mathcal{R}^* = A_\mathcal{R}^k$ and we have $\mathcal{L}(A_\mathcal{R}^*) \supseteq \mathcal{R}^*(\mathcal{L}(A_\mathcal{R}^0))$ [5]. To build $A_\mathcal{R}^{i+1}$ from $A_\mathcal{R}^i$, we achieve a *completion step* which consists of finding *critical pairs* between $\to_\mathcal{R}$ and $\to_{A_\mathcal{R}^i}$. To define the notion of critical pair, we extend the definition of substitutions to the terms of $\mathcal{T}(\mathcal{F} \cup Q)$. For a substitution $\sigma : \mathcal{X} \mapsto Q$ and a rule $l \to r \in \mathcal{R}$, a critical pair is an instance $l\sigma$ of l such that there exists $q \in Q$ satisfying $l\sigma \to_{A_\mathcal{R}^i}^* q$ and $l\sigma \to_\mathcal{R} r\sigma$. Note that since $\mathcal{R}$, $A_\mathcal{R}^i$ and the set Q of states of $A_\mathcal{R}^i$ are finite, there is only a finite number of critical pairs. For every critical pair detected between $\mathcal{R}$ and $A_\mathcal{R}^i$ such that we do not have a state q' for which $r\sigma \to_{A_\mathcal{R}^i}^{\not\varepsilon} q'$

$$\begin{tikzcd}
l\sigma \arrow[r, "\mathcal{R}"'] \arrow[d, "A^i_{\mathcal{R}}"', "*"] & r\sigma \arrow[d, "\not\varepsilon"', "A^{i+1}_{\mathcal{R}}"] \\
q & q' \arrow[l, "A^{i+1}_{\mathcal{R}}"]
\end{tikzcd}$$

Figure 1: A critical pair solved

and $q' \rightarrow q \in \Delta^i_\varepsilon$, the tree automaton $A^{i+1}_{\mathcal{R}}$ is constructed by adding new transitions $r\sigma \rightarrow^{\not\varepsilon} q'$ to Δ^i and $q' \rightarrow q$ to Δ^i_ε such that $A^{i+1}_{\mathcal{R}}$ recognizes $r\sigma$ in q, i.e. $r\sigma \rightarrow^*_{A^{i+1}_{\mathcal{R}}} q$, see Figure 1. It is important to note that we consider the critical pair only if the last step of the reduction $l\sigma \rightarrow^*_{A^i_{\mathcal{R}}} q$, is the last step of rewriting is not a ε-transition. Without this condition, the completion computes the transitive closure of the expected relation Δ_ε, and thus looses precision. The transition $r\sigma \rightarrow q'$ is not necessarily a normalized transition of the form $f(q_1,\ldots,q_n) \rightarrow q'$ and so it has to be normalized first. Instead of adding $r\sigma \rightarrow q'$ we add $\downarrow (r\sigma \rightarrow q')$ to transitions of Δ^i. Here is the $\downarrow$ function used to normalize transitions. Note that, in this function, transitions are normalized using new states of Q_{new}.

Definition 4 ($\downarrow$). *Let $A = \langle \mathcal{F}, Q, Q_F, \Delta \cup \Delta_\varepsilon \rangle$ be a tree automaton, Q_{new} a set of* new *states such that $Q \cap Q_{new} = \emptyset$, $s \in \mathcal{T}(\mathcal{F} \cup Q)$ and $q' \in Q$. The normalization of the transition $s \rightarrow q'$ is done in two mutually inductive steps. The first step denoted by $\downarrow (s \rightarrow q' \mid \Delta)$, we rewrite s by Δ until rewriting is impossible: we obtain a unique configuration t if Δ respects the property 1. The second step $\downarrow'$ is inductively defined by:*

