Computing and Informatics, Vol. 29, 2010, 407–426

LIVENESS VERIFICATION IN TRSS USING TREE AUTOMATA AND TERMINATION ANALYSIS

Mousa Mousazadeh, Behrouz Tork Ladani

Department of Computer Engineering
University of Isfahan
Hezar Jerib Av., Isfahan, Iran
e-mail: {mousazadeh, ladani}@eng.ui.ac.ir

Hans Zantema

Department of Computer Science
TU Eindhoven
P. O. Box 513, 5600 MB Eindhoven, The Netherlands
&
Institute for Computing and Information Sciences
Radboud University
Nijmegen, P. O. Box 9010, 6500 GL Nijmegen, The Netherlands
e-mail: H.Zantema@tue.nl

Manuscript received 20 January 2009; revised 21 September 2009
Communicated by Tomáš Vojnar

Abstract. This paper considers verification of the liveness property $\mathsf{Live}(R, I, G)$ for a term rewrite system (TRS) R, where I (Initial states) and G (Good states) are two sets of ground terms represented by finite tree automata. Considering I and G, we transform R to a new TRS R' such that termination of R' proves the property $\mathsf{Live}(R, I, G)$.

Keywords: Term rewriting systems, liveness properties, finite tree automata, termination

1 INTRODUCTION

The notion of safety and liveness properties have been first introduced by Lamport [12]. Informally, safety properties assert that nothing bad ever happens while liveness properties assert that something good happens eventually. This paper considers the verification of liveness properties in term rewrite systems (TRSs). In TRSs, informally we say the liveness property $\mathsf{Live}(R, I, G)$ holds if every maximal reduction by TRS R starting in initial states I contains an element of good states G. In [9] the relationship between liveness and termination was studied, and it was observed that conversely liveness can be seen as termination of a modified relation. Since various techniques have been developed to prove termination automatically, an obvious goal is to apply these techniques in order to prove liveness properties.

The techniques which are presented in previous work [9, 10, 11] can only be used for some verifying specific types of liveness properties. Indeed, they can verify liveness properties where the set of good states has some specific form. Actually, by previous techniques the liveness property $\mathsf{Live}(R, I, G)$ can be verified where $G = \{t \mid t \text{ does not contain an instance of } p\}$ or $G = \{t \mid t \text{ contains an instance of } p\}$ for some term p. Also, in these methods the initial states (I) simply contains all the ground terms while sometimes it is required to consider I as a proper subset of the set of all ground terms. Furthermore, they concentrate on a particular case of rewrite systems called *top rewrite systems*.

In this paper we consider that R is an arbitrary TRS, and G is an arbitrary regular set of ground terms represented by a *finite tree automaton*. A tree automaton is a type of state machine. Tree automata deal with tree structures (terms), rather than with the strings of more conventional state machines. By using finite tree automata one can provide a finite representation for any arbitrary regular set of terms (finite or infinite). In the general case, we consider that the set of the initial states contains all the ground terms, but under some conditions we extend the method to consider I as a proper subset of the set of all ground terms represented by a tree automaton.

Considering I and G, we transform R to a new TRS R' such that termination of R' proves the property $\mathsf{Live}(R, I, G)$. We show these transformations are sound indeed, i.e. termination of R' implies $\mathsf{Live}(R, I, G)$. Furthermore, we show under some specific conditions these transformations are complete, i.e. $\mathsf{Live}(R, I, G)$ also implies termination of R'.

Moreover, we show by some examples how this transformation can be used. To prove termination of transformed TRSs automatic tools for termination have been used. We have succeeded in automatically proving defined properties for all of our examples using the AProVE 1.2 tool [8]. The system AProVE 1.2 can be used for automated termination and innermost termination proofs of (conditional) TRSs, Prolog programs, functional, and imperative programs.

This paper is organized as follows. In Section 2 we present some theoretical preliminaries. Section 3 is the main part of this paper and investigates the verification

method for liveness properties in term rewrite systems. Section 4 discusses liveness in the framework of fairness. We briefly review related work in Section 5. Finally we conclude the paper in Section 6.

2 PRELIMINARIES

2.1 Liveness in Rewriting

In this section we briefly give a formal definition of liveness using the framework of abstract reduction and term rewriting as presented in [9]. We assume a set S of states and a notion of computation that can be expressed by a binary relation $\rightarrow \subseteq S \times S$. So "$t \rightarrow u$" means that a computation step from t to u is possible. A computation sequence or reduction is defined to be a finite sequence $t_1, t_2, \ldots, t_n$ or an infinite sequence $t_1, t_2, t_3, \ldots$ with $t_i \rightarrow t_{i+1}$. We write $\rightarrow^*$ for the reflexive transitive closure of $\rightarrow$, i.e., $\rightarrow^*$ represents zero or more computation steps.

To define liveness we assume a set $G \subseteq S$ of *'good'* states and a set $I \subseteq S$ of initial states. A reduction is *maximal* if it is either infinite or if its last element is in the set of *normal forms* $\mathsf{NF} = \{t \in S \mid \neg\, \exists u : t \rightarrow u\}$. The liveness property $\mathsf{Live}(\rightarrow, I, G)$ holds if every maximal reduction starting in I contains an element of G. Thus, the notion of liveness describes eventuality properties.

Definition 1 (Liveness)**.** Let S be a set of states, $\rightarrow \subseteq S \times S$, and $G, I \subseteq S$. Let "$t_1, t_2, t_3, \ldots$" denote an infinite sequence of states. Then $\mathsf{Live}(\rightarrow, I, G)$ holds *iff*

1. $t_1, t_2, t_3, \ldots : (t_1 \in I \wedge \forall i : t_i \rightarrow t_{i+1}) \Rightarrow \exists i : t_i \in G$, and
2. $t_1, t_2, \ldots, t_n : (t_1 \in I \wedge t_n \in \mathsf{NF} \wedge \forall i : t_i \rightarrow t_{i+1}) \Rightarrow \exists i : t_i \in G$.

It has been shown in [10] that this notion of liveness specializes the "standard" definition of Alpern and Schneider [1] in the framework of rewriting. In Alpern and Schneider's framework, a property P is a set of infinite sequences of states. Terminating executions of a program are represented by repeating the final state infinitely often.

Now we focus on liveness in rewriting, i.e., we study the property $\mathsf{Live}(R, I, G)$ for TRS R. For an introduction to term rewriting, the reader is referred to [14], for example.

Let Σ be a signature containing a constant and let $\mathcal{V}$ be a set of variables. We write $\mathcal{T}(\Sigma, \mathcal{V})$ for the set of terms over Σ and $\mathcal{V}$, and $\mathcal{T}(\Sigma)$ is the set of ground terms. Now $\mathcal{T}(\Sigma, \mathcal{V})$ represents computation states and $I, G \subseteq \mathcal{T}(\Sigma, \mathcal{V})$.

2.2 Finite Tree Automata

There are two main classes of finite tree automata: *top-down* finite tree automata and *bottom-up* finite tree automata. A non-deterministic top-down finite tree automaton (top-down NFTA) over Σ is a tuple $\mathcal{A} = (Q, \Sigma, Q_I, \Delta)$ where Q is a set of

states (states are unary symbols), $Q_I \subseteq Q$ is a set of initial states, and Δ is a set of rewrite rules of the following type:

$$q(f(x_1, \ldots, x_n)) \to f(q_1(x_1), \ldots, q_n(x_n)),$$

where $n \geq 0$, $f \in \Sigma$ is of arity n, $q, q_1, \ldots, q_n \in Q$, and $x_1, \ldots, x_n \in \mathcal{V}$.

