Electronic Notes in Theoretical Computer Science 86 No. 4 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume86.html 16 pages

# Simulating Liveness by Reduction Strategies

Jürgen Giesl [1]

*LuFG Informatik II, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany*

Hans Zantema [2]

*Department of Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

**Abstract**

We define a general framework to handle liveness and related properties by reduction strategies in abstract reduction and term rewriting. Classically, reduction strategies in rewriting are used to simulate the evaluation process in programming languages. The aim of our work is to use reduction strategies to also study liveness questions which are of high importance in practice (e.g., in protocol verification for distributed processes). In particular, we show how the problem of verifying liveness is related to termination of term rewrite systems (TRSs). Using our results, techniques for proving termination of TRSs can be used to verify liveness properties.

## 1 Introduction

In this paper, we give a formal definition of *safety* and *liveness* using the framework of abstract reduction (Sect. 2). In particular, liveness is formalized by imposing a suitable reduction strategy. In Sect. 3 we show how the reduction-based definitions of safety and liveness correspond to standard definitions from the literature [1]. Then in Sect. 4 the notion of liveness is specialized to the framework of term rewriting. In Sect. 5 we investigate the connection between liveness and termination. More precisely, we show how termination of ordinary rewriting is related to termination under the reduction strategy required for liveness properties. To this end, we present a transformation such that termination of the transformed TRS is equivalent to the liveness property of the original TRS. In [11], a similar transformation was presented for liveness properties of a certain form (global liveness), but we show that such transformations can also be given for other liveness properties (local liveness). With

[1] Email: giesl@informatik.rwth-aachen.de
[2] Email: h.zantema@tue.nl

these results, (existing) termination techniques for TRSs can be used to infer liveness. So our approach differs from most previous applications of rewriting techniques in process verification which were mainly concerned with the verification of other properties (e.g., reachability [4,7] or equivalence [5,13]).

## 2 Formalizing Safety and Liveness

We define safety and liveness using the framework of abstract reduction. Let $S$ be a set of states and let $\rightarrow \subseteq S \times S$ where "$t \rightarrow u$" means that a computation step from $t$ to $u$ is possible. A *computation sequence* or *reduction* is a finite sequence $t_0, t_1, \ldots, t_n$ or an infinite sequence $t_0, t_1, t_2, \ldots$ with $t_i \rightarrow t_{i+1}$. As usual, $\rightarrow^*$ is the reflexive transitive closure of $\rightarrow$. To define safety and liveness we assume a set $G \subseteq S$ of *goal states* and a set $I \subseteq S$ of *initial states*.

### *2.1 Formalizing Safety*

For safety, $G$ represents "good" states and safety means that nothing bad will happen. So in every reduction starting in initial states, all states are good.

**Definition 2.1 (Safety)** $\mathsf{Safe}(I, \rightarrow, G)$ iff $\forall t \in I, u \in S : (t \rightarrow^* u) \Rightarrow u \in G$.

Usually safety properties are proved by choosing an invariant. Then one proves that the invariant is true initially and that the invariant holds for every state $u$ where $t \rightarrow u$ for some state $t$ satisfying the invariant. In this way safety is proved in a purely local way: only one-step reductions are considered in the proof. In other words, safety is proved by induction on the length of the reduction, and a claim about arbitrary reductions can be made by analyzing only one-step reductions. It is a natural question whether this approach covers all safety properties. The next theorem answers this question positively. Here, the set $G'$ represents the set of all states satisfying the invariant.

**Theorem 2.2 (Proving Safety)** $\mathsf{Safe}(I, \rightarrow, G)$ *iff there is a set* $G' \subseteq S$ *with*

- $I \subseteq G' \subseteq G$, *and*
- $\forall t \in G', u \in S : (t \rightarrow u) \Rightarrow u \in G'$.

**Proof.** For the "if"-part, let $G'$ satisfy the properties above and $t = t_0 \rightarrow t_1 \rightarrow \cdots \rightarrow t_n = u$ for $t \in I$. By induction on $i$, one can show that $t_i \in G'$ for every $i = 0, \ldots, n$. This implies $t_n = u \in G' \subseteq G$, which we had to prove.

For the "only if"-direction we assume $\mathsf{Safe}(I, \rightarrow, G)$ and let $G' = \{u \in S \mid \exists t \in I : t \rightarrow^* u\}$. Now we show that $G'$ satisfies the properties above.

- $I \subseteq G'$ holds since $t \rightarrow^* t$,
- $G' \subseteq G$ is implied by $\mathsf{Safe}(I, \rightarrow, G)$, and
- $\forall t \in G', u \in S : (t \rightarrow u) \Rightarrow u \in G'$ follows from the definition of $G'$. □

Note that the "only if"-direction cannot be proved simply by defining $G' = G$, since in general, $G$ is not invariant. For instance, if $S = \{1, 2, 3\}$, $\rightarrow =$

$\{(1,1),(2,3)\}$, $I = \{1\}$ and $G = \{1,2\}$, then $\mathsf{Safe}(I, \to, G)$, but $u \in G$ does not always hold if $t \in G$ and $t \to u$. Essentially this corresponds to the difference between "always true" and "invariant" as pointed out in [14].

### *2.2 Formalizing Liveness*

Liveness properties state that some goal will eventually be reached. To formalize "eventuality" we consider maximal reductions that continue as long as possible. A reduction is *maximal* if it is infinite or its last element is a *normal form* from $\mathsf{NF} = \{t \in S \mid \neg\exists u : t \to u\}$. The liveness property $\mathsf{Live}(I, \to, G)$ holds if every maximal reduction starting in $I$ contains an element of $G$.

**Definition 2.3 (Liveness)** $\mathsf{Live}(I, \to, G)$ holds iff

(i) $\forall t_0, t_1, t_2, \ldots : (t_0 \in I \wedge \forall i \in \mathbb{N} : t_i \to t_{i+1}) \Rightarrow \exists i \in \mathbb{N} : t_i \in G$, and

(ii) $\forall t_0, t_1, \ldots, t_n : (t_0 \in I \wedge t_n \in \mathsf{NF} \wedge \forall i \in \{0, \ldots, n-1\} : t_i \to t_{i+1}) \Rightarrow \exists i \in \{0, \ldots, n\} : t_i \in G$.

