Liveness in Rewriting*

Jürgen Giesl[1] and Hans Zantema[2]

[1] LuFG Informatik II, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany,
giesl@informatik.rwth-aachen.de

[2] Department of Computer Science, TU Eindhoven, P.O. Box 513,
5600 MB Eindhoven, The Netherlands, h.zantema@tue.nl

Abstract. In this paper, we show how the problem of verifying liveness properties is related to termination of term rewrite systems (TRSs). We formalize liveness in the framework of rewriting and present a sound and complete transformation to transform particular liveness problems into TRSs. Then the transformed TRS terminates if and only if the original liveness property holds. This shows that liveness and termination are essentially equivalent. To apply our approach in practice, we introduce a simpler sound transformation which only satisfies the 'only if'-part. By refining existing techniques for proving termination of TRSs we show how liveness properties can be verified automatically. As examples, we prove a liveness property of a waiting line protocol for a network of processes and a liveness property of a protocol on a ring of processes.

1 Introduction

Usually, *liveness* is roughly defined as: "*something will eventually happen*" [1] and it is often remarked that "*termination is a particular case of liveness*". In this paper we present liveness in the general but precise setting of abstract reduction and TRSs and we study the relationship between liveness and termination. While classically, TRSs are applied to model evaluation in programming languages, we use TRSs to study liveness questions which are of high importance in practice (e.g., in protocol verification for distributed processes). In particular, we show how to verify liveness properties by existing termination techniques for TRSs.

In Sect. 2 we define a suitable notion of liveness to express eventuality properties using abstract reduction. Sect. 3 specializes this notion to the framework of term rewriting. In Sect. 4 we investigate the connection between a particular kind of liveness and termination, and present a sound and complete transformation which allows us to express liveness problems as termination problems of ordinary TRSs. Now techniques for proving termination of TRSs can also be used to infer liveness properties. To apply this approach in practice, based on our preceding results we present a sound (but incomplete) technique to perform termination proofs for liveness properties in Sect. 5, which is significantly easier to mechanize. In contrast to methods like model checking, our technique does not require finite state space. Our approach differs from other applications of term rewriting techniques to parameterized systems or infinite state spaces, where the emphasis is on verification of other properties like reachability [4]. We

* *Proceedings of the 14th International Conference on Rewriting Techniques and Applications (RTA-03)*, Valencia, Spain, LNCS, Springer-Verlag, 2003.

demonstrate our approach on two case studies of network protocols.

2 Liveness in Abstract Reduction

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

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

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

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

For example, *termination* (or *strong normalization* $\mathsf{SN}(I, \to)$) is a special liveness property describing the non-existence of infinite reductions, i.e.,

$$\mathsf{SN}(I, \to) \;=\; \neg(\exists t_1, t_2, t_3, \ldots : \; t_1 \in I \wedge \forall i : t_i \to t_{i+1}).$$

Theorem 2. *The property $\mathsf{SN}(I, \to)$ holds if and only if $\mathsf{Live}(I, \to, \mathsf{NF})$ holds.*

Proof. For the 'if'-part, if $\mathsf{SN}(I, \to)$ does not hold, then there is an infinite reduction $t_1 \to t_2 \to \cdots$ with $t_1 \in I$. Due to NF's definition, this infinite reduction does not contain elements of NF, contradicting Property 1 in Def. 1.

Conversely, if $\mathsf{SN}(I, \to)$ holds, then Property 1 in the definition of $\mathsf{Live}(I, \to, \mathsf{NF})$ holds trivially. Property 2 also holds, since $G = \mathsf{NF}$. □

Thm. 2 states that termination is a special case of liveness. The next theorem proves a kind of converse. For that purpose, we restrict the computation relation $\to$ such that it may only proceed if the current state is not in G.

Definition 3 ($\to_G$). *Let S, $\to$, G be as in Def. 1. Then $\to_G \subseteq S \times S$ is the relation where $t \to_G u$ holds if and only if $t \to u$ and $t \notin G$.*

Now we show that $\mathsf{Live}(I, \to, G)$ is equivalent to $\mathsf{SN}(I, \to_G)$. The 'only if'-part holds without any further conditions. However, for the 'if'-part we have to demand that G contains all normal forms $\mathsf{NF}(I)$ reachable from I, where $\mathsf{NF}(I) = \{u \in \mathsf{NF} \mid \exists t \in I : t \to^* u\}$. Otherwise, if there is a terminating sequence $t_1 \to \ldots \to t_n$ with all $t_i \notin G$, we might have $\mathsf{SN}(I, \to_G)$ but not $\mathsf{Live}(I, \to, G)$.

Theorem 4. *Let $\mathsf{NF}(I) \subseteq G$. Then $\mathsf{Live}(I, \to, G)$ holds iff $\mathsf{SN}(I, \to_G)$ holds.*

Proof. For the 'if'-part assume $\mathsf{SN}(I, \to_G)$. Property 2 of Def. 1 holds since

$\mathsf{NF}(I) \subseteq G$. If Property 1 does not hold then there is an infinite reduction without elements of G starting in I, contradicting $\mathsf{SN}(I, \to_G)$.

Conversely assume that $\mathsf{Live}(I, \to, G)$ holds and that $\mathsf{SN}(I, \to_G)$ does not hold. Then there is an infinite sequence $t_1, t_2, \ldots$ with $t_1 \in I \wedge \forall i : t_i \to_G t_{i+1}$. Hence, $t_i \notin G$ and $t_i \to t_{i+1}$ for all i, contradicting Property 1 in Def. 1. □

Thm. 4 allows us to verify actual liveness properties: if $\mathsf{NF}(I) \subseteq G$, then one can instead verify termination of $\to_G$. If $\mathsf{NF}(I) \not\subseteq G$, then $\mathsf{SN}(I, \to_G)$ still implies the liveness property for all infinite computations. In Sect. 4 and 5 we show how techniques to prove termination of TRSs can be used for termination of $\to_G$.

3 Liveness in Term Rewriting

Now we focus on liveness in rewriting, i.e., we study the property $\mathsf{Live}(I, \to_R, G)$ where $\to_R$ is the rewrite relation corresponding to a TRS R. For an introduction to term rewriting, the reader is referred to [3], for example.

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

By Thm. 4, $\mathsf{Live}(I, \to, G)$ is equivalent to $\mathsf{SN}(I, \to_G)$, if $\mathsf{NF}(I) \subseteq G$. To verify liveness, we want to prove $\mathsf{SN}(I, \to_G)$ by approaches for termination proofs of ordinary TRSs. However, depending on the form of G, different techniques are required. In the remainder we restrict ourselves to sets G of the following form:

$$G \;=\; \{t \mid t \text{ does not contain an instance of } p\} \qquad \text{for some term } p.$$

In other words, G contains all terms which cannot be written as $C[p\sigma]$ for any context C and substitution σ. As before, $t \to_G u$ holds iff $t \to_R u$ and $t \notin G$. So a term t may be reduced whenever it contains an instance of the term p.

A typical example of a liveness property is that eventually all processes requesting a resource are granted access to the resource (see Sect. 5.3). If a process waiting for the resource is represented by the unary function symbol old and if terms are used to denote the state of the whole network, then we would define $G = \{t \mid t \text{ does not contain an instance of } \mathsf{old}(x)\}$. Now $\mathsf{Live}(I, \to_R, G)$ means that eventually one reaches a term without the symbol old.

However, for arbitrary terms and TRSs, the notion $\to_G$ is not very useful: if there is a symbol f of arity > 1 or if p contains a variable x (i.e., if p can be written as $C[x]$ for some context C), then termination of $\to_G$ implies termination of the full rewrite relation $\to_R$. The reason is that any infinite reduction $t_1 \to_R t_2 \to_R \cdots$ gives rise to an infinite reduction $f(t_1, p, \ldots) \to_R f(t_2, p, \ldots) \to_R \cdots$ or $C[t_1] \to_R C[t_2] \to_R \cdots$ where in both cases none of the terms is in G. Therefore we concentrate on the particular case of *top rewrite systems* in which there is a designated symbol top. (These TRSs can be regarded as special forms of *typed* rewrite systems [11].)