When $n = 0$, i.e. when the symbol is a constant symbol c, a transition rule of top-down NFTA is of the form $q(c) \to c$. A top-down automaton starts at the root and moves downward, associating along a run a state with each subterm inductively. The tree language $L(\mathcal{A})$ recognized by $\mathcal{A}$ is the set of all ground terms t for which there is an initial state $q_0 \in Q_I$ such that

$$q_0(t) \to_\Delta^* t.$$

A non-deterministic bottom-up finite Tree Automaton (bottom-up NFTA) over Σ is a tuple $\mathcal{A} = (Q, \Sigma, Q_F, \Delta)$ where Q is a set of (unary) states, $Q_F \subseteq Q$ is a set of final states, and Δ is a set of transition rules of the following type:

$$f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n)),$$

where $n \geq 0$, $f \in \Sigma$ is of arity n, $q, q_1, \ldots, q_n \in Q$, and $x_1, \ldots, x_n \in \mathcal{V}$.

A bottom-up automaton starts at the leaves and moves upward, associating along a run a state with each subterm inductively. There is no initial state in a bottom-up NFTA, but, when $n = 0$, i.e. when the symbol is a constant symbol c, a transition rule is of the form $c \to q(c)$. Therefore, the transition rules for the constant symbols can be considered as the "initial rules". The tree language $L(\mathcal{A})$ recognized by a bottom-up NFTA $\mathcal{A}$ is the set of all ground terms t for which there is an final state $q_f \in Q_F$ such that

$$t \to_\Delta^* q_f(t).$$

A tree automaton is deterministic (DFTA) if there are no two rules with the same left-hand side. The expressive power of nondeterministic bottom-up and nondeterministic top-down tree automata is the same, but deterministic top-down tree automata are strictly less powerful than nondeterministic top-down tree automata. For an introduction to tree automata, the reader is referred to [3], for example.

We say the finite tree automaton $\mathcal{A}$ is closed under the TRS R *iff* for all terms $s \in L(\mathcal{A})$ if $s \to_R t$, then also $t \in L(\mathcal{A})$.

3 VERIFICATION METHOD

In this section we first present the basic model that is used for liveness verification in Section 3.1. Next in 3.2 and 3.3 we introduce two sound transformations that use the basic model to transform an original TRS to a new TRS such that termination of the new TRS validates the liveness property. Also, we show under some conditions

these transformations are complete, i.e., if the liveness property holds then the transformed TRS terminates.

3.1 Analysis Model

In this section we introduce the basic model that is used for liveness verification. This model considers that the set of initial states containing all the ground terms, i.e. $I = \{t \mid t \in \mathcal{T}(\Sigma)\}$. Section 3.4 presents a technique to verify liveness properties with arbitrary initial states (arbitrary tree automata) if one specific condition is satisfied.

Suppose $\mathcal{A}$ is a finite tree automaton which accepts the set of good states. Our model composes rewrite system R and transition function of tree automaton (Δ) to define a new rewrite system $L(R, \mathcal{A})$.

It is desired that $L(R, \mathcal{A})$ follows reduction steps which are presented in Figure 1. R performs one step of reduction on a term t to produce a new term t', then Δ performs a few steps of reductions on term t'. If Δ finds $t' \in L(\mathcal{A})$, then the transformed TRS terminates, else allows R to perform a new reduction on t' and so forth.

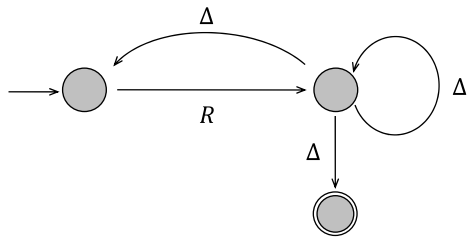

Fig. 1. Reduction sequence in basic model

Note that this model is ideal, i.e. if a transformation follows this model exactly, then it is sound and complete. To do this, some conditions should hold. The next sections discuss the issue in detail.

3.2 A Sound Transformation for Top-Down NFTA

In this section we suppose $\mathcal{A}$ is a top-down NFTA, and define a sound transformation that uses the basic model to transform the original TRS R to a new TRS $L(R, \mathcal{A})$ such that termination of the new TRS validates the liveness property $\mathsf{Live}(R, \mathcal{T}(\Sigma), L(\mathcal{A}))$. In this section we suppose $\mathcal{A}$ is a top-down NFTA.

Definition 2 ($L(R, \mathcal{A})$ for top-down NFTA)**.** Let $\mathcal{A} = (Q, \Sigma, Q_I, \Delta)$ be a top-down finite tree automaton, and R be a TRS over Σ. Also, suppose `root`, `turnA`, `turnR`, and `letR` be new unary symbols. Then term rewrite system $L(R, \mathcal{A})$ over $\Sigma \cup Q \cup \{\texttt{root}, \texttt{turnA}, \texttt{turnR}, \texttt{letR}\}$ consists of the following rules:

$$\texttt{letR}(l) \to \texttt{turnA}(r) \tag{1}$$

for all rules $l \to r$ in R

$$f(x_1, \ldots, \texttt{turnA}(x_i), \ldots, x_n) \to \texttt{turnA}(f(x_1, \ldots, x_n)) \tag{2}$$

for all $f \in \Sigma$ of arity $n \geq 1, i = 1, \ldots, n$

$$\texttt{root}(\texttt{turnA}(x)) \to \texttt{root}(q_0(x)) \tag{3}$$

for all $q_0 \in Q_I$

$$l \to r \tag{4}$$

for all rules $l \to r$ in Δ

$$q(f(x_1, \ldots, x_n)) \to \texttt{turnR}(f(x_1, \ldots, x_n)) \tag{5}$$

for all $q(f(x_1, \ldots, x_n))$ such that $q \in Q$, $f \in \Sigma$ of arity $n \geq 0$, and $q(f(x_1, \ldots, x_n))$ is not an instance of the left hand side of any rule of Δ.

$$f(x_1, \ldots, \texttt{turnR}(x_i), \ldots, x_n) \to \texttt{turnR}(f(x_1, \ldots, x_n)) \tag{6}$$

for all $f \in \Sigma$ of arity $n \geq 1, i = 1, \ldots, n$

$$\texttt{turnR}(\texttt{turnR}(x)) \to \texttt{turnR}(x) \tag{7}$$

$$\texttt{root}(\texttt{turnR}(x)) \to \texttt{root}(\texttt{letR}(x)) \tag{8}$$

$$\texttt{letR}(f(x_1, \ldots, x_n)) \to f(x_1, \ldots, \texttt{letR}(x_i), \ldots, x_n) \tag{9}$$

for all $f \in \Sigma$ of arity $n \geq 1, i = 1, \ldots, n$

The following theorem shows that the above transformation is sound, i.e. termination of $L(R, \mathcal{A})$ implies validation of $\mathsf{Live}(R, \mathcal{T}(\Sigma), L(\mathcal{A}))$.

Theorem 1 (Soundness)**.** Let R be a TRS, and $\mathcal{A}$ be a top-down finite tree automaton. If $\mathsf{NF} \subseteq L(\mathcal{A})$, then termination of $L(R, \mathcal{A})$ implies $\mathsf{Live}(R, \mathcal{T}(\Sigma), L(\mathcal{A}))$.

Proof. Assume $\mathsf{Live}(R, \mathcal{T}(\Sigma), L(A))$ does not hold. Then one of the following conditions holds:

1. There is a finite sequence $t_0, t_1, \ldots, t_n$ such that $t_0 \in \mathcal{T}(\Sigma)$, $t_n \in \mathsf{NF}$, $\forall i : t_i \to_R t_{i+1}$, $t_i \notin L(\mathcal{A})$, which contradicts $\mathsf{NF} \subseteq L(\mathcal{A})$.

2. There is an infinite sequence $t_0, t_1, t_2, \ldots$, such that $t_0 \in \mathcal{T}(\Sigma)$, $\forall i : t_i \to_R t_{i+1}$, $t_i \notin L(\mathcal{A})$. To prove the theorem, we show:

$$\forall i : \mathtt{root}(\mathtt{letR}(t_i)) \to^+_{L(R,\mathcal{A})} \mathtt{root}(\mathtt{letR}(t_{i+1})),$$

which gives an infinite $L(R,\mathcal{A})$-reduction, in contradiction to termination of $L(R,\mathcal{A})$.