For example, *termination* (or *strong normalization*) is a special liveness property describing the non-existence of infinite reductions, i.e.,

$$\mathsf{SN}(I, \to) \;=\; \neg(\exists t_0, t_1, t_2, \ldots : \; t_0 \in I \wedge \forall i \in \mathbb{N} : t_i \to t_{i+1}).$$

**Theorem 2.4 ($\mathsf{SN}$ is a Liveness Prop. [11])** $\mathsf{SN}(I, \to)$ *iff* $\mathsf{Live}(I, \to, \mathsf{NF})$.

The next theorem states a kind of converse. Here, we impose a reduction strategy such that "$\to$" may only proceed if the current state is not in $G$.

**Definition 2.5 ($\to_G$)** Let $\to_G \subseteq S \times S$ where $t \to_G u$ iff $t \to u$ and $t \notin G$.

Now one can show that $\mathsf{Live}(I, \to, G)$ is equivalent to $\mathsf{SN}(I, \to_G)$. The "only if"-part holds without any further conditions. For the "if"-part, $G$ must contain 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 $t_0 \to \ldots \to t_n$ with all $t_i \notin G$, we might have $\mathsf{SN}(I, \to_G)$ but not $\mathsf{Live}(I, \to, G)$.

**Theorem 2.6 (Equivalence of Liveness and Termination [11])**
*Let* $\mathsf{NF}(I) \subseteq G$. *Then* $\mathsf{Live}(I, \to, G)$ *holds iff* $\mathsf{SN}(I, \to_G)$ *holds.*

By Thm. 2.6 we can 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 liveness for all infinite computations. In Sect. 5 we discuss how techniques to prove full termination of TRSs can be used for termination of $\to_G$.

## 3 Connection to Previous Formalization

In this section we show that our notions of *safety* and *liveness* specialize the "standard" definitions of Alpern and Schneider [1]. In their framework, a property $P$ is a set of infinite sequences of states. Terminating executions of a program are represented by repeating the final state infinitely often. According

to [1], a property $P$ is a *safety property* iff the following condition holds:

(1) If $\langle t_0, t_1, \ldots\rangle \notin P$, then there is an $i \in \mathbb{N}$ such that for all $\langle t_{i+1}, t_{i+2}, \ldots\rangle$ we have $\langle t_0, \ldots, t_i, t_{i+1}, t_{i+2}, \ldots\rangle \notin P$.

In other words, if an infinite sequence $\langle t_0, t_1, \ldots\rangle$ does not satisfy a safety property $P$, then there is a finite prefix $\langle t_0, \ldots, t_i\rangle$ of the sequence which already violates it. So irrespective of how this finite prefix is extended to an infinite sequence $\langle t_0, \ldots, t_i, t_{i+1}, t_{i+2}, \ldots\rangle$, the property $P$ is not fulfilled.

In [1], for a *liveness property* $P$, every finite prefix of states can be extended into an infinite sequence satisfying $P$. So $P$ is a liveness property iff

(2) For all $\langle t_0, \ldots, t_i\rangle$ there exist $\langle t_{i+1}, t_{i+2}, \ldots\rangle$ such that $\langle t_0, \ldots, t_i, t_{i+1}, t_{i+2}, \ldots\rangle \in P$.

In contrast to safety, this liveness definition does not require that there is a discrete point in the infinite sequence from which on the liveness condition is always fulfilled. To ease the checking of a liveness property (in the rewriting framework), it turns out to be useful to add this demand. We call $P$ a *discrete* property iff it satisfies the following discreteness condition:

(3) If $\langle t_0, t_1, \ldots\rangle \in P$, then there is an $i \in \mathbb{N}$ such that for all $\langle t_{i+1}, t_{i+2}, \ldots\rangle$ we have $\langle t_0, \ldots, t_i, t_{i+1}, t_{i+2}, \ldots\rangle \in P$.

In other words, if an infinite sequence satisfies the property $P$, then this is due to a finite prefix $\langle t_0, \ldots, t_i\rangle$. No matter how this prefix is extended, the property will always be preserved. A *discrete liveness* property is a property satisfying both (2) and (3). An example for a non-discrete liveness property is starvation freedom which states that a process makes progress infinitely often.

There is a natural correspondence between safety and discreteness: $P$ is a safety property iff $P$'s complement is a discreteness property. This indicates that it is often more intuitive to use discreteness as the definition of "liveness". Indeed, one may argue that non-discrete liveness properties like starvation freedom should rather be called *fairness* instead of *liveness properties*.

Now we compare the definitions of [1] to our formalizations in Sect. 2. We defined safety and liveness not as properties of arbitrary sequences of states, but of sequences representing computation. So if $P_2$ is a safety or liveness property of sequences of states and $P_1$ is the property that a sequence of states corresponds to a particular computation, then we formulate statements like "$P_1 \subseteq P_2$" expressing that $P_1$ implies $P_2$. Moreover, in our approach the "*goals*" to be reached are made explicit as properties of states. We demonstrate that our notions of safety and liveness nevertheless correspond to the notions of [1] provided that their liveness definition is restricted to *discrete* liveness.

First, we show that our safety and liveness concepts can be simulated by the concepts of Alpern and Schneider. In other words, our definitions of safety and liveness are also safety and (discrete) liveness properties according to [1]. Here, $P_{I\rightarrow}$ is the property that a sequence of states corresponds to a computation starting in $I$. Then $\mathsf{Safe}(I, \rightarrow, G)$ can be formulated as "$P_{I\rightarrow} \subseteq P_G^{\mathsf{safe}}$" and $\mathsf{Live}(I, \rightarrow, G)$ can be formulated as "$P_{I\rightarrow} \subseteq P_G^{\mathsf{live}}$" for suitable safety and

liveness properties $P_G^{\mathsf{safe}}$ and $P_G^{\mathsf{live}}$ in the framework of Alpern and Schneider.

