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. We show how the problem of verifying liveness properties in fair computations is related to relative termination of term rewrite systems (TRSs), extending [3]. We present a new transformation that is stronger than the sound transformation introduced in [3]. On the one hand we show a completeness result under some mild conditions. On the other hand we show how our transformation applies to examples where the previous approach fails. In particular we succeed in automatically 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 [3] this relation 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. Assuming that computation steps can be described by term rewriting and that the set of good states representing the "eventual happenings" can be represented as a set of terms not containing a particular term, in [3] a method for transforming liveness problems to problems of termination of term rewrite systems (TRSs) has been proposed. For alternative sets of good states, in [4] another approach was proposed.

Two transformations were given in [3]. The first one, sound and complete, results in TRSs that turn out to be intractable even for extremely simple liveness problems. 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 [2]. In applications one typically is interested in behavior of infinite fair computations rather than 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 never a client is 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*.

The second extension is the following. It turns out that the simple transformation presented in [3] 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 [3], but much simpler than the sound and complete transformation from [3]. 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.

To this end a tool — TPA(`http://www.win.tue.nl/tpa`), part of the TORPA project — was developed for proving relative termination of TRSs automatically, based on polynomial interpretations and semantic labelling ([6]).

This paper is organized as follows. In Section 2 the general framework from [3] 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 examples are given 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 [3] 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 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 usually we write $\rightarrow^*$ for its reflexive transitive closure and $\rightarrow^+$ for its transitive closure. We define a set of states in *normal forms* as $\mathrm{NF} \equiv \{s \in S \mid \neg\exists_{s' \in S} : \ s \rightarrow s'\}$ and a set of normal forms 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, that is: $\mathrm{SN}(I, \rightarrow) \equiv \neg\exists_{t_1,t_2,\ldots} : t_1 \in I \wedge \forall_i : t_i \rightarrow t_{i+1}$.

With respect to a set of initial states $I \subseteq S$, 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*

- $\forall_{t_1,t_2,\ldots} : \left.\begin{array}{r} t_1 \in I \\ \forall_i : t_i \rightarrow t_{i+1} \end{array}\right\} \implies \exists_i : t_i \in G$, *and*
- $\forall_{t_1,t_2,\ldots,t_n} : \left.\begin{array}{r} t_1 \in I \\ t_n \in \mathrm{NF} \\ \forall_{i\in\{1,\ldots,n-1\}} : t_i \rightarrow t_{i+1} \end{array}\right\} \implies \exists_{i\in\{1,\ldots,n\}} : t_i \in G.$

We define the restricted computation relation $\rightarrow_G \equiv \{(s,t) \mid s \rightarrow t \wedge s \notin G\}$. The following theorem from [3] 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 $\rightarrow, \overset{=}{\rightarrow} \subseteq S \times S$. 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*

- $\forall_{t_1,t_2,\ldots} : \left.\begin{array}{r} t_1 \in I \\ \forall_i : t_i \rightarrow t_{i+1} \vee t_i \overset{=}{\rightarrow} t_{i+1} \\ \forall_i \exists_{j>i} : t_j \rightarrow t_{j+1} \end{array}\right\} \implies \exists_i : t_i \in G$

Our definition is based on the notion of relative termination. Let us recall that we say that $\rightarrow$ terminates relatively to $\overset{=}{\rightarrow}$ if every (possibly infinite) $\rightarrow \cup \overset{=}{\rightarrow}$ computation contains only finitely many $\rightarrow$ steps. We introduce the relation $\rightarrow / \overset{=}{\rightarrow} \;\equiv\; \overset{=}{\rightarrow}^* \cdot \rightarrow \cdot \overset{=}{\rightarrow}^*$ and observe that relative termination of $\rightarrow$ to $\overset{=}{\rightarrow}$ is equivalent to $\mathrm{SN}(\rightarrow / \overset{=}{\rightarrow})$.

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

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 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 [5].

Now, again after [3], 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* 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. We call this rule a* top rule *then.*
- *or both ℓ and r do not contain an instance of* top *symbol. Then the rule is called a* non-top rule*.*

Clearly every reduction starting in a top term only contains top terms. In the remaining 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 is of the form:

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

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:

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

for some top TRSes 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 (relative) termination question of an ordinary TRS. This allows us to employ all the existing 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 a number of interesting and practical examples, some 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 system implies that the liveness property in question holds (even better if we can have equivalence). In [3] 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. It can deal with a much broader class of liveness problems than the sound transformation from [3] and, although somehow more complicated, it is still suitable for automatic termination provers. But before we do that we will shortly introduce the sound transformation LS from [3], in which $P = \{p\}$.

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

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

While $\mathrm{LS}(R,p)$ is sound, it is not complete as we now see in a simple example not being really meaningful as a liveness example. Some more realistic examples, none of which can be treated by LS, will be presented in Sect. 4.

Example 1. Consider the following TRS:

$$\begin{aligned} \mathsf{top}(f(x,b)) &\to \mathsf{top}(f(b,b)) \\ a &\to b \end{aligned}$$

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:

$$\mathsf{top}(\mathsf{check}(f(b,b))) \rightleftarrows \mathsf{top}(f(\mathsf{check}(b),b))$$

The reason why things go wrong here is that the idea of the transformation is that when p does not occur in the term then every application of top rule produces check symbol that cannot be removed. It can only be propagated downwards in the term and the hope is that this will block further reductions and hence infer termination. But in the above example and in the examples that we will see in Sect. 4, this is not the case.

3.2 Definition of the Transformation

We give a new transformation inspired by the sound and complete transformation presented in [3] but significantly simpler so that obtained systems can still be treated with tools for automatic termination proving. We present it for only one unary top symbol but generalization to more top symbols and/or different arities is straightforward. We extend the setting from [3] by verifying liveness with respect to a set of terms (P) instead of a single term (p).