We have $t_i \to_R t_{i+1}$, then there is a rule $l \to r$ in R, some context C_i, and some substitution σ such that, $t_i = C_i[l\sigma]$ and $t_{i+1} = C_i[r\sigma]$. Then,

$$\begin{aligned}\mathtt{root}(\mathtt{letR}(t_i)) = \mathtt{root}(\mathtt{letR}(C_i[l\sigma])) &\to^*_{L(R,\mathcal{A})} \mathtt{root}(C_i[\mathtt{letR}(l\sigma)])\\ &\to_{L(R,\mathcal{A})} \mathtt{root}(C_i[\mathtt{turnA}(r\sigma)])\\ &\to^*_{L(R,\mathcal{A})} \mathtt{root}(\mathtt{turnA}(C_i[r\sigma]))\\ &= \mathtt{root}(\mathtt{turnA}(t_{i+1}))\\ &\to_{L(R,\mathcal{A})} \mathtt{root}(q_0(t_{i+1}))\end{aligned}$$

$t_{i+1} \notin L(\mathcal{A})$, i.e. $\mathcal{A}$ does not accept t_{i+1}, then there is some context C'_i, a term $f(s_1, \ldots, s_n)$, and at least one $q \in Q$, such that

$$\mathtt{root}(q_0(t_{i+1})) \to^*_\Delta \mathtt{root}(C'_i[q(f(s_1, \ldots, s_n))]),$$

and $q(f(s_1, \ldots, s_n))$ can not reduce further by Δ. $L(R,\mathcal{A})$ contains all rules of Δ, then

$$\mathtt{root}(q_0(t_{i+1})) \to^*_{L(R,\mathcal{A})} \mathtt{root}(C'_i[q(f(s_1, \ldots, s_n))])$$

$q(f(s_1, \ldots, s_n))$ can not reduce further by Δ, hence $q(f(x_1, \ldots, x_n))$ is not an instance of left hand side of any rule of Δ, so $q(f(x_1, \ldots, x_n)) \to \mathtt{turnR}(f(x_1, \ldots, x_n))$ is in $L(R,\mathcal{A})$, then

$$\mathtt{root}(C'_i[q(f(s_1, \ldots, s_n))]) \to_{L(R,\mathcal{A})} \mathtt{root}(C'_i[\mathtt{turnR}(f(s_1, \ldots, s_n))]).$$

If there is more than one `turnR` symbol in the context it can be reduced to one, using the rule $\mathtt{turnR}(\mathtt{turnR}(x)) \to \mathtt{turnR}(x)$, so

$$\begin{aligned}\mathtt{root}(C'_i[\mathtt{turnR}(f(s_1, \ldots, s_n))]) &\to^*_{L(R,\mathcal{A})} \mathtt{root}(\mathtt{turnR}(C'_i[f(s_1, \ldots, s_n)]))\\ &= \mathtt{root}(\mathtt{turnR}(t_{i+1}))\\ &\to_{L(R,\mathcal{A})} \mathtt{root}(\mathtt{letR}(t_{i+1})).\end{aligned}$$

Then, $\forall i : \mathtt{root}(\mathtt{letR}(t_i)) \to^+_{L(R,\mathcal{A})} \mathtt{root}(\mathtt{letR}(t_{i+1}))$. □

The following example shows how this transformation can be used.

Example 1. Let $\Sigma = \{f(), c\}$ and $R = \{f(x) \to f(f(x))\}$. Also suppose $G = \{f^n(c) \mid n \geq 3\} \cup \{c\}$. Then the tree automaton $\mathcal{A}$ in which $L(\mathcal{A}) = G$ is defined as follows:

$$Q = \{q_0, q_1, q_2, q_3\}, \Sigma = \{f(), c\}, Q_I = \{q_0\}, \Delta :$$

$$
\begin{array}{ll}
q_0(f(x)) \to f(q_1(x)) & q_2(f(x)) \to f(q_3(x)) \\
q_0(c) \to c & q_3(f(x)) \to f(q_3(x)) \\
q_1(f(x)) \to f(q_2(x)) & q_3(c) \to c
\end{array}
$$

Then $L(R, \mathcal{A})$ consists of the following rules:

$L(R, \mathcal{A})$:

$$
\begin{array}{ll}
\mathtt{letR}(f(x)) \to \mathtt{turnA}(f(f(x))) & q_1(c) \to \mathtt{turnR}(c) \\
 & q_2(c) \to \mathtt{turnR}(c) \\
f(\mathtt{turnA}(x)) \to \mathtt{turnA}(f(x)) & \\
 & f(\mathtt{turnR}(x)) \to \mathtt{turnR}(f(x)) \\
\mathtt{root}(\mathtt{turnA}(x)) \to \mathtt{root}(q_0(x)) & \\
 & \mathtt{turnR}(\mathtt{turnR}(x)) \to \mathtt{turnR}(x) \\
 & \\
q_0(f(x)) \to f(q_1(x)) & \mathtt{root}(\mathtt{turnR}(x)) \to \mathtt{root}(\mathtt{letR}(x)) \\
q_0(c) \to c & \\
q_1(f(x)) \to f(q_2(x)) & \mathtt{letR}(f(x)) \to f(\mathtt{letR}(x)) \\
q_2(f(x)) \to f(q_3(x)) & \\
q_3(f(x)) \to f(q_3(x)) & \\
q_3(c) \to c &
\end{array}
$$

Note that $\mathsf{NF} = \{c\} \subseteq L(\mathcal{A})$. The TRS $L(R, \mathcal{A})$ is terminating; this can be proved by the tool AProVE [8]. Thus the liveness property holds in the TRS. □

Now we show under some conditions our transformation is complete, i.e. if $L(R, \mathcal{A})$ is non-terminating, liveness property $\mathsf{Live}(R, \mathcal{T}(\Sigma), L(\mathcal{A}))$ does not hold. Indeed we have the following theorem.

Theorem 2 (Completeness)**.** Let $\mathcal{A}$ be a top-down finite tree automata, and R be a TRS such that

1. Q_I contains only one initial state, i.e. $Q_I = \{q_0\}$,
2. $\mathcal{A}$ is *deterministic*, and
3. R is *non-duplicating*, i.e., for every rule $l \to r$, no variable occurs more often in r than in l,

Then if $\mathsf{Live}(R, \mathcal{T}(\Sigma), \mathrm{L}(\mathcal{A}))$ holds, then $L(R, \mathcal{A})$ terminates.

To prove completeness of the transformation, we need several auxiliary lemmas. At first we recall the following lemma from [9]. For a function symbol $f \in \Sigma$ and a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$, let $\| \ t \ \|_f$ be the number of f-symbols occurring in t. For $\emptyset \neq \Sigma' \subseteq \Sigma$, let $\| \ t \ \|_{\Sigma'} = \sum_{f \in \Sigma'} \| \ t \ \|_f$.

Lemma 1. Let R be a non-duplicating TRS, and for some $\Sigma' \subseteq \Sigma$,

$$\| l \|_{\Sigma'} \geq \| r \|_{\Sigma'} \text{ for all rules } l \to r \text{ in } R$$

Let R' consist of those rules $l \to r$ from R which satisfy $\| l \|_{\Sigma'} > \| r \|_{\Sigma'}$. Then R is terminating if and only if $R \setminus R'$ is terminating.

Lemma 2. Let R be a non-duplicating TRS, $\mathcal{A}$ be a finite tree automaton, and $L'(R, \mathcal{A}) = L(R, \mathcal{A}) \setminus \{(5)\}$. Then $L'(R, \mathcal{A})$ is terminating.

