HAL open science

Verifying Temporal Regular properties of Abstractions of Term Rewriting Systems

Benoît Boyer, Thomas Genet

To cite this version:

Benoît Boyer, Thomas Genet. Verifying Temporal Regular properties of Abstractions of Term Rewriting Systems. RULE'09, 2009, France. pp.99-108. hal-00780491

HAL Id: hal-00780491

https://hal.science/hal-00780491v1

Submitted on 24 Jan 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Verifying Temporal Regular Properties on 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 on systems which can be modeled by a term rewriting system. This representation and verification technique works well for proving properties on 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 into the same state by the tree automaton.

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

We propose to extend the algorithm to produce a completed 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 a regular set 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 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 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 focusion 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 verification technique used in [2], called Tree Automata Completion [4], is abble 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

Submitted to Electronic Proceedings
in Theoretical Computer Science.

© Boyer & Genet

ordering is lost. This prevents, in particular, to prove temporal properties on such models. However, using approximations make it possible to prove unreachability properties on infinite state systems.

In this preliminary work, we propose to improve the Tree Automata Completion method so as to prove temporal properties on 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 [3, 6] 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 extended uniquely 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 $t \in \mathscr{X}$
- $\mathscr{P}os(f(t_1,\ldots,t_n)) = \{\lambda\} \cup \{i.p \mid 1 \leq i \leq n \text{ and } p \in \mathscr{P}os(t_i)\}$

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 write generally write $s \to_{\mathscr{R}} t$. The reflexive transitive closure of $\to_{\mathscr{R}}$ is denoted by $\to^\star_{\mathscr{R}}$. The set of $\mathscr{R}$-descendants of a set of ground terms E is $\mathscr{R}^*(E) = \{t \in \mathscr{T}(\mathscr{F}) \mid \exists s \in E \text{ s.t. } s \to^\star_{\mathscr{R}} t\}$.

The *verification technique* defined in [5, 4] 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 [6]. However, it is possible to over-approximate it [5, 4, 7] 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* represent respectively the set of initial configurations and the set of "bad" configurations that should not be reached. Then, using tree automata completion, we construct a tree automaton B whose language $\mathscr{L}(B)$ is such that $\mathscr{L}(B) \supseteq \mathscr{R}^*(E)$. Then 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 $Q \cap \mathscr{F} = \emptyset$. $\mathscr{T}(\mathscr{F} \cup \mathscr{Q})$ 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. $c \in \mathscr{T}(\mathscr{F} \cup \mathscr{Q})$ and $q \in Q$. A* normalized transition *is a transition $c \to q$ where $c = f(q_1,\ldots,q_n)$, $f \in \mathscr{F}$ 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 $A = \langle \mathscr{F}, Q, Q_F, \Delta \cup \Delta_\varepsilon \rangle$, 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 $\mathscr{T}(\mathscr{F} \cup \mathscr{Q})$ induced by the transitions of A (the set $\Delta \cup \Delta_\varepsilon$) is denoted by $\rightarrow_{\Delta \cup \Delta_\varepsilon}$. When Δ is clear from the context, $\rightarrow_{\Delta \cup \Delta_\varepsilon}$ will also be denoted by $\rightarrow_A$. We also introduce $\rightarrow_A^{\not\varepsilon}$ the 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 $\mathscr{L}(A,q) = \{t \in \mathscr{T}(\mathscr{F}) \mid t \rightarrow_A^\star q\}$. The language recognized by A is $\mathscr{L}(A) = \bigcup_{q \in Q_F} \mathscr{L}(A,q)$. 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 $t \rightarrow_A^{\not\varepsilon} q$.*

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

3 The Tree Automata Completion with ε-transitions

Given a tree automaton A and a TRS $\mathscr{R}$, the tree automata completion algorithm, proposed in [5, 4], computes a tree automaton $A^*_{\mathscr{R}}$ such that $\mathscr{L}(A^*_{\mathscr{R}}) = \mathscr{R}^*(\mathscr{L}(A))$ when it is possible (for some of the classes of TRSs where an exact computation is possible, see [4]) and such that $\mathscr{L}(A^*_{\mathscr{R}}) \supseteq \mathscr{R}^*(\mathscr{L}(A))$ otherwise. In this paper, we just consider the exact case.

