Proving Liveness with Fairness using Rewriting

Adam Koprowski and Hans Zantema

Technical University of Eindhoven
Department of Computer Science
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
{A.Koprowski, H.Zantema}@tue.nl

Abstract. In this paper we combine rewriting techniques with verification issues. More precisely, we show how techniques for proving relative termination of term rewrite systems (TRSs) can be applied to prove liveness properties in fair computations. We do this using a new transformation which is stronger than the sound transformation from [5] but still is suitable for automation. On the one hand we show completeness of this approach under some mild conditions. On the other hand we show how this approach applies to some examples completely automatically, using the TPA tool designed for proving relative termination of TRSs. In particular we succeed in proving liveness in the classical readers-writers synchronization problem.

1 Introduction

Usually, *liveness* is roughly defined as: "*something will eventually happen*" and it is often remarked that "*termination is a particular case of liveness*". In [5] the relationship between liveness and termination was investigated in more detail, 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. In [5] a method for transforming a class of liveness problems to problems of termination of term rewrite systems (TRSs) has been proposed. For a slightly different setting in [6] another approach was proposed.

Two transformations were given in [5]. The first one, sound and complete, even for extremely simple liveness problems results in complicated TRSs for which proving termination, especially in an automated way, is very difficult. That was the motivation for another, much simpler, transformation, which is sound but not complete.

In this paper this approach is extended in two ways. First we extend the basic framework to fair computations. That means that we do not restrict to the basic notion of liveness stating that any infinite computation eventually reaches a good state, but we do this for infinite fair computations, being infinite computations containing some essential computation steps infinitely often. Fairness has been studied extensively in [4]. In applications one is often interested in the behavior of infinite fair computations rather than of arbitrary infinite computations. For instance, in a waiting line protocol one may want to prove that eventually all old clients will be served. If it is allowed that infinitely many new clients come in, one may think of an infinite computation in which this does not hold: infinitely many new clients come in but no client is ever served. However, if serving of clients is defined to be the essential computation step, in a corresponding fair computation it can be proved that eventually all old clients will be served. It turns out that just like liveness corresponds to termination, liveness in fair computations corresponds to *relative termination*. So combining liveness and fairness is a main issue of this paper.

The second extension is the following. It turns out that the simple transformation presented in [5] often results in non-terminating TRSs, and therefore is not applicable, also in liveness problems not involving fairness. Therefore we propose a new transformation. Our new transformation is slightly more complicated than the simple transformation from [5], but much simpler than the sound and complete transformation from [5]. However, assuming some mild conditions, in this paper we show that our new transformation is sound and complete too. Moreover, we show in two examples that our new transformation results in TRSs for which (relative) termination can be proved fully automatically. In particular we consider the classical readers-writers synchronization problem, in which the priority of access is controlled in an obvious way. The desired liveness property states that every process in the system eventually gets access to the resource. Using our technique we succeed in automatically proving this liveness property. Both examples involve infinite state spaces and hence the standard model checking techniques are not applicable to them.

To this end a tool — TPA (*Termination Proved Automatically*, `http://www.win.tue.nl/tpa`) — was developed for proving relative termination of TRSs automatically, based on polynomial interpretations [9], semantic labelling with booleans and with natural numbers [13], recursive path ordering [3] and a simple version of dependency pairs [1]. Most of those well-known termination techniques, except dependency pairs, were extended in a straightforward way to deal also with relative termination. TPA took part in the annual termination competition in 2005 (`http://www.lri.fr/~marche/termination-competition/2005`) where it got 3rd place out of 6 participating tools.

This paper is organized as follows. In Section 2 the general framework from [5] is extended in order to deal with liveness with fairness. Next in Section 3 the new transformation is introduced and the corresponding theorems on soundness and completeness are given. Finally in Section 4 two examples are presented in which this new approach has been applied.

2 Liveness with Fairness Conditions

2.1 Liveness in Abstract Reduction

First we present the framework as described in [5] with no more than necessary details to understand its extension given later. For a more elaborate description we refer to the original article.

We give the model of the system that should be verified in the framework of abstract reduction. We assume a set of states S and a binary relation on states expressing computation steps, $\rightarrow \subseteq S \times S$. As usual we write $\rightarrow^*$ for its reflexive transitive closure and $\rightarrow^+$ for its transitive closure. We define a set of states in *normal form* as $\mathrm{NF} \equiv \{s \in S \mid \neg\exists s' \in S : \; s \rightarrow s'\}$ and a set of terms in normal form reachable from a given set of states I as $\mathrm{NF}(I) \equiv \{s \in \mathrm{NF} \mid \exists t \in I : t \rightarrow^* s\}$. We call a reduction sequence *maximal* if it is either infinite or its last element is in NF. By $\mathrm{SN}(I, \rightarrow)$ we denote termination of reduction sequences starting in I and by $\mathrm{SN}(\rightarrow)$ termination of arbitrary sequences. That is: $\mathrm{SN}(I, \rightarrow) \equiv \neg\exists t_1, t_2, \ldots : t_1 \in I \wedge \forall i : t_i \rightarrow t_{i+1}$ and $\mathrm{SN}(\rightarrow) \equiv \mathrm{SN}(S, \rightarrow)$.

With respect to a set of initial states $I \subseteq S$ and a set of good states $G \subseteq S$, we say that the liveness property $\mathrm{Live}(I, \rightarrow, G)$ holds if all maximal $\rightarrow$-reductions starting in I contain an element from G. More precisely:

Definition 1 (Liveness) *Let S be a set of states, $\rightarrow \subseteq S \times S$; $G, I \subseteq S$. Then* $\mathrm{Live}(I, \rightarrow, G)$ *holds iff*

– $\forall t_1, t_2, \ldots : \left\{ \begin{array}{r} t_1 \in I \\ \forall i : t_i \rightarrow t_{i+1} \end{array} \right\} \Longrightarrow \exists i : t_i \in G$, *and*

– $\forall t_1, \ldots, t_n : \left\{ \begin{array}{r} t_1 \in I \\ t_n \in \mathrm{NF} \\ \forall i \in \{1, \ldots, n-1\} : t_i \rightarrow t_{i+1} \end{array} \right\} \Longrightarrow \exists i \in \{1, \ldots, n\} : t_i \in G.$

We define the restricted computation relation $\rightarrow_G \equiv \{(s,t) \mid s \rightarrow t \wedge s \notin G\}$. The following theorem from [5] relates liveness to termination of $\rightarrow_G$.

Theorem 2 *If* $\mathrm{NF}(I) \subseteq G$ *then* $\mathrm{Live}(I, \rightarrow, G)$ *iff* $\mathrm{SN}(I, \rightarrow_G)$.