Proof. Let $\Sigma' = \{\texttt{turnR}\}$, then by Lemma 1, $L'(R, \mathcal{A}) = L(R, \mathcal{A}) \setminus \{(5)\}$ is terminating if and only if $L(R, \mathcal{A}) \setminus \{(5), (7), (8)\}$ is terminating. Now let $\Sigma' = \{\texttt{letR}\}$, then similarly by Lemma 1, $L(R, \mathcal{A}) \setminus \{(5), (7), (8)\}$ is terminating if and only if $L(R, \mathcal{A}) \setminus \{(5), (7), (8), (1)\}$ is terminating. Now let $\Sigma' = \{\texttt{turnA}\}$, then similarly by Lemma 1, $L(R, \mathcal{A}) \setminus \{(5), (7), (8), (1)\}$ is terminating if and only if $L(R, \mathcal{A}) \setminus \{(5), (7), (8), (1), (3)\}$ is terminating.

Then $L'(R, \mathcal{A})$ is terminating if and only if $\{(2), (4), (6), (9)\}$ is terminating. All rules of (2) can be removed using polynomial ordering. To remove rule $f(x_1, \ldots, \texttt{turnA}(x_i), \ldots, x_n) \to \texttt{turnA}(f(x_1, \ldots, x_n))$, the following polynomial ordering can be used [2]:

$[\texttt{letR}](X_1) = X_1$
$[\texttt{turnR}](X_1) = X_1$
$[\texttt{turnA}](X_1) = 1 + X_1$
$[q](X_1) = X_1$ for all $q \in Q$
$[f](X_1, \ldots, X_i, \ldots, X_n) = X_1 + \ldots + 2 * X_i + \ldots + X_n$ for all $i \in \{1, ..., n\}$
$[g](X_1, \ldots, X_m) = X_1 + \ldots + X_m$ for all function symbols $g \in \Sigma,\ g \neq f$
$[a] = 0$ for all constant symbols $a \in \Sigma$.

Similarly, all rules of (6) and (9) can be removed using polynomial ordering. Then $L'(R, \mathcal{A})$ are terminating if and only if all rules of type (4), i.e. rules of Δ, are terminating. Obviously Δ is terminating, then $L'(R, \mathcal{A})$ are terminating. □

Lemma 3. Let R be a non-duplicating TRS, $\mathcal{A}$ be a finite tree automaton, and $L'(R, \mathcal{A}) = L(R, \mathcal{A}) \setminus \{(3)\}$. Then $L'(R, \mathcal{A})$ is terminating.

Proof. Let $L'(R, \mathcal{A})$ be non-terminating, then there is an infinite sequence $t_0 \to t_1 \to t_2 \to \ldots$ of $L'(R, \mathcal{A})$-reductions.

By definition, $L'(R, \mathcal{A})$ does not contain any rule of form $\texttt{root}(\texttt{turnA}(x)) \to \texttt{root}(q_0(x))$. Also, Δ is terminating and R is non-duplicating, then in the above reduction only finite numbers of $q \in Q$ symbols can be generated. On the other hand, due to Lemma 2, this reduction contains infinitely many applications of rules (5). In every application of rules (5) one $q \in Q$ symbol disappears. Then, to generate sufficient $q \in Q$ symbols, this reduction must contain infinitely many applications of rules (3). Hence, $L(R, \mathcal{A}) \setminus \{(3)\}$ is terminating. □

To introduce the next lemma, first we recall the notion $\overline{t}$ for a term $t \in \mathcal{T}(\Sigma \cup Q \cup \{\mathtt{turnA}, \mathtt{turnR}, \mathtt{letR}\})$ from [11]. $\overline{t}$ denotes term t after removing all occurrences of symbols of $Q \cup \{\mathtt{turnA}, \mathtt{turnR}, \mathtt{letR}\}$). Formally:

$\overline{\mathtt{turnA}(t)} = \overline{t}, \quad \overline{\mathtt{turnR}(t)} = \overline{t}, \quad \overline{\mathtt{letR}(t)} = \overline{t}, \quad \overline{q(t)} = \overline{t} \quad$ for all $q \in Q$,

$\overline{f(t_1, \ldots, t_n)} = f(\overline{t_1}, \ldots, \overline{t_n}) \quad$ for all $f \in \Sigma$.

Lemma 4. Let R be a non-duplicating TRS, $\mathcal{A} = (Q, \Sigma, Q_I, \Delta)$ be a deterministic finite tree automaton, and $t_0 \in \mathcal{T}(\Sigma \cup Q \cup \{\mathtt{turnA}, \mathtt{turnR}, \mathtt{letR}\})$. If $\overline{t_0} \in L(\mathcal{A})$ and $Q_I = \{q_0\}$, then there is no infinite sequence of $L(R, \mathcal{A})$-reductions, starting with $\mathtt{root}(q_0(t_0))$.

Proof. Suppose there exists infinite sequence $\mathtt{root}(q_0(t_0)) \to \mathtt{root}(t_1) \to \mathtt{root}(t_2) \to \cdots$ of $L(R, \mathcal{A})$-reductions. Then one of the following occurs:

I) $\overline{t_0} = t_0$, i.e. the term t_0 does not contain none of symbols of $Q \cup \{\mathtt{turnA}, \mathtt{turnR}, \mathtt{letR}\}$. In this case there is only one option for reducing the term $\mathtt{root}(q_0(t_0))$, i.e., $\mathtt{root}(q_0(t_0)) \to^+ \mathtt{root}(t_0)$ using Δ rules, due to theorem hypothesis $\overline{t_0} \in L(\mathcal{A})$. However, the term $\mathtt{root}(t_0)$ can not be reduced further using $L(R, \mathcal{A})$. Hence this case can not result in an infinite sequence of $L(R, \mathcal{A})$-reductions.

II) The term t_0 contains one of the symbols $\mathtt{turnA}$, $\mathtt{turnR}$, or $\mathtt{letR}$. So, all of the consequence symbols of q_0 that will be obtained using Δ rules and reach one of the symbols $\mathtt{turnA}$, $\mathtt{turnR}$, or $\mathtt{letR}$ – that we show them by q_i – can not be removed, because this symbol can not reach a constant symbol to be removed using Δ rules. Indeed, we have one of the following reductions for each symbol of $\{\mathtt{turnA}, \mathtt{turnR}, \mathtt{letR}\}$ that appears in reduction sequence:

$$\mathtt{root}(q_0(t_0)) \to^* \mathtt{root}(C[q_i(\mathtt{turnA}(s))]),$$

$$\mathtt{root}(q_0(t_0)) \to^* \mathtt{root}(C[q_i(\mathtt{turnR}(s))]),$$

$$\mathtt{root}(q_0(t_0)) \to^* \mathtt{root}(C[q_i(\mathtt{letR}(s))]),$$

for some context C and term s. In all of the above reductions a symbol $\mathtt{turnA}$ can not reach $\mathtt{root}$. So, rules of type (3) can not be used in the reduction. But, due to Lemma 3, $L(R, \mathcal{A}) \backslash \{(3)\}$ is terminating. Hence this case can not results in an infinite sequence of $L(R, \mathcal{A})$-reductions. Note that occurring or not occurring of $q \in Q$ symbols in t_0 does not care for the proof in this case.

III) The term t_0 does not contain any of the symbols of $\{\mathtt{turnA}, \mathtt{turnR}, \mathtt{letR}\}$, but contains at least one of the symbols of Q. In this case, if symbols of Q can be removed using Δ rules then this case is the same as case I, else it is the same as case II. Hence this case can not result in an infinite sequence of $L(R, \mathcal{A})$-reductions again. This contradiction proves the lemma.

□

Now we can prove Theorem 2.