Definition 5 (Top Rewrite System). *Let Σ be a signature and let $\mathsf{top} \notin \Sigma$ be a new unary function symbol. A term $t \in \mathcal{T}(\Sigma \cup \{\mathsf{top}\}, \mathcal{V})$ is a* top term *if*

its root is top *and* top *does not occur below the root. Let* $\mathcal{T}_{\mathsf{top}}$ *denote the set of all ground top terms. A TRS* R *over the signature* $\Sigma \cup \{\mathsf{top}\}$ *is a* top rewrite system *iff for all rules* $l \to r \in R$ *either*

- l *and* r *are top terms (in this case, we speak of a* top rule*) or*
- l *and* r *do not contain the symbol* top *(then we have a* non-top rule*)*

Top rewrite systems typically suffice to model networks of processes, since the whole network is represented by a top term [6]. Clearly, in top rewrite systems, top terms can only be reduced to top terms again. In such systems we consider liveness properties $\mathsf{Live}(\mathcal{T}_{\mathsf{top}}, \to_R, G)$. So we want to prove that every maximal reduction of ground top terms contains a term without an instance of p.

Example 6 (Simple liveness example). Consider the following two-rule TRS R.

$$\mathsf{top}(\mathsf{c}) \to \mathsf{top}(\mathsf{c}) \qquad\qquad \mathsf{f}(x) \to x$$

Clearly, R is not terminating and we even have infinite reductions within $\mathcal{T}_{\mathsf{top}}$:

$$\mathsf{top}(\mathsf{f}(\mathsf{f}(\mathsf{c}))) \to_R \mathsf{top}(\mathsf{f}(\mathsf{c})) \to_R \mathsf{top}(\mathsf{c}) \to_R \mathsf{top}(\mathsf{c}) \to_R \dots$$

However, in every reduction one eventually reaches a term without f. Hence, if $p = \mathsf{f}(x)$, then the liveness property is fulfilled for all ground top terms. Note that for $\Sigma = \{\mathsf{c}, \mathsf{f}\}$, we have $\mathsf{NF}(\mathcal{T}_{\mathsf{top}}) = \varnothing$ and thus, $\mathsf{NF}(\mathcal{T}_{\mathsf{top}}) \subseteq G$. Hence, by Thm. 4 it is sufficient to verify that $\to_G$ is terminating on $\mathcal{T}_{\mathsf{top}}$. Indeed, the above reduction is not possible with $\to_G$, since $\mathsf{top}(\mathsf{c})$ is a normal form w.r.t. $\to_G$.

4 Liveness and Termination

In this section we investigate the correspondence between liveness and termination in the framework of term rewriting. As in the previous section, we consider liveness properties $\mathsf{Live}(\mathcal{T}_{\mathsf{top}}, \to_R, G)$ for top rewrite systems R where G consists of those terms that do not contain instances of some subterm p. Provided that $\mathsf{NF}(\mathcal{T}_{\mathsf{top}}) \subseteq G$, by Thm. 4 the liveness property is equivalent to $\mathsf{SN}(\mathcal{T}_{\mathsf{top}}, \to_G)$.

Our aim is to prove termination of $\to_G$ on $\mathcal{T}_{\mathsf{top}}$ by means of termination of TRSs. In this way one can use all existing techniques for termination proofs of term rewrite systems (including future developments) in order to prove liveness properties. A first step into this direction was taken in [6], where the termination proof technique of *dependency pairs* was used to verify certain liveness properties of telecommunication processes. However, now our aim is to develop an approach to connect liveness and termination in general.

Given a TRS R and a term p, we define a TRS $L(R, p)$ such that $L(R, p)$ terminates (on all terms) if and only if $\mathsf{SN}(\mathcal{T}_{\mathsf{top}}, \to_G)$. A transformation where the 'only if'-direction holds is called *sound* and if the 'if'-direction holds, it is called *complete*. The existence of the sound and complete transformation $L(R, p)$ shows that for rewrite relations, liveness and termination are essentially equivalent.

The construction of $L(R, p)$ is motivated by an existing transformation [7, 8] which was developed for a completely different purpose (termination of context-sensitive rewriting). We introduce a number of new function symbols resulting in an extended signature Σ_G. Here, $\mathsf{proper}(t)$ checks whether t is a ground term over

the original signature Σ (Lemma 9) and $\mathsf{match}(p,t)$ checks in addition whether p matches t (Lemma 10). In this case, $\mathsf{proper}(t)$ and $\mathsf{match}(p,t)$ reduce to $\mathsf{ok}(t)$. To ease the formulation of the match-rules, we restrict ourselves to *linear* terms p, i.e., a variable occurs at most once in p. Moreover, for every variable x in p we introduce a fresh constant denoted by the corresponding upper-case letter X. We write $\overline{p}$ for the ground term obtained by replacing every variable in p by its corresponding fresh constant and in this way, it suffices to handle ground terms $\overline{p}$ in the match-rules. The new symbol check investigates whether its argument is a ground term over Σ which contains an instance of p (Lemma 11). In this case, $\mathsf{check}(t)$ reduces to $\mathsf{found}(t)$ and to find the instance of p, check may be propagated downwards through the term until one reaches the instance of p.

Finally, $\mathsf{active}(t)$ denotes that t may be reduced, since it contains an instance of p. Therefore, active may be propagated downwards to any desired redex of the term. After the reduction step, active is replaced by mark which is then propagated upwards to the top of the term. Now one checks whether the resulting term still contains an instance of p and none of the newly introduced function symbols. To this end, mark is replaced by check. If an instance of p is found, check is turned into found and found is propagated to the top of the term where it is replaced by active again. The TRS $L(R,p)$ has been designed in such a way that infinite reductions are only possible if this process is repeated infinitely often and Lemmata 12–14 investigate $L(R,p)$'s behavior formally.

Definition 7 ($L(R,p)$). *Let R be a top rewrite system over $\Sigma \cup \{\mathsf{top}\}$ with $\mathsf{top} \notin \Sigma$ and let $p \in \mathcal{T}(\Sigma,\mathcal{V})$ be linear. The TRS $L(R,p)$ over the signature $\Sigma_G = \Sigma \cup \{\mathsf{top},\mathsf{match},\mathsf{active},\mathsf{mark},\mathsf{check},\mathsf{proper},\mathsf{start},\mathsf{found},\mathsf{ok}\} \cup \{X \mid x \in \mathcal{V}(p)\}$ consists of the following rules for all non-top rules $l \to r \in R$, all top rules $\mathsf{top}(t) \to \mathsf{top}(u) \in R$, all $f \in \Sigma$ of arity $n > 0$ and $1 \le i \le n$, and all constants $c \in \Sigma_G$:*