**Theorem 3.1 (Simulating Def. 2.1 and 2.3 in [1])** *Let $G \neq \varnothing$. We define the following properties (i.e., sets of infinite sequences of states):*

- $P_{I\rightarrow} = \{\langle t_0, t_1, \ldots\rangle \mid t_0 \in I,\ \forall i \in \mathbb{N} : t_i \rightarrow t_{i+1}$ *or* $t_i \in \mathsf{NF}$ *and* $t_i = t_{i+1}\}$
- $P_G^{\mathsf{safe}} = \{\langle t_0, t_1, \ldots\rangle \mid t_i \in G$ *for all* $i \in \mathbb{N}\}$
- $P_G^{\mathsf{live}} = \{\langle t_0, t_1, \ldots\rangle \mid t_i \in G$ *for some* $i \in \mathbb{N}\}$

*(a)* $P_{I\rightarrow}$ *and* $P_G^{\mathsf{safe}}$ *are safety properties,* $P_G^{\mathsf{live}}$ *is a discrete liveness property.*

*(b)* $\mathsf{Safe}(I, \rightarrow, G)$ *iff* $P_{I\rightarrow} \subseteq P_G^{\mathsf{safe}}$

*(c)* $\mathsf{Live}(I, \rightarrow, G)$ *iff* $P_{I\rightarrow} \subseteq P_G^{\mathsf{live}}$

**Proof.**

(a) If $\langle t_0, t_1, \ldots\rangle \notin P_{I\rightarrow}$, then we have $t_0 \notin I$ or $t_i \notin \mathsf{NF}$ and $t_i \not\rightarrow t_{i+1}$ or $t_i \in \mathsf{NF}$ and $t_i \neq t_{i+1}$ for some $i$. In the first case, every infinite sequence starting with the prefix $t_0$ will not be in $P_{I\rightarrow}$. In the second and third case, every infinite sequence starting with $\langle t_0, \ldots, t_i, t_{i+1}\rangle$ will not be in $P_{I\rightarrow}$. Hence, $P_{I\rightarrow}$ is indeed a safety property (i.e., it satisfies (1)).

If $\langle t_0, t_1, \ldots\rangle \notin P_G^{\mathsf{safe}}$, then there is a $t_i \notin G$. So every infinite sequence starting with $\langle t_0, \ldots, t_i\rangle$ is not in $P_G^{\mathsf{safe}}$ and thus, $P_G^{\mathsf{safe}}$ is a safety property.

Let $\langle t_0, \ldots, t_i\rangle$ be arbitrary states. Since $G \neq \varnothing$, there is a $t_{i+1} \in G$. So every finite prefix $\langle t_0, \ldots, t_i\rangle$ can be extended to an infinite sequence $\langle t_0, \ldots, t_i, t_{i+1}, \ldots\rangle$ satisfying $P_G^{\mathsf{live}}$. Hence, $P_G^{\mathsf{live}}$ is a liveness property.

Let $\langle t_0, t_1, \ldots\rangle \in P_G^{\mathsf{live}}$. Then there is a $t_i \in G$. Hence, if one extends $\langle t_0, \ldots, t_i\rangle$ by an arbitrary sequence $\langle t_{i+1}, t_{i+2}, \ldots\rangle$ we again have $\langle t_0, \ldots, t_i, t_{i+1}, t_{i+2}, \ldots\rangle \in P_G^{\mathsf{live}}$. Thus, $P_G^{\mathsf{live}}$ is a discrete liveness property.

(b) For "only if", let $\langle t_0, t_1, \ldots\rangle \in P_{I\rightarrow}$. So $t_0 \in I$ and for all $i$ we have $t_0 \rightarrow^* t_i$. By $\mathsf{Safe}(I, \rightarrow, G)$ this implies $t_i \in G$. So we obtain $\langle t_0, t_1, \ldots\rangle \in P_G^{\mathsf{safe}}$.

For the "if"-direction, let $t_0 \in I$ and $t_0 \rightarrow^* t_i$. Then there exists an infinite sequence $\langle t_0, \ldots, t_i, \ldots\rangle \in P_{I\rightarrow}$. So $P_{I\rightarrow} \subseteq P_G^{\mathsf{safe}}$ implies $t_i \in G$.

(c) For the "only if"-direction, let $\langle t_0, t_1, \ldots\rangle \in P_{I\rightarrow}$ and therefore $t_0 \in I$. If $t_i \rightarrow t_{i+1}$ for all $i$, then by $\mathsf{Live}(I, \rightarrow, G)$ there is a $t_i \in G$. If there is a $t_n \in \mathsf{NF}$ and $t_n = t_m$ for all $m > n$, then by $\mathsf{Live}(I, \rightarrow, G)$ there is a $t_i \in G$ for $i \leq n$. Therefore, in both cases we obtain $\langle t_0, t_1, \ldots\rangle \in P_G^{\mathsf{live}}$.

For "if', let $t_0 \in I$. If $t_0 \rightarrow t_1 \rightarrow ...$ is infinite, then $\langle t_0, t_1, ...\rangle \in P_{I\rightarrow} \subseteq P_G^{\mathsf{live}}$ and $t_i \in G$ for some $i$. Similarly, if $t_0 \rightarrow ... \rightarrow t_n$ for $t_n \in \mathsf{NF}$, then $\langle t_0, ..., t_n, t_n, t_n, \ldots\rangle \in P_{I\rightarrow} \subseteq P_G^{\mathsf{live}}$ and therefore $t_i \in G$ for some $i$. □

Now we prove the converse, i.e., all safety and (discrete) liveness properties of [1] can also be expressed in our framework. One can even simulate every discrete property by a liveness property in our framework. So if $P^{\mathsf{safe}}$ is a safety and $P^{\mathsf{live}}$ is a discrete property in [1] (i.e., $P^{\mathsf{safe}}$ satisfies (1), $P^{\mathsf{live}}$ satisfies (3)), then for any property $P$, "$P \subseteq P^{\mathsf{safe}}$" is a safety and "$P \subseteq P^{\mathsf{live}}$" is a liveness property in our framework. Thus, both frameworks are equally power-

ful for examining properties "$P_1 \subseteq P_2$" of computations (if the liveness definition of [1] is restricted to *discrete* liveness). In [1], a *goal* not only depends on a state itself but also on the way that state is reached. In contrast, in our approach being a *goal* is just a property of the state alone. However, the concept of *goals* in [1] can be simulated in our framework as well. For that purpose, the notion of state is extended until it covers all relevant information for being a goal. So instead of the original states we now consider finite tuples of states $(t_0, \ldots, t_i)$ which stand for "state $t_i$, if reached by the sequence $\langle t_0, \ldots, t_i \rangle$". The initial "tuple-state" is the empty tuple $(\,)$. To encode questions like "$P_1 \subseteq P_2$" in our framework, we simulate the first property $P_1$ by a relation $\rightarrow_{P_1}$ on tuple-states. Here, $\rightarrow_{P_1}$ builds up elements of $P_1$ step by step.