2.2 Liveness with Fairness in Abstract Reduction

In liveness we are mainly interested in the behavior of infinite reduction sequences, or shortly, infinite reductions. However, in many applications one is not interested in arbitrary infinite reductions but in infinite fair reductions, defined as follows. Instead of a single rewrite relation $\rightarrow$ we have two relations $\rightarrow, \overset{=}{\rightarrow} \subseteq S \times S$ [1]. An infinite reduction in $\rightarrow \cup \overset{=}{\rightarrow}$ is called *fair* (with respect to $\rightarrow$) if it contains infinitely many $\rightarrow$-steps. Finally we say that liveness for fair reductions starting in I with respect to $\rightarrow$, $\overset{=}{\rightarrow}$ and G holds, denoted as $\mathrm{Live}(I, \rightarrow, \overset{=}{\rightarrow}, G)$, if any fair $\rightarrow \cup \overset{=}{\rightarrow}$ reduction starting in I contains an element of G. Note that all fair reductions are infinite, hence in investigating liveness with fairness we are only interested in systems with infinite behavior.

Definition 3 (Liveness with fairness) *Let S be a set of states, $\rightarrow, \overset{=}{\rightarrow} \subseteq S \times S$; $G, I \subseteq S$. Then liveness for fair reductions with respect to $I, \rightarrow, \overset{=}{\rightarrow}$ and G,* $\mathrm{Live}(I, \rightarrow, \overset{=}{\rightarrow}, G)$*, holds iff*

[1] The notation for $\rightarrow$ and $\overset{=}{\rightarrow}$ is chosen to be consistent with the notation for relative termination problems as used in TPDB (Termination Problem Database, `http://www.lri.fr/~marche/tpdb/`). The database serves as a base of problems for Termination Competitions.

– $\forall t_1, t_2, \ldots : \left\{ \begin{array}{r} t_1 \in I \\ \forall i : t_i \to t_{i+1} \vee t_i \overset{=}{\to} t_{i+1} \\ \forall i \exists j > i : t_j \to t_{j+1} \end{array} \right\} \implies \exists i : t_i \in G$

Our definition is based on the notion of relative termination. We define that $\to$ terminates relatively to $\overset{=}{\to}$ if every (possibly infinite) $\to \cup \overset{=}{\to}$ computation contains only finitely many $\to$ steps. We introduce the relation $\to / \overset{=}{\to} \equiv \overset{=}{\to}^* \cdot \to \cdot \overset{=}{\to}^*$ and observe that relative termination of $\to$ to $\overset{=}{\to}$ is equivalent to $\mathrm{SN}(\to / \overset{=}{\to})$. Also observe that $\mathrm{SN}(\to / \emptyset) \equiv SN(\to)$ so termination is a special case of relative termination.

The result of the next theorem gives us a method of verifying liveness with fairness requirements.

Theorem 4 $\mathrm{Live}(I, \to, \overset{=}{\to}, G)$ *holds iff* $\mathrm{SN}(I, \to_G / \overset{=}{\to}_G)$.

Proof. ($\Rightarrow$) Assume that $\mathrm{Live}(I, \to, \overset{=}{\to}, G)$ holds and $\mathrm{SN}(I, \to_G / \overset{=}{\to}_G)$ does not hold. From the latter we get that there is an infinite, fair reduction sequence $t_1, t_2, \ldots$ with $t_1 \in I$ and $\forall i : t_i \to_G t_{i+1} \vee t_i \overset{=}{\to}_G t_{i+1}$. From the definition of $\to_G$ all $t_i \notin G$. But then this reduction sequence is a counter-example for $\mathrm{Live}(I, \to, \overset{=}{\to}, G)$.

($\Leftarrow$) Since $\mathrm{SN}(I, \to_G / \overset{=}{\to}_G)$ then in every infinite, fair $\to \cup \overset{=}{\to}$ reduction starting in I there is an element from G (which blocks further $\to_G \cup \overset{=}{\to}_G$ reductions) and that is exactly what the definition of $\mathrm{Live}(I, \to, \overset{=}{\to}, G)$ calls for. $\square$

2.3 Liveness with Fairness in Term Rewriting

In previous sections we described the transition relation by means of abstract reductions, and related liveness of $\to$ to termination of $\to_G$. Our goal is to employ techniques for proving termination of rewriting in order to prove liveness properties. To that end a transformation is required since usually $\to_G$ is not a rewrite relation even if $\to$ is a rewrite relation.

For a signature Σ and a set $\mathcal{V}$ of variables, we denote the set of terms over Σ and $\mathcal{V}$ by $\mathcal{T}(\Sigma, \mathcal{V})$. Now we represent the computation states by terms, so S becomes $\mathcal{T}(\Sigma, \mathcal{V})$ and $I, G \subseteq \mathcal{T}(\Sigma, \mathcal{V})$. Abstract reduction relations $\to$ and $\overset{=}{\to}$ now correspond to two TRSs over the same signature Σ: R and $R^=$, respectively. As a shorthand for $\to_R$ we write $\to$ and for $\to_{R^=}$ we simply write $\overset{=}{\to}$. Just like it is usual to write $\mathrm{SN}(R)$ rather than $\mathrm{SN}(\to_R)$, we will write $\mathrm{Live}(I, R, R^=, G)$ rather than $\mathrm{Live}(I, \to_R, \to_{R^=}, G)$.

For an introduction to term rewriting the reader is referred, for instance, to [12].

Now, again after [5], we will introduce the notion of top TRSs, which we are going to use to model liveness problems.

Definition 5 (Top TRSs) *Let Σ be a signature and* top *be a fresh unary symbol in this signature, that is* top $\notin \Sigma$. *A term $t \in \mathcal{T}(\Sigma \cup \{\mathsf{top}\}, \mathcal{V})$ is called a* top term *if it contains exactly one instance of the* top *symbol, at the root of the term. We denote the set of top terms by $\mathcal{T}_{top}(\Sigma, \mathcal{V})$.*

A TRS over $\Sigma \cup \{\mathsf{top}\}$ is called a top term rewrite system *(top TRS) if for all its rules $\ell \to r$ one of the following holds:*

– *Both ℓ and r are top terms. Then we call this rule a* top rule.
– *Both ℓ and r do not contain an instance of the* top *symbol. Then the rule is called a* non-top rule.

Clearly for top TRSs every reduction starting in a top term only contains top terms. In the remainder we restrict ourselves to liveness with respect to

– reduction relations described by top TRSs,
– the set of initial states consisting of all top terms, and
– the set of good states of the form:

$$G(P) = \{t \in \mathcal{T}_{top}(\Sigma, \mathcal{V}) \mid \neg\exists p \in P,\ \sigma,\ C : t = C[p\sigma]\},$$

for some set $P \subseteq \mathcal{T}(\Sigma, \mathcal{V})$, that is $G(P)$ represents top terms not containing an instance of any of the terms from P.

So we are going to investigate liveness properties of the form:

$$\mathrm{Live}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R, R^{=}, G(P))$$

for some top TRSs R and $R^{=}$. This is equivalent to proving that every infinite fair reduction of top terms contains a term which does not contain an instance of any of the terms from P.

As we will show later this type of question can be transformed to a relative termination question of an ordinary TRS. This allows us to employ the techniques for proving relative termination for TRSs to verify liveness properties. Also, while quite restricted, this setting seems to be general enough to be able to cope with some interesting and practical examples, two of which will be presented at the end of this paper.