- $\downarrow' (f(t_1,\ldots,t_n) \rightarrow q \mid \Delta) = \Delta \cup \{f(t_1,\ldots,t_n) \rightarrow q\}$ *if* $\forall i = 1\ldots n:\ t_i \in Q$
- $\downarrow' (f(t_1,\ldots,t_n) \rightarrow q \mid \Delta) = \downarrow (f(t_1,\ldots,q_i,\ldots,t_n) \rightarrow q \mid \ \downarrow' (t_i \rightarrow q_i \mid \Delta)\ )$ *where t_i is subterm s.t. $t_i \in \mathcal{T}(\mathcal{F} \cup Q) \setminus Q$ and $q_i \in Q_{new}$.*

Lemma 1. *If the property 1 holds for $A^i_{\mathcal{R}}$ then it holds also for $A^{i+1}_{\mathcal{R}}$.*

Intuition. The determinism of $\rightarrow^{\not\varepsilon}$ is preserved by Δ, since when a new set of transitions is added to Δ for a subterm t_i, we rewrite all other subterms t_j with the new Δ until rewriting is impossible before resuming the normalization. Then, if we try to add to Δ a transition $f(q_1,\ldots,q_n) \rightarrow q$ though there exists a transition $f(q_1,\ldots,q_n) \rightarrow q' \in \Delta$, it means that the configuration $f(q_1,\ldots,q_n)$ can be rewritten by Δ. This is a contradiction : when we resume the normalization all subterms t_i can not be rewritten by the current Δ. So, we never add a such transition to Δ. The normalization produces a new set of transitions Δ that preserves the property 1. □

It is very important to remark that the transition $q' \rightarrow q$ in Figure 1 creates an order between the language recognized by q and the one recognized by q'. Intuitively, we know that for all substitution $\sigma' : \mathcal{X} \rightarrow \mathcal{T}(\mathcal{F})$ such that $l\sigma'$ is a term recognized by q, it is rewritten by $\mathcal{R}$ into a canonical term ($r\sigma'$) of q'. By duality, the term $r\sigma'$ has a parent ($l\sigma'$) in the state q. Extending this reasoning, Δ_ε defines a relation between canonical terms. This relation follows rewriting steps at the top position and forgets rewriting in the subterms.

Definition 5 ($\dashrightarrow$). *Let $\mathcal{R}$ be a TRS. For all terms u v, we have $u \dashrightarrow_{\mathcal{R}} v$ iff there exists w such that $u \rightarrow^*_{\mathcal{R}} w$, $w \rightarrow^\lambda_{\mathcal{R}} v$ and there is not rewriting on top position λ on the sequence denoted by $u \rightarrow^*_{\mathcal{R}} w$.*

In the following, we show that the completion builds a tree automaton where the set Δ_ε is an *abstraction* $\dashrightarrow_{\mathcal{R}_i}$ of the rewriting relation $\rightarrow_{\mathcal{R}}$, for any relevant set $\mathcal{R}_i$.

Theorem 1 (Correctness)**.** *Let be $A^*_{\mathcal{R}}$ a complete tree automaton such that $q' \rightarrow q$ is a ε-transition of $A^*_{\mathcal{R}}$. Then, for all canonical terms u v of states q and q' respectively s.t. $q' \rightarrow q$, we have :*

$$\begin{tikzcd}
u \arrow[r, dashed, "\mathcal{R}"'] \arrow[d, "A^*_{\mathcal{R}}"', "\not\varepsilon"] & v \arrow[d, "A^*_{\mathcal{R}}"', "\not\varepsilon"] \\
q & q' \arrow[l]
\end{tikzcd}$$

First, we have to prove that the property 1 is preserved by completion. To prove theorem 1, we need a stronger lemma.

Lemma 2. *Let be $A^*_{\mathcal{R}}$ a complete tree automaton, q a state of $A^*_{\mathcal{R}}$ and $v \in \mathcal{L}(A^*_{\mathcal{R}}, q)$. Then, for all canonical term u of q, we have $u \rightarrow^*_{\mathcal{R}} v$.*

Proof sketch. The proof is done by induction on the number of completion steps to reach the post-fixpoint $A^*_{\mathcal{R}}$: we are going to show that if $A^i_{\mathcal{R}}$ respects the property of lemma 2, then $A^{i+1}_{\mathcal{R}}$ also does.