$$
\begin{aligned}
\mathsf{active}(l) &\to \mathsf{mark}(r)\\
\mathsf{top}(\mathsf{active}(t)) &\to \mathsf{top}(\mathsf{mark}(u))\\
\mathsf{top}(\mathsf{mark}(x)) &\to \mathsf{top}(\mathsf{check}(x))\\
\mathsf{check}(f(x_1,..,x_n)) &\to f(\mathsf{proper}(x_1),..,\mathsf{check}(x_i),..,\mathsf{proper}(x_n))\\
\mathsf{check}(x) &\to \mathsf{start}(\mathsf{match}(\overline{p},x))\\
\mathsf{match}(f(x_1,..,x_n),f(y_1,..,y_n)) &\to f(\mathsf{match}(x_1,y_1),..,\mathsf{match}(x_n,y_n)),\ \textit{if } f \in \Sigma(p)\\
\mathsf{match}(c,c) &\to \mathsf{ok}(c), \qquad \textit{if } c \in \Sigma(p)\\
\mathsf{match}(c,x) &\to \mathsf{proper}(x), \qquad \textit{if } c \notin \Sigma \textit{ and } c \in \Sigma(\overline{p})\\
\mathsf{proper}(c) &\to \mathsf{ok}(c), \qquad \textit{if } c \in \Sigma\\
\mathsf{proper}(f(x_1,\ldots,x_n)) &\to f(\mathsf{proper}(x_1),\ldots,\mathsf{proper}(x_n))\\
f(\mathsf{ok}(x_1),\ldots,\mathsf{ok}(x_n)) &\to \mathsf{ok}(f(x_1,\ldots,x_n))\\
\mathsf{start}(\mathsf{ok}(x)) &\to \mathsf{found}(x)\\
f(\mathsf{ok}(x_1),..,\mathsf{found}(x_i),..,\mathsf{ok}(x_n)) &\to \mathsf{found}(f(x_1,..,x_n))\\
\mathsf{top}(\mathsf{found}(x)) &\to \mathsf{top}(\mathsf{active}(x))\\
\mathsf{active}(f(x_1,\ldots,x_i,\ldots,x_n)) &\to f(x_1,\ldots,\mathsf{active}(x_i),\ldots,x_n)\\
f(x_1,\ldots,\mathsf{mark}(x_i),\ldots,x_n) &\to \mathsf{mark}(f(x_1,\ldots,x_n))
\end{aligned}
\qquad
\begin{aligned}
&(1)\\
&\\
&\\
&\\
&\\
&\\
&\\
&\\
&\\
&\\
&\\
&\\
&\\
&(2)\\
&\\
&
\end{aligned}
$$

Example 8 (Transformation of simple liveness example). Recall the TRS from Ex. 6 again. Here, the transformation yields the following TRS $L(R,p)$.

$$
\begin{aligned}
\mathsf{active}(\mathsf{f}(x)) &\to \mathsf{mark}(x) & \mathsf{proper}(\mathsf{c}) &\to \mathsf{ok}(\mathsf{c})\\
\mathsf{top}(\mathsf{active}(\mathsf{c})) &\to \mathsf{top}(\mathsf{mark}(\mathsf{c})) & \mathsf{proper}(\mathsf{f}(x)) &\to \mathsf{f}(\mathsf{proper}(x))\\
\mathsf{top}(\mathsf{mark}(x)) &\to \mathsf{top}(\mathsf{check}(x)) & \mathsf{f}(\mathsf{ok}(x)) &\to \mathsf{ok}(\mathsf{f}(x))\\
\mathsf{check}(\mathsf{f}(x)) &\to \mathsf{f}(\mathsf{check}(x)) & \mathsf{start}(\mathsf{ok}(x)) &\to \mathsf{found}(x)\\
\mathsf{check}(x) &\to \mathsf{start}(\mathsf{match}(\mathsf{f}(X), x)) & \mathsf{f}(\mathsf{found}(x)) &\to \mathsf{found}(\mathsf{f}(x))\\
\mathsf{match}(\mathsf{f}(x), \mathsf{f}(y)) &\to \mathsf{f}(\mathsf{match}(x, y)) & \mathsf{top}(\mathsf{found}(x)) &\to \mathsf{top}(\mathsf{active}(x))\\
\mathsf{match}(X, x) &\to \mathsf{proper}(x) & \mathsf{active}(\mathsf{f}(x)) &\to \mathsf{f}(\mathsf{active}(x))\\
& & \mathsf{f}(\mathsf{mark}(x)) &\to \mathsf{mark}(\mathsf{f}(x))
\end{aligned}
$$

Note that it is really necessary to introduce the symbol proper and to check whether the whole term does not contain any new symbols from $\Sigma_G \setminus \Sigma$. If the proper-rules were removed, all remaining proper-terms were replaced by their arguments, and in $f(\mathsf{ok}(x_1), \ldots, \mathsf{found}(x_i), \ldots, \mathsf{ok}(x_n)) \to \mathsf{found}(f(x_1, \ldots, x_n))$, the terms $\mathsf{ok}(x_i)$ were replaced by x_i, then the transformation would not be complete any more. As a counterexample, regard $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{f}\}$ and the TRS

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

and let $p = \mathsf{a}$. The TRS satisfies the liveness property since for any ground top term, after at most two steps one reaches a term without a (one obtains either $\mathsf{top}(\mathsf{b})$ or $\mathsf{top}(\mathsf{f}(\mathsf{b}, \mathsf{b}, \mathsf{b}))$). However, with the modified transformation we would get the following non-terminating cyclic reduction where u is the term $\mathsf{found}(\mathsf{b})$:

$$
\begin{aligned}
&\mathsf{top}(\mathsf{mark}(\mathsf{f}(u, u, u))) &&\to\ \mathsf{top}(\mathsf{check}(\mathsf{f}(u, u, u))) &&\to\\
&\mathsf{top}(\mathsf{f}(u, \mathsf{check}(u), u)) &&\to\ \mathsf{top}(\mathsf{found}(\mathsf{f}(\mathsf{b}, \mathsf{check}(u), u))) &&\to\\
&\mathsf{top}(\mathsf{active}(\mathsf{f}(\mathsf{b}, \mathsf{check}(u), u))) &&\to\ \mathsf{top}(\mathsf{mark}(\mathsf{f}(u, u, u))) &&\to\ \ldots
\end{aligned}
$$

To prove soundness and completeness of our transformation, we need several auxiliary lemmata about reductions with $L(R, p)$. The first lemma states that proper really checks whether its argument does not contain symbols from $\Sigma_G \setminus \Sigma$.

Lemma 9 (Reducing proper). *For $t \in \mathcal{T}(\Sigma_G)$ we have $\mathsf{proper}(t) \to^+_{L(R,p)} \mathsf{ok}(u)$ if and only if $t, u \in \mathcal{T}(\Sigma)$ and $t = u$.*

Proof. The proof is identical to the one in [7, Lemma 2] and [8]. □

Now we show that $\mathsf{match}(\overline{p}, t)$ checks whether p matches t and $t \in \mathcal{T}(\Sigma)$.

Lemma 10 (Reducing match). *Let $p \in \mathcal{T}(\Sigma, \mathcal{V})$, let $q \in \mathcal{T}(\Sigma(p), \mathcal{V})$ be linear, and let $t \in \mathcal{T}(\Sigma_G)$. We have $\mathsf{match}(\overline{q}, t) \to^+_{L(R,p)} \mathsf{ok}(u)$ iff $t = u \in \mathcal{T}(\Sigma)$ and $q\sigma = t$ for some σ.*

Proof. The 'if'-direction is an easy induction on the structure of the term t, see [9]. The 'only if'-direction is proved by induction on the length of the reduction. If the first reduction step is in t, then $\mathsf{match}(\overline{q}, t) \to_{L(R,p)} \mathsf{match}(\overline{q}, t') \to^+_{L(R,p)} \mathsf{ok}(u)$ for a term t' with $t \to_{L(R,p)} t'$. The induction hypothesis states $t' = u \in \mathcal{T}(\Sigma)$ and $q\sigma = t'$. Note that $t' \in \mathcal{T}(\Sigma)$ implies $t = t'$ which proves the lemma.

Otherwise, the first reduction step is on the root position (since $\overline{q}$ is in normal form). If q is a variable, then q obviously matches t and we obtain $\mathsf{match}(\overline{q}, t) \to_{L(R,p)} \mathsf{proper}(t) \to^+_{L(R,p)} \mathsf{ok}(u)$ and $t = u \in \mathcal{T}(\Sigma)$ by Lemma 9. If q is a constant c, then a root reduction is only possible if $t = c$. We obtain $\mathsf{match}(\overline{q}, t) = \mathsf{match}(\overline{q}, c) \to_{L(R,p)} \mathsf{ok}(c)$. So in this case the lemma also holds.