Definition 7 (LT) *Let R and $R^=$ be top TRSs systems 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:*

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

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

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

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

For readability concerns we will write $\rightarrow_{\mathrm{LT}}$ instead of $\rightarrow_{\mathrm{LT}(R)}$ and $\overset{=}{\rightarrow}_{\mathrm{LT}}$ instead of $\rightarrow_{\mathrm{LT}^{=}(R^{=},P)}$.

In order to apply automatical 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 $\mathrm{SN}(\mathrm{LT}(R)/\,\mathrm{LT}^{=}(R^{=}, P))$ holds and $\mathrm{Live}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R, R^{=}, G(P))$ does not hold. By Theorem 4, $\mathrm{SN}(\mathcal{T}_{top}(\Sigma, \mathcal{V}),\ \rightarrow_G / \overset{=}{\rightarrow}_G)$ 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, top $\rightarrow_G / \overset{=}{\rightarrow}_G$ reduction. We will show that this infinite reduction can be mapped to an infinite $\rightarrow_{\mathrm{LT}} / \overset{=}{\rightarrow}_{\mathrm{LT}}$ reduction, thus contradicting $\mathrm{SN}(\mathrm{LT}(R)/\,\mathrm{LT}^{=}(R^{=}, P))$. For that purpose it is sufficient to show that:

$$\mathsf{top}(t)\ \rightarrow_G / \overset{=}{\rightarrow}_G\ \mathsf{top}(u) \implies \mathsf{top}(\mathsf{ok}(t))\ \rightarrow_{\mathrm{LT}} / \overset{=}{\rightarrow}_{\mathrm{LT}}\ \mathsf{top}(\mathsf{ok}(u))$$

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

(i) whenever $\mathsf{top}(t) \rightarrow_G \mathsf{top}(u)$ then $\mathsf{top}(\mathsf{ok}(t))\ \rightarrow_{\mathrm{LT}} / \overset{=}{\rightarrow}_{\mathrm{LT}}\ \mathsf{top}(\mathsf{ok}(u))$, and

(ii) whenever $\mathsf{top}(t) \overset{=}{\rightarrow}_G \mathsf{top}(u)$ then $\mathsf{top}(\mathsf{ok}(t))\ \overset{=}{\rightarrow}^{*}_{\mathrm{LT}}\ \mathsf{top}(\mathsf{ok}(u))$.

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

If on the other hand $\mathsf{top}(t) \rightarrow_G \mathsf{top}(u)$ 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 $\mathsf{top}(\ell) \rightarrow \mathsf{top}(r)$. Also from the definition of $\rightarrow_G$ we get that $\mathsf{top}(u)$ does contain an instance of some $p \in P$ (as it is part of infinite reduction), which means that we have $\mathsf{top}(u) = \mathsf{top}(C[p\gamma])$ for some context C and some substitution γ. Then we have:

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

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

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 the use of LS can also be dealt with the use of 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 $p \in \mathcal{T}(\Sigma, \mathcal{V})$. If* $\mathrm{SN}(\mathrm{LS}(R,p))$ *then* $\mathrm{SN}(\mathrm{LT}^{=}(R, \{p\}))$.

Proof. Assume $\mathrm{SN}(\mathrm{LS}(R,p))$ and not $\mathrm{SN}(\mathrm{LT}^{=}(R, \{p\}))$. Then there is an infinite reduction sequence in $\mathrm{LT}^{=}(R, \{p\})$. It follows easily from definitions of $\mathrm{LT}^{=}$ and LS that by dropping all ok symbols from terms in this reduction we obtain a valid reduction sequence in $\mathrm{LS}(R,p)$. Only applications of $f(x_1, \ldots, \mathsf{ok}(x_i), \ldots, x_n) \rightarrow \mathsf{ok}(f(x_1, \ldots, x_n))$ become equalities but since this rule itself is terminating, in that infinite reduction there must be infinitely many steps not using this rule. So in this way we map an infinite reduction in $\mathrm{LT}^{=}(R, \{p\})$ to an infinite reduction in $\mathrm{LS}(R,p)$ thus contradicting $\mathrm{SN}(\mathrm{LS}(R,p))$. □

There is a good reason why the sound and complete transformation presented in [3] is so complicated, so clearly enough we cannot hope that as simple transformation 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 2. Let us begin with a very simple example of TRS, namely:

$$\begin{gathered} \mathsf{top}(a) \rightarrow \mathsf{top}(b) \\ b \rightarrow a \end{gathered}$$

It is an easy observation that in every infinite reduction those two rules have to be applied interchangeably, which means in particular that we have liveness with $P = \{a\}$. But after transformation we obtain the following system:

$$\begin{gathered} \mathsf{top}(\mathsf{ok}(a)) \rightarrow \mathsf{top}(\mathsf{check}(b)) \\ b \rightarrow a \\ \mathsf{check}(a) \xrightarrow{=} \mathsf{ok}(a) \end{gathered}$$

The above system allows an infinite reduction, namely:

$$\mathsf{top}(\mathsf{ok}(a)) \rightarrow \mathsf{top}(\mathsf{check}(b)) \rightarrow \mathsf{top}(\mathsf{check}(a)) \xrightarrow{=} \mathsf{top}(\mathsf{ok}(a)) \rightarrow \ldots$$

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.

Now we move on to another example showing another property that can destroy liveness.