The initial $A^0_{\mathcal{R}}$ respects the expected property : we consider any state q and a canonical term t of q: since no completion step was done, $A^0_{\mathcal{R}}$ has no ε-transitions. It means that for all term $t' \rightarrow^{\not\varepsilon} q$. Thanks to the property 2, we have $t = t'$ and obviously $t \rightarrow^*_{\mathcal{R}} t'$.

Now, we consider the normalization of a transition of the form $r\sigma \rightarrow^{\not\varepsilon} q'$ such that $l\sigma \rightarrow^*_{A^i_{\mathcal{R}}} q$ with Δ the ground transition set and Δ_ε the ε-transition set of $A^i_{\mathcal{R}}$. We show that the property is true for all new states (including q'). Then, in a second time, we will show that it is true for state q, if we add the second transitions of completion: $q' \rightarrow q$.

Let us focus on the normalization of $\downarrow' (r\sigma \rightarrow q' \mid \Delta)$ where for any existing state q and for all u $v \in \mathcal{T}(\mathcal{F})$ such that $v \rightarrow_{\Delta\cup\Delta_\varepsilon} q$ and $u \rightarrow_{\Delta} q$, we have $u \rightarrow^*_{\mathcal{R}} v$. We show that if we have $\Delta' = \downarrow' (t \rightarrow q' \mid \Delta)$, for all u $v \in \mathcal{T}(\mathcal{F})$ such that $v \rightarrow_{\Delta'\cup\Delta_\varepsilon} q'$ and $u \rightarrow_{\Delta'} q$, we have $u \rightarrow^*_{\mathcal{R}} v$. The induction is done over the decreasing number of symbols of $\mathcal{F}$ used to build t.

First case $\downarrow' (t \rightarrow q \mid \Delta)$ where $t = f(q_1, \ldots, q_n)$: we define Δ' by adding the transition $f(q_1, \ldots, q_n) \rightarrow q$ to Δ, where q is a new state. Then, for all substitution $\sigma' : Q \mapsto \mathcal{T}(\mathcal{F})$ such that $t\sigma' \rightarrow_{\Delta\cup\Delta_\varepsilon} q$, and all substitution $\sigma'' : Q \mapsto \mathcal{T}(\mathcal{F})$ such that $t\sigma'' \rightarrow_{\Delta'} q$ we aim at proving that $t\sigma'' \rightarrow^*_{\mathcal{R}} t\sigma'$. Since each state q_i is already defined, using the hypothesis on Δ we deduce that $\sigma''(q_i) \rightarrow^*_{\mathcal{R}} \sigma'(q_i)$. This implies that $t\sigma'' \rightarrow^*_{\mathcal{R}} t\sigma'$, the property also holds for Δ'.

Second case $\downarrow' (t \rightarrow q \mid \Delta)$ where $f(t_1, \ldots, t_n)$: we select t_i a subterm of t, obviously the number of symbols is strictly lower to the number of symbols of t. By induction for the normalization of $\downarrow' (t_i \rightarrow q_i \mid \Delta)$ we have a new set Δ' that respects the expected property. Then, we normalize $t' = f(t'_1, \ldots, q_i, \ldots, t'_n)$, the term obtained after rewriting with Δ' thanks to $\downarrow$. Since $t_i \notin Q$, the number of symbols in $f(t_1, \ldots, q_i, \ldots, t_n)$ is strictly lower to the number of symbols of t. By rewriting the term $f(t_1, \ldots, q_i, \ldots, t_n)$ with Δ', we obtain the term t' for which the number of symbols is lower or equal to the one of $f(t_1, \ldots, q_i, \ldots, t_n)$. Since t' has a decreasing number of symbols and Δ' respects the property we can deduce by induction that we have $\Delta'' = \downarrow' (t' \rightarrow q \mid \Delta')$ such that for all $v \rightarrow_{\Delta''\cup\Delta_\varepsilon} q'$ and $u \rightarrow_{\Delta''} q$, $u \rightarrow^*_{\mathcal{R}} v$.