Finally, if $q = f(q_1, \ldots, q_n)$, for a root reduction we have $t = f(t_1, \ldots, t_n)$. Then $\mathsf{match}(\overline{q}, t) = \mathsf{match}(f(\overline{q_1}, \ldots, \overline{q_n}), f(t_1, \ldots, t_n)) = f(\mathsf{match}(\overline{q_1}, t_1), \ldots, \mathsf{match}(\overline{q_n}, t_n)) \to^+_{L(R,p)} \mathsf{ok}(u)$. To reduce $f(\ldots)$ to $\mathsf{ok}(\ldots)$, all arguments of f must reduce to ok-terms. Hence, $\mathsf{match}(\overline{q_i}, t_i) \to^+_{L(R,p)} \mathsf{ok}(u_i)$ for all i where these reductions are shorter than the reduction $\mathsf{match}(\overline{q}, t) \to^+_{L(R,p)} \mathsf{ok}(u)$. The induction hypothesis implies $t_i = u_i \in \mathcal{T}(\Sigma)$ and that there are substitutions σ_i with $q_i\sigma_i = t_i$. Since q is linear, we can combine these σ_i to one σ such that $q\sigma = t$. Moreover, this implies $u = f(u_1, \ldots, u_n)$ which proves the lemma. □

Based on the previous two lemmata, one can show that check works properly, i.e., it checks whether its argument is a term from $\mathcal{T}(\Sigma)$ containing an instance of p. The proof is similar to the one of Lemma 10 and can be found in [9].

Lemma 11 (Reducing check). *Let $p \in \mathcal{T}(\Sigma, \mathcal{V})$ be linear and $t \in \mathcal{T}(\Sigma_G)$. We have $\mathsf{check}(t) \to^+_{L(R,p)} \mathsf{found}(u)$ iff $t = u \in \mathcal{T}(\Sigma)$ and t contains a subterm $p\sigma$.*

Lemma 12 shows that the top-rules (1), (2) are applied in an alternating way.

Lemma 12 (Reducing active and check). *For all $t, u \in \mathcal{T}(\Sigma_G)$ we have*

(a) $\mathsf{active}(t) \not\to^+_{L(R,p)} \mathsf{found}(u)$ *and* $\mathsf{active}(t) \not\to^+_{L(R,p)} \mathsf{ok}(u)$
(b) $\mathsf{check}(t) \not\to^+_{L(R,p)} \mathsf{mark}(u)$ *and* $\mathsf{proper}(t) \not\to^+_{L(R,p)} \mathsf{mark}(u)$

Proof. For (a), by induction on $n \in \mathbb{N}$, we show that there is no reduction from $\mathsf{active}(t)$ to $\mathsf{found}(u)$ or to $\mathsf{ok}(u)$ of length n. If the first reduction step is in t, then the claim follows from the induction hypothesis. Otherwise, the reduction starts with a root step. This first step cannot be $\mathsf{active}(t) \to_{L(R,p)} \mathsf{mark}(u)$, since the root symbol mark can never be reduced again. Hence, we must have $t = f(t_1, \ldots, t_i, \ldots, t_n)$ and $\mathsf{active}(t) = \mathsf{active}(f(t_1, \ldots, t_i, \ldots, t_n)) \to_{L(R,p)} f(t_1, \ldots, \mathsf{active}(t_i), \ldots, t_n)$. In order to rewrite this term to a found- or ok-term, in particular $\mathsf{active}(t_i)$ must be rewritten to a found- or ok-term which contradicts the induction hypothesis. For the (similar) proof of (b), we refer to [9]. □

We now prove that the top-rules are crucial for $L(R, p)$'s termination behavior.

Lemma 13. *Let $L'(R, p) = L(R, p) \setminus \{(1), (2)\}$. Then $L'(R, p)$ is terminating.*

Proof. Termination of $L'(R, p)$ can be proved by the recursive path order [5] using the precedence $\mathsf{active} > \mathsf{check} > \mathsf{match} > \mathsf{proper} > \mathsf{start} > f > \mathsf{ok} > \mathsf{found} > \mathsf{mark}$ for all $f \in \Sigma \cup \{X \mid x \in \mathcal{V}(p)\}$. □

Before relating $L(R, p)$ and $\to_G$, we study the connection of $L(R, p)$ and $\to_R$.

Lemma 14. *Let $t, u \in \mathcal{T}(\Sigma)$. Then we have $\mathsf{active}(t) \to^+_{L(R,p)} \mathsf{mark}(u)$ iff $t \to_R u$ and $\mathsf{top}(\mathsf{active}(t)) \to^+_{L(R,p)} \mathsf{top}(\mathsf{mark}(u))$ iff $\mathsf{top}(t) \to_R \mathsf{top}(u)$.*

Proof. The 'if'-direction is easy by induction on t. For the 'only if'-direction, we prove that $\mathsf{active}(t) \to^+_{L(R,p)} \mathsf{mark}(u)$ implies $t \to_R u$ by induction on the length of the reduction. The proof that $\mathsf{top}(\mathsf{active}(t)) \to^+_{L(R,p)} \mathsf{top}(\mathsf{mark}(u))$ implies $\mathsf{top}(t) \to_R \mathsf{top}(u)$ is analogous [9]. Since $t \in \mathcal{T}(\Sigma)$, the first reduction step must be on the root position. If $\mathsf{active}(t) \to_{L(R,p)} \mathsf{mark}(u)$ on root position, then $t = l\sigma$ and $u = r\sigma$ for a rule $l \to r \in R$ and thus, $t \to_R u$. Otherwise, $t = f(t_1, \ldots, t_n)$ and $\mathsf{active}(t) = \mathsf{active}(f(t_1, \ldots, t_n)) \to_{L(R,p)} f(t_1, \ldots, \mathsf{active}(t_i), \ldots, t_n) \to^+_{L(R,p)} \mathsf{mark}(u)$. Thus, $\mathsf{active}(t_i) \to^+_{L(R,p)} \mathsf{mark}(u_i)$ and $u = f(t_1, \ldots, u_i, \ldots, t_n)$. The induction hypothesis implies $t_i \to_R u_i$ and hence, $t \to_R u$. □

Theorem 15 (Soundness and Completeness). *Let R be a top rewrite system over $\Sigma \cup \{\mathsf{top}\}$ with $\mathsf{top} \notin \Sigma$ and let $p \in \mathcal{T}(\Sigma, \mathcal{V})$ be linear. The TRS $L(R,p)$ is terminating (on all terms) iff the relation $\to_G$ is terminating on $\mathcal{T}_{\mathsf{top}}$.*

Proof. We first show the 'only if'-direction. If $\to_G$ does not terminate on $\mathcal{T}_{\mathsf{top}}$ then there is an infinite reduction $\mathsf{top}(t_1) \to_G \mathsf{top}(t_2) \to_G \ldots$ where $t_1, t_2, \ldots \in \mathcal{T}(\Sigma)$. By Lemma 14 we have $\mathsf{top}(\mathsf{active}(t_i)) \to^+_{L(R,p)} \mathsf{top}(\mathsf{mark}(t_{i+1}))$. Lemma 11 implies $\mathsf{check}(t_{i+1}) \to^+_{L(R,p)} \mathsf{found}(t_{i+1})$, since each t_{i+1} contains an instance of p. So we obtain the following contradiction to the termination of $L(R,p)$.

$$\begin{aligned}\mathsf{top}(\mathsf{active}(t_1)) \to^+_{L(R,p)} \mathsf{top}(\mathsf{mark}(t_2)) \to_{L(R,p)} \mathsf{top}(\mathsf{check}(t_2)) \to^+_{L(R,p)} \\ \mathsf{top}(\mathsf{found}(t_2)) \to_{L(R,p)} \mathsf{top}(\mathsf{active}(t_2)) \to^+_{L(R,p)} \ldots\end{aligned}$$