3 A New Transformation

3.1 Motivation

We are seeking a transformation with the property that relative termination of the transformed pair of systems implies that the liveness property in question holds (even better if we can have equivalence). In [5] two such transformations were proposed: the first one sound and complete (equivalence between termination and liveness holds) and the second one only sound (termination implies liveness but not the other way around) but significantly simpler. Experiments with them show that the former is so complex that, although it is a nice theoretical result, in practice it leads to TRSs far too complicated for present termination techniques to deal with, especially in an automated way. The sound transformation does not have this disadvantage but in several examples it is not powerful enough, leading to non-terminating TRSs, while the desired liveness property does hold.

In this section we propose a new transformation avoiding the aforementioned problems. But before we do that we will shortly introduce the sound transformation LS from [5] where $P = \{p\}$. As in our presentation also LS can be easily generalized to allow P to contain more than one element, as remarked in [5].

Definition 6 (LS) *Let R be a top TRS over $\Sigma \cup \{\mathsf{top}\}$ and $p \in \mathcal{T}(\Sigma, \mathcal{V})$. We define* $\mathrm{LS}(R, p)$ *to consist of the following rules:*

$$\begin{array}{rcll}
\ell & \to & r & \textit{for all non-top rules } \ell \to r \textit{ in } R \\
\mathsf{top}(\ell) & \to & \mathsf{top}(\mathsf{check}(r)) & \textit{for all top rules } \mathsf{top}(\ell) \to \mathsf{top}(r) \textit{ in } R \\
\mathsf{check}(f(x_1, \dots, x_n)) & \to & f(x_1, \dots, \mathsf{check}(x_i), \dots, x_n) & \\
 & & & \textit{for all } f \in \Sigma \textit{ of arity } n \geq 1, 1 \leq i \leq n \\
\mathsf{check}(p) & \to & p &
\end{array}$$

While *LS* is sound, it is not complete. This is illustrated by the following TRS $R = \{\mathsf{top}(f(x, b)) \to \mathsf{top}(f(b, b)),\ a \to b\}$. Normal forms do not contain symbol a and in every infinite reduction after finitely many steps only term $\mathsf{top}(f(b, b))$ occurs, so liveness for $p = a$ holds. However, $\mathrm{LS}(R, p)$ admits an infinite reduction, namely: $\mathsf{top}(\mathsf{check}(f(b, b))) \rightleftarrows \mathsf{top}(f(\mathsf{check}(b), b))$.

3.2 Definition of the Transformation

We give a new transformation inspired by the sound and complete transformation presented in [5] but significantly simpler so that obtained systems can still be treated with tools for automatic termination proving. It can deal with a much broader class of liveness problems than the sound transformation from [5]. We present it for only one unary top symbol but generalization to more top symbols and/or different arities is straightforward.

Definition 7 (LT) *Let R and $R^=$ be top TRSs over $\Sigma \cup \{\mathsf{top}\}$ and $P \subseteq \mathcal{T}(\Sigma, \mathcal{V})$. The transformed systems* $\mathrm{LT}(R)$ *and* $\mathrm{LT}^=(R^=, P)$ *over $\Sigma \cup \{\mathsf{top}, \mathsf{ok}, \mathsf{check}\}$ are defined as follows:*

$$\boxed{\mathrm{LT}(R)}$$

$$\begin{array}{rcll}
\ell & \to & r & \textit{for all non-top rules } \ell \to r \textit{ in } R \\
\mathsf{top}(\mathsf{ok}(\ell)) & \to & \mathsf{top}(\mathsf{check}(r)) & \textit{for all top rules } \mathsf{top}(\ell) \to \mathsf{top}(r) \textit{ in } R
\end{array}$$

$$\boxed{\mathrm{LT}^=(R^=, P)}$$

$$\begin{array}{rcll}
\ell & \to & r & \textit{for all non-top rules } \ell \to r \textit{ in } R^= \\
\mathsf{top}(\mathsf{ok}(\ell)) & \to & \mathsf{top}(\mathsf{check}(r)) & \textit{for all top rules } \mathsf{top}(\ell) \to \mathsf{top}(r) \textit{ in } R^= \\
\mathsf{check}(p) & \to & \mathsf{ok}(p) & \textit{for all } p \in P \\
\mathsf{check}(f(x_1, \dots, x_n)) & \to & f(x_1, \dots, \mathsf{check}(x_i), \dots, x_n) & \\
 & & & \textit{for all } f \in \Sigma \textit{ of arity } n \geq 1, 1 \leq i \leq n \\
f(x_1, \dots, \mathsf{ok}(x_i), \dots, x_n) & \to & \mathsf{ok}(f(x_1, \dots, x_n)) & \\
 & & & \textit{for all } f \in \Sigma \textit{ of arity } n \geq 1, 1 \leq i \leq n
\end{array}$$

The idea behind this transformation is that the presence of an ok symbol at the root of the term is intended to indicate existence of an instance of $p \in P$. Every time a top rule is applied this ok symbol is transformed to a check symbol. This check symbol can traverse toward the leaves and upon reaching an instance of some term $p \in P$ is transformed back into an ok symbol. This ok symbol can move up again and allow further top reductions upon reaching the root of the term.

Few remarks concerning the transformation:

- For readability concerns we will write $\to_{\mathrm{LT}}$ instead of $\to_{\mathrm{LT}(R)}$ and $\overset{=}{\to}_{\mathrm{LT}}$ instead of $\to_{\mathrm{LT}^=(R^=,P)}$.
- In order to apply automatic techniques the set P should be finite, otherwise the TRS $\mathrm{LT}^=(R^=, P)$ is infinite.
- If the liveness problem does not involve fairness, so it is modelled by single TRS R, then we define the result of the transformation to be also a single TRS, namely $\mathrm{LT}^=(R, P)$.

3.3 Soundness

Now we show soundness, that is relative termination of the transformed system implies liveness of the original one.

Theorem 8 (Soundness) *Let $R, R^=$ be top TRSs over $\Sigma \cup \{\mathsf{top}\}$, let $P \subseteq \mathcal{T}(\Sigma, \mathcal{V})$. Then:*

$$\mathrm{SN}(\mathrm{LT}(R)/\,\mathrm{LT}^=(R^=, P)) \implies \mathrm{Live}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R, R^=, G(P))$$

Proof. Assume that $\mathrm{SN}(\mathrm{LT}(R)/\,\mathrm{LT}^=(R^=, P))$ holds and $\mathrm{Live}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R, R^=, G(P))$ does not hold. By Theorem 4, $\mathrm{SN}(\mathcal{T}_{top}(\Sigma, \mathcal{V}),\ \to_G / \overset{=}{\to}_G)$ does not hold as it is equivalent to $\mathrm{Live}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R, R^=, G(P))$. That means that there is an infinite $\to_G\ /\ \overset{=}{\to}_G$ reduction of top terms. We will show that this infinite reduction can be mapped to an infinite $\to_{\mathrm{LT}} / \overset{=}{\to}_{\mathrm{LT}}$ reduction, thus contradicting $\mathrm{SN}(\mathrm{LT}(R)/\,\mathrm{LT}^=(R^=, P))$. For that purpose it is sufficient to show that:

$$\mathsf{top}(t)\ \to_G / \overset{=}{\to}_G\ \mathsf{top}(u) \implies \mathsf{top}(\mathsf{ok}(t))\ \to_{\mathrm{LT}} / \overset{=}{\to}_{\mathrm{LT}}\ \mathsf{top}(\mathsf{ok}(u))$$

that is that any step in $\to_G / \overset{=}{\to}_G$ can be mimicked by a step in $\to_{\mathrm{LT}} / \overset{=}{\to}_{\mathrm{LT}}$. It easily follows if we can show that:

(i) whenever $\mathsf{top}(t) \to_G \mathsf{top}(u)$ then $\mathsf{top}(\mathsf{ok}(t))\ \to_{\mathrm{LT}} / \overset{=}{\to}_{\mathrm{LT}}\ \mathsf{top}(\mathsf{ok}(u))$, and
(ii) whenever $\mathsf{top}(t) \overset{=}{\to}_G \mathsf{top}(u)$ then $\mathsf{top}(\mathsf{ok}(t))\ \overset{=}{\to}{}^*_{\mathrm{LT}}\ \mathsf{top}(\mathsf{ok}(u))$.

(i) First observe that if $\mathsf{top}(t) \to_G \mathsf{top}(u)$ by the application of a non-top rule $\ell \to r$ then the same rule is present in $\mathrm{LT}(R)$ so we trivially have $\mathsf{top}(\mathsf{ok}(t)) \to_{\mathrm{LT}} / \overset{=}{\to}_{\mathrm{LT}} \mathsf{top}(\mathsf{ok}(u))$.

If on the other hand $\mathsf{top}(t) \to_G \mathsf{top}(u)$ by application of a top rule then from the definition of top TRSs we have that $t = \ell\delta$ and $u = r\delta$ for some substitution δ and some rule $\mathsf{top}(\ell) \to \mathsf{top}(r)$ from R. Note that $\mathsf{top}(u)$ is part of an infinite

$\to_G$ / $\overset{=}{\to}_G$-reduction so $\mathsf{top}(u) \to_G \mathsf{top}(w)$ or $\mathsf{top}(u) \overset{=}{\to}_G \mathsf{top}(w)$ for some term w. Then from the definition of $\to_G$ we get that $\mathsf{top}(u)$ does contain an instance of some $p \in P$ which means that we have $u = C[p\gamma]$ for some context C and some substitution γ. Then we have:

$$\begin{array}{rcl} \mathsf{top}(\mathsf{ok}(t)) & = & \mathsf{top}(\mathsf{ok}(\ell\delta)) \\ & \to_{\mathrm{LT}} & \mathsf{top}(\mathsf{check}(r\delta)) \\ & = & \mathsf{top}(\mathsf{check}(C[p\gamma])) \\ & \overset{=}{\to}{}^*_{\mathrm{LT}} & \mathsf{top}(C[\mathsf{check}(p)\gamma]) \\ & \overset{=}{\to}_{\mathrm{LT}} & \mathsf{top}(C[\mathsf{ok}(p)\gamma]) \\ & \overset{=}{\to}{}^*_{\mathrm{LT}} & \mathsf{top}(\mathsf{ok}(C[p\gamma])) \qquad = \mathsf{top}(\mathsf{ok}(u)) \end{array}$$

The reasoning for (ii) is similar, just the first step is from $\overset{=}{\to}_{\mathrm{LT}(R)}$ instead of $\to_{\mathrm{LT}(R)}$. □

3.4 Completeness Results

In the previous subsection we proved that our approach is correct, that is that the proposed transformation is sound. Now we will try to address the question of its power. First we show (Theorem 9) that any liveness problem that could be dealt with using LS can also be dealt with using LT. Then we show that under some restrictions our new approach is even complete.

Theorem 9 *Let R be a top TRS over $\Sigma \cup \{\mathsf{top}\}$ and let $p \in \mathcal{T}(\Sigma, \mathcal{V})$. Then* $\mathrm{SN}(\mathrm{LS}(R, p))$ *implies* $\mathrm{SN}(\mathrm{LT}^{=}(R, \{p\}))$.

Proof. Follows from the observation that any $\mathrm{LT}^{=}(R, \{p\})$ reduction can be mapped to $\mathrm{LS}(R, p)$ reduction by dropping all ok symbols and the rule for propagating ok symbol is terminating in itself.

Note however that there is no 'if and only if' in Theorem 9, which means that LT may succeed in case LS fails. A very simple example showing that is the TRS $R = \{\mathsf{top}(f(x, b)) \to \mathsf{top}(f(b, b)),\ a \to b\}$ used at the end of Section 3.1 to show incompleteness of LT. We concluded there that $\mathrm{LS}(R, p)$ is not terminating, however it is not difficult to see that $\mathrm{LT}^{=}(R, \{p\})$ is terminating. Two more complex and practical examples will be presented in Section 4.

There is a good reason why the sound and complete transformation presented in [5] is so complicated, so clearly enough we cannot hope that a transformation as simple as LT would be complete. The best we can hope for is completeness under some additional restrictions on the shape of TRSs modelling the liveness problem. Indeed that is the case. First we present three such requirements along with examples showing that if they do not hold completeness is lost. However, for the setting of liveness problems, these requirements are quite mild. Then we will prove completeness for the restricted set of systems for which they do hold.

Example 1. Let us begin with a very simple example, namely:

$$R = \{\mathsf{top}(a) \to \mathsf{top}(b), \qquad b \to a\} \qquad\qquad R^{=} = \emptyset.$$

Consider liveness with $P = \{a\}$, meaning that in every reduction eventually a term without a is reached. It is an easy observation that in every infinite reduction of this TRS its two rules have to be applied interchangeably, so after at most one step the term without a is reached and liveness holds. But the transformation yields:

$$\begin{aligned} \mathrm{LT}(R) &= \{\mathsf{top}(\mathsf{ok}(a)) \to \mathsf{top}(\mathsf{check}(b)),\ b \to a\} \\ \mathrm{LT}^{=}(R^{=}, P) &= \{\mathsf{check}(a) \overset{=}{\to} \mathsf{ok}(a)\}. \end{aligned}$$

The above system allows an infinite $\to_{\mathrm{LT}} / \overset{=}{\to}_{\mathrm{LT}}$ reduction, namely:

$$\mathsf{top}(\mathsf{ok}(a)) \to_{\mathrm{LT}} \mathsf{top}(\mathsf{check}(b)) \to_{\mathrm{LT}} \mathsf{top}(\mathsf{check}(a)) \overset{=}{\to}_{\mathrm{LT}} \mathsf{top}(\mathsf{ok}(a)) \to_{\mathrm{LT}} \dots$$

The reason why things go wrong here is that some term from P (being a in this case) can be created, that is there are reductions from terms not containing an instance of p (for any $p \in P$) to terms containing an instance of p (for some $p \in P$). We can mend that by forbidding this kind of behavior. Let us note that this means restricting to liveness problems for which if the desired property holds at some point it will hold from that point onwards.