Example 3. Consider the following TRS over $\{f, g, \mathsf{top}, a, b\}$:

$$\begin{aligned} \mathsf{top}(g(x,y,a)) &\to \mathsf{top}(f(x)) \\ f(x) &\to g(x,x,x) \end{aligned}$$

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 is possible if the term contains the symbol b. So liveness with $P = \{b\}$ holds. The transformed system reads:

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

and allows an infinite reduction, namely:

$$\begin{array}{c}
\mathsf{top}(\mathsf{check}(f(\mathsf{ok}(a)))) \xrightarrow{(2)} \mathsf{top}(\mathsf{check}(g(\mathsf{ok}(a),\mathsf{ok}(a),\mathsf{ok}(a)))) \overset{(6)}{\stackrel{\equiv}{\to}} \\
\mathsf{top}(g(\mathsf{ok}(a),\mathsf{check}(\mathsf{ok}(a)),\mathsf{ok}(a))) \overset{(11)}{\stackrel{\equiv}{\to}} \mathsf{top}(\mathsf{ok}(g(\mathsf{ok}(a),\mathsf{check}(\mathsf{ok}(a)),a))) \xrightarrow{(1)} \\
\mathsf{top}(\mathsf{check}(f(\mathsf{ok}(a)))) \to \ldots
\end{array}$$

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

Example 4. Finally consider the following simple TRS:

$$\begin{aligned} f(a) &\to b \\ b &\to b \end{aligned}$$

Clearly liveness with $P = \{a\}$ holds but after transformation we obtain:

$$\begin{aligned} a &\to b \\ b &\to b \\ \mathsf{check}(a) &\stackrel{\equiv}{\to} \mathsf{ok}(a) \end{aligned}$$

being non-terminating since b rewrites 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 in case none of the three properties mentioned above holds, 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:

$$\begin{aligned}
\overline{\mathsf{check}(t)} &= \overline{t} \\
\overline{\mathsf{ok}(t)} &= \overline{t} \\
\overline{f(x_1,\ldots,x_n)} &= f(\overline{x_1},\ldots,\overline{x_n}) \quad \text{for } f \notin \{\mathsf{check},\mathsf{ok}\}
\end{aligned}$$

We need two auxiliary lemmas. 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:*

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

Proof. (i) It follows from an easy observation that for any rule $\ell \rightarrow r \in \mathrm{LT}(R)$ we have $\overline{\ell} \rightarrow \overline{r} \in R$.

(ii) Similarly if $\ell \rightarrow r \in \mathrm{LT}^=(R^=, P)$ then $\overline{\ell} \rightarrow \overline{r} \in R^=$ or $\overline{\ell} = \overline{r}$, but in any way $\overline{t} \overset{=}{\rightarrow}{}^{*}\, \overline{u}$ in zero or one step.

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 ($\mathsf{top} \notin \Sigma$). Let S consist of the following rules:*

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

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

Proof. Assume $\mathrm{SN}(R/R^=)$ and not $\mathrm{SN}(R/(R^= \cup S))$. So there is an infinite $\rightarrow_R /(\rightarrow_{R^=} \cup \rightarrow_S)$ reduction. Because both R and $R^=$ do not contain top symbol then $R = \mathrm{LT}(R)$ and $R^= \cup S = \mathrm{LT}^=(R^=, P)$. So with the use of Lemma 10 we can map this infinite $\rightarrow_{\mathrm{LT}(R)} / \rightarrow_{\mathrm{LT}^=(R^=,P)}$ reduction to infinite $\rightarrow_R / \rightarrow_{R^=}$ reduction which contradicts $\mathrm{SN}(R/R^=)$. □

Now we will present the theorem stating that for non-duplicating TRSes 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 5. Let us consider the following TRS:

$$\begin{aligned} \mathsf{top}(f(x)) &\to \mathsf{top}(a) \\ f(x) &\overset{=}{\to} g(f(x), f(x)) \end{aligned}$$

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:

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

Next we prove that in case of non-duplication the above mentioned equivalence does hold. First we need a simple lemma (for which the proof is left to reader).

Lemma 12 *Let R be a non-duplicating top TRS over Σ. Let C, D be n-hole contexts not containing the symbol* top*, and let $t_1, \ldots, t_n, u_1, \ldots, u_n$ be terms possibly containing* top *for which*

$$\mathsf{top}(C[\mathsf{top}(t_1), \ldots, \mathsf{top}(t_n)]) \ \to_R \mathsf{top}(D[\mathsf{top}(u_1), \ldots, \mathsf{top}(u_n)]).$$

Then either

- *$C = D$ and $\mathsf{top}(t_i) \to_R \mathsf{top}(u_i)$ for some i and $t_j = u_j$ for all $j \neq i$, or*
- *$\mathsf{top}(C[x, \ldots, x]) \to_R \mathsf{top}(D[x, \ldots, x])$ and the multiset $\{\mathsf{top}(t_1), \ldots, \mathsf{top}(t_n)\}$ is equal to the multiset $\{\mathsf{top}(u_1), \ldots, \mathsf{top}(u_n)\}$.*

Now we formulate the theorem stating equivalence of relative termination on top terms with relative termination on arbitrary terms for non-duplicating TRSes.

Theorem 13 *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 'only if'-part is trivial. For the 'if'-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. 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. Due to non-duplication of the rules and non-decreasingness of the number of top symbols we conclude that 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 $\rightarrow \cup \overset{=}{\rightarrow}$ reduction of $\mathsf{top}(t_i)$ contains only finitely many $\rightarrow$-steps, for $i = 1, \ldots, n$. Due to Lemma 12 all steps are either in the context or in descendants of $\mathsf{top}(t_i)$. Since the descendants of $\mathsf{top}(t_i)$ allow only finitely many $\rightarrow$-steps and there are infinitely many $\rightarrow$-steps in total, we conclude that there are infinitely many $\rightarrow$-steps in the contexts. More precisely, we arrive at an infinite top reduction of $\mathsf{top}(C[x, \ldots, x])$ containing infinitely many $\rightarrow$-steps, contradicting $\mathrm{SN}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R/R^{=})$. □

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

Theorem 14 (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 $t \rightarrow u$ or $t \overset{=}{\rightarrow} u$ 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 13 we get that there is an infinite, top $\rightarrow_{\mathrm{LT}} / \overset{=}{\rightarrow}_{\mathrm{LT}}$ reduction.