For the 'if'-direction assume that $L(R,p)$ is not terminating. By type introduction [11] one can show that there exists an infinite $L(R,p)$-reduction of ground top terms. Due to Lemma 13 the reduction contains infinitely many applications of the rules (1) and (2). These rules must be applied in alternating order, since $\mathsf{active}(t)$ can never reduce to $\mathsf{found}(u)$ and $\mathsf{check}(t)$ can never reduce to $\mathsf{mark}(u)$ by Lemma 12. So the reduction has the following form where all reductions with the rules (1) and (2) are displayed.

$$\begin{aligned}\ldots \to^*_{L(R,p)} \mathsf{top}(\mathsf{mark}(t_1)) \to_{L(R,p)} \mathsf{top}(\mathsf{check}(t_1)) \to^+_{L(R,p)} \\ \mathsf{top}(\mathsf{found}(u_1)) \to_{L(R,p)} \mathsf{top}(\mathsf{active}(u_1)) \to^+_{L(R,p)} \\ \mathsf{top}(\mathsf{mark}(t_2)) \to_{L(R,p)} \mathsf{top}(\mathsf{check}(t_2)) \to^+_{L(R,p)} \\ \mathsf{top}(\mathsf{found}(u_2)) \to_{L(R,p)} \mathsf{top}(\mathsf{active}(u_2)) \to^+_{L(R,p)} \ldots\end{aligned}$$

By Lemma 11 we have $t_i = u_i \in \mathcal{T}(\Sigma)$ and that t_i contains an instance of p. Lemma 14 implies $\mathsf{top}(u_i) \to_R \mathsf{top}(t_{i+1})$. Together, we obtain $\mathsf{top}(t_1) \to_G \mathsf{top}(t_2) \to_G \ldots$ in contradiction to the termination of $\to_G$ on $\mathcal{T}_{\mathsf{top}}$. □

By Thm. 15, one can now use existing techniques for termination proofs of TRSs to verify liveness of systems like Ex. 6. For instance, termination of the transformed TRS from Ex. 8 is easy to show with dependency pairs [2], cf. [9].

5 Proving Liveness

In Sect. 5.1 we present a sound transformation which is more suitable for mechanizing liveness proofs than the complete transformation from Sect. 4. The reason

is that for this new transformation, termination of the transformed TRS is much easier to show. On the other hand, the approach in this section is incomplete, i.e., it cannot succeed for all examples. Subsequently, in Sect. 5.2 we introduce an automatic preprocessing technique based on *semantic labelling* [12] to simplify these termination proofs further. In this way, rewriting techniques can be used to mechanize the verification of liveness properties. To illustrate the use of our approach, in Sect. 5.3 we show how to verify liveness properties of a network of processes with a shared resource and of a token ring protocol.

5.1 A Sound Transformation for Liveness

To obtain a simple sound transformation, the idea is to introduce only one new symbol check. A new occurrence of check is created in every application of a top rule. If check finds an instantiation of p then check may be removed. Otherwise, check remains in the term where it may block further reductions.

Definition 16 ($LS(R,p)$). *For a top rewrite system R over $\Sigma \cup \{\mathsf{top}\}$ with $\mathsf{top} \notin \Sigma$ and $p \in \mathcal{T}(\Sigma,\mathcal{V})$, let $LS(R,p)$ consist of the following rules.*

$$\begin{aligned} l &\to r && \text{for all non-top rules } l \to r \text{ in } R\\ \mathsf{top}(t) &\to \mathsf{top}(\mathsf{check}(u)) && \text{for all top rules } \mathsf{top}(t) \to \mathsf{top}(u)\\ \mathsf{check}(f(x_1,..,x_n)) &\to f(x_1,..,\mathsf{check}(x_i),..,x_n) && \text{for } f\in\Sigma \text{ of arity } n\geq 1,\ i=1,..,n\\ \mathsf{check}(p) &\to p \end{aligned}$$

Example 17 (Simple example revisited). To illustrate the transformation, reconsider the system from Ex. 6. Here, $LS(R,\mathsf{f}(x))$ is the following TRS whose termination can be proved by dependency pairs and the recursive path order.

$$\begin{aligned} \mathsf{top}(\mathsf{c}) &\to \mathsf{top}(\mathsf{check}(\mathsf{c})) && (3) & \qquad \mathsf{check}(\mathsf{f}(x)) &\to \mathsf{f}(\mathsf{check}(x)) && (5)\\ \mathsf{f}(x) &\to x && (4) & \qquad \mathsf{check}(\mathsf{f}(x)) &\to \mathsf{f}(x) && (6) \end{aligned}$$

Now we show that this transformation is indeed sound. In other words, the above termination proof verifies the liveness property of our example.

Theorem 18 (Soundness). *Let R be a top rewrite system over $\Sigma\cup\{\mathsf{top}\}$ with $\mathsf{top}\notin\Sigma$, let $p\in\mathcal{T}(\Sigma,\mathcal{V})$, and let $G=\{t \mid t \text{ does not contain an instance of } p\}$. If $LS(R,p)$ is terminating then there is no infinite $\to_G$-reduction of top terms.*

Proof. Assume there is an infinite $\to_G$-reduction of top terms $\mathsf{top}(t_1)\to_G \mathsf{top}(t_2) \to_G \ldots$ Since top does not occur in p, every t_i has the form $C_i[p\sigma_i]$ for some context C_i and substitution σ_i. To prove the theorem, we show that $\mathsf{top}(t_i)\to^+_{LS(R,p)} \mathsf{top}(t_{i+1})$ for every i, by which we obtain an infinite $LS(R,p)$-reduction.

If $\mathsf{top}(t_i)\to_R \mathsf{top}(t_{i+1})$ by the application of a non-top rule $l\to r$ then we also have $\mathsf{top}(t_i)\to_{LS(R,p)} \mathsf{top}(t_{i+1})$ since $l\to r$ is also contained in $LS(R,p)$. Otherwise, $\mathsf{top}(t_i)\to_R \mathsf{top}(t_{i+1})$ by a top rule $\mathsf{top}(t)\to\mathsf{top}(u)$. Hence, $t_i = t\sigma$ and $t_{i+1}=u\sigma$ for some σ. Since $LS(R,p)$ contains the rules $\mathsf{check}(f(x_1,\ldots,x_n)) \to f(x_1,\ldots,\mathsf{check}(x_i),\ldots,x_n)$ for all f with arity ≥ 1, we obtain

$$\begin{aligned}\mathsf{top}(t_i) = \mathsf{top}(t\sigma) &\to_{LS(R,p)} \mathsf{top}(\mathsf{check}(u\sigma)) &&= \mathsf{top}(\mathsf{check}(C_{i+1}[p\sigma_{i+1}]))\\ &\to^*_{LS(R,p)} \mathsf{top}(C_{i+1}[\mathsf{check}(p)\sigma_{i+1}])\\ &\to_{LS(R,p)} \mathsf{top}(C_{i+1}[p\sigma_{i+1}]) &&= \mathsf{top}(t_{i+1}) \qquad \square\end{aligned}$$

Example 19 (Sound transformation is not complete). However, this transformation is incomplete as can be shown by the following top rewrite system R

$$\mathsf{top}(\mathsf{f}(x,\mathsf{b})) \to \mathsf{top}(\mathsf{f}(\mathsf{b},\mathsf{b})) \qquad\qquad \mathsf{a} \to \mathsf{b}$$

where $\Sigma = \{\mathsf{a},\mathsf{b},\mathsf{f}\}$ and $p = \mathsf{a}$. In this example, normal forms do not contain a any more and every infinite reduction of top terms reaches the term $\mathsf{top}(\mathsf{f}(\mathsf{b},\mathsf{b}))$ which does not contain the symbol a either. Hence, the liveness property holds. However, $LS(R,p)$ admits the following infinite reduction:

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

Thus, the transformation of Def. 16 is incomplete, because even if check remains in a term, this does not necessarily block further (infinite) reductions.