From now on, for readability concerns, we will assume that rules from R are given as $l \to r$ and rules from $R^{=}$ as $l \overset{=}{\to} r$ and we will just present a set of rules leaving the separation to R and $R^{=}$ implicit. Now we move on to another example showing another property that can destroy liveness.

Example 2. Consider the TRS over $\{f, g, \mathsf{top}, a, b\}$ consisting of the following rules:

$$\mathsf{top}(g(x, y, a)) \to \mathsf{top}(f(x)), \qquad\qquad f(x) \to g(x, x, x).$$

In any infinite top reduction the second rule is applied infinitely often, and a straightforward analysis shows that after applying the second rule in a top reduction, no infinite reduction from a term containing the symbol b is possible. So liveness with $P = \{b\}$ holds. The transformed system reads:

$$\begin{array}{lrcllrcl}
(1) & \mathsf{top}(\mathsf{ok}(g(x,y,a))) & \to & \mathsf{top}(\mathsf{check}(f(x))) & (7) & \mathsf{check}(g(x,y,z)) & \overset{=}{\to} & g(x,y,\mathsf{check}(z)) \\
(2) & f(x) & \to & g(x,x,x) & (8) & f(\mathsf{ok}(x)) & \overset{=}{\to} & \mathsf{ok}(f(x)) \\
(3) & \mathsf{check}(b) & \overset{=}{\to} & \mathsf{ok}(b) & (9) & g(\mathsf{ok}(x),y,z) & \overset{=}{\to} & \mathsf{ok}(g(x,y,z)) \\
(4) & \mathsf{check}(f(x)) & \overset{=}{\to} & f(\mathsf{check}(x)) & (10) & g(x,\mathsf{ok}(y),z) & \overset{=}{\to} & \mathsf{ok}(g(x,y,z)) \\
(5) & \mathsf{check}(g(x,y,z)) & \overset{=}{\to} & g(\mathsf{check}(x),y,z) & (11) & g(x,y,\mathsf{ok}(z)) & \overset{=}{\to} & \mathsf{ok}(g(x,y,z)) \\
(6) & \mathsf{check}(g(x,y,z)) & \overset{=}{\to} & g(x,\mathsf{check}(y),z) & & & &
\end{array}$$

and allows an infinite reduction, namely:

$$\begin{aligned}
&\mathsf{top}(\mathsf{check}(f(\mathsf{ok}(a)))) \overset{(2)}{\to} \mathsf{top}(\mathsf{check}(g(\mathsf{ok}(a),\mathsf{ok}(a),\mathsf{ok}(a)))) \overset{\overset{(6)}{=}}{\to} \\
\mathsf{top}(g(\mathsf{ok}(a),\mathsf{check}(\mathsf{ok}(a)),\mathsf{ok}(a))) &\overset{\overset{(11)}{=}}{\to} \mathsf{top}(\mathsf{ok}(g(\mathsf{ok}(a),\mathsf{check}(\mathsf{ok}(a)),a))) \overset{(1)}{\to} \\
&\mathsf{top}(\mathsf{check}(f(\mathsf{ok}(a)))) \to \dots
\end{aligned}$$

This time completeness was harmed by duplicating rules in the original system.

Example 3. Finally consider the following simple TRS consisting of two rules: $top(f(a)) \to top(b)$ and $b \to b$. Clearly liveness with $P = \{a\}$ holds but after transformation we obtain a non-terminating TRS since b can be rewritten to itself. This gives rise to the third, and last, requirement, namely that the signature of the TRS for which we consider liveness problem must contain at least one symbol of arity ≥ 2. This is a really weak requirement: it is not required that this symbol occurs in the rewrite rules.

Now we will prove that if all three restrictions are satisfied, that is there are no duplicating rules, terms from P cannot be created and Σ contains some symbol of arity ≥ 2, then the completeness holds.

Before we state the completeness theorem we need some auxiliary results. First let us denote by $\overline{t}$ the term t after removing all occurrences of ok and check symbols. Formally:

$$\begin{aligned}
\overline{\mathsf{check}(t)} &= \overline{t} \\
\overline{\mathsf{ok}(t)} &= \overline{t} \\
\overline{f(t_1, \ldots, t_n)} &= f(\overline{t_1}, \ldots, \overline{t_n}) \quad \text{for } f \notin \{\mathsf{check}, \mathsf{ok}\}
\end{aligned}$$

We need two lemmas for which the proofs are easy and can be found in [8]. First we will state the lemma which shows that the reduction steps in a transformed system can be mimicked in the original system after removing extra ok and check symbols.

Lemma 10 *Given two TRSs R and $R^=$ over the same signature Σ and arbitrary terms t, u, we have the following implications:*

$$\begin{aligned}
\text{(i) } t \to_{\mathrm{LT}} u &\implies \overline{t} \to \overline{u} \\
\text{(ii) } t \overset{=}{\to}_{\mathrm{LT}} u &\implies \overline{t} \overset{=}{\to}^{*} \overline{u}
\end{aligned}$$

Later on we will need the following lemma stating that extending TRS with administrative rules for check and ok preserves termination.

Lemma 11 *Given two TRSs R and $R^=$ over Σ ($\mathsf{top}, \mathsf{ok}, \mathsf{check} \notin \Sigma$). Let S consist of the following rules:*

$$\begin{aligned}
\mathsf{check}(p) &\to \mathsf{ok}(p) \qquad \textit{for all } p \in P \\
\mathsf{check}(f(x_1, \ldots, x_n)) &\to f(x_1, \ldots, \mathsf{check}(x_i), \ldots, x_n) \\
&\qquad \textit{for all } f \in \Sigma \textit{ of arity } n \geq 1,\ i = 1, \ldots, n \\
f(x_1, \ldots, \mathsf{ok}(x_i), \ldots, x_n)) &\to \mathsf{ok}(f(x_1, \ldots, x_n)) \\
&\qquad \textit{for all } f \in \Sigma \textit{ of arity } n \geq 1,\ i = 1, \ldots, n
\end{aligned}$$

Now if $\mathrm{SN}(R/R^=)$ *then* $\mathrm{SN}(R/(R^= \cup S))$.

Proof. Easy using Lemma 10.

Now we will present the theorem stating that for non-duplicating TRSs relative termination on top terms is equivalent to relative termination on arbitrary terms. We start by an example showing that non-duplication is essential for that.