**Theorem 3.2 (Simulating [1] in Def. 2.1 and 2.3)** *Let $P \neq \varnothing$ be an arbitrary property, let $P^{\mathsf{safe}}$ satisfy (1), and let $P^{\mathsf{live}}$ satisfy (3). Let $S' = \{(t_0, \ldots, t_n) \mid t_i \in S, n \geq 0\}$ and let $G^{\mathsf{safe}}, G^{\mathsf{live}} \subseteq S'$ with*

- $G^{\mathsf{safe}} = \{(t_0, \ldots, t_i) \mid \langle t_0, \ldots, t_i, t_{i+1}, \ldots \rangle \in P^{\mathsf{safe}} \text{ for some } \langle t_{i+1}, \ldots \rangle\}$
- $G^{\mathsf{live}} = \{(t_0, \ldots, t_i) \mid \langle t_0, \ldots, t_i, t_{i+1}, \ldots \rangle \in P^{\mathsf{live}} \text{ for all } \langle t_{i+1}, \ldots \rangle\}$

*We define the relation $\rightarrow_P \subseteq S' \times S'$ as $(t_0, \ldots, t_i) \rightarrow_P (t_0, \ldots, t_i, t_{i+1})$ iff there exist $\langle t_{i+2}, \ldots \rangle$ such that $\langle t_0, \ldots, t_i, t_{i+1}, t_{i+2}, \ldots \rangle \in P$. Then we have:*

*(a)* $P \subseteq P^{\mathsf{safe}}$ *iff* $\mathsf{Safe}((\,), \rightarrow_P, G^{\mathsf{safe}})$

*(b)* $P \subseteq P^{\mathsf{live}}$ *iff* $\mathsf{Live}((\,), \rightarrow_P, G^{\mathsf{live}})$

**Proof.**

(a) For the "if"-direction, let $\langle t_0, t_1, \ldots \rangle \in P$. So we have $(\,) \rightarrow_P (t_0) \rightarrow_P (t_0, t_1) \rightarrow_P \ldots$ By $\mathsf{Safe}((\,), \rightarrow_P, G^{\mathsf{safe}})$ this implies $(t_0, \ldots, t_i) \in G^{\mathsf{safe}}$ for all $i$. Assume that $\langle t_0, t_1, \ldots \rangle \notin P^{\mathsf{safe}}$. Since $P^{\mathsf{safe}}$ is a safety property, there is an $i$ such that for all $\langle t_{i+1}, t_{i+2}, \ldots \rangle$ we have $\langle t_0, \ldots, t_i, t_{i+1}, t_{i+2}, \ldots \rangle \notin P^{\mathsf{safe}}$. This implies $(t_0, \ldots, t_i) \notin G^{\mathsf{safe}}$ which is a contradiction.

For "only if", let $(\,) \rightarrow_P^* (t_0, \ldots, t_i)$. By definition of $\rightarrow_P$, there is $\langle t_{i+1}, t_{i+2}, \ldots \rangle$ with $\langle t_0, \ldots, t_i, t_{i+1}, \ldots \rangle \in P \subseteq P^{\mathsf{safe}}$. Hence, $(t_0, \ldots, t_i) \in G^{\mathsf{safe}}$.

(b) For "if", let $\langle t_0, t_1, \ldots \rangle \in P$, i.e., $(\,) \rightarrow_P (t_0) \rightarrow_P (t_0, t_1) \rightarrow_P \ldots$ By definition, $\rightarrow_P$ has no normal forms. By $\mathsf{Live}((\,), \rightarrow_P, G^{\mathsf{live}})$ there is an $i$ with $(t_0, \ldots, t_i) \in G^{\mathsf{live}}$. So for all infinite extensions of $\langle t_0, \ldots, t_i \rangle$, the resulting sequence is in $P^{\mathsf{live}}$. In particular, we obtain $\langle t_0, t_1, \ldots \rangle \in P^{\mathsf{live}}$.

For "only if", let $(\,) \rightarrow_P (t_0) \rightarrow_P (t_0, t_1) \rightarrow_P \ldots$ By definition of $\rightarrow_P$ we have $\langle t_0, \ldots \rangle \in P \subseteq P^{\mathsf{live}}$. Since $P^{\mathsf{live}}$ satisfies (3), there is an $i$ such that *for all* $\langle t_{i+1}, t_{i+2}, \ldots \rangle$ we have $\langle t_0, \ldots, t_i, t_{i+1}, t_{i+2}, \ldots \rangle \in P^{\mathsf{live}}$. In other words, $(t_0, \ldots, t_i) \in G^{\mathsf{live}}$ for some $i$ and thus $\mathsf{Live}((\,), \rightarrow_P, G^{\mathsf{live}})$. □

## 4 Liveness in Term Rewriting

Now we focus on liveness in rewriting. More precisely, we study the property $\mathsf{Live}(I, \rightarrow_R, G)$ where $\rightarrow_R$ is the rewrite relation corresponding to a TRS $R$.

For an introduction to term rewriting, the reader is referred to [3], for example.

Let $\Sigma$ be a signature with at least one constant and let $\mathcal{V}$ be a set of variables. $\mathcal{T}(\Sigma, \mathcal{V})$ is the set of terms over $\Sigma$ and $\mathcal{V}$ and $\mathcal{T}(\Sigma)$ is the set of ground terms. Now $\mathcal{T}(\Sigma, \mathcal{V})$ represents computation states and $G \subseteq \mathcal{T}(\Sigma, \mathcal{V})$.

By Thm. 2.6, $\mathsf{Live}(I, \to, G)$ is equivalent to $\mathsf{SN}(I, \to_G)$, if $\mathsf{NF}(I) \subseteq G$. To verify liveness, we want to prove termination of $\to_G$ by approaches for termination proofs of ordinary TRSs. But due to the reduction strategy in the definition of $\to_G$, classical termination techniques are not applicable directly. In Sect. 5 we present a transformation from a TRS $R$ and a set $G$ of terms to a TRS $R'$ such that $\to_G$ terminates iff the rewrite relation $\to_{R'}$ terminates. A transformation where "if" holds is called *sound* and if the "only if"-direction holds, it is called *complete*. The existence of a sound and complete transformation means that liveness and termination are essentially equivalent.

Depending on the form of $G$, different transformations have to be developed. We concentrate on two kinds of liveness properties: *local liveness* where $G$ is closed under contexts and substitutions and *global liveness* where the complement of $G$ is closed under contexts and substitutions. In global liveness, the property of the term to be reached eventually is that a certain pattern does not occur *anywhere* in the term, which is a global property of the term. In local liveness, the desired property is that a certain pattern occurs *somewhere* in the term, being a local property. Clearly, there exist liveness properties which do not belong to our classes of local or global liveness. However, in [11] and in the following sections, we demonstrate that local and global liveness indeed capture many interesting liveness properties.