5.2 A Preprocessing Procedure for Verifying Liveness

The aim of our sound transformation from Def. 16 is to simplify (and possibly automate) the termination proofs which are required in order to show liveness properties. Since the TRSs resulting from our transformation have a particular form, we now present a method to preprocess such TRSs. This preprocessing is especially designed for this form of TRSs and in this way, their termination proofs can often be simplified significantly. The method consists of four steps which can be performed automatically:

(a) First one deletes rules which cannot cause non-termination.
(b) Then one applies the well-known transformation technique of *semantic labelling* [12] with a particularly chosen model and labelling. (This restricted form of semantic labelling can be done automatically.)
(c) Then one again deletes rules which cannot cause non-termination.
(d) Finally one uses an existing automatic technique (e.g., the recursive path order or dependency pairs) to prove termination of the resulting TRS.

To delete rules in Step (a) and (c) we use the following lemma. For a function symbol $f \in \Sigma$ and a term $t \in \mathcal{T}(\Sigma,\mathcal{V})$, let $\#_f(t)$ be the number of f-symbols occurring in t. For $\varnothing \neq \Sigma' \subseteq \Sigma$ let $\#_{\Sigma'}(t) = \sum_{f\in\Sigma'} \#_f(t)$.

Lemma 20. *Let R be a TRS such that*

- *R is* non-duplicating*, i.e., for every rule $l \to r$, no variable occurs more often in r than in l, and*
- *$\#_{\Sigma'}(l) \geq \#_{\Sigma'}(r)$ for all rules $l \to r$ in R*

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

Proof. The 'only if'-part holds since $R \setminus R' \subseteq R$. For the 'if'-part assume that $R \setminus R'$ is terminating and that we have an infinite R-reduction. Due to the conditions of the lemma we have $\#_{\Sigma'}(t) \geq \#_{\Sigma'}(u)$ for every step $t \to_R u$ and

$\#_{\Sigma'}(t) > \#_{\Sigma'}(u)$ for every step $t \to_{R'} u$. Hence, due to well-foundedness of the natural numbers, the infinite R-reduction contains only finitely many R'-steps. After removing the finite initial part containing all these R'-steps, the remaining part is an infinite $R \setminus R'$-reduction, which gives a contradiction. □

The application of Lemma 20 is easily automated as follows: for all sets $\Sigma' \subseteq \Sigma$ with $|\Sigma'| \leq n$ for some (small) $n \in \mathbb{N}$, it is checked whether $\#_{\Sigma'}(l) \geq \#_{\Sigma'}(r)$ for all rules $l \to r$. If so, then all rules $l \to r$ satisfying $\#_{\Sigma'}(l) > \#_{\Sigma'}(r)$ are removed. This process is repeated until no rule can be removed any more.

As a first example, we apply Lemma 20 to the TRS from Ex. 17. By counting the occurrences of f, we note that the number of f-symbols strictly decreases in Rule (4) and it remains the same in all other rules. Hence, due to Lemma 20 we can drop this rule when proving termination of the TRS. It turns out that in this case repetition of this process does not succeed in removing more rules.

In our termination procedure, in Step (b) we apply a particular instance of *semantic labelling* [12]. Before describing this instance we briefly explain how semantic labelling works as a tool to prove termination of a TRS R over the signature Σ: One starts by choosing a *model* for the TRS R. Thus, one defines a non-empty carrier set M and for every function symbol $f \in \Sigma$ of arity n, an interpretation $f_M : M^n \to M$ is chosen. As usual, every variable assignment $\alpha : \mathcal{V} \to M$ can be extended to terms from $\mathcal{T}(\Sigma, \mathcal{V})$ by inductively defining $\alpha(f(t_1, \ldots, t_n)) = f_M(\alpha(t_1), \ldots, \alpha(t_n))$. The interpretation is a *model* for R if $\alpha(l) = \alpha(r)$ for every rule $l \to r$ in R and every variable assignment $\alpha : \mathcal{V} \to M$.

Using this model, the TRS R over the signature Σ is transformed into a *labelled* TRS $\overline{R}$ over the *labelled* signature $\overline{\Sigma}$. Here, every function symbol $f \in \Sigma$ of arity n may be labelled by n elements from M, i.e., $\overline{\Sigma} = \{f_{a_1,\ldots,a_n} \mid f \in \Sigma, n = \text{arity}(f), a_i \in M\}$ where the arity of $f_{a_1,\ldots,a_n}$ is the same as the arity of f. For any variable assignment $\alpha : \mathcal{V} \to M$, we define a function $\mathsf{lab}_\alpha : \mathcal{T}(\Sigma, \mathcal{V}) \to \mathcal{T}(\overline{\Sigma}, \mathcal{V})$ which labels every function symbol by the interpretations of its arguments:

$$\begin{aligned} \mathsf{lab}_\alpha(x) &= x, \text{ for } x \in \mathcal{V} \\ \mathsf{lab}_\alpha(f(t_1, \ldots, t_n)) &= f_{\alpha(t_1),\ldots,\alpha(t_n)}(\mathsf{lab}_\alpha(t_1), \ldots, \mathsf{lab}_\alpha(t_n)) \end{aligned}$$

Now the TRS $\overline{R}$ is defined to consist of all rules $\mathsf{lab}_\alpha(l) \to \mathsf{lab}_\alpha(r)$ for all variable assignments $\alpha : \mathcal{V} \to M$ and all rules $l \to r$ in R. The main theorem of semantic labelling states that R is terminating if and only if $\overline{R}$ is terminating.

In general, semantic labelling permits a lot of freedom and is hard to automate, since one may choose arbitrary models. Moreover, in full semantic labelling one may also use arbitrary labellings. However, we will restrict ourselves to the case where $M = \{0, 1\}$. Now there are only finitely many possibilities for the interpretations f_M in the model. This means that with this restriction the termination method consisting of the steps (a) - (d) is fully decidable.

To improve efficiency and to avoid checking all possibilities of a two-element model for semantic labelling, we now propose heuristics for choosing the interpretations f_M in such a model. These heuristics are adapted to the special form of TRSs resulting from our transformation in Def. 16 when verifying liveness properties. The main objective is that we want to distinguish between terms

that contain instances of p and terms that do not. Therefore, our aim is to interpret the former terms by 0 and the latter terms by 1. Since the intention of check is that an occurrence of p should be found, $\mathsf{check}(x)$ will be interpreted as the constant function 0. Since top only occurs at the top, for $\mathsf{top}(x)$ we may also choose a constant function. Having these objectives in mind, we arrive at the following heuristic for choosing the operations f_M in the model $M = \{0, 1\}$:

- $\mathsf{top}_M(x) = \mathsf{check}_M(x) = f_M(x_1, \ldots, x_n) = 0$ for $x = 0, 1$, where f is the root symbol of p;
- $c_M = 1$ for every constant c, except if $p = c$;
- $f_M(x_1, \ldots, x_n) = \min(x_1, \ldots, x_n)$ for all other symbols f as long as this does not conflict with the model requirement $\alpha(l) = \alpha(r)$. In particular, for the remaining unary symbols f one tries to choose $f_M(x) = x$.