So, we conclude that the normalization $\downarrow' (r\sigma \rightarrow q' \mid \Delta)$ computes Δ' the set of ground transitions for $A^{i+1}_{\mathcal{R}}$. For all terms u v such that $u \rightarrow_{\Delta'\cup\Delta_\varepsilon} q'$ and $u \rightarrow_{\Delta'} q'$ we have $u \rightarrow^*_{\mathcal{R}} v$.

Now, let us consider the second added transition $q' \rightarrow q$ to Δ_ε, all canonical terms $r\sigma''$ of q', and all terms $l\sigma''' \in \mathcal{L}(A^i_{\mathcal{R}}, q)$ such that $l\sigma''' \rightarrow_{\mathcal{R}} r\sigma'''$ and $r\sigma''' = r\sigma''$. By hypothesis on $A^i_{\mathcal{R}}$, we know that every canonical term u of q we have $u \rightarrow^*_{\mathcal{R}} l\sigma'''$. By transitivity, we have $u \rightarrow^*_{\mathcal{R}} r\sigma''$. The last step consists in proving that for all terms of all states of $A^{i+1}_{\mathcal{R}}$, the property holds: this can be done by induction on the depth of the recognized terms. □

The theorem 1 is shown by considering the introduction of the transition $q' \to q$. By construction, there exists a substitution $\sigma : \mathscr{X} \mapsto Q$ and a rule $l \to r \in \mathscr{R}$ such that we have $l\sigma \to^*_{A^*_{\mathscr{R}}} q$ and $r\sigma \to^{\not\varepsilon}_{A^*_{\mathscr{R}}} q'$. We consider all substitution $\sigma' : \mathscr{X} \mapsto \mathscr{T}(\mathscr{F})$ such that for each variable $x \in \mathscr{V}(l)$, $\sigma'(x)$ is a canonical term of the state $\sigma(x)$. Obviously, using the result of the lemma 2, for all canonical term u of q we have $u \to^*_{\mathscr{R}} l\sigma'$. Since the last step of rewriting in the reduction $l\sigma \to^*_{A^*_{\mathscr{R}}} q$ is not a ε-transition, we also deduce that $l\sigma'$ is not produced by a rewriting at the top position of u whereas it is the case for $r\sigma'$ and we have $u \dashrightarrow_{\mathscr{R}} r\sigma'$.

Theorem 2 (Completeness)**.** *Let $A^*_{\mathscr{R}}$ be a complete tree automaton, q, q' states of $A^*_{\mathscr{R}}$ and $u, v \in \mathscr{T}(\mathscr{F})$ such that u is a canonical term of q and v is a canonical term of q'. If $u \dashrightarrow_{\mathscr{R}} v$ then there exists a ε-transition $q' \to q$ in $A^*_{\mathscr{R}}$.*

Proof sketch. By definition of $u \dashrightarrow_{\mathscr{R}} v$ there exists a term w such that $u \to^*_{\mathscr{R}} w$ and and there exists a rule $l \to r \in \mathscr{R}$ and a substitution $\sigma : \mathscr{X} \mapsto \mathscr{T}(\mathscr{F})$ such that $w = l\sigma$ and $v = r\sigma$. Since $A^*_{\mathscr{R}}$ is a complete tree automaton, it is closed by rewriting. This means that any term obtained by rewriting any term of $\mathscr{L}(A^*_{\mathscr{R}}, q)$ is also in $\mathscr{L}(A^*_{\mathscr{R}}, q)$. This property is true in particular for the terms u and w. Since w is rewritten in q by transitions of $A^*_{\mathscr{R}}$, we can define a second substitution $\sigma' : \mathscr{X} \mapsto Q$ such that $l\sigma \to^*_{A^*_{\mathscr{R}}} l\sigma' \to^*_{A^*_{\mathscr{R}}} q$. Using again the closure property of $A^*_{\mathscr{R}}$, we know that the critical pair $l\sigma' \to_{\mathscr{R}} r\sigma'$ and $l\sigma' \to^*_{A^*_{\mathscr{R}}} q$ is solved by adding the transitions $r\sigma' \to^{\not\varepsilon}_{A^*_{\mathscr{R}}} q''$ and $q'' \to q$. Since the property 1 is preserved by completion steps, we can deduce that $q'' = q'$ which means $q' \to q$. □