First 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)). Then $\mathsf{top}(f(p, \Box, \ldots))$ with $p \in P$ and this reduction sequence in second argument is 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 also relatively terminating to non-top rules of $\mathrm{LT}^{=}(R^{=}, P)$.

From $\mathrm{Live}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R, R^{=}, G(P))$ we have that in every infinite, fair top reduction in R we finally reach the term in which p does not occur for all $p \in P$. We can conclude the same about $\mathrm{LT}(R)$ as, due to Lemma 10, every infinite, fair top reduction in $\mathrm{LT}(R)$ not containing p can be mapped to an infinite, fair top reduction in R not containing p (note that $\overline{p} = p$). Because of (i) p will not occur from that point onwards, which means that it can only occur in some finite prefix. After dropping this prefix we obtain an infinite, fair top reduction sequence without p. But non-top rules of $\rightarrow_{\mathrm{LT}} / \overset{=}{\rightarrow}_{\mathrm{LT}}$ are relatively terminating which means that in this sequence top rules have to be applied infinitely often.

Every top reduction removes one occurrence of ok symbol, so it should also be created infinitely often. Note that because of (ii) the only way to create ok symbol is by application of $\mathsf{check}(p) \overset{=}{\to} \mathsf{ok}(p)$ rule. But since p does not occur, this rule is not applicable which leads to contradiction and ends the proof. □

Examples 2, 3 and 4 show that conditions (i)-(iii) of this theorem are essential.

4 Examples

In this section we present two examples illustrating applicability of the proposed transformation. None of them could be treated with the use of the LS transformation described in [3]. Both relative termination proofs of the transformed systems were found completely automatically by TPA.

Example 6 (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 after some time all the cars will 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 unary *top* symbol which contains either binary symbol *left* or *right* indicating which side has a green light. Their arguments are unary symbols *new* and *old* representing cars waiting to cross the bridge. 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:

Rules	Semantics
(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.

Using the transformation as described in Sect. 3.2 with $P = \{\mathsf{old}(x)\}$ we obtain:

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{top}(\mathsf{ok}(\mathsf{left}(\mathsf{old}(x), y))) & \to & \mathsf{top}(\mathsf{check}(\mathsf{right}(x, y))) & \mathsf{bot} & \overset{=}{\to} & \mathsf{new}(\mathsf{bot}) \\
\mathsf{top}(\mathsf{ok}(\mathsf{left}(\mathsf{new}(x), y))) & \to & \mathsf{top}(\mathsf{check}(\mathsf{right}(x, y))) & \mathsf{new}(\mathsf{ok}(x)) & \overset{=}{\to} & \mathsf{ok}(\mathsf{new}(x)) \\
\mathsf{top}(\mathsf{ok}(\mathsf{right}(x, \mathsf{old}(y)))) & \to & \mathsf{top}(\mathsf{check}(\mathsf{left}(x, y))) & \mathsf{old}(\mathsf{ok}(x)) & \overset{=}{\to} & \mathsf{ok}(\mathsf{old}(x)) \\
\mathsf{top}(\mathsf{right}(x, \mathsf{new}(y))) & \to & \mathsf{top}(\mathsf{check}(\mathsf{left}(x, y))) & \mathsf{left}(\mathsf{ok}(x), y) & \overset{=}{\to} & \mathsf{ok}(\mathsf{left}(x, y)) \\
\mathsf{top}(\mathsf{ok}(\mathsf{left}(\mathsf{bot}, y))) & \to & \mathsf{top}(\mathsf{check}(\mathsf{right}(\mathsf{bot}, y))) & \mathsf{left}(x, \mathsf{ok}(y)) & \overset{=}{\to} & \mathsf{ok}(\mathsf{left}(x, y)) \\
\mathsf{top}(\mathsf{ok}(\mathsf{right}(x, \mathsf{bot}))) & \to & \mathsf{top}(\mathsf{check}(\mathsf{left}(x, \mathsf{bot}))) & \mathsf{right}(\mathsf{ok}(x), y) & \overset{=}{\to} & \mathsf{ok}(\mathsf{right}(x, y)) \\
\mathsf{top}(\mathsf{ok}(\mathsf{left}(\mathsf{old}(x), y))) & \overset{=}{\to} & \mathsf{top}(\mathsf{check}(\mathsf{left}(x, y))) & \mathsf{right}(x, \mathsf{ok}(y)) & \overset{=}{\to} & \mathsf{ok}(\mathsf{right}(x, y)) \\
\mathsf{top}(\mathsf{ok}(\mathsf{left}(\mathsf{new}(x), y))) & \overset{=}{\to} & \mathsf{top}(\mathsf{check}(\mathsf{left}(x, y))) & \mathsf{check}(\mathsf{old}(x)) & \overset{=}{\to} & \mathsf{ok}(\mathsf{old}(x)) \\
\mathsf{top}(\mathsf{ok}(\mathsf{right}(x, \mathsf{old}(y)))) & \overset{=}{\to} & \mathsf{top}(\mathsf{check}(\mathsf{right}(x, y))) & \mathsf{check}(\mathsf{new}(x)) & \overset{=}{\to} & \mathsf{new}(\mathsf{check}(x)) \\
\mathsf{top}(\mathsf{ok}(\mathsf{right}(x, \mathsf{new}(y)))) & \overset{=}{\to} & \mathsf{top}(\mathsf{check}(\mathsf{right}(x, y))) & & & \\
 & & & \mathsf{check}(\mathsf{left}(x, y)) & \overset{=}{\to} & \mathsf{left}(\mathsf{check}(x), y) \\
 & & & \mathsf{check}(\mathsf{left}(x, y)) & \overset{=}{\to} & \mathsf{left}(x, \mathsf{check}(y)) \\
 & & & \mathsf{check}(\mathsf{right}(x, y)) & \overset{=}{\to} & \mathsf{right}(\mathsf{check}(x), y) \\
 & & & \mathsf{check}(\mathsf{right}(x, y)) & \overset{=}{\to} & \mathsf{right}(x, \mathsf{check}(y))
\end{array}$$

All requirements of Theorem 14 are satisfied which means that by answering, either positively or negatively, the question of relative termination of the above TRS we can, respectively, prove or disapprove the liveness property.