### *4.1 Global Liveness*

A liveness property $\mathsf{Live}(I, \to_R, G)$ is called *global* if $G$ has the form [3]

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

In other words, $G$ consists of all terms which cannot be written as $C[p\sigma]$ for any context $C$ and substitution $\sigma$. 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 $p$. Note that for sets $G$ as above, the relation $\to_G$ is a rewrite relation (i.e., it is closed under substitutions and contexts). This also makes clear that ground termination of $\to_G$ is equivalent to full termination of $\to_G$.

A typical global liveness property is that eventually all processes requesting a resource are granted access, cf. [11, Sect. 5.3]. Here, the network of processes is described by a term $t$ and processes that have not yet gained access to the resource are represented as an instance of the subterm $\mathsf{old}(x)$. The aim is to prove that eventually, $t$ reduces to a term without $\mathsf{old}$. This form of liveness is called "global" since the goal situation is stated as a condition on *all* processes.

For arbitrary terms and TRSs, $\to_G$ is not useful: if there is a symbol $f$ of

[3] An easy extension is to permit $G = \{t | t \text{ contains no instances of terms } p_1, \ldots, \text{or } p_n\}$.

arity $> 1$ or if $p$ contains a variable $x$ (i.e., if $p = C[x]$ for some context $C$), then termination of $\rightarrow_G$ implies termination of the full rewrite relation $\rightarrow_R$. The reason is that any infinite reduction $t_0 \rightarrow_R t_1 \rightarrow_R \cdots$ gives rise to an infinite reduction $f(t_0, p, \ldots) \rightarrow_R f(t_1, p, \ldots) \rightarrow_R \cdots$ or $C[t_0] \rightarrow_R C[t_1] \rightarrow_R \cdots$ 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 unary symbol $\mathsf{tp}$ which may only occur on the root position of terms. Moreover, if the root of one side of a rule is $\mathsf{tp}$, then the other side must also start with $\mathsf{tp}$.

Top rewrite systems typically suffice to model networks of processes, since the whole network is represented by a top term [8]. Clearly, in top rewrite systems, top terms can only be reduced to top terms again. In such systems we consider properties $\mathsf{Live}(\mathcal{T}_{\mathsf{top}}, \rightarrow_R, G)$, where $\mathcal{T}_{\mathsf{top}}$ is the set of ground terms with $\mathsf{tp}$ on root position. So the goal is to prove that every maximal reduction of ground top terms contains a term without an instance of $p$. Transformations by which this can be treated are elaborated extensively in [11].

### *4.2 Local Liveness*

In the remainder we concentrate on *local liveness*, where $G$ has the form [4]

$$G \;=\; \{t \mid t \text{ contains an instance of } p\} \quad \text{for some term } p.$$

So in local liveness, $G$ is closed under substitutions and contexts, whereas in global liveness the complement of $G$ is closed under substitutions and contexts. Now $t \rightarrow_G u$ holds if $t \rightarrow_R u$ and $t$ contains no instance of $p$. A typical local liveness property is that eventually at least *one* process requesting a resource is granted access, rather than requiring this for all processes as in global liveness.

**Example 4.1 (Waiting Lines)** This TRS describes the behavior of two waiting lines of processes. The combination of the lines has a bounded size, i.e., a new process can only enter a waiting line if some process was "served". The processes in the lines are served on a "first in - first out" basis. So at the front end of a waiting line, a process may be served, where serving is denoted by a constant $\mathsf{serve}$. If a process is served, its place in the line is replaced by a free place, denoted by $\mathsf{free}$. 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 a line has a free place at its back end, a new process $\mathsf{new}$ may enter *any* waiting line and the free place is deleted. Apart from new processes represented by $\mathsf{new}$ we also consider old processes represented by $\mathsf{old}$, which were already in the line initially. Introducing the binary symbol $\mathsf{tp}$ having the representations of the waiting lines as its arguments, this network is described by the following top rewrite system $R$. Here, the leftmost symbol of a term represents the "back

[4] Again, the approach can easily be extended to sets $G$ of the form $\{t \mid t$ contains an instance of $p_1, p_2$, or $p_n\}$ for terms $p_1, \ldots, p_n$.

end" of the waiting line and the rightmost symbol is the "front end".

$$\mathsf{tp}(\mathsf{free}(x), y) \to \mathsf{tp}(\mathsf{new}(x), y) \qquad \mathsf{new}(\mathsf{free}(x)) \to \mathsf{free}(\mathsf{new}(x))$$
$$\mathsf{tp}(\mathsf{free}(x), y) \to \mathsf{tp}(x, \mathsf{new}(y)) \qquad \mathsf{old}(\mathsf{free}(x)) \to \mathsf{free}(\mathsf{old}(x))$$
$$\mathsf{tp}(x, \mathsf{free}(y)) \to \mathsf{tp}(\mathsf{new}(x), y) \qquad \mathsf{new}(\mathsf{serve}) \to \mathsf{free}(\mathsf{serve})$$
$$\mathsf{tp}(x, \mathsf{free}(y)) \to \mathsf{tp}(x, \mathsf{new}(y)) \qquad \mathsf{old}(\mathsf{serve}) \to \mathsf{free}(\mathsf{serve})$$

For various variations of this system we proved global liveness with respect to $p = \mathsf{old}(x)$ in [11], stating that eventually *all* old clients will be served. However, for this version this global liveness property does not hold: we have an infinite reduction of top terms all containing the symbol $\mathsf{old}$:

$$\mathsf{tp}(\mathsf{new}(\mathsf{serve}), \mathsf{old}(\mathsf{serve})) \to_R \mathsf{tp}(\mathsf{free}(\mathsf{serve}), \mathsf{old}(\mathsf{serve})) \to_R$$
$$\mathsf{tp}(\mathsf{new}(\mathsf{serve}), \mathsf{old}(\mathsf{serve})) \to_R \ldots$$

But we can prove the weaker local liveness property that eventually *some* client will be served. In our formalism this is done by choosing $G = \{t \mid t$ contains an instance of $\mathsf{free}(\mathsf{serve})\}$, since $\mathsf{free}(\mathsf{serve})$ always occurs after serving a client.

Now $\to_G$ is not a rewrite relation since it is neither closed under contexts nor under substitutions. Moreover, ground termination of $\to_G$ does not imply termination of $\to_G$. To permit the restriction to ground terms, when regarding local liveness, we always assume that our signature contains at least an extra constant $\mathsf{c}$ and an extra unary function symbol $\mathsf{h}$ which do not appear in $p$. In this case, ground termination and full termination of $\to_G$ are equivalent.