Example 2. *To illustrate this result, we give a completed tree automaton for a small TRS. We define $\mathscr{R}$ as the union of the two sets of rules $\mathscr{R}_1 = \{a \to b,\ b \to c\}$ and $\mathscr{R}_2 = \{f(c) \to g(a),\ g(c) \to h(a),\ h(c) \to f(a)\}$. We define initial set $E = \{f(a)\}$. We obtain the following tree automaton fixpoint :*

$$A^*_{\mathscr{R}} = \left\langle Q_F = \{q_f\}, \quad \Delta = \left\{ \begin{array}{rcl} a & \to & q_a \\ b & \to & q_b \\ c & \to & q_c \\ f(q_a) & \to & q_f \\ g(q_a) & \to & q_g \\ h(q_a) & \to & q_h \end{array} \right\} \Delta_\varepsilon = \left\{ \begin{array}{rcl} q_b & \to & q_a \\ q_c & \to & q_b \\ q_g & \to & q_f \\ q_h & \to & q_g \\ q_f & \to & q_h \end{array} \right\} \right\rangle$$

If we consider the transition $q_h \to q_g$, and its canonical terms $h(a)$ and $g(a)$ respectively, we can deduce $g(a) \dashrightarrow_{\mathscr{R}} h(a)$. This is obviously an abstraction since we have $g(a) \to^1_{\mathscr{R}} g(b) \to^1_{\mathscr{R}} g(c) \to^\lambda_{\mathscr{R}} h(a)$.

In the following, we use the notation $\dashrightarrow_{\mathscr{R}_i}$ to specify the relation for a relevant subset $\mathscr{R}_i$ of $\mathscr{R}$. For instance, $u \dashrightarrow_{\mathscr{R}_i} v$ denotes that there exists w such that $u \to^*_{\mathscr{R}} w$ with no rewriting at the λ position of u and $w \to^\lambda_{\mathscr{R}_i} v$. In example 2, we can say that $g(a) \dashrightarrow_{\mathscr{R}_2} h(a)$.

4 From Tree Automaton to Kripke Structure

Let $A^*_{\mathscr{R}} = \langle \mathscr{T}(\mathscr{F}), Q, Q_F, \Delta \cup \Delta_\varepsilon \rangle$ be a complete tree automaton, for a given TRS $\mathscr{R}$ and an initial language recognized by A. A Kripke structure is a four tuple $K = (S, S_0, R, L)$ where S is a set of states, $S_0 \subseteq S$ initial states, $R \subseteq S \times S$ a left-total transition relation and L a function that labels each state with a set of predicates which are true in that state. In our case, the set of true predicates is a regular set of terms.

Definition 6 (Labelling Function)**.** *Let $A_P = \langle \mathcal{T}(\mathcal{F}), Q, \Delta \rangle$ be the structure defined from $A^*_{\mathcal{R}}$ by removing ε-transitions and final states. We define the labelling function $L : q \mapsto \langle \mathcal{T}(\mathcal{F}), Q, \{q\}, \Delta \rangle$ as the function which associates to a state q the automaton A_P where q is the unique final state. We obviously have the property for all state state q :*

$$\forall t \in \mathcal{L}(L(q)), \quad t \rightarrow^{\not\varepsilon}_{A^*_{\mathcal{R}}} q$$

Now, we can build the Kripke structure for the subset $\mathcal{R}_i$ of $\mathcal{R}$ on which we want to prove some temporal properties.