Applying these heuristics to our example results in the following interpretation:

$$\mathsf{top}_M(x) = \mathsf{check}_M(x) = \mathsf{f}_M(x) = 0 \quad \text{for } x \in M = \{0, 1\} \quad \text{and} \quad \mathsf{c}_M = 1$$

One checks that this is a model for the TRS. Here it is essential that we first removed Rule (4), since $\mathsf{f}_M(x) = 0 \neq x$ if $x = 1$. The labelling results in the TRS

$$\begin{array}{rll} \mathsf{top}_1(\mathsf{c}) & \to \mathsf{top}_0(\mathsf{check}_1(\mathsf{c})) & \\ \mathsf{check}_0(\mathsf{f}_i(x)) & \to \mathsf{f}_0(\mathsf{check}_i(x)) & \text{for } i \in \{0, 1\} \\ \mathsf{check}_0(\mathsf{f}_i(x)) & \to \mathsf{f}_i(x) & \text{for } i \in \{0, 1\} \end{array}$$

In Step (c) of our termination procedure, we apply Lemma 20 again. By counting the occurrences of top_1, we can drop the first rule. By counting f_1, the second rule can be removed if i is 1, and by counting check_0 we can delete the third rule. So the remaining TRS just contains the rule $\mathsf{check}_0(\mathsf{f}_0(x)) \to \mathsf{f}_0(\mathsf{check}_0(x))$ whose termination is trivial to prove by the recursive path order.

This example indicates that preprocessing a TRS according to Steps (a) - (c) often simplifies the termination proof considerably. For the original TRS of Ex. 17, one needs dependency pairs for the termination proof, whereas after the transformation a very simple recursive path order is sufficient.

5.3 Two Case Studies of Liveness

To demonstrate the applicability of our approach, we regard two case studies. The first one is motivated by verification problems of protocols similar to the *bakery protocol* [10]. We describe a network of processes which want to gain access to a shared resource. The processes waiting for the resource are served one after another. Since the maximal size of the waiting line is fixed, a new process can only enter the waiting line if a process in the current line has been "served" (i.e., if it has been granted access to the resource). The maximal length n of the waiting line is arbitrary, and we will show that the liveness property holds for all $n \in \mathbb{N}$. Hence, techniques like classical model checking are not applicable here.

The processes in the line are served on a "first in - first out" basis (this corresponds to the serving of clients in a shop). So at the front end of the waiting line, a process may be served, where serving is denoted by a constant serve. If a process is served, its place in the line is replaced by a free place, denoted by free.

If the place in front of some process is free, this process may take the free place, creating a free place on its original position. If the line has a free place at its back end, a new process new may enter the waiting line, taking over the position of the free place. Apart from new processes represented by new we also consider old processes represented by old, which were already in the line initially. We want to verify the liveness property that eventually all old processes will be served. To model protocols with TRSs, we represent the state of the whole network by a top term. Introducing the symbol top at the back end of the waiting line, this network is described by the following top rewrite system R:

$$\begin{array}{rlrl} \mathsf{top}(\mathsf{free}(x)) &\to \mathsf{top}(\mathsf{new}(x)) & \mathsf{new}(\mathsf{serve}) &\to \mathsf{free}(\mathsf{serve}) \\ \mathsf{new}(\mathsf{free}(x)) &\to \mathsf{free}(\mathsf{new}(x)) & \mathsf{old}(\mathsf{serve}) &\to \mathsf{free}(\mathsf{serve}) \\ \mathsf{old}(\mathsf{free}(x)) &\to \mathsf{free}(\mathsf{old}(x)) & & \end{array}$$

Note that the above TRS admits infinite reductions of top terms. For instance,

$$\mathsf{top}(\mathsf{new}(\mathsf{serve})) \to_R \mathsf{top}(\mathsf{free}(\mathsf{serve})) \to_R \mathsf{top}(\mathsf{new}(\mathsf{serve})) \to_R \dots$$

describes that the protocol for serving processes and for letting new processes enter may go on forever. But we will prove that after finitely many steps one reaches a term without the symbol old, i.e., eventually all old processes are served. In our terminology this liveness property is represented by $\mathsf{Live}(\mathcal{T}_{\mathsf{top}}, \to_R, G)$ where $G = \{t \mid t \text{ does not contain an instance of } \mathsf{old}(x)\}$. Note that this liveness property does not hold for various variations of this system. For instance, if processes are allowed to swap by $\mathsf{new}(\mathsf{old}(x)) \to \mathsf{old}(\mathsf{new}(x))$, or if new processes are always allowed to line up by $\mathsf{top}(x) \to \mathsf{top}(\mathsf{new}(x))$, then liveness is destroyed.

Since top(serve) is the only ground top term that is in normal form, we conclude that $\mathsf{NF}(\mathcal{T}_{\mathsf{top}}) \subseteq G$. Hence by Thm. 4 the required liveness property is equivalent to $\mathsf{SN}(\mathcal{T}_{\mathsf{top}}, \to_G)$. To prove this termination property of $\to_G$, according to Thm. 18 we may prove termination of the TRS $LS(R, p)$:

$$\begin{array}{rlrl} \mathsf{top}(\mathsf{free}(x)) &\to \mathsf{top}(\mathsf{check}(\mathsf{new}(x))) \quad (7) & \mathsf{check}(\mathsf{free}(x)) &\to \mathsf{free}(\mathsf{check}(x)) \quad (12) \\ \mathsf{new}(\mathsf{free}(x)) &\to \mathsf{free}(\mathsf{new}(x)) \quad (8) & \mathsf{check}(\mathsf{new}(x)) &\to \mathsf{new}(\mathsf{check}(x)) \quad (13) \\ \mathsf{old}(\mathsf{free}(x)) &\to \mathsf{free}(\mathsf{old}(x)) \quad (9) & \mathsf{check}(\mathsf{old}(x)) &\to \mathsf{old}(\mathsf{check}(x)) \quad (14) \\ \mathsf{new}(\mathsf{serve}) &\to \mathsf{free}(\mathsf{serve}) \quad (10) & \mathsf{check}(\mathsf{old}(x)) &\to \mathsf{old}(x) \quad (15) \\ \mathsf{old}(\mathsf{serve}) &\to \mathsf{free}(\mathsf{serve}) \quad (11) & & \end{array}$$

While standard techniques for automated termination proofs of TRSs do not succeed for this TRS, with the preprocessing steps (a) - (c) termination can easily be shown automatically.

According to (a), we first delete rules which do not influence termination. By counting the occurrences of old, with Lemma 20 we can remove Rule (11). Then in Step (b), we apply the heuristics for semantic labelling and arrive at

$$\mathsf{top}_M(x) = \mathsf{check}_M(x) = \mathsf{old}_M(x) = 0, \;\; \mathsf{new}_M(x) = \mathsf{free}_M(x) = x, \;\; \mathsf{serve}_M = 1$$

for $x \in M = \{0, 1\}$. Indeed this is a model for the TRS. For that purpose, we had to remove Rule (11) since $\mathsf{old}_M(\mathsf{serve}_M) = 0 \neq 1 = \mathsf{free}_M(\mathsf{serve}_M)$. The corresponding labelled TRS $\overline{R}$ is

$$\mathsf{top}_i(\mathsf{free}_i(x)) \to \mathsf{top}_0(\mathsf{check}_i(\mathsf{new}_i(x))) \quad (7_i)$$
$$\mathsf{new}_i(\mathsf{free}_i(x)) \to \mathsf{free}_i(\mathsf{new}_i(x)) \quad (8_i)$$
$$\mathsf{old}_i(\mathsf{free}_i(x)) \to \mathsf{free}_0(\mathsf{old}_i(x)) \quad (9_i)$$
$$\mathsf{new}_1(\mathsf{serve}) \to \mathsf{free}_1(\mathsf{serve}) \quad (10)$$
$$\mathsf{check}_i(\mathsf{free}_i(x)) \to \mathsf{free}_0(\mathsf{check}_i(x)) \quad (12_i)$$
$$\mathsf{check}_i(\mathsf{new}_i(x)) \to \mathsf{new}_0(\mathsf{check}_i(x)) \quad (13_i)$$
$$\mathsf{check}_0(\mathsf{old}_i(x)) \to \mathsf{old}_0(\mathsf{check}_i(x)) \quad (14_i)$$
$$\mathsf{check}_0(\mathsf{old}_i(x)) \to \mathsf{old}_i(x), \quad (15_i)$$

for $i \in \{0, 1\}$. It remains to prove termination of this TRS of 15 rules. According to Step (c) we repeatedly apply Lemma 20. By consecutively choosing $\Sigma' = \{f\}$ for f being top_1, old_1, new_1, free_1, free_0, and check_0, the rules (7_1), (14_1), (10) and (13_1), (9_1) and (12_1), (7_0), and finally (15_0) and (15_1) are removed. Termination of the remaining system consisting of the rules (8_0), (8_1), (9_0), (12_0), (13_0), and (14_0) is easily proved by the recursive path order, using a precedence satisfying $\mathsf{check}_0 > \mathsf{old}_0 > \mathsf{free}_0$, $\mathsf{check}_0 > \mathsf{new}_0 > \mathsf{free}_0$, and $\mathsf{new}_1 > \mathsf{free}_1$. Hence, the liveness property of this example can be proved automatically.