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

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

If we find a fixpoint automaton $A^k_{\mathscr{R}}$ such that $\mathscr{R}^*(\mathscr{L}(A^k_{\mathscr{R}})) = \mathscr{L}(A^k_{\mathscr{R}})$, then we note $A^*_{\mathscr{R}} = A^k_{\mathscr{R}}$ and we have $\mathscr{L}(A^*_{\mathscr{R}}) = \mathscr{R}^*(\mathscr{L}(A^0_{\mathscr{R}}))$ [4]. To build $A^{i+1}_{\mathscr{R}}$ from $A^i_{\mathscr{R}}$, we achieve a *completion step* which consists of finding *critical pairs* between $\rightarrow_{\mathscr{R}}$ and $\rightarrow_{A^i_{\mathscr{R}}}$. To define the notion of critical pair, we extend the definition of substitutions to terms of $\mathscr{T}(\mathscr{F} \cup \mathscr{Q})$. For a substitution $\sigma : \mathscr{X} \mapsto Q$ and a rule $l \rightarrow r \in \mathscr{R}$, a critical pair is an instance $l\sigma$ of l such that there exists $q \in Q$ satisfying $l\sigma \rightarrow^*_{A^i_{\mathscr{R}}} q$ and $l\sigma \rightarrow_{\mathscr{R}} r\sigma$. Note that since $\mathscr{R}$, $A^i_{\mathscr{R}}$ and the set Q of states of $A^i_{\mathscr{R}}$ are finite, there is only a finite number of critical pairs. For every critical pair detected between $\mathscr{R}$ and $A^i_{\mathscr{R}}$ such that we do not have a state q' for which $r\sigma \rightarrow^{\not\varepsilon}_{A^i_{\mathscr{R}}} q'$ and $q' \rightarrow q \in \Delta^i_\varepsilon$, the tree automaton $A^{i+1}_{\mathscr{R}}$ 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 $A^{i+1}_{\mathscr{R}}$ recognizes $r\sigma$ in q, i.e. $r\sigma \rightarrow^*_{A^{i+1}_{\mathscr{R}}} q$, see Figure 1. However, the

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

Figure 1: A critical pair solved

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. Thus, 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 $A = \langle \mathscr{F}, Q, \mathscr{Q}_f, \Delta \cup \Delta_\varepsilon \rangle$ be a tree automaton, Q_{new} a set of* new *states such that $Q \cap Q_{new} = \emptyset$, $t \in \mathscr{T}(\mathscr{F} \cup \mathscr{Q})$ and $q \in Q$. The normalisation of the transition $s \to q'$ is done in two steps. 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:*

- $\downarrow' (t \to^{\varepsilon'} q') = \emptyset$ *if* $t = q'$,
- $\downarrow' (t \to^{\varepsilon'} q') = \{c \to q \mid c \to t \in \Delta\}$ *if* $t \in Q$ *and* $t \neq q'$
- $\downarrow' (f(t_1, \ldots, t_n) \to^{\varepsilon'} q) = \bigcup_{i=1\ldots n} \downarrow' (t_i \to^{\varepsilon'} q_i) \cup \{f(q_1, \ldots, q_n) \to^{\varepsilon'} q\}$ *where* $\forall i = 1 \ldots n : (t_i \in Q \Rightarrow q_i = t_i) \wedge (t_i \in \mathscr{T}(\mathscr{F} \cup \mathscr{Q}) \setminus Q \Rightarrow q_i \in Q_{new})$.

It is very important to remark that the introduction of the transition $q' \to q$ creates an order between the language recognized by q and the one recognized by q'. More precisely, we know that there exists a configuration $(l\sigma)$ of q which is rewritten by $\mathscr{R}$ into a canonical configuration $(r\sigma)$ of q'. By duality, the configuration $r\sigma$ has a parent $(l\sigma)$ in the state q.

In the following, we show that the completion builds an abstraction of the rewriting relation.

Definition 5 ($\dashrightarrow$) *Let $\mathscr{R}$ be a TRS. For all terms u v, we have $u \dashrightarrow_{\mathscr{R}} v$ iff there exists w such that $u \to^*_{\mathscr{R}} w$, $w \to^{\lambda}_{\mathscr{R}} v$ and there is not rewriting on top position λ between u and w.*

Theorem 1 *Let be $A^*_{\mathscr{R}}$ a complete tree automaton such that $q' \to q$ is a ε-transition of $A^*_{\mathscr{R}}$. Then, there exists two canonical terms u v such we have the following commutative diagram :*

$$\begin{tikzcd} u \arrow[r, dashed, "\mathscr{R}"'] \arrow[d, "A^*_{\mathscr{R}}"', "\varepsilon'"] & v \arrow[d, "A^*_{\mathscr{R}}"', "\varepsilon'"] \\ q & q' \arrow[l] \end{tikzcd}$$

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 :*

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

If we consider the transition $q_h \to q_g$, and their 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 $g(a) \to^1_{\mathscr{R}} g(b) \to^1_{\mathscr{R}} g(c) \to^{\lambda}_{\mathscr{R}} h(a)$.