## 5 Transformations for Local Liveness

Now we investigate the correspondence between liveness and termination in the framework of term rewriting. Then all existing techniques for termination proofs of TRSs (including future developments) can be used for liveness properties. A first step into this direction was taken in [8], where the termination proof technique of *dependency pairs* was used to verify certain liveness properties of telecommunication processes. However, we now develop an approach to connect liveness and termination in general. While in [11] we considered global liveness, we now show that a similar approach is possible for local liveness, although $\to_G$ is no longer a rewrite relation. Hence, let $G = \{t|$ $t$ contains an instance of $p\}$. Our goal is to present a transformation $L$ such that $\mathsf{SN}(\to_G)$ iff $\mathsf{SN}(L(R,p))$, i.e., iff all $L(R,p)$-reductions are terminating. [5]

As a first approach we choose a sound transformation $L_s$ defined by

$$L_s(R,p) = \{\ l \to r \in R \mid p \text{ does not occur in } r\ \}.$$

Clearly, rules creating instances of $p$ may not be applied in infinite $\to_G$-reductions. Hence, $\mathsf{SN}(L_s(R,p))$ implies $\mathsf{SN}(\to_G)$, i.e., $L_s$ is sound. To prove

[5] To ease the presentation, we only present transformations without regarding initial states $I$. However, our transformations can easily be extended to take $I$ into account.

liveness in Ex. 4.1, for $p = \mathsf{free}(\mathsf{serve})$, $L_s(R,p)$ consists of $R$'s first 6 rules and termination is easily proved by dependency pairs [2].

However, $L_s$ is not complete: $\mathsf{SN}(\to_G)$ does not imply $\mathsf{SN}(L_s(R,p))$. As an example let $R = \{\mathsf{f}(x) \to \mathsf{f}(\mathsf{f}(x))\}$ and let $p = \mathsf{f}^{10}(x)$. In the sequel we will see that $\mathsf{SN}(\to_G)$ holds. However, $L_s(R,p) = R$ is clearly not terminating. The rest of this section is devoted to a sound and complete transformation $L$.

$L(R,p)$'s construction is motivated by an existing transformation [9,10] developed for a completely different purpose (termination of context-sensitive rewriting). We introduce a fresh binary symbol $\mathsf{mat}$, fresh unary symbols $\mathsf{tp}$, $\mathsf{chk}$, $\mathsf{active}$, $\mathsf{mark}$, $\mathsf{no}$, and for every variable in $p$ we introduce one fresh constant. Let $\overline{p}$ be the ground term obtained by replacing every variable in $p$ by its corresponding fresh constant and $\Sigma_G$ denotes the resulting extended signature. As in [9,10], the symbol $\mathsf{active}$ is used to specify potential next redexes and $\mathsf{mark}$ means that its argument must be inspected in order to identify those subterms which may be reduced next. In [9,10], this depends on the position of a subterm (only subterms in "active" positions of all function symbols above them are marked with $\mathsf{active}$). In contrast, now all subterms that do not contain an instance of $p$ may be marked with $\mathsf{active}$.

To simplify the $\mathsf{mat}$-rules, we restrict ourselves to linear terms $p$ where any variable occurs at most once. Here, $\mathsf{mat}(\overline{p},t)$ checks whether $p$ does not match $t$. A corresponding transformation is also possible for non-linear terms $p$, but then $\mathsf{mat}$ would would also fail if two occurrences of a variable would have to be instantiated differently. The TRS $L(R,p)$ consists of the following rules.

$$\begin{aligned}
\mathsf{active}(l) &\to \mathsf{mark}(r) \quad \text{for all rules } l \to r \text{ in } R\\
\mathsf{tp}(\mathsf{mark}(x)) &\to \mathsf{tp}(\mathsf{chk}(\mathsf{mat}(\overline{p},x)))\\
\mathsf{mat}(f(x_1,\ldots,x_n), f(y_1,\ldots,y_n)) &\to f(y_1,\ldots,\mathsf{mat}(x_i,y_i),\ldots,y_n)\\
&\quad \text{for } f \in \Sigma \text{ of arity } n>0 \text{ in } p,\ 1 \le i \le n\\
\mathsf{mat}(f(x_1,\ldots,x_n), g(y_1,\ldots,y_m)) &\to \mathsf{no}(g(y_1,\ldots,y_m))\\
&\quad \text{for } f,g \in \Sigma,\ f \text{ occurs in } p,\ f \neq g\\
f(x_1,\ldots,\mathsf{no}(x_i),\ldots,x_n) &\to \mathsf{no}(f(x_1,\ldots,x_n))\\
&\quad \text{for } f \in \Sigma \text{ of arity } n>0,\ 1 \le i \le n\\
\mathsf{chk}(\mathsf{no}(f(x_1,\ldots,x_n))) &\to f(\mathsf{chk}(\mathsf{mat}(\overline{p},x_1)),\ldots,\mathsf{chk}(\mathsf{mat}(\overline{p},x_n)))\\
&\quad \text{for } f \in \Sigma \text{ of arity } n>0\\
\mathsf{chk}(\mathsf{no}(c)) &\to \mathsf{active}(c) \quad \text{for } c \in \Sigma \text{ of arity } 0\\
f(\mathsf{active}(x_1),\ldots,\mathsf{active}(x_n)) &\to \mathsf{active}(f(x_1,..,x_n)) \text{ for } f\in\Sigma, \text{ arity } n>0\\
f(\mathsf{active}(x_1),..,\mathsf{mark}(x_i),..,\mathsf{active}(x_n)) &\to \mathsf{mark}(f(x_1,\ldots,x_n))\\
&\quad \text{for } f \in \Sigma \text{ of arity } n>0,\ 1 \le i \le n
\end{aligned}$$