As a second case study we consider the following protocol on a ring of processes (similar to a token ring protocol). Every process is in one of the three states sent, rec (received), or no (nothing). Initially at least one of the processes is in state rec which means that it has received a message (token). Now the protocol is defined as follows:

> If a process is in state rec then it may send its message to its right neighbor which then will be in state rec, while the process itself then will be in state sent.

Clearly, at least one process will always be in state rec, and this procedure can go on forever; we will prove that eventually no process will be in state no. This means that eventually all processes have received the message; a typical liveness property to be proved. The requirement $\mathsf{NF}(I) \subseteq G$ and in fact $\mathsf{NF}(I) = \varnothing$ (for I consisting of all configurations containing rec) is easily seen to hold on the protocol level. According to Thm. 4, for proving the desired liveness property it suffices to show $\mathsf{SN}(I, \to_G)$. The protocol is encoded by unary symbols sent, rec, and no, where the right neighbor of each of these symbols corresponds to the root of its argument. To obtain a ring topology we add a unary symbol top and a constant bot. For a symbol with the argument bot, its right neighbor is defined to be the symbol just below top. So again the state of the whole ring network is represented by a top-term $\mathsf{top}(f_1(\ldots(f_n(\mathsf{bot}))\ldots))$. Here the size n of the ring is arbitrary. In order to pass messages from the bot-process n to the top-process 1, an auxiliary unary symbol up is introduced.

$$\mathsf{rec}(\mathsf{rec}(x)) \to \mathsf{sent}(\mathsf{rec}(x)) \quad (16)$$
$$\mathsf{rec}(\mathsf{sent}(x)) \to \mathsf{sent}(\mathsf{rec}(x)) \quad (17)$$
$$\mathsf{rec}(\mathsf{no}(x)) \to \mathsf{sent}(\mathsf{rec}(x)) \quad (18)$$
$$\mathsf{rec}(\mathsf{bot}) \to \mathsf{up}(\mathsf{sent}(\mathsf{bot})) \quad (19)$$
$$\mathsf{rec}(\mathsf{up}(x)) \to \mathsf{up}(\mathsf{rec}(x)) \quad (20)$$
$$\mathsf{sent}(\mathsf{up}(x)) \to \mathsf{up}(\mathsf{sent}(x)) \quad (21)$$
$$\mathsf{no}(\mathsf{up}(x)) \to \mathsf{up}(\mathsf{no}(x)) \quad (22)$$
$$\mathsf{top}(\mathsf{rec}(\mathsf{up}(x))) \to \mathsf{top}(\mathsf{rec}(x)) \quad (23)$$
$$\mathsf{top}(\mathsf{sent}(\mathsf{up}(x))) \to \mathsf{top}(\mathsf{rec}(x)) \quad (24)$$
$$\mathsf{top}(\mathsf{no}(\mathsf{up}(x))) \to \mathsf{top}(\mathsf{rec}(x)) \quad (25)$$

Now we prove that every infinite top reduction reaches a term without no, proving the desired liveness property. Applying Thm. 18 for $p = \mathsf{no}(x)$, this can be done by proving termination of $LS(R, p)$, which consists of Rules (16) - (22) and

$$\mathsf{top}(\mathsf{rec}(\mathsf{up}(x))) \to \mathsf{top}(\mathsf{check}(\mathsf{rec}(x))) \quad (23a)$$
$$\mathsf{top}(\mathsf{sent}(\mathsf{up}(x))) \to \mathsf{top}(\mathsf{check}(\mathsf{rec}(x))) \quad (24a)$$
$$\mathsf{check}(\mathsf{sent}(x)) \to \mathsf{sent}(\mathsf{check}(x)) \quad (27)$$
$$\mathsf{check}(\mathsf{rec}(x)) \to \mathsf{rec}(\mathsf{check}(x)) \quad (28)$$

$$\mathsf{top}(\mathsf{no}(\mathsf{up}(x))) \to \mathsf{top}(\mathsf{check}(\mathsf{rec}(x))) \quad (25a)$$
$$\mathsf{check}(\mathsf{up}(x)) \to \mathsf{up}(\mathsf{check}(x)) \quad (26)$$
$$\mathsf{check}(\mathsf{no}(x)) \to \mathsf{no}(\mathsf{check}(x)) \quad (29)$$
$$\mathsf{check}(\mathsf{no}(x)) \to \mathsf{no}(x) \quad (30)$$

Termination is easily proved completely automatically according to our heuristics: by respectively choosing Σ' to be $\{\mathsf{no}\}$ and $\{\mathsf{rec}, \mathsf{up}\}$ in Lemma 20, the rules (16), (18), (23a), and (25a) can be removed. After applying labelling according to our heuristics a TRS is obtained for which termination is proved automatically by applying Lemma 20 and the recursive path order, cf. [9].

6 Conclusion and Further Research

We showed how to relate liveness and termination of TRSs and presented a sound and complete transformation such that liveness holds iff the transformed TRS is terminating. By a simpler sound transformation and by refining termination techniques for TRSs we developed an approach to verify liveness mechanically.

Our results can be refined in several ways. For instance, instead of one unary top symbol one can regard several top symbols of arbitrary arity and one can extend the framework to liveness w.r.t. several terms $p_1, \ldots, p_n$ instead of just one p. Such refinements and further examples of liveness properties verified by our method can be found in [9]. For example, we show liveness in a network with several waiting lines of processes which want to gain access to a shared resource. This problem is considerably more difficult than the waiting line protocol in Sect. 5.3, since liveness only holds if the lines are synchronized in a suitable way.

References

1. B. Alpern and F. B. Schneider. Defining liveness. *Inf. Pr. Lett.*, 21:181–185, 1985.
2. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
3. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambr. Univ. Pr., 1998.
4. A. Bouajjani. Languages, rewriting systems, and verification of infinite-state systems. In *Proc. ICALP '01*, volume 2076 of *LNCS*, pages 24–39, 2001.
5. N. Dershowitz. Termination of rewriting. *J. Symb. Comp.*, 3:69–116, 1987.
6. J. Giesl and T. Arts. Verification of Erlang processes by dependency pairs. *Applicable Algebra in Engineering, Communication and Comp.*, 12(1,2):39–72, 2001.
7. J. Giesl and A. Middeldorp. Transforming context-sensitive rewrite systems. In *Proc. 10th RTA*, volume 1631 of *Lecture Notes in Comp. Sc.*, pages 271–285, 1999.
8. J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 2003. To appear. Preliminary extended version in Technical Report AIB-2002-02, RWTH Aachen, Germany.
9. J. Giesl and H. Zantema. Liveness in rewriting. Technical Report AIB-2002-11, RWTH Aachen, Germany, 2002. `http://aib.informatik.rwth-aachen.de`.
10. L. Lamport. A new solution to Dijkstra's concurrent programming problem. *Communications of the ACM*, 17(8):453–455, 1974.
11. H. Zantema. Termination of term rewriting: Interpretation and type elimination. *Journal of Symbolic Computation*, 17:23–50, 1994.
12. H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.