Example 4. Let us consider the following TRS:

$$\mathsf{top}(f(x)) \to \mathsf{top}(a) \qquad\qquad f(x) \overset{=}{\to} g(f(x), f(x))$$

Here relative termination on top terms follows from the observation that any $\to$-step on any top term always yields the normal form $\mathsf{top}(a)$. However, this system admits the following fair reduction:

$$f(\mathsf{top}(f(x)) \overset{=}{\to} g(f(\mathsf{top}(f(x)), f(\mathsf{top}(f(x))) \to g(f(\mathsf{top}(a), \underbrace{f(\mathsf{top}(f(x)))}_{\text{initial term}} \overset{=}{\to} \cdots .$$

Theorem 12 *Let $R, R^=$ be non-duplicating top TRSs over Σ. Then we have:*

$$\mathrm{SN}(\mathcal{T}(\Sigma, \mathcal{V}), R/R^=) \iff \mathrm{SN}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R/R^=)$$

Proof. The ($\Rightarrow$)-part is trivial. For the ($\Leftarrow$)-part assume we have an arbitrary infinite fair reduction; we have to prove that there is also an infinite fair top reduction. By putting a top symbol around all terms we force that all terms in the infinite fair reduction have top as the root symbol. Next among all infinite fair reductions having top as the root symbol we choose one in which the number N of top symbols occurring in the initial term is minimal. Due to non-duplication in every term in this reduction at most N top symbols occur; due to minimality of N we conclude that each of these terms contains exactly N top symbols. We write $\mathsf{top}(C[\mathsf{top}(t_1), \ldots, \mathsf{top}(t_n)])$ for the initial term in the reduction for a context C not containing the symbol top. Since the number of top-symbols remains unchanged every term in the reduction is of the same shape, having the same number n of holes in the context. Due to minimality every infinite $\to \cup \overset{=}{\to}$ reduction of $\mathsf{top}(t_i)$ contains only finitely many $\to$-steps, for $i = 1, \ldots, n$. Due to definition of top TRSs all steps are either in the context C or in descendants of $\mathsf{top}(t_i)$. Since the descendants of $\mathsf{top}(t_i)$ allow only finitely many $\to$-steps and there are infinitely many $\to$-steps in total, we conclude that there are infinitely many $\to$-steps in the contexts. More precisely, we arrive at an infinite top reduction of $\mathsf{top}(C[x, \ldots, x])$ containing infinitely many $\to$-steps, contradicting $\mathrm{SN}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R/R^=)$. $\square$

Now we formulate the theorem which states that, under the three extra requirements introduced before, the transformation defined in Sect. 3.2 is complete.

Theorem 13 (Completeness) *Let R, $R^=$ be top TRSs over $\Sigma \cup \{\mathsf{top}\}$. If the following conditions are satisfied:*

(i) if u contains an instance of some $p \in P$ and $t \to u$ or $t \overset{=}{\to} u$ then t also contains an instance of p,

(ii) both R and $R^=$ are non-duplicating,

(iii) there is at least one function symbol of arity ≥ 2 in Σ.

then:

$$\mathrm{Live}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R, R^{=}, G(P)) \implies \mathrm{SN}(\mathrm{LT}(R)/\,\mathrm{LT}^{=}(R^{=}, P))$$

Proof. Assume $\mathrm{Live}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R, R^{=}, G(P))$ and conditions (i)-(iii) hold and $\mathrm{SN}(\mathrm{LT}(R)/\,\mathrm{LT}^{=}(R^{=}, P))$ does not hold. Then there is an infinite $\rightarrow_{\mathrm{LT}} / \overset{=}{\rightarrow}_{\mathrm{LT}}$ reduction. Due to non-duplication of R and $R^{=}$, $\mathrm{LT}(R)$ and $\mathrm{LT}^{=}(R^{=}, P)$ are also non-duplicating and by application of Theorem 12 we get that there is an infinite top $\rightarrow_{\mathrm{LT}} / \overset{=}{\rightarrow}_{\mathrm{LT}}$ reduction, call it ω.

Assume infinitely many terms in ω contain instances of terms from P. By the observation that an instance of p occurs in $\overline{C[p\delta]}$, Lemma 10 applied to ω gives rise to an infinite top reduction in $R/R^{=}$ having infinitely many terms containing instances of $p \in P$. Due to (i) all terms in this infinite reduction contain instances of $p \in P$ contradicting $\mathrm{Live}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R, R^{=}, G(P))$. Hence only finitely many terms in ω contain instances of terms from P. So removing this finite prefix of ω yields an infinite top $\rightarrow_{\mathrm{LT}} / \overset{=}{\rightarrow}_{\mathrm{LT}}$ reduction ω' in which no instance of $p \in P$ occurs at all.

Note that non-top rules of R are relatively terminating to non-top rules of $R^{=}$. Assume they are not. Then there is an infinite $\rightarrow / \overset{=}{\rightarrow}$ reduction sequence obtained using non-top rules of R and $R^{=}$. Let $f \in \Sigma$ be a function symbol of arity ≥ 2 (its existence is ensured by (iii)). Put the infinite $\rightarrow / \overset{=}{\rightarrow}$ reduction in context $\mathsf{top}(f(p, \Box, \ldots))$. This yields an infinite, fair top reduction containing p and thus contradicting $\mathrm{Live}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R, R^{=}, G(P))$. Now, by application of Lemma 11, we conclude that non-top rules of $\mathrm{LT}(R)$ are relatively terminating to non-top rules of $\mathrm{LT}^{=}(R^{=}, P)$.

In ω' top rules are applied infinitely often as non-top rules of $\mathrm{LT}(R)$ are relatively terminating to non-top rules of $\mathrm{LT}^{=}(R^{=}, P)$. Note that because of (ii) the only way to create an ok symbol is by application of the rule $\mathsf{check}(p) \overset{=}{\rightarrow} \mathsf{ok}(p)$. Every top reduction removes one occurrence of the ok symbol, so the aforementioned rule should be applied infinitely often. But since p does not occur in ω' this rule is not applicable which leads to a contradiction and ends the proof. $\Box$

Examples 1, 2 and 3 show that conditions (i)-(iii) of this theorem are essential.

4 Examples

In this section we present two examples illustrating the applicability of the proposed transformation. None of them could be treated with the use of the LS transformation described in [5]. Both relative termination proofs of the transformed systems were found completely automatically by TPA .

Example 5 (Cars over a bridge). There is a road with cars going in two directions. But on their way there is a bridge which is only wide enough to permit a single lane of traffic. So there are lights indicating which side of the bridge is allowed to cross it. We want to verify the liveness property, namely that every car will eventually be able to cross the bridge. For that clearly we need some assumptions about the lighting system. We want to be as general as possible so instead of assuming some particular algorithm of switching lights we just require them to change in a fair way, that is in the infinite observation of the system there must be infinitely many light switches (or equivalently: no matter when we start watching the road after some, arbitrary, time we will see the change of lights). Also we assume that before a light switches at least one car will pass (otherwise liveness is lost as lights can change all the time without any cars passing).