**Theorem 5.1** *Let $p \in \mathcal{T}(\Sigma,\mathcal{V})$ be linear. The relation $\to_G$ is terminating if*

*and only if the TRS $L(R,p)$ is terminating.*

The proof of Thm. 5.1 is given in the appendix. As an example, let $R$ again consist of the rule $\mathsf{f}(x) \to \mathsf{f}(\mathsf{f}(x))$ and let $p = \mathsf{f}^{10}(x)$. We required $\Sigma$ to contain at least one extra constant $\mathsf{c}$ and one extra unary symbol $\mathsf{h}$. To ease the presentation, here we omit $\mathsf{h}$ and let $\Sigma = \{\mathsf{c}, \mathsf{f}\}$. We obtain $\overline{p} = \mathsf{f}^{10}(X)$ where $X$ is a fresh constant. $L(R,p)$ consists of the rules

$$\begin{array}{rlrl}
\mathsf{active}(\mathsf{f}(x)) &\to \mathsf{mark}(\mathsf{f}(\mathsf{f}(x))) & \mathsf{chk}(\mathsf{no}(\mathsf{f}(x))) &\to \mathsf{f}(\mathsf{chk}(\mathsf{mat}(\mathsf{f}^{10}(X), x)))\\
\mathsf{mat}(\mathsf{f}(x), \mathsf{f}(y)) &\to \mathsf{f}(\mathsf{mat}(x,y)) & \mathsf{chk}(\mathsf{no}(\mathsf{c})) &\to \mathsf{active}(\mathsf{c})\\
\mathsf{mat}(\mathsf{f}(x), \mathsf{c}) &\to \mathsf{no}(\mathsf{c}) & \mathsf{f}(\mathsf{active}(x)) &\to \mathsf{active}(\mathsf{f}(x))\\
\mathsf{f}(\mathsf{no}(x)) &\to \mathsf{no}(\mathsf{f}(x)) & \mathsf{f}(\mathsf{mark}(x)) &\to \mathsf{mark}(\mathsf{f}(x))\\
\mathsf{tp}(\mathsf{mark}(x)) &\to \mathsf{tp}(\mathsf{chk}(\mathsf{mat}(\mathsf{f}^{10}(X), x)))
\end{array}$$

Now the step $\mathsf{f}(\mathsf{c}) \to_G \mathsf{f}(\mathsf{f}(\mathsf{c}))$ transforms to the following reduction in $L(R,p)$:

$$\begin{array}{llll}
\mathsf{tp}(\mathsf{mark}(\mathsf{f}(\mathsf{c}))) & \to \mathsf{tp}(\mathsf{chk}(\mathsf{mat}(\mathsf{f}^{10}(X), \mathsf{f}(\mathsf{c})))) & \to\\
\mathsf{tp}(\mathsf{chk}(\mathsf{f}(\mathsf{mat}(\mathsf{f}^{9}(X), \mathsf{c})))) & \to \mathsf{tp}(\mathsf{chk}(\mathsf{f}(\mathsf{no}(\mathsf{c})))) & \to\\
\mathsf{tp}(\mathsf{chk}(\mathsf{no}(\mathsf{f}(\mathsf{c})))) & \to \mathsf{tp}(\mathsf{f}(\mathsf{chk}(\mathsf{mat}(\mathsf{f}^{10}(X), \mathsf{c})))) & \to\\
\mathsf{tp}(\mathsf{f}(\mathsf{chk}(\mathsf{no}(\mathsf{c})))) & \to \mathsf{tp}(\mathsf{f}(\mathsf{active}(\mathsf{c}))) & \to\\
\mathsf{tp}(\mathsf{active}(\mathsf{f}(\mathsf{c}))) & \to \mathsf{tp}(\mathsf{mark}(\mathsf{f}(\mathsf{f}(\mathsf{c}))))
\end{array}$$

In general, for a ground term $t \in \mathcal{T}(\Sigma)$, an $L(R,p)$-reduction of $\mathsf{tp}(\mathsf{mark}(t))$ starts with checking whether $p$ matches $t$. If not, then on every position below the root, a $\mathsf{chk}$ is created, after which a similar matching check can be done on the direct subterms. This goes on until leaves are reached. Hence, $\mathsf{active}$ is created only if none of the subterms on the path to the leaf is matched by $p$. Then these $\mathsf{active}$-symbols may move back to the root, while during this process exactly one $R$-step may be applied, changing $\mathsf{active}$ into $\mathsf{mark}$. (Several $R$-steps are impossible, since $\mathsf{mark}$ can only be propagated upwards in terms $f(\mathsf{active}(x_1), ..., \mathsf{mark}(x_i), ..., \mathsf{active}(x_n))$ where only one argument of $f$ has the root $\mathsf{mark}$.) The TRS $L(R,p)$ is designed in such a way that infinite reductions are only possible if this process is repeated infinitely often. In our example one can prove termination of $L(R,p)$ by the dependency pair method [2].

## 6 Conclusion

In this paper, we showed how to define safety and liveness in terms of abstract reduction and rewriting. In particular, liveness is defined by imposing a suitable reduction strategy. We have shown that our definitions are comparable with existing definitions of safety and liveness in the literature [1]. While safety properties can be proved in a local way (Thm. 2.2), for proving liveness properties, it is useful to investigate the connection between liveness and ter-

mination: It turns out that liveness and termination are essentially equivalent. Depending on the form of the liveness condition, different transformations can be given such that termination of the transformed TRS is equivalent to the desired liveness property to be proved. This means that techniques for termination analysis of ordinary TRSs can be used to verify liveness properties.

While the formalization of liveness using abstract reduction was already presented in [11], we extended the results of [11] by regarding safety (Sect. 2), by comparing our notions with the ones of Alpern and Schneider [1] (Sect. 3), and by introducing local liveness (Sect. 4 and 5).

To increase the applicability of this approach, instead of the sound and complete transformation, one may use simpler sound (but incomplete) transformations like $L_s$, since proving termination of $L(R,p)$ can be very hard in general. Then termination of the transformed TRS still implies the desired local liveness property, but not vice versa. In [11] we already presented such techniques for global liveness which permit automated proofs of liveness properties for interesting process protocols (e.g., networks with shared resources and a token ring protocol). In contrast to model checking and related methods, this approach does not require finiteness of the state space. We plan to refine and to develop such approaches further in future work.

# A Appendix: The Proof of Theorem 5.1

We first present auxiliary lemmata required in the proof of Thm. 5.1. Lemma A.1 describes the behavior of $\mathsf{mat}$ and it is the only lemma where the linearity restriction on $p$ is used. As explained in Sect. 5, a similar transformation (with more complicated $\mathsf{mat}$-rules) is also possible for non-linear terms $p$.