Proof. [Completeness] Let $\Sigma' = \Sigma \cup Q \cup \{\mathtt{root}, \mathtt{turnA}, \mathtt{turnR}, \mathtt{letR}\}$, and set of *root terms* be $\mathcal{T}_{\mathtt{root}} = \{\mathtt{root}(t) \mid t \in \mathcal{T}(\Sigma' \backslash \{\mathtt{root}\}, \mathcal{V})\}$. Suppose $\mathsf{Live}(R, \mathcal{T}(\Sigma), L(\mathcal{A}))$ holds, but $L(R, \mathcal{A})$ does not terminate. By type introduction [15] it can be shown that there exists an infinite $L(R, \mathcal{A})$-reduction of ground root terms. We show this sequence of reductions by $\mathtt{root}(t_0) \to \mathtt{root}(t_1) \to \mathtt{root}(t_2) \to \ldots$. Let $M = \{i \mid t_i = \mathtt{root}(q_0(t'_i))$ for some term $t'_i\}$. By Lemma 3, $L(R, \mathcal{A}) \setminus \{(3)\}$ is terminating, and hence M is infinite.

We claim there exist infinitely many j such that $j \in M$ and $\overline{t'_j} \in L(\mathcal{A})$. Suppose for all $j \in M$, $\overline{t'_j} \notin L(\mathcal{A})$. Due to liveness property, there exist infinitely many $\mathtt{root}(t_k)$ in the sequence such that $\overline{t_k} \in L(\mathcal{A})$, because if there exist only finitely many $\mathtt{root}(t_k)$ such that $\overline{t_k} \in L(\mathcal{A})$, removing finite prefix of sequence yields an infinite $L(R, \mathcal{A})$ reduction in which no member of $L(\mathcal{A})$ occurs at all. To generate such $\mathtt{root}(t_k)$ the following reduction must appear in the sequence:

$$\ldots \mathtt{root}(C[\mathtt{letR}(s_0)]) \to \mathtt{root}(t_k) \to^+ \mathtt{root}(s_1) \to^+ \mathtt{root}(\mathtt{turnA}(s_2)) \\ \to \mathtt{root}(q_0(s_3)) \to^+ \mathtt{root}(\mathtt{letR}(s_4)) \ldots$$

for some context C and terms $s_0, \ldots$, and s_4.

To reduce $\mathtt{root}(s_1)$ to $\mathtt{root}(\mathtt{letR}(s_4))$, we must have $\overline{s_1} \notin L(\mathcal{A})$; but $\overline{t_k} \in L(\mathcal{A})$, then to reduce $\mathtt{root}(t_k)$ to $\mathtt{root}(s_1)$ the following reduction must appear in the sequence:

$$\mathtt{root}(t_k) = \mathtt{root}(C'[\mathtt{letR}(t'_k)]) \to \mathtt{root}(C'[\mathtt{turnA}(s'_1)]) \to^+ \mathtt{root}(s_1)$$

for some context C' and term s'_1.

Then, for reducing each $\mathtt{root}(t_k)$ two $\mathtt{letR}$ disappear and one $\mathtt{letR}$ symbol appears in the sequence. R is non-duplicating, then this subsequence of reduction can be done for finitely many of t_k terms. Thus there exist infinitely many t_k such that $\overline{t_k} \in L(\mathcal{A})$ and $t_k = q_0(t'_k)$; but this result contradicts Lemma 4, because Lemma 4 states that there is not any infinite sequence of $L(R, \mathcal{A})$-reductions starting with $\mathtt{root}(q_0(t'_k))$. This contradiction proves the theorem. □

The following examples show that conditions (1) and (2) of this theorem are essential, but we have not any example to prove condition (3) is also essential.

Example 2. Let $\Sigma = \{f(), a, b, c\}$ and $R = \{a \to b,\ b \to c,\ c \to a\}$. Also suppose $G = \{t \mid t$ contains symbol a or symbol $b\}$. Then tree automaton $\mathcal{A}$ in which $L(\mathcal{A}) = G$ can be defined as follows:

$Q = \{q_0, q_1\}, \Sigma = \{f(), a, b, c\}, Q_I = \{q_0, q_1\}, \Delta :$

$$\begin{array}{ll} q_0(f(x)) \to f(q_0(x)) & q_1(f(x)) \to f(q_1(x)) \\ q_0(a) \to a & q_1(b) \to b \end{array}$$

Then $L(R, \mathcal{A})$ consists of the following rules:

$L(R, \mathcal{A})$:

$$\begin{array}{ll}
\texttt{letR}(a) \to \texttt{turnA}(b) & q_0(b) \to \texttt{turnR}(b) \\
\texttt{letR}(b) \to \texttt{turnA}(c) & q_0(c) \to \texttt{turnR}(c) \\
\texttt{letR}(c) \to \texttt{turnA}(a) & q_1(a) \to \texttt{turnR}(a) \\
 & q_1(c) \to \texttt{turnR}(c) \\
f(\texttt{turnA}(x)) \to \texttt{turnA}(f(x)) & \\
 & f(\texttt{turnR}(x)) \to \texttt{turnR}(f(x)) \\
\texttt{root}(\texttt{turnA}(x)) \to \texttt{root}(q_0(x)) & \\
\texttt{root}(\texttt{turnA}(x)) \to \texttt{root}(q_1(x)) & \texttt{turnR}(\texttt{turnR}(x)) \to \texttt{turnR}(x) \\
 & \\
q_0(f(x)) \to f(q_0(x)) & \texttt{root}(\texttt{turnR}(x)) \to \texttt{root}(\texttt{letR}(x)) \\
q_0(a) \to a & \\
q_1(f(x)) \to f(q_1(x)) & \texttt{letR}(f(x)) \to f(\texttt{letR}(x)) \\
q_1(b) \to b &
\end{array}$$

Clearly liveness property $\mathsf{Live}(R, \mathcal{T}(\Sigma), L(\mathcal{A}))$ holds in the TRS, but $L(R, \mathcal{A})$ is not terminating. We have the following loop:

$\texttt{root}(q_0(b)) \to \texttt{root}(\texttt{turnR}(b)) \to \texttt{root}(\texttt{letR}(b)) \to \texttt{root}(\texttt{turnA}(c)) \to$
$\texttt{root}(q_0(c)) \to \texttt{root}(\texttt{turnR}(c)) \to \texttt{root}(\texttt{letR}(c)) \to \texttt{root}(\texttt{turnA}(a)) \to$
$\texttt{root}(q_1(a)) \to \texttt{root}(\texttt{turnR}(a)) \to \texttt{root}(\texttt{letR}(a)) \to \texttt{root}(\texttt{turnA}(b)) \to$
$\texttt{root}(q_0(b)) \to \ldots$ □

Example 3. Let $\Sigma = \{f(,), a, b\}$ and $R = \{a \to b,\ b \to a\}$. Also suppose $G = \{t \mid t$ contains symbol $a\}$. Then tree automaton $\mathcal{A}$ in which $L(\mathcal{A}) = G$ is defined as follows:

$Q = \{q_0, q_1\}, \Sigma = \{f(,), a, b\}, Q_I = \{q_0\}, \Delta$:

$$\begin{array}{ll}
q_0(f(x,y)) \to f(q_0(x), q_1(y)) & q_1(f(x,y)) \to f(q_1(x), q_1(y)) \\
q_0(f(x,y)) \to f(q_1(x), q_0(y)) & q_1(a) \to a \\
q_0(a) \to a & q_1(b) \to b.
\end{array}$$

Obviously $\mathcal{A}$ is not deterministic. $L(R, \mathcal{A})$ consists of the following rules:

$L(R, \mathcal{A})$:

$$\begin{array}{ll}
\texttt{letR}(a) \to \texttt{turnA}(b) & q_0(b) \to \texttt{turnR}(b) \\
\texttt{letR}(b) \to \texttt{turnA}(a) & \\
 & f(\texttt{turnR}(x), y) \to \texttt{turnR}(f(x,y)) \\
f(\texttt{turnA}(x), y) \to \texttt{turnA}(f(x,y)) & f(x, \texttt{turnR}(y)) \to \texttt{turnR}(f(x,y)) \\
f(x, \texttt{turnA}(y)) \to \texttt{turnA}(f(x,y)) & \\
 & \texttt{turnR}(\texttt{turnR}(x)) \to \texttt{turnR}(x) \\
\texttt{root}(\texttt{turnA}(x)) \to \texttt{root}(q_0(x)) & \\
 & \texttt{root}(\texttt{turnR}(x)) \to \texttt{root}(\texttt{letR}(x)) \\
q_0(f(x,y)) \to f(q_0(x), q_1(y)) & \\
q_0(f(x,y)) \to f(q_1(x), q_0(y)) & \texttt{letR}(f(x,y)) \to f(\texttt{letR}(x), y)
\end{array}$$

$$q_0(a) \to a \qquad \texttt{letR}(f(x,y)) \to f(x, \texttt{letR}(y))$$
$$q_1(f(x,y)) \to f(q_1(x), q_1(y))$$
$$q_1(a) \to a$$
$$q_1(b) \to b.$$

Clearly liveness property $\mathsf{Live}(R, \mathcal{T}(\Sigma), L(\mathcal{A}))$ holds in the TRS, but $L(R, \mathcal{A})$ is not terminating. We have the following loop:

$$\begin{array}{llll}
\texttt{root}(q_0(f(a,b))) & \to \texttt{root}(f(q_1(a), q_0(b))) & \to \texttt{root}(f(a, q_0(b))) & \to \\
\texttt{root}(f(a, \texttt{turnR}(b))) & \to \texttt{root}(\texttt{turnR}(f(a,b))) & \to \texttt{root}(\texttt{letR}(f(a,b))) & \to \\
\texttt{root}(f(\texttt{letR}(a), b)) & \to \texttt{root}(f(\texttt{turnA}(b), b)) & \to \texttt{root}(\texttt{turnA}(f(b,b))) & \to \\
\texttt{root}(q_0(f(b,b))) & \to \texttt{root}(f(q_1(b), q_0(b))) & \to \texttt{root}(f(b, q_0(b))) & \to \\
\texttt{root}(f(b, \texttt{turnR}(b))) & \to \texttt{root}(\texttt{turnR}(f(b,b))) & \to \texttt{root}(\texttt{letR}(f(b,b))) & \to \\
\texttt{root}(f(\texttt{letR}(b), b)) & \to \texttt{root}(f(\texttt{turnA}(a), b)) & \to \texttt{root}(\texttt{turnA}(f(a,b))) & \to \\
\texttt{root}(q_0(f(a,b))) & \to \ldots \square &&
\end{array}$$

3.3 A Sound Transformation for Bottom-Up NFTA

Although the languages accepted by bottom-up NFTAs are the same as the languages accepted by top-down NFTAs, for particular examples using bottom-up NFTAs may be simpler than using top-down NFTAs. Therefore it makes sense to consider an alternative transformation based on bottom-up NFTA. For this we propose the following definition:

Definition 3 ($L(R, \mathcal{A})$ for bottom-up NFTA)**.** Let $\mathcal{A} = (Q, \Sigma, Q_F, \Delta)$ be a bottom-up finite tree automaton, and R be a TRS over Σ. Also, suppose `root`, `turnA`, `letA`, `turnR`, and `letR` be new unary symbols. Then term rewrite system $L(R, \mathcal{A})$ over $\Sigma \cup Q \cup \{\texttt{root}, \texttt{turnA}, \texttt{letA}, \texttt{turnR}, \texttt{letR}\}$ consists of the following rules:

$$\texttt{letR}(l) \to \texttt{turnA}(r) \tag{10}$$

for all rules $l \to r$ in R

$$f(x_1, \ldots, \texttt{turnA}(x_i), \ldots, x_n) \to \texttt{turnA}(f(x_1, \ldots, x_n)) \tag{11}$$

for all $f \in \Sigma$ of arity $n \geq 1, i = 1, \ldots, n$

$$\texttt{root}(\texttt{turnA}(x)) \to \texttt{root}(\texttt{letA}(x)) \tag{12}$$

$$\texttt{letA}(f(x_1, \ldots, x_n)) \to f(\texttt{letA}(x_1), \ldots, \texttt{letA}(x_n)) \tag{13}$$

for all $f \in \Sigma$ of arity $n \geq 1, i = 1, \ldots, n$

$$\texttt{letA}(c) \to q_c(c) \tag{14}$$

for all $c \to q_c(c)$ in Δ

$$l \to r \tag{15}$$

for all $l \to r$ in Δ except rules $c \to q_c(c)$

$$\mathtt{root}(q(x)) \to \mathtt{root}(\mathtt{turnR}(x)) \tag{16}$$

for all $q \in Q \backslash Q_F$

$$f(q_1(x_1), \ldots, q_n(x_n))) \to \mathtt{turnR}(f(x_1, \ldots, x_n)) \tag{17}$$

for all $f(q_1(x_1), \ldots, q_n(x_n)))$ such that $q_i \in Q$, $f \in \Sigma$ of arity $n \geq 1$, and $f(q_1(x_1), \ldots, q_n(x_n)))$ is not an instance of left hand side of any rule of Δ.

$$f(x_1, \ldots, \mathtt{turnR}(x_i), \ldots, x_n) \to \mathtt{turnR}(f(x_1, \ldots, x_n)) \tag{18}$$

for all $f \in \Sigma$ of arity $n \geq 1, i = 1, \ldots, n$

$$q(x) \to x \tag{19}$$

for all $q \in Q$

$$\mathtt{root}(\mathtt{turnR}(x)) \to \mathtt{root}(\mathtt{letR}(x)) \tag{20}$$

$$\mathtt{letR}(f(x_1, \ldots, x_n)) \to f(x_1, \ldots, \mathtt{letR}(x_i), \ldots, x_n) \tag{21}$$

for all $f \in \Sigma$ of arity $n \geq 1, i = 1, \ldots, n$

The following theorem shows this transformation is sound.

Theorem 3 (Soundness)**.** Let R be a TRS, and $\mathcal{A}$ be a bottom-up finite tree automaton. If $\mathsf{NF} \subseteq L(\mathcal{A})$, then termination of $L(R, \mathcal{A})$ implies $\mathsf{Live}(R, \mathcal{T}(\Sigma), L(\mathcal{A}))$.

The above theorem can be proved similar to Theorem 1. Also, the following theorem shows under some conditions this transformation is complete.

Theorem 4 (Completeness)**.** Let $\mathcal{A}$ be a bottom-up finite tree automata, and R be a TRS such that

- $\mathcal{A}$ is *deterministic*, and
- R is *non-duplicating*,

Then if $\mathsf{Live}(R, \mathcal{T}(\Sigma), L(\mathcal{A}))$ holds, $L(R, \mathcal{A})$ terminates.

The proof of this theorem needs explaining several details, but the general procedure for its proving is similar to proof of Theorem 2. Then, here we ignore proving this theorem. Indeed to prove the theorem, we need to rewrite Lemmas 2, 3, and 4 with respect to the new setting, and then by type introduction [15] to show that there is not any infinite $L(R, \mathcal{A})$-reduction of ground root terms.

For every bottom-up NFTA, we can construct an equivalent bottom-up DFTA [3]. So the first condition above could be handled easily. Then this transformation leads us to better completeness results.

3.4 Initial States

Sometimes we need to consider initial states as a proper subset of the set of all ground terms. Techniques for proving *local termination* [4] can be used to prove this type of liveness properties. Local termination considers proving termination on a restricted set of terms rather than set of all ground terms.

Beside techniques for proving locall termination, if $\mathcal{A}_I$ is closed under R, we present a technique to verify liveness properties with initial states $L(\mathcal{A}_I)$. In the following theorem $L^C(\mathcal{A})$ shows the complement of $L(\mathcal{A})$, and $\mathsf{NF}(S) = \{t \in \mathsf{NF} \mid \exists s \in S : s \to_R^* t\}$, where S is a set of terms.