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 [3], 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 $\to$ rules thus proving relative termination. The detailed, automatically generated relative termination proof can be found in Appendix A.

The next example we investigate is commonly known as "the readers-writers problem" and goes back to Courtois et al. [1]. It is considered as a classical synchronization problem.

Example 7 (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 a time).

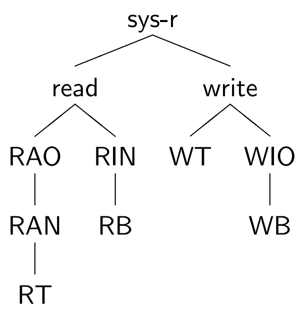
It is very usual in literature to concentrate only on safety requirement and propose a solution with priority for readers (writers) which can clearly lead to starvation of writers (readers). We will present a starvation-free solution and we will verify that indeed starvation is not possible, corresponding to liveness.

To achieve that we introduce a flag indicating which group of processes 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, 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 6 we distinguish between old and new processes and verify that finally there are no old processes in the system. We model readers processes by unary function symbols: RAO, RAN, RIO, RIN where the second character indicates whether reader is currently **A**ctive (performs reading) or **I**nactive (waits for access to the resource) and the third character indicates whether 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 only 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 symbol 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 binary operator read with first argument containing list of active processes terminated by constant RT and the second argument containing list of processes waiting for the resource terminated by constant RB. Similarly binary operator write describes writers processes with first argument being either WT — no active writer, WAO — active old writer or WAN — active new writer. The second argument of write describes a list of inactive writers.

Let us take a look at the following term tree:



It describes a system with two processes currently performing reading; one of them is old and the other is new. There is also one new reader waiting for the resource. There are no processes performing writing but there is one old writing process waiting for the resource. The priority is set for readers.

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 situation when the reading operation always takes some fixed interval of time. Now we are ready to present the model of the system.

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

$$\begin{array}{rrcl}
(7) & \mathsf{top}(\mathsf{sys\text{-}r}(\mathsf{read}(r_1, \mathsf{RIO}(r_2)), \mathsf{write}(\mathsf{WT}, \mathsf{WB}))) & \overset{=}{\rightarrow} & \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}))) & \overset{=}{\rightarrow} & \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}))) & \overset{=}{\rightarrow} & \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}))) & \overset{=}{\rightarrow} & \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)))) & \overset{=}{\rightarrow} & \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)))) & \overset{=}{\rightarrow} & \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)))) & \overset{=}{\rightarrow} & \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)))) & \overset{=}{\rightarrow} & \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))) & \overset{=}{\rightarrow} & \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))) & \overset{=}{\rightarrow} & \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)))) & \overset{=}{\rightarrow} & \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)))) & \overset{=}{\rightarrow} & \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:

Rules	Semantics
$(1-2)$	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 $P = \{\mathsf{RAO}(x), \mathsf{RIO}(x), \mathsf{WAO}, \mathsf{WIO}(x)\}$.

In this way we obtain the following TRS:

$$\begin{array}{rrcl@{\qquad}rrcl}
(1) & \mathsf{RB} & \overset{=}{\rightarrow} & \mathsf{RIN}(\mathsf{RB}) & & & & \\
(2) & \mathsf{WB} & \overset{=}{\rightarrow} & \mathsf{WIN}(\mathsf{WB}) & (19) & \mathsf{check}(\mathsf{RIO}(x)) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{RIO}(x)) \\
(3) & \mathsf{RAO}(\mathsf{RT}) & \rightarrow & \mathsf{RT} & (20) & \mathsf{check}(\mathsf{RAO}(x)) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{RAO}(x)) \\
(4) & \mathsf{RAN}(\mathsf{RT}) & \rightarrow & \mathsf{RT} & (21) & \mathsf{check}(\mathsf{WIO}(x)) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{WIO}(x)) \\
(5) & \mathsf{WAO} & \rightarrow & \mathsf{WT} & (22) & \mathsf{check}(\mathsf{WAO}) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{WAO}) \\
(6) & \mathsf{WAN} & \rightarrow & \mathsf{WT} & & & &
\end{array}$$

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

$$\begin{array}{rrclrrcl}
(23) & \mathsf{check}(\mathsf{RIO}(x)) & \overset{=}{\rightarrow} & \mathsf{RIO}(\mathsf{check}(x)) & (33) & \mathsf{RIO}(\mathsf{ok}(x)) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{RIO}(x))\\
(24) & \mathsf{check}(\mathsf{RIN}(x)) & \overset{=}{\rightarrow} & \mathsf{RIN}(\mathsf{check}(x)) & (34) & \mathsf{RIN}(\mathsf{ok}(x)) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{RIN}(x))\\
(25) & \mathsf{check}(\mathsf{RAO}(x)) & \overset{=}{\rightarrow} & \mathsf{RAO}(\mathsf{check}(x)) & (35) & \mathsf{RAO}(\mathsf{ok}(x)) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{RAO}(x))\\
(26) & \mathsf{check}(\mathsf{RAN}(x)) & \overset{=}{\rightarrow} & \mathsf{RAN}(\mathsf{check}(x)) & (36) & \mathsf{RAN}(\mathsf{ok}(x)) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{RAN}(x))\\
(27) & \mathsf{check}(\mathsf{WIO}(x)) & \overset{=}{\rightarrow} & \mathsf{WIO}(\mathsf{check}(x)) & (37) & \mathsf{WIO}(\mathsf{ok}(x)) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{WIO}(x))\\
(28) & \mathsf{check}(\mathsf{WIN}(x)) & \overset{=}{\rightarrow} & \mathsf{WIN}(\mathsf{check}(x)) & (38) & \mathsf{WIN}(\mathsf{ok}(x)) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{WIN}(x))\\
(29) & \mathsf{check}(\mathsf{sys\text{-}r}(x,y)) & \overset{=}{\rightarrow} & \mathsf{sys\text{-}r}(\mathsf{check}(x),y) & (39) & \mathsf{sys\text{-}r}(\mathsf{ok}(x),y) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{sys\text{-}r}(x,y))\\
(30) & \mathsf{check}(\mathsf{sys\text{-}r}(x,y)) & \overset{=}{\rightarrow} & \mathsf{sys\text{-}r}(x,\mathsf{check}(y)) & (40) & \mathsf{sys\text{-}r}(x,\mathsf{ok}(y)) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{sys\text{-}r}(x,y))\\
(31) & \mathsf{check}(\mathsf{sys\text{-}w}(x,y)) & \overset{=}{\rightarrow} & \mathsf{sys\text{-}w}(\mathsf{check}(x),y) & (41) & \mathsf{sys\text{-}w}(\mathsf{ok}(x),y) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{sys\text{-}w}(x,y))\\
(32) & \mathsf{check}(\mathsf{sys\text{-}w}(x,y)) & \overset{=}{\rightarrow} & \mathsf{sys\text{-}w}(x,\mathsf{check}(y)) & (42) & \mathsf{sys\text{-}w}(x,\mathsf{ok}(y)) & \overset{=}{\rightarrow} & \mathsf{ok}(\mathsf{sys\text{-}w}(x,y))
\end{array}$$