We model the system with a unary *top* symbol whose arguments start with a binary symbol *left* or *right* indicating which side has a green light. The arguments of *left* and *right* start with unary symbols *new* and *old* representing cars waiting to cross the bridge. The constant *bot* stands for the end of the queue. New cars are allowed to arrive at the end of the queue at any time. What we want to prove is that finally no old car remains.

$$
\begin{array}{lrcl}
(1) & \mathsf{top}(\mathsf{left}(\mathsf{old}(x), y)) & \to & \mathsf{top}(\mathsf{right}(x, y)) \\
(2) & \mathsf{top}(\mathsf{left}(\mathsf{new}(x), y)) & \to & \mathsf{top}(\mathsf{right}(x, y)) \\
(3) & \mathsf{top}(\mathsf{right}(x, \mathsf{old}(y))) & \to & \mathsf{top}(\mathsf{left}(x, y)) \\
(4) & \mathsf{top}(\mathsf{right}(x, \mathsf{new}(y))) & \to & \mathsf{top}(\mathsf{left}(x, y)) \\
(5) & \mathsf{top}(\mathsf{left}(\mathsf{bot}, y)) & \to & \mathsf{top}(\mathsf{right}(\mathsf{bot}, y)) \\
(6) & \mathsf{top}(\mathsf{right}(x, \mathsf{bot})) & \to & \mathsf{top}(\mathsf{left}(x, \mathsf{bot})) \\
(7) & \mathsf{top}(\mathsf{left}(\mathsf{old}(x), y)) & \overset{=}{\to} & \mathsf{top}(\mathsf{left}(x, y)) \\
(8) & \mathsf{top}(\mathsf{left}(\mathsf{new}(x), y)) & \overset{=}{\to} & \mathsf{top}(\mathsf{left}(x, y)) \\
(9) & \mathsf{top}(\mathsf{right}(x, \mathsf{old}(y))) & \overset{=}{\to} & \mathsf{top}(\mathsf{right}(x, y)) \\
(10) & \mathsf{top}(\mathsf{right}(x, \mathsf{new}(y))) & \overset{=}{\to} & \mathsf{top}(\mathsf{right}(x, y)) \\
(11) & \mathsf{bot} & \overset{=}{\to} & \mathsf{new}(\mathsf{bot})
\end{array}
$$

We have the following semantics of the rules:

(1) – (4)	Car passes and the light changes.
(5) – (6)	No car waiting, light can change.
(7) – (10)	Car passes, light remains the same.
(11)	New car arriving.

We want to prove liveness with $P = \{\mathsf{old}(x)\}$. For that purpose we need to show relative termination of the transformed system. It is an easy observation that the following procedure is termination-preserving: if for every rule the number of occurrences of some symbol is bigger or equal in the left hand side than in the right hand side, then remove the rules for which it is strictly bigger. This approach, already presented in [5], corresponds to proving termination with polynomial orderings with successor as interpretation for symbol begin counted and identity for all the other symbols.

The proof of relative termination can be given as follows. First count occurrences of old to remove four rules. Then apply semantic labelling over $\{0, 1\}$ taking constant 1 for old, identity for remaining unary symbols, disjunction for all binary symbols and constant 0 for bot. In the resulting system repeatedly apply counting argument to remove all the $\rightarrow$ rules thus proving relative termination. The details including the proof generated automatically by TPA can be found in [8].

The next example we investigate is commonly known as "the readers-writers problem" and goes back to Courtois et al. [2]. It is considered as a classical synchronization problem.

Example 6 (The readers and writers problem). Some resource is to be shared among a number of processes. There are two types of processes: "readers", which perform only reading operation and "writers" which can perform both reading and writing. The safety requirement is that writers must have exclusive access to the resource (that is when a writer has access to the resource no other process can have it) whereas readers can share the access (as long as there is no writer active at the same time).

It is usual in literature ([11], [10]) to concentrate only on safety requirements and propose a solution with priority for readers (writers) which can clearly lead to starvation of writers (readers). In [7] a fair solution to this problem has been proposed. We will present another variant of this starvation-free solution, where the access to the resource is controlled in a first-come first-served manner and we will verify that indeed starvation is not possible, corresponding to liveness.

To achieve that we introduce a flag indicating which group of processes (either readers or writers) has priority. If only one group claims the resource it is simply allowed to use it. But in case of a conflict, that is two groups interested in use of the resource, the group having priority is allowed to access it and then the priority is changed. Without adding this priority flag obviously the desried liveness property does not hold.

As in example 5 we distinguish between old and new processes and verify that finally there are no old processes in the system. We model reader processes by unary function symbols: RAO, RAN, RIO, RIN where the second character indicates whether the reader is currently **A**ctive (performs reading) or **I**nactive (waits for access to the resource) and the third character indicates whether the reader is **O**ld or **N**ew. The argument is used to organize processes into lists. Similarly for writers we have WAO, WAN, WIO, WIN. However WAO and WAN are constants as there can be at most one active writing process at a time and there is no need to keep a list of such processes.

The whole system is then modelled by means of binary function symbols sys-r or sys-w indicating priority for readers or writers respectively. The first argument describes all readers in the system and the second one models writers. Readers are modelled by a binary operator read whose first argument contains the list of active processes terminated by constant RT and the second argument contains the list of processes waiting for the resource terminated by constant RB.

Similarly, the binary operator write describes writers processes. Its first argument can be either WT (no active writer), WAO (active old writer) or WAN (active new writer). The second argument describes a list of inactive writers.

Due to using lists to represent active processes, we make one additional restriction that simplifies the modelling substantially, namely we assume that reading processes free the resource in the same order as they got access to it. It corresponds to the situation when the reading operation always takes some fixed interval of time. Now we are ready to present the model of the system.

$$\begin{array}{lrcl@{\qquad}lrcl}
(1) & \mathsf{RB} & \xrightarrow{\equiv} & \mathsf{RIN}(\mathsf{RB}) & (4) & \mathsf{RAN}(\mathsf{RT}) & \rightarrow & \mathsf{RT} \\
(2) & \mathsf{WB} & \xrightarrow{\equiv} & \mathsf{WIN}(\mathsf{WB}) & (5) & \mathsf{WAO} & \rightarrow & \mathsf{WT} \\
(3) & \mathsf{RAO}(\mathsf{RT}) & \rightarrow & \mathsf{RT} & (6) & \mathsf{WAN} & \rightarrow & \mathsf{WT}
\end{array}$$