Theorem 5 (Initial States)**.** Let $\mathcal{A}$, $\mathcal{A}_I$, and $\mathcal{A}_G$ be three finite tree automata, such that $L(\mathcal{A}) = L^C(\mathcal{A}_I) \cup L(\mathcal{A}_G)$. For a TRS R if $\mathsf{NF}(L(\mathcal{A}_I)) \subseteq L(\mathcal{A}_G)$, and $\mathcal{A}_I$ be closed under R, then termination of $L(R, \mathcal{A})$ implies $\mathsf{Live}(R, L(\mathcal{A}_I), L(\mathcal{A}_G))$.

Proof. Suppose $\mathsf{Live}(R, L(\mathcal{A}_I), L(\mathcal{A}_G))$ does not hold. Then one of the following conditions holds:

1. There is a finite sequence $t_0, t_1, \ldots, t_n$ such that $t_0 \in L(\mathcal{A}_I)$,$t_n \in \mathsf{NF}(L(\mathcal{A}_I))$, $\forall i : t_i \to_R t_{i+1}$, $t_i \notin L(\mathcal{A}_G)$, in contradiction to $\mathsf{NF}(L(\mathcal{A}_I)) \subseteq L(\mathcal{A}_G)$.
2. There is an infinite sequence $t_0, t_1, t_2, \ldots$, such that $t_0 \in L(\mathcal{A}_I)$, $\forall i : t_i \to_R t_{i+1}$, $t_i \notin L(\mathcal{A}_G)$. Let $k = min\{j | t_j \in L(\mathcal{A})\}$. By Theorems 1 and 3, there exists such k. We have $L(\mathcal{A}) = L^C(\mathcal{A}_I) \cup L(\mathcal{A}_G)$, and $\forall i : t_i \notin L(\mathcal{A}_G)$, then $t_k \in L^C(\mathcal{A}_I)$, so $t_k \notin L(\mathcal{A}_I)$.

Also, $\forall i, i = 0, \ldots, k-1 : t_i \notin L^C(\mathcal{A}_I) \cup L(\mathcal{A}_G)$, then $\forall i, i = 0, \ldots, k-1 : t_i \notin L^C(\mathcal{A}_I)$, so $t_{k-1} \in L(\mathcal{A}_I)$.

$t_{k-1} \in L(\mathcal{A}_I)$, $t_{k-1} \to_R t_k$,$t_k \notin L(\mathcal{A}_I)$, then $\mathcal{A}_I$ is not closed under R. This contradiction proves the theorem. □

The following example shows how this transformation can be used.

Example 4. Let $\Sigma = \{f(), g(), c\}$ and $R = \{f(x) \to f(f(x)),\ g(x) \to g(g(x))\}$. Also suppose $I = \{f^n(c) \mid n \geq 1\} \cup \{c\}$ and $G = \{f^n(c) \mid n \geq 3\} \cup \{c\}$. Then tree automaton $\mathcal{A}$ in which $L(\mathcal{A}) = L^C(\mathcal{A}_I) \cup L(\mathcal{A}_G)$ is defined as follows:

$$Q = \{q_0, q_1, q_2, q_3\}, \Sigma = \{f(), g(), c\}, Q_I = \{q_0\}, \Delta :$$

$$\begin{array}{ll}
q_0(f(x)) \to f(q_1(x)) & q_2(f(x)) \to f(q_3(x)) \\
q_0(g(x)) \to g(q_3(x)) & q_2(g(x)) \to g(q_3(x)) \\
q_0(c) \to c & q_3(f(x)) \to f(q_3(x)) \\
q_1(f(x)) \to f(q_2(x)) & q_3(g(x)) \to g(q_3(x)) \\
q_1(g(x)) \to g(q_3(x)) & q_3(c) \to c.
\end{array}$$

Then $L(R, \mathcal{A})$ consists of the following rules:

$L(R, \mathcal{A})$:

$$\begin{array}{ll}
\texttt{letR}(f(x)) \to \texttt{turnA}(f(f(x))) & q_1(c) \to \texttt{turnR}(c) \\
\texttt{letR}(g(x)) \to \texttt{turnA}(g(g(x))) & q_2(c) \to \texttt{turnR}(c) \\
\\
f(\texttt{turnA}(x)) \to \texttt{turnA}(f(x)) & f(\texttt{turnR}(x)) \to \texttt{turnR}(f(x)) \\
g(\texttt{turnA}(x)) \to \texttt{turnA}(g(x)) & g(\texttt{turnR}(x)) \to \texttt{turnR}(g(x)) \\
\\
\texttt{root}(\texttt{turnA}(x)) \to \texttt{root}(q_0(x)) & \texttt{turnR}(\texttt{turnR}(x)) \to \texttt{turnR}(x) \\
\\
q_0(f(x)) \to f(q_1(x)) & \texttt{root}(\texttt{turnR}(x)) \to \texttt{root}(\texttt{letR}(x)) \\
q_0(g(x)) \to g(q_3(x)) & \\
q_0(c) \to c & \texttt{letR}(f(x)) \to f(\texttt{letR}(x)) \\
q_1(f(x)) \to f(q_2(x)) & \texttt{letR}(g(x)) \to g(\texttt{letR}(x)) \\
q_1(g(x)) \to g(q_3(x)) & \\
q_2(f(x)) \to f(q_3(x)) & \\
q_2(g(x)) \to g(q_3(x)) & \\
q_3(f(x)) \to f(q_3(x)) & \\
q_3(g(x)) \to g(q_3(x)) & \\
q_3(c) \to c. &
\end{array}$$

It can be shown easily that conditions of Theorem 5 hold and $L(R, \mathcal{A})$ is terminating, thus liveness property $\mathsf{Live}(R, L(\mathcal{A}_I), L(\mathcal{A}_G))$ holds in the TRS. □

In general, using Theorem 5 for proving liveness is somewhat difficult, because checking whether $\mathsf{NF}(L(\mathcal{A}_I)) \subseteq L(\mathcal{A}_G)$ is not decidable in the general case; also the size of the tree automata that accepts all terms but those of I can be exponentially bigger than A_I. However, as it was observed, in some problems using this technique can be helpful.

4 LIVENESS WITH FAIRNESS

The notion of liveness in TRSs has been extended in [11] to include fair computations, i.e., that liveness is not restricted to its basic notion stating that any infinite computation eventually reaches a good state, but it can be done for infinite fair computations, where infinite computations contain some essential steps infinitely often.

There are different notions of fairness in term rewriting literature. In [13] Lucas and Meseguer have presented some comparisons between existing notions of fairness. In this paper we use Koprowski and Zantema's definition [11].

Based on Koprowski and Zantema's definition of fairness, in fair computations, instead of a single rewrite relation $\to$, we have two relations $\to, \xrightarrow{=} \subseteq S \times S$ which are presented by rewrite systems R and $R^{=}$. An infinite reduction in $\to \cup \xrightarrow{=}$ is called *fair* (with respect to $\xrightarrow{=}$) if it contains infinitely many $\xrightarrow{=}$-steps [11]. Then liveness for fair reductions with respect to R and $R^{=}$, initial states I and good states G,

$\mathsf{Live}(R, R^{=}, I, G)$, holds iff for any infinite fair reduction $t_1 \to t_2 \to t_3 \to \ldots$ with $t_1 \in I$, we have $\exists i : t_i \in G$.

In previous sections we saw how liveness corresponds to termination. Similarly, liveness in fair computations corresponds to *relative termination* [11]. A rewrite system R is said to terminate relatively to $R^{=}$ if every (possibly infinite) $\to \cup \xrightarrow{=}$ computation contains only finitely many $\to$ steps [7].

The presented transformation can be extended to contain the notion of fairness. When we study liveness in fair computations, there are two different rewrite systems. Thus it is required to transform both R and $R^{=}$.