The proof output produced by TPA consists of more than 1000 lines which prevents us from presenting it here in full detail. Instead we give a brief description of it. We present it exactly as generated by TPA without claim that this is in any sense the simplest termination proof of that system. The proof proceeds by repeating a number of times the following procedure:

1. Apply semantic labelling over booleans with a given interpretation.
2. Use counting argument a number of times to remove some rules in labelled system.
3. Unlabel to obtain TRS with few rules less.

The following table shows the details to be used in subsequent steps. Every row corresponds to applying this procedure once. The first column is just a sequential number to identify row in a table. The second column shows interpretation to be used in step 1. Every function symbol not mentioned there is assigned the default interpretation which is 0 (false) in case of a constant, identity function for unary symbol and disjunction for binary symbol. Unless stated otherwise all the function symbols get labelled by the value of their arguments. The third column lists symbols to be counted in step 2. Finally the last column lists the rules of the original TRS that can be removed after unlabelling (step 3).

No.	Labelling	Counted symbols	Removed rules
1	No labelling	WIO; RIO; WAO; RAO	(3), (5), (7)-(8), (13)-(15), (17)
2	[WAO] $= 1$ [read(x,y)] $= x \wedge y$	sys-r$_{11}$; WIN$_1$; sys-r$_{01}$; sys-r$_{10}$; RIN$_1$; check$_1$	(22)
3	[WIO] $= 1$ [read(x,y)] $= x \wedge y$	as above	(21)

4	$[\mathsf{RAO}] = 1$ $[\mathsf{read}(x,y)] = x \wedge y$	as above	(20)
5	$[\mathsf{RIO}] = 1$ $[\mathsf{read}(x,y)] = x \wedge y$	WIN_1; $\mathsf{sys\text{-}r}_{01}$; sys-r11; ok_0	(11)-(12)
6	$[\mathsf{RIO}] = 1$	WIN_1; $\mathsf{sys\text{-}r}_{01}$; $\mathsf{sys\text{-}r}_{11}$; ok_0; RIN_1; RAN_0	(4)
7	$[\mathsf{RIO}] = 1$ $[\mathsf{read}(x,y)] = 0$ **label only check symbol**	ok and check_1	(9)-(10)
8	$[\mathsf{RT}] = 1$ $[\mathsf{RAN}(x)] = 0$	write_{11}; write_{10}; WIO_1; RAO_1; RIO_1; WIN_1; top_1; write_{01}; $\mathsf{sys\text{-}r}_{11}$; $\mathsf{sys\text{-}w}_{11}$; $\mathsf{sys\text{-}r}_{01}$; $\mathsf{sys\text{-}w}_{01}$; read_{11}; RAN_1; read_{10}; read_{01}; RIN_1; $\mathsf{sys\text{-}w}_{10}$	(18)
9	No labelling	sys-r; check; WAN	(6), (16), (19)

All the interpretations yield a model except for the one used in step 8 which yields a quasi-model.

5 Concluding Remarks

This paper describes a technique to transform some liveness problems with fairness to the problems of proving relative termination of a transformed TRS. In a number of 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. Typically, finding the proof of relative transformation of the transformed TRS may be a hard job, and the computer generated proofs may be complicated, and of a shape that it is unlikely that they are found by a human.

Typically, the liveness problems that we consider involve infinite state spaces in two ways: they are not about a single set of initial states but involve infinitely many possible sets of initial states, and even for a single set of initial states, the set of reachable states is infinite. Due to these properties standard model checking techniques are not applicable for this kind of liveness problems.

References

1. P. J. Courtois, F. Heymans, and D. L. Parnas. Concurrent control with “readers” and “writers”. *Commun. ACM*, 14(10):667–668, 1971.
2. Nissim Francez. *Fairness*. Springer-Verlag New York, Inc., 1986.
3. Jürgen Giesl and Hans Zantema. Liveness in rewriting. In *Proc. 14th RTA, LNCS 2706*, pages 321–336, 2003.
4. Jürgen Giesl and Hans Zantema. Simulating liveness by reduction strategies. *Electr. Notes Theor. Comput. Sci.*, 86(4), 2003.
5. TeReSe. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.
6. Hans Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24(1/2):89–105, 1995.