$$\begin{array}{lrcl}
(7) & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(r_1, \mathsf{RIO}(r_2)), \mathsf{write}(\mathsf{WT}, \mathsf{WB}))) & \xrightarrow{\equiv} & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(\mathsf{RAO}(r_1), r_2), \mathsf{write}(\mathsf{WT}, \mathsf{WB}))) \\
(8) & \mathsf{top}(\mathsf{sys\text{-}w}(\mathsf{read}(r_1, \mathsf{RIO}(r_2)), \mathsf{write}(\mathsf{WT}, \mathsf{WB}))) & \xrightarrow{\equiv} & \mathsf{top}(\mathsf{sys\text{-}w}(\mathsf{read}(\mathsf{RAO}(r_1) r_2), \mathsf{write}(\mathsf{WT}, \mathsf{WB}))) \\
(9) & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(r_1, \mathsf{RIN}(r_2)), \mathsf{write}(\mathsf{WT}, \mathsf{WB}))) & \xrightarrow{\equiv} & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(\mathsf{RAN}(r_1), r_2), \mathsf{write}(\mathsf{WT}, \mathsf{WB}))) \\
(10) & \mathsf{top}(\mathsf{sys\text{-}w}(\mathsf{read}(r_1, \mathsf{RIN}(r_2)), \mathsf{write}(\mathsf{WT}, \mathsf{WB}))) & \xrightarrow{\equiv} & \mathsf{top}(\mathsf{sys\text{-}w}(\mathsf{read}(\mathsf{RAN}(r_1), r_2), \mathsf{write}(\mathsf{WT}, \mathsf{WB}))) \\
(11) & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(\mathsf{RT}, \mathsf{RB}), \mathsf{write}(\mathsf{WT}, \mathsf{WIN}(w)))) & \xrightarrow{\equiv} & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(\mathsf{RT}, \mathsf{RB}), \mathsf{write}(\mathsf{WAN}, w))) \\
(12) & \mathsf{top}(\mathsf{sys\text{-}w}(\mathsf{read}(\mathsf{RT}, \mathsf{RB}), \mathsf{write}(\mathsf{WT}, \mathsf{WIN}(w)))) & \xrightarrow{\equiv} & \mathsf{top}(\mathsf{sys\text{-}w}(\mathsf{read}(\mathsf{RT}, \mathsf{RB}), \mathsf{write}(\mathsf{WAN}, w))) \\
(13) & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(\mathsf{RT}, \mathsf{RB}), \mathsf{write}(\mathsf{WT}, \mathsf{WIO}(w)))) & \xrightarrow{\equiv} & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(\mathsf{RT}, \mathsf{RB}), \mathsf{write}(\mathsf{WAO}, w))) \\
(14) & \mathsf{top}(\mathsf{sys\text{-}w}(\mathsf{read}(\mathsf{RT}, \mathsf{RB}), \mathsf{write}(\mathsf{WT}, \mathsf{WIO}(w)))) & \xrightarrow{\equiv} & \mathsf{top}(\mathsf{sys\text{-}w}(\mathsf{read}(\mathsf{RT}, \mathsf{RB}), \mathsf{write}(\mathsf{WAO}, w))) \\
(15) & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(r_1, \mathsf{RIO}(r_2)), \mathsf{write}(\mathsf{WT}, w))) & \xrightarrow{\equiv} & \mathsf{top}(\mathsf{sys\text{-}w}(\mathsf{read}(\mathsf{RAO}(r_1), r_2), \mathsf{write}(\mathsf{WT}, w))) \\
(16) & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(r_1, \mathsf{RIN}(r_2)), \mathsf{write}(\mathsf{WT}, w))) & \xrightarrow{\equiv} & \mathsf{top}(\mathsf{sys\text{-}w}(\mathsf{read}(\mathsf{RAN}(r_1), r_2), \mathsf{write}(\mathsf{WT}, w))) \\
(17) & \mathsf{top}(\mathsf{sys\text{-}w}(\mathsf{read}(\mathsf{RT}, r_2), \mathsf{write}(\mathsf{WT}, \mathsf{WIO}(w)))) & \xrightarrow{\equiv} & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(\mathsf{RT}, r_2), \mathsf{write}(\mathsf{WAO}, w))) \\
(18) & \mathsf{top}(\mathsf{sys\text{-}w}(\mathsf{read}(\mathsf{RT}, r_2), \mathsf{write}(\mathsf{WT}, \mathsf{WIN}(w)))) & \xrightarrow{\equiv} & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(\mathsf{RT}, r_2), \mathsf{write}(\mathsf{WAN}, w)))
\end{array}$$

The meaning of the rules is as follows:

$(1-2)$	New inactive process appears in the system and is queued to wait for the resource.
$(3-6)$	Active process finishes reading/writing.
$(7-10)$	Nobody is writing nor waiting for write access — inactive reading process is allowed to start reading; priority does not change
$(11-14)$	Nobody is reading nor waiting for read access and nobody is writing — writer is allowed to start writing; priority does not change.
$(15-16)$	Nobody is writing and priority is for readers — reader is allowed to start reading; priority is switched.
$(17-18)$	Nobody is reading nor writing and priority is for writers — writer is allowed to start writing; priority is switched.

What we want to prove is that finally no old process remains in the system. This corresponds to verifying liveness with the set $P = \{\mathsf{RAO}(x), \mathsf{RIO}(x), \mathsf{WAO}, \mathsf{WIO}(x)\}$.

To prove this liveness property we need to show relative termination of the transformed system. This was done completely automatically by TPA . The proof produced by TPA consists of more than 1000 lines. It proceeds by repeating a number of times the following procedure: apply semantic labelling, remove some rules using polynomial interpretations, unlabel the system to obtain a TRS with few rules less than before labelling. The proof applies semantic labelling with different interpretations 7 times. For a more detailed description we again refer to [8].

5 Conclusions

This paper describes a technique to transform a liveness problem with fairness to the problem of proving relative termination of a transformed TRS. In two presented examples the latter could be done fully automatically. The only human activity in this approach is modelling the original problem in the language of term rewriting. The advantage of this approach compared to standard model checking is that it can easily deal with liveness problems involving infinite state spaces.

In modelling liveness problem as TRS there is usually a lot of choice. This choice can influence the difficulty of the corresponding termination problem. This holds for instance for readers-writers example presented in Section 4 for which we considered a number of variations not all of which could be proved by TPA. Therefore improving TPA to be able to handle the broader class of relative termination problems is an obvious goal. Therefore we consider work on adopting existing termination techniques (like dependency pairs) for proving relative termination as well as developing techniques specifically for proving relative termination to be an interesting subject of further research.

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theor. Comput. Sci.*, 236(1-2):133–178, 2000.
2. P. J. Courtois, F. Heymans, and D. L. Parnas. Concurrent control with "readers" and "writers". *Commun. ACM*, 14(10):667–668, 1971.
3. N. Dershowitz. Orderings for term-rewriting systems. *Theor. Comput. Sci.*, 17:279–301, 1982.
4. N. Francez. *Fairness.* Springer-Verlag, 1986.
5. J. Giesl and H. Zantema. Liveness in rewriting. In *Proc. 14th RTA, LNCS 2706*, pages 321–336, 2003.
6. J. Giesl and H. Zantema. Simulating liveness by reduction strategies. *Electr. Notes TCS*, 86(4), 2003.
7. C. A. R. Hoare. Monitors: an operating system structuring concept. *Commun. ACM*, 17(10):549–557, 1974.
8. A. Koprowski and H. Zantema. Proving liveness with fairness using rewriting. Technical Report CS-Report 05-06, Tech. Univ. of Eindhoven, March 2005. Available from `http://w3.tue.nl/en/services/library/digilib/tue_publications/reports/`.
9. D. S. Lankford. On proving term rewriting systems are noetherian. Technical Report MTP-3, Math. Dept., Louisiana Tech. Univ., Ruston, May 1979.
10. M. Raynal and D. Beeson. *Algorithms for mutual exclusion.* MIT Press, Cambridge, MA, USA, 1986.
11. A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating system concepts.* John Wiley & Sons, Inc., 2004.
12. TeReSe. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in TCS.* Cambridge University Press, 2003.
13. H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24(1/2):89–105, 1995.