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 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 $A_P = \langle \mathscr{T}(\mathscr{F}), Q, \Delta \rangle$ be the tree automaton defined from $A^*_{\mathscr{R}}$ by removing ε-transitions. We knowingly omit the set of final states. We define the labelling function $L(q) = \langle \mathscr{T}(\mathscr{F}), Q, \{q\}, \Delta \rangle$ 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 :*

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

Now, we can build the Kripke structure for the subset of $\mathscr{R}$ on which we want to prove some temporal properties. In Example 2, $\mathscr{R}$ is split : if we want verify properties on $\mathscr{R}_1$ or $\mathscr{R}_2$, we need to consider a different subset of Δ_ε corresponding to the abstraction of the relation rewriting $\dashrightarrow_{\mathscr{R}_i}$ as shown in figure 2 and 3.

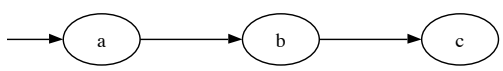

Figure 2: $\dashrightarrow_{\mathscr{R}_1}$

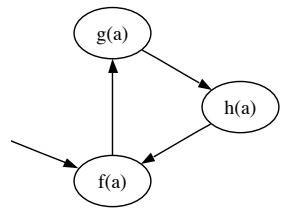

Figure 3: $\dashrightarrow_{\mathscr{R}_2}$

Commentaire : REFAIRE LES SCHEMAS : REMPLACER LES TERMES PAR DES ETATS CORRESPONDANTS OU ETATS (TERMES)? Moi je laisserais les termes sur cet exemple car ca permet de comprendre mieux ou on va. Quite a mettre les etats dans la suite de l'exemple et expliquer qu'ils reconnaissent ces termes.

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

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

Theorem 2 *Le be $K = (S, S_0, R, L)$ a Kripke structure built from $A^*_{\mathscr{R}}$. For any states s, s' such that $R^*(s, s')$ holds, exist two terms $u \in L(s)$ and $v \in L(s')$ such that $u \dashrightarrow^*_{\mathscr{R}_i} v$.*

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 if and only if one term recognized by Ap must be recognized by P to satisfy the property. More formally:

$$K(Q,\ Q_F,\ \Delta^{\leftarrow}_{\varepsilon},\ L),\ q \models P \quad \Longleftrightarrow \quad \mathscr{L}(L(q)) \cap \mathscr{L}(P) \neq \emptyset$$

We also add the operators ($\wedge$, $\vee$, $\neg$, **X**, **F**, **G**, **U**, **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 [[**?**]].

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 $\mathbf{G}(\{f(a)\} \implies \mathbf{X}\{g(a)\})$ has to be interpreted as $f(a) \dashrightarrow_{\mathscr{R}_2} g(a)$.

We use the Büchi automata framework to perform model checking. A survey of this technique can be found in the chapter 9 of [[**?**]]. LTL (or R-LTL) formulas and Kripke structures can be translated into Büchi automata. We construct two Büchi automata : B_K obtained form 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 its all behaviors are recognized by the automaton B_L. It means checking $\mathscr{L}(B_K) \subseteq \mathscr{L}(B_L)$. To do it, 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.

B_M and B_K are classically defined as 5-tuples: alphabet, states, initial states, final states and transition relation. Since we use tree automata to define predicates, the alphabet of B_K and B_L is a set of tree automata. 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 [**?**].

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 $B_K = \langle Q,\, Q_i,\, \Delta,\, Q\rangle$ by $\overline{B_L} = \langle Q',\, Q'_i,\, \Delta',\, F\rangle$ is defined as*

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

where $\Delta_\times$ is the set of transitions $(q_K, q_L) \xrightarrow{A_K \cap A_L} (q'_K, q'_L)$ 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 $A_K \cap A_L$ must be non empty as expected by the interpretation 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.

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 is lost in the original algorithm. Another contribution is the mechanism making it possible to prove LTL-like temporal properties on such abstractions of sets of reachable terms. In this paper, we only deal with exact tree automata completion results. Future plans are to extend this result so as to prove temporal properties on over-approximations. A similar objective has already been tackled in [**?**]. However, this was done in a pure rewriting framework where abstractions are more heavily constrained than in tree automata completion [4]. Hence, by extending LTL formula checking on tree automata over-approximations, we hope to ease the verification of temporal formula on infinite state systems.

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] 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`.

[4] 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`.

[5] 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.

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

[7] 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.