$L(R, \mathcal{A})$:

$$\mathtt{letR}(l) \to \mathtt{turnA}(r) \quad \text{for all rules } l \to r \text{ in } R$$

$L(R^{=}, \mathcal{A})$ is also obtained for both top-down and bottom-up NFTAs using the definitions 2 and 3 respectively, i.e.,

$L(R^{=}, \mathcal{A})$:

$$\mathtt{letR}(l) \xrightarrow{=} \mathtt{turnA}(r) \quad \text{for all rules } l \xrightarrow{=} r \text{ in } R^{=}$$
$$\vdots$$

Then if $L(R, \mathcal{A})$ terminates relatively to $L(R^{=}, \mathcal{A})$, desired liveness property holds in the system.

The soundness of the above transformation can be proved in the same way as that of $L(R, \mathcal{A})$, also if $\mathcal{A}$ is a top-down or bottom-up NFTA, completeness of this transformation depends on satisfying the conditions of Theorem 2 or 4, respectively.

5 RELATED WORK

In this section we briefly review related work in the domain of safety and liveness verification.

When we verify a safety property, we indeed verify if none of the bad states are *reachable*; more precisely, if we could enumerate all reachable states, then we also could verify if bad states belong to it or not. Thus *reachability analysis* can be used for safety verification.

In the domain of reachability analysis over term rewriting systems there are some publications, and some automatic tools have been introduced, even some techniques of reachability analysis have been used for cryptographic protocol verification [6]. The reference [4] surveys some techniques and tools for achieving reachability analysis over term rewriting systems. The core of those techniques is a generic tree automata completion algorithm used to compute in an exact or approximated way the set of descendants (or reachable terms).

In the domain of liveness verification over term rewriting systems some work was done before. In this work the focus was on a special class of TRSs called *top rewrite systems*. Giesl and Zantema in [9] have defined a specific type of liveness

properties in TRSs called *global* liveness. The good states of global liveness is defined as $G = \{t \mid t \text{ does not contain an instance of } p\}$ for some term p. They have proposed two transformations for global liveness. The first one, sound and complete, results in complicated TRSs even for extremely simple liveness problems for which proving termination is very difficult. The second one is only sound, thus simple. Also, in [10] they have defined a new type of liveness properties called *local* liveness, as well as two transformations, similar to global liveness. The good states of local liveness is defined as $G = \{t \mid t \text{ contains an instance of } p\}$ for some term p.

In [11] Koprowski and Zantema have extended the notion of liveness to include fair computations. Furthermore, they have introduced a new transformation for verifying global liveness in the framework of fairness. Recently, in [13] Lucas and Meseguer have considered applying rewriting termination technology – enjoying a quite mature set of termination results and tools – to the problem of proving automatically the termination of concurrent systems under fairness assumptions. These new introduced techniques could be used to obtain new results for dealing with liveness verification in fair conditions.

6 CONCLUSIONS

In this paper the problem of liveness verification in term rewrite systems was considered. Our general idea is to specify the desired liveness property using two finite tree automata which represent the corresponding sets of good states and initial states. Then to verify the liveness property, considering these two finite tree automata we transform the original TRS to a new TRS such that termination of the new TRS validates the desired liveness property. We have shown that the provided transformations are indeed sound and under some specific conditions are complete.

Although the general approach used in our method is not new, we have some novel extensions over previous work. While in earlier works the type of original TRS is restricted to a special kind of systems named top rewrite systems, we have removed this limitation to consider any arbitrary TRS. Furthermore, using prior methods the liveness property can be verified where the set of good states has some specific form. To remove this limitation we have used the notion of finite tree automata to represent good states by arbitrary regular language. Although in the general case this method considers that the set of initial states contains all the ground terms, but under some conditions we have extend the method to consider initial states as a proper subset of the set of all ground terms represented by a finite tree automaton.

Although termination problem is a non-decidable problem in general, i.e. there is not any algorithm to prove or deny termination of all TRSs, this reality is not unpromising, because many termination problems can be solved by existing methods. In this way, automatic tools for proving termination could be helpful.

REFERENCES

[1] Alpern, B.—Schneider, F. B.: Defining Liveness. Information Processing Letters 21, 1985, pp. 181–185.

[2] Ben Cherifa, A.—Lescanne, P.: Termination of Rewriting Systems by Polynomial Interpretations and Its Implementation. Science of Computer Programming, Vol. 9, 1987, No. 2, pp. 137–159.

[3] Comon, H.—Dauchet, M.—Gilleron, R.—Jacquemard, F.—Lugiez, D.—Tison, S.—Tommasi, M.: Tree Automata Techniques and Applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2007.

[4] Feuillade, G.—Genet, T.—Viet Triem Tong, V.: Reachability Analysis over Term Rewriting Systems. Journal of Automated Reasoning 33, 2004, pp. 341–383.

[5] Endrullis, J.—Vrije, R.—Waldmann, J.: Local Termination. In Proc. 20th RTA, LNCS 5595, pp. 270–284, 2009.

[6] Genet, T.—Klay, F.: Rewriting for Cryptographic Protocol Verification. Proceedings of the 17th CADE Conference, LNAI 1831, pp. 271–290, 2000.

[7] Geser, A.: Relative Termination. Ph. D. thesis, Universitt Passau, Germany, 1990.

[8] Giesl, J.—Schneider-Kamp, P.—Thiemann, R.: AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In Proc. 3rd IJCAR, LNAI 4130, pp. 281–286, 2006. Available on: `http://aprove.informatik.rwth-aachen.de/`.

[9] Giesl, J.—Zantema, H.: Liveness in Rewriting. Proceedings of the 14th RTA, LNCS 2706, pp. 321–336, 2003.

[10] Giesl, J.—Zantema, H.: Simulating Liveness by Reduction Strategies. Proceedings of the 3rd WRS, Electronic Notes in TCS, Vol. 86, 2003, No. 4, pp. 641–656.

[11] Koprowski, A.—Zantema, H.: Proving Liveness With Fairness Using Rewriting. LNAI 3717, pp. 232–247, 2005.

[12] Lamport, L.: Proving the Correctness of Multiprocess Programs. IEEE Transactions on Software Engineering SE-3, Vol. 2, 1997, pp. 125–143.

[13] Lucas, S.—Meseguer, J.: Termination of Just/Fair Computations in Term Rewriting. Information and Computation, Vol. 206, 2008, pp. 652–675.

[14] Terese: Term Rewriting Systems. Cambridge Tracts in Theoretical Computer Science 55, Cambridge University Press, 2003.

[15] Zantema, H.: Termination of Term Rewriting: Interpretation and Type Elimination. Journal of Symbolic Computation, Vol. 17, 1994, pp. 23–50.

Mousa Mousazadeh received his B. Sc. in applied mathematics from Sharif University of Technology, Teharn, Iran in 2004, and his M. Sc. in artificial intelligence from University of Isfahan in 2008. Currently, he is a Ph. D. student in artificial intelligence at University of Isfahan. His research interests include formal verification, artificial intelligence, access control, and web service security.

Behrouz Tork Ladani received his B. Sc. in software engineering from University of Isfahan (Isfahan, Iran) in 1996, his M. Sc. in software engineering from Amir-Kabir University of Technology (Tehran, Iran) in 1998, and his Ph. D. in computer Engineering from Tarbiat-Modarres University (Tehran, Iran) in 2005. He is currently an Assistant Professor at the Department of Computer Engineering, University of Isfahan. His research interests include formal specification and verification, cryptographic protocols, multi agent security, and web services. He is a member of Iranian Society of Cryptology (ISC).

Hans Zantema did his Ph. D. in pure mathematics in 1983. After a few years in industry he was employed at various computer science departments: from 1987 to 2000 at Utrecht University, since 2000 at the University of Technology in Eindhoven, and since 2007 as a part time Full Professor at Radboud University in Nijmegen, all in The Netherlands. He is the author of over 70 refereed papers, many of which are about rewriting.