Definition 7 (Construction of a Kripke Structure)**.** *We build the 4-tuple (S, S_0, R, L) from a tree automaton such that we have $S = Q$, $S_0 \subseteq S$ is a set of initial states, $R(q, q')$ if $q' \rightarrow q \in \Delta_\varepsilon$ and the labelling function L as just defined previously.*

Kripke structures must have a complete relation R. For any state q whose have no successor by R, we had a loop such that $R(q,q)$ holds. Note that this is a classical transformation of Kripke structures [3]. A Kripke structure is parametrized by the set S_0. It defines which connected component of R we are interested to analyze. For instance, to analyze the abstract rewriting at the top position of terms in $\mathcal{L}(A^*_{\mathcal{R}})$, we define set $S_0 = Q_F$ (the set of final states of $A^*_{\mathcal{R}}$), since all canonical terms of final states are initial terms. For all abstract rewriting at a deeper position p, we need to define a set Sub of initial subterms considered as the beginning of the rewriting at the position p. Then the set S_0 will be defined as $S_0 = \{q \mid \exists t \in Sub,\ t \rightarrow^{\not\varepsilon}_{A^*_{\mathcal{R}}} q\}$.

Kripke structure models exactly the abstract rewriting relation $\dashrightarrow^*_{\mathcal{R}_i}$ for the corresponding subset $\mathcal{R}_i \subseteq \mathcal{R}$.

Theorem 3. *Le be $K = (S, S_0, R, L)$ a Kripke structure built from $A^*_{\mathcal{R}}$. For any states s, s' such that $R(s,s')$ holds, there exists two terms $u \in L(s)$ and $v \in L(s')$ such that $u \dashrightarrow_{\mathcal{R}_i} v$.*

Proof. Here, the proof is quite trivial. It is a consequence of the theorem 1 which can be applied on the relation R of the Kripke structure. □

In Example 2, if we want to verify properties of $\mathcal{R}_1$ or $\mathcal{R}_2$, we need to consider a different subset of Δ_ε corresponding to the abstraction of the relation rewriting $\dashrightarrow_{\mathcal{R}_i}$. Figures 2 and 3 show the Kripke structures corresponding to those abstractions. Note that in figure 2, a loop is needed on state c to have a total relation for K_1.

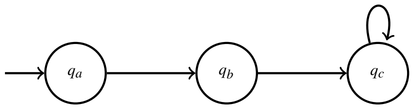

Figure 2: Kripke structure K_1 for $\dashrightarrow_{\mathcal{R}_1}$

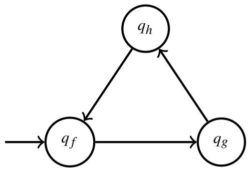

Figure 3: Kripke structure K_2 for $\dashrightarrow_{\mathcal{R}_2}$

The set S_0 of initial states depends of the abstract rewriting relation selected. For example, if we want to analyze $\dashrightarrow_{\mathcal{R}_2}$ (or $\dashrightarrow_{\mathcal{R}_1}$), we define $S_0 = \{q_f\}$ (resp. $S_0 = \{q_a\}$).

5 Verification of R-LTL properties

To express our properties, we propose to define the Regular Linear Temporal Logic (R-LTL). R-LTL is LTL where predicates are replaced by a tree automaton. The language of such a tree automaton characterizes a set of admissible terms. A state q of a Kripke structure validates the atomic property P characterized by a tree automaton A_P if and only if one term recognized by $L(q)$ must be recognized by A_P to satisfy the property. More formally:

$$K(Q,\ Q_F,\ R,\ L),\ q \models P \quad \iff \quad \mathscr{L}(L(q)) \cap \mathscr{L}(A_P) \neq \emptyset$$

We also add the operators ($\wedge$, $\vee$, $\neg$, $\mathbf{X}$, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{U}$, $\mathbf{R}$) with their standard semantics as in LTL to keep the expressiveness of the temporal logic. More information about these operators can be found in [3]. Note that temporal properties do not range over the rewriting relation $\rightarrow_{\mathscr{R}}$ but over its abstraction $\dashrightarrow_{\mathscr{R}}$. It means that the semantics of the temporal operators has to be interpreted w.r.t. this specific relation. For example, the formula $\mathbf{G}(\{f(a)\} \implies \mathbf{X}\{g(a)\})$ on K_2 (for more clarity, we note predicates as sets of terms): the formula has to be interpreted as : for all $q\ q'$, if $K_2,\ q \models \{f(a)\}$ and $R(q,q')$ then we have $K_2,\ q' \models \{g(a)\}$. In the rewriting interpretation the only term u such that $f(a) \dashrightarrow_{\mathscr{R}_2} u$ is $u = g(a)$.