A Termination proof of Example 6

```
  TRS-TORPA v.1.0

Result [9]: TRS is relatively terminating.

Default interpretations for symbols are not printed. For polynomial
interpretations and semantic labelling over N\{0,1} defaults are 2
for constants, identity for unary symbols and x+y-2 for binary symbols.
For semantic labelling over {0,1} (booleans) defaults are 0 for
constants, identity for unary symbols and disjunction for binary symbols.

[0]  TRS loaded from input file:
(1) top(ok(left(car(x,y),z))) -> top(check(right(y,z)))
(2) top(ok(right(x,car(y,z)))) -> top(check(left(x,z)))
(3) top(ok(left(bot,x))) -> top(check(right(bot,x)))
(4) top(ok(right(x,bot))) -> top(check(left(x,bot)))
(5) top(ok(left(car(x,y),z))) ->= top(check(left(y,z)))
(6) top(ok(right(x,car(y,z)))) ->= top(check(right(x,z)))
(7) bot ->= car(new,bot)
(8) check(old) ->= ok(old)
(9) check(car(x,y)) ->= car(check(x),y)
(10) check(car(x,y)) ->= car(x,check(y))
(11) check(left(x,y)) ->= left(check(x),y)
(12) check(left(x,y)) ->= left(x,check(y))
(13) check(right(x,y)) ->= right(check(x),y)
(14) check(right(x,y)) ->= right(x,check(y))
(15) car(ok(x),y) ->= ok(car(x,y))
(16) car(x,ok(y)) ->= ok(car(x,y))
(17) left(ok(x),y) ->= ok(left(x,y))
(18) left(x,ok(y)) ->= ok(left(x,y))
(19) right(ok(x),y) ->= ok(right(x,y))
(20) right(x,ok(y)) ->= ok(right(x,y))

[1]  Label this TRS using following interpretation that is a model:
[top(x)] = 0
[old] = 1
rest default
  thus obtaining new TRS:
(1a) top$1(ok$1(left$11(car$11(x,y),z))) -> top$1(check$1(right$11(y,z)))
(1b) top$1(ok$1(left$10(car$11(x,y),z))) -> top$1(check$1(right$10(y,z)))
(1c) top$1(ok$1(left$11(car$10(x,y),z))) -> top$1(check$1(right$01(y,z)))
(1d) top$1(ok$1(left$10(car$10(x,y),z))) -> top$0(check$0(right$00(y,z)))
(1e) top$1(ok$1(left$11(car$01(x,y),z))) -> top$1(check$1(right$11(y,z)))
(1f) top$1(ok$1(left$10(car$01(x,y),z))) -> top$1(check$1(right$10(y,z)))
(1g) top$1(ok$1(left$01(car$00(x,y),z))) -> top$1(check$1(right$01(y,z)))
(1h) top$0(ok$0(left$00(car$00(x,y),z))) -> top$0(check$0(right$00(y,z)))
(2a) top$1(ok$1(right$11(x,car$11(y,z)))) -> top$1(check$1(left$11(x,z)))
(2b) top$1(ok$1(right$11(x,car$10(y,z)))) -> top$1(check$1(left$10(x,z)))
(2c) top$1(ok$1(right$01(x,car$11(y,z)))) -> top$1(check$1(left$01(x,z)))
(2d) top$1(ok$1(right$01(x,car$10(y,z)))) -> top$0(check$0(left$00(x,z)))
(2e) top$1(ok$1(right$11(x,car$01(y,z)))) -> top$1(check$1(left$11(x,z)))
(2f) top$1(ok$1(right$10(x,car$00(y,z)))) -> top$1(check$1(left$10(x,z)))
(2g) top$1(ok$1(right$01(x,car$01(y,z)))) -> top$1(check$1(left$01(x,z)))
(2h) top$0(ok$0(right$00(x,car$00(y,z)))) -> top$0(check$0(left$00(x,z)))
(3a) top$1(ok$1(left$01(bot,x))) -> top$1(check$1(right$01(bot,x)))
(3b) top$0(ok$0(left$00(bot,x))) -> top$0(check$0(right$00(bot,x)))
(4a) top$1(ok$1(right$10(x,bot))) -> top$1(check$1(left$10(x,bot)))
(4b) top$0(ok$0(right$00(x,bot))) -> top$0(check$0(left$00(x,bot)))
(5a) top$1(ok$1(left$11(car$11(x,y),z))) ->= top$1(check$1(left$11(y,z)))
(5b) top$1(ok$1(left$10(car$11(x,y),z))) ->= top$1(check$1(left$10(y,z)))
(5c) top$1(ok$1(left$11(car$10(x,y),z))) ->= top$1(check$1(left$01(y,z)))
(5d) top$1(ok$1(left$10(car$10(x,y),z))) ->= top$0(check$0(left$00(y,z)))
(5e) top$1(ok$1(left$11(car$01(x,y),z))) ->= top$1(check$1(left$11(y,z)))
(5f) top$1(ok$1(left$10(car$01(x,y),z))) ->= top$1(check$1(left$10(y,z)))
(5g) top$1(ok$1(left$01(car$00(x,y),z))) ->= top$1(check$1(left$01(y,z)))
(5h) top$0(ok$0(left$00(car$00(x,y),z))) ->= top$0(check$0(left$00(y,z)))
(6a) top$1(ok$1(right$11(x,car$11(y,z)))) ->= top$1(check$1(right$11(x,z)))
(6b) top$1(ok$1(right$11(x,car$10(y,z)))) ->= top$1(check$1(right$10(x,z)))
(6c) top$1(ok$1(right$01(x,car$11(y,z)))) ->= top$1(check$1(right$01(x,z)))
(6d) top$1(ok$1(right$01(x,car$10(y,z)))) ->= top$0(check$0(right$00(x,z)))
(6e) top$1(ok$1(right$11(x,car$01(y,z)))) ->= top$1(check$1(right$11(x,z)))
(6f) top$1(ok$1(right$10(x,car$00(y,z)))) ->= top$1(check$1(right$10(x,z)))