We use the Büchi automata framework to perform model checking. A survey of this technique can be found in the chapter 9 of [3]. LTL (or R-LTL) formulas and Kripke structures can be translated into Büchi automata. We construct two Büchi automata : B_K obtained from the Kripke structure and B_L defined by the LTL formula. Since the set of behaviors of the Kripke structure is the language of the automaton B_K, the Kripke structure satisfies the R-LTL formula if all its behaviors are recognized by the automaton B_L. It means checking $\mathscr{L}(B_K) \subseteq \mathscr{L}(B_L)$. For this purpose, we construct the automaton $\overline{B_L}$ that recognizes the language $\overline{\mathscr{L}(B_L)}$ and we check the emptiness of the automaton $B_\cap$ that accepts the intersection of languages $\mathscr{L}(B_K)$ and $\overline{\mathscr{L}(B_L)}$. If this intersection is empty, the term rewriting system satisfies the property. This is the standard model-checking technique.

B_M and B_K are classically defined as 5-tuples: alphabet, states, initial states, final states and transition relation. Generally, the alphabet of Büchi automata is a set of predicates. Since we use here tree automata to define predicates, the alphabet of B_K and B_L is Σ the set of tree automata that can be defined over $\mathscr{T}(\mathscr{F})$. Actually, a set of behaviors is a word which describes a sequence of states: if $\pi = s_0 s_1 s_2 s_3 \ldots$ denotes a valid sequence of states in the Kripke structure, then the word $\pi' = L(s_0)L(s_1)L(s_2)\ldots$ is recognized by B_K. The algorithms used to build B_M and B_K can be found in [3].

The automaton intersection $B_\cap$ is obtained by computing the product of B_K by $\overline{B_L}$. By construction all states of B_K have to be final. Intuitively any infinite path over the Kripke structure must be recognized by B_K. This case allows to use a simpler version of the general Büchi automata product.

Definition 8 ($B_K \times \overline{B_L}$)**.** *The product of* $B_K = \langle \Sigma,\ Q,\ Q_i,\ \Delta,\ Q\rangle$ *by* $\overline{B_L} = \langle \Sigma,\ Q',\ Q'_i,\ \Delta',\ F\rangle$ *is defined as*

$$\langle \Sigma,\ Q \times Q',\ Q_i \times Q'_i,\ \Delta_\times,\ Q \times F\rangle$$

where $\Delta_\times$ *is the set of transitions* $(q_K, q_L) \xrightarrow{(A_K,A_L)} (q'_K, q'_L)$ *such that* $q_K \xrightarrow{A_K} q'_K$ *is a transition of* B_K *and* $q_L \xrightarrow{A_L} q'_L$ *is a transition of* $\overline{B_L}$*. Moreover, the transition is only valid if the intersection between the languages of* A_K *and* A_L *is non empty as expected by the satisfiability of the R-LTL atomic formula.*

Finally the emptiness of the language $\mathscr{L}(B_\cap)$ can be checked using the standard algorithm based on depth first search to check if final states are reachable.

Example 3. *To illustrate the approach, we propose to check the formula* $P = \mathbf{G}(\{f(a)\} \implies \mathbf{X}\{g(a)\})$ *on example 2. The automaton* $\overline{B_L}$ *(fig. 4) recognizes the negation of the formula P expressed as* $\mathbf{F}(\{f(a)\} \wedge \mathbf{X}\neg\{g(a)\})$ *and* B_K *(fig. 5) recognizes the all behaviors of the Kripke structure* K_2 *(fig. 3). The notation* A_α *denotes the tree automaton such that its language is described by* α *(*$A_{\neg g(a)}$ *recognizes the complement of the language* $\mathscr{L}(A_{g(a)})$ *and* A_* *recognizes all term in* $\mathscr{T}(\mathscr{F})$*). Figure 6 shows the result of* *intersection $B_\cap$ between B_K and $\overline{B_L}$. Only reachable states and valid transitions (labeled by non empty tree automata intersection) are showed. Since no reachable states of $B_\cap$ are final, its language is empty. It means that all behaviors of K_2 satisfy P : the only successor of $f(a)$ for the relation $\dashrightarrow_{\mathcal{R}_2}$ is $g(a)$.*

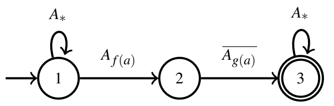

Figure 4: Automaton $\overline{B_L}$

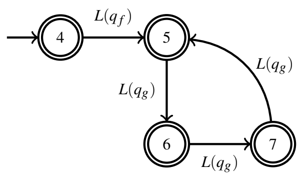

Figure 5: Automaton B_K

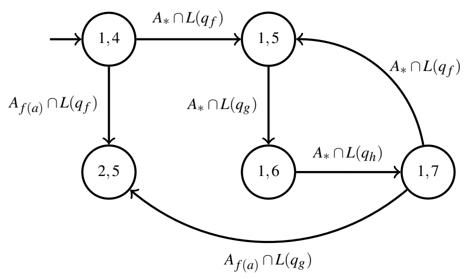

Figure 6: Automaton $B_\cap$

6 Conclusion, Discussion

In this paper, we show how to improve the tree automata completion mechanism to keep the ordering between reachable terms. This ordering was lost in the original algorithm [5]. Another contribution is the mechanism making it possible to prove LTL-like temporal properties on such abstractions of sets of reachable terms. The work presented here only deals with finite state systems and exact tree automata completion results. Future plans are to extend this result so as to prove temporal properties on over-approximations of infinite state systems. A similar objective has already been tackled in [8]. However, this was done in a pure rewriting framework where abstractions are more heavily constrained than in tree automata completion [5]. Hence, by extending LTL formula checking on tree automata over-approximations, we hope to ease the verification of temporal formula on infinite state systems.

Acknowledgements

Many thanks to Axel Legay and Vlad Rusu for fruitful discussions on this work and to anonymous referees for their comments.

References

[1] F. Baader & T. Nipkow (1998): *Term Rewriting and All That*. Cambridge University Press.

[2] Y. Boichut, T. Genet, T. Jensen & L. Leroux (2007): *Rewriting Approximations for Fast Prototyping of Static Analyzers*. In: *RTA*, *LNCS* 4533. Springer Verlag, pp. 48–62.

[3] Edmund M. Clarke, Orna Grumberg & Doron A. Peled (2000): *Model Checking*. MIT Press.

[4] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison & M. Tommasi (2008): *Tree Automata Techniques and Applications*. `http://tata.gforge.inria.fr`.

[5] G. Feuillade, T. Genet & V. Viet Triem Tong (2004): *Reachability Analysis over Term Rewriting Systems*. *Journal of Automated Reasonning* 33 (3-4), pp. 341–383. Available at `http://www.irisa.fr/lande/genet/publications.html`.

[6] T. Genet (1998): *Decidable Approximations of Sets of Descendants and Sets of Normal forms*. In: *Proc. 9th RTA Conf., Tsukuba (Japan)*, *LNCS* 1379. Springer-Verlag, pp. 151–165.

[7] R. Gilleron & S. Tison (1995): *Regular Tree Languages and Rewrite Systems*. *Fundamenta Informaticae* 24, pp. 157–175.

[8] J. Meseguer, M. Palomino & N. Martí-Oliet (2008): *Equational abstractions*. *TCS* 403(2-3), pp. 239–264.

[9] T. Takai (2004): *A Verification Technique Using Term Rewriting Systems and Abstract Interpretation*. In: *Proc. 15th RTA Conf., Aachen (Germany)*, *LNCS* 3091. Springer, pp. 119–133.