(6g) top$1(ok$1(right$01(x,car$01(y,z)))) ->= top$1(check$1(right$01(x,z)))
(6h) top$0(ok$0(right$00(x,car$00(y,z)))) ->= top$0(check$0(right$00(x,z)))
(7a) bot ->= car$00(new,bot)
(8a) check$1(old) ->= ok$1(old)
(9a) check$1(car$11(x,y)) ->= car$11(check$1(x),y)
(9b) check$1(car$10(x,y)) ->= car$10(check$1(x),y)
(9c) check$1(car$01(x,y)) ->= car$01(check$0(x),y)
```

```
(9d) check$0(car$00(x,y)) ->= car$00(check$0(x),y)
(10a) check$1(car$11(x,y)) ->= car$11(x,check$1(y))
(10b) check$1(car$10(x,y)) ->= car$10(x,check$0(y))
(10c) check$1(car$01(x,y)) ->= car$01(x,check$1(y))
(10d) check$0(car$00(x,y)) ->= car$00(x,check$0(y))
(11a) check$1(left$11(x,y)) ->= left$11(check$1(x),y)
(11b) check$1(left$10(x,y)) ->= left$10(check$1(x),y)
(11c) check$1(left$01(x,y)) ->= left$01(check$0(x),y)
(11d) check$0(left$00(x,y)) ->= left$00(check$0(x),y)
(12a) check$1(left$11(x,y)) ->= left$11(x,check$1(y))
(12b) check$1(left$10(x,y)) ->= left$10(x,check$0(y))
(12c) check$1(left$01(x,y)) ->= left$01(x,check$1(y))
(12d) check$0(left$00(x,y)) ->= left$00(x,check$0(y))
(13a) check$1(right$11(x,y)) ->= right$11(check$1(x),y)
(13b) check$1(right$10(x,y)) ->= right$10(check$1(x),y)
(13c) check$1(right$01(x,y)) ->= right$01(check$0(x),y)
(13d) check$0(right$00(x,y)) ->= right$00(check$0(x),y)
(14a) check$1(right$11(x,y)) ->= right$11(x,check$1(y))
(14b) check$1(right$10(x,y)) ->= right$10(x,check$0(y))
(14c) check$1(right$01(x,y)) ->= right$01(x,check$1(y))
(14d) check$0(right$00(x,y)) ->= right$00(x,check$0(y))
(15a) car$11(ok$1(x),y) ->= ok$1(car$11(x,y))
(15b) car$10(ok$1(x),y) ->= ok$1(car$10(x,y))
(15c) car$01(ok$0(x),y) ->= ok$1(car$01(x,y))
(15d) car$00(ok$0(x),y) ->= ok$0(car$00(x,y))
(16a) car$11(x,ok$1(y)) ->= ok$1(car$11(x,y))
(16b) car$10(x,ok$0(y)) ->= ok$1(car$10(x,y))
(16c) car$01(x,ok$1(y)) ->= ok$1(car$01(x,y))
(16d) car$00(x,ok$0(y)) ->= ok$0(car$00(x,y))
(17a) left$11(ok$1(x),y) ->= ok$1(left$11(x,y))
(17b) left$10(ok$1(x),y) ->= ok$1(left$10(x,y))
(17c) left$01(ok$0(x),y) ->= ok$1(left$01(x,y))
(17d) left$00(ok$0(x),y) ->= ok$0(left$00(x,y))
(18a) left$11(x,ok$1(y)) ->= ok$1(left$11(x,y))
(18b) left$10(x,ok$0(y)) ->= ok$1(left$10(x,y))
(18c) left$01(x,ok$1(y)) ->= ok$1(left$01(x,y))
(18d) left$00(x,ok$0(y)) ->= ok$0(left$00(x,y))
(19a) right$11(ok$1(x),y) ->= ok$1(right$11(x,y))
(19b) right$10(ok$1(x),y) ->= ok$1(right$10(x,y))
(19c) right$01(ok$0(x),y) ->= ok$1(right$01(x,y))
(19d) right$00(ok$0(x),y) ->= ok$0(right$00(x,y))
(20a) right$11(x,ok$1(y)) ->= ok$1(right$11(x,y))
(20b) right$10(x,ok$0(y)) ->= ok$1(right$10(x,y))
(20c) right$01(x,ok$1(y)) ->= ok$1(right$01(x,y))
(20d) right$00(x,ok$0(y)) ->= ok$0(right$00(x,y))

[2]  Use following polynomial interpretation:
[ok$0(x)] = x + 1
rest default
  Remove rules with left hand side strictly bigger than right hand side:
(1h), (2h), (3b), (4b), (5h), (6h), (15c), (16b), (17c), (18b), (19c), (20b)

[3]  Use following polynomial interpretation:
[car$01(x,y)] = x + y + 1
rest default
  Remove rules with left hand side strictly bigger than right hand side:
(1e)-(1f), (2e), (2g), (5e)-(5f), (6e), (6g)

[4]  Use following polynomial interpretation:
[car$10(x,y)] = x + y + 1
rest default
  Remove rules with left hand side strictly bigger than right hand side:
(1c)-(1d), (2b), (2d), (5c)-(5d), (6b), (6d)

[5]  Use following polynomial interpretation:
[car$11(x,y)] = x + y + 1
rest default
  Remove rules with left hand side strictly bigger than right hand side:
(1a)-(1b), (2a), (2c), (5a)-(5b), (6a), (6c)

[6]  Use following polynomial interpretation:
[right$10(x,y)] = x + y + 1
rest default
  Remove rules with left hand side strictly bigger than right hand side:
(2f), (4a)

[7]  Unlabel this TRS to obtain the one consisting of the rules:
(1), (3), (5)-(20)

[8]  Use following polynomial interpretation:
[left(x,y)] = x + y + 1
rest default
  Remove rules with left hand side strictly bigger than right hand side:
(1), (3)

[9]  Since there are no remaining strict rules, relative termination is proved!
```