**Lemma A.1 (Reductions with $\mathsf{mat}$)** *For $p \in \mathcal{T}(\Sigma,\mathcal{V})$, $p$ linear, $t \in \mathcal{T}(\Sigma)$, we have $\mathsf{mat}(\overline{p},t) \to^+_{L(R,p)} \mathsf{no}(u)$ iff $t = u$ and no substitution $\sigma$ satisfies $p\sigma = t$.*

**Proof.** We apply induction on the structure of $p$. The cases where $p$ is a constant or a variable, or $p$ and $t$ have distinct root symbols follow directly from the analysis of the shape of the rules. For the remaining case $p = f(p_1,\ldots,p_n)$, $t = f(t_1,\ldots,t_n)$ we need the induction hypothesis and the property that for a linear term $p = f(p_1,\ldots,p_n)$ we have $\exists\sigma\forall i : p_i\sigma = t_i \iff \forall i\exists\sigma : p_i\sigma = t_i$. □

The next lemma shows that the rule $\mathsf{tp}(\mathsf{mark}(x)) \to \mathsf{tp}(\mathsf{chk}(\mathsf{mat}(\overline{p},x)))$ is crucial for the termination behavior of $L(R,p)$.

**Lemma A.2 (Termination Behavior of $L(R,p)$)** *Let $L'(R,p) = L(R,p) \setminus \{\mathsf{tp}(\mathsf{mark}(x)) \to \mathsf{tp}(\mathsf{chk}(\mathsf{mat}(\overline{p},x)))\}$. Then $L'(R,p)$ is terminating.*

**Proof.** Let $R_1$ consist of the $\mathsf{mat}$-rules and the rules of the form $f(x_1,\ldots,\mathsf{no}(x_i),\ldots,x_n) \to \mathsf{no}(f(x_1,\ldots,x_n))$ from $L(R,p)$. Termination of $R_1$ can be shown by the RPO [6] with the precedence $\mathsf{mat} > f > \mathsf{no}$ for all $f \in \Sigma$.

We define a filtering $\mathsf{fil}$ where $\mathsf{fil}(t)$ is the normal form with respect to the two rules $\mathsf{no}(x) \to x$ and $\mathsf{mat}(x,y) \to y$. Moreover, let $R_2$ consist of

$$\begin{aligned}
\mathsf{active}(l) &\to \mathsf{mark}(r)\\
\mathsf{chk}(f(x_1,\ldots,x_n)) &\to f(\mathsf{chk}(x_1),\ldots,\mathsf{chk}(x_n))\\
\mathsf{chk}(c) &\to \mathsf{active}(c)\\
f(\mathsf{active}(x_1),\ldots,\mathsf{active}(x_n)) &\to \mathsf{active}(f(x_1,\ldots,x_n))\\
f(\mathsf{active}(x_1),..,\mathsf{mark}(x_i),..,\mathsf{active}(x_n)) &\to \mathsf{mark}(f(x_1,\ldots,x_n))
\end{aligned}$$

for all $l \to r$ in $R$, all $f \in \Sigma$ with arity $n > 0$, and all constants $c \in \Sigma$.

If $t \to_{L'(R,p)} u$ then either $\mathsf{fil}(t) \to_{R_2} \mathsf{fil}(u)$ or both $\mathsf{fil}(t) = \mathsf{fil}(u)$ and $t \to_{R_1} u$. Hence, for termination of $L'(R,p)$ it suffices to prove termination of $R_2$.

For any term $t$ we define the multiset $M(t)$ over the natural numbers as follows: for every path from the root to a leaf of $t$ we count the total number of $\mathsf{active}$ and $\mathsf{chk}$ symbols on this path, and put the resulting number in $M(t)$. For example, if $t = \mathsf{chk}(\mathsf{active}(c), \mathsf{chk}(\mathsf{active}(c)))$ then $M(t) = \{2,3\}$. Now:

$M(t) > M(u)$ if the step $t \to_{R_2} u$ was done by applying a rule of the form $\mathsf{active}(l) \to \mathsf{mark}(r)$ (for a duplicating rule $l \to r$ it is possible that $M(u)$ is obtained from $M(t)$ by replacing one element of $M(t)$ by a number of smaller elements)

$M(t) \geq M(u)$ if the step $t \to_{R_2} u$ was done by applying another rule of $R_2$.

By the well-foundedness of the multiset order we conclude that in an infinite $R_2$-reduction, rules of the form $\mathsf{active}(l) \to \mathsf{mark}(r)$ are applied only finitely many times. Hence it remains to prove termination of the rest of $R_2$:

$$\begin{aligned}
\mathsf{chk}(f(x_1,\ldots,x_n)) &\to f(\mathsf{chk}(x_1),\ldots,\mathsf{chk}(x_n))\\
\mathsf{chk}(c) &\to \mathsf{active}(c)\\
f(\mathsf{active}(x_1),\ldots,\mathsf{active}(x_n)) &\to \mathsf{active}(f(x_1,\ldots,x_n))\\
f(\mathsf{active}(x_1),..,\mathsf{mark}(x_i),..,\mathsf{active}(x_n)) &\to \mathsf{mark}(f(x_1,\ldots,x_n))
\end{aligned}$$

We use RPO with precedence $\mathsf{chk} > f > \mathsf{active} > \mathsf{mark}$ for all $f \in \Sigma$. □

Now we can formulate a lemma which relates $L(R,p)$ to $\to_R$.

**Lemma A.3 (Relationship between $L(R,p)$ and $\to_R$)** *If $t,u \in \mathcal{T}(\Sigma_G \setminus \{\mathsf{tp}\})$ and $\mathsf{chk}(\mathsf{mat}(\overline{p},t)) \to^+_{L(R,p)} \mathsf{mark}(u)$ then $t,u \in \mathcal{T}(\Sigma)$ and $t \to_R u$. Moreover, if $\mathsf{chk}(\mathsf{mat}(\overline{p},t)) \to^+_{L(R,p)} \mathsf{active}(u)$, then $t,u \in \mathcal{T}(\Sigma)$ and $t = u$.*

**Proof.** Since $t$ does not contain $\mathsf{tp}$, in the reduction one can only use rules from $L'(R,p)$ ($L'(R,p)$ is defined as in Lemma A.2). Let $t \gtrsim u$ hold iff $\mathsf{fil}(t)$ $(\to_{R_2} \cup \rhd_\Sigma)^*$ $\mathsf{fil}(u)$. Here, $\mathsf{fil}$ and $R_2$ are defined as in the proof of Lemma A.2 and $\rhd_\Sigma$ is the subterm relation for symbols from $\Sigma$, i.e., it is the rewrite relation of the TRS with the rules $f(x_1,\ldots,x_i,\ldots,x_n) \to x_i$ for all $f \in \Sigma$. Let $\succ = \gtrsim \setminus \precsim$ be the strict part of $\gtrsim$. The well-foundedness of $\succ$ follows from the well-foundedness of $R_2$, which was shown in the proof of Lemma A.2.

We use induction on $\succ$. When reducing $\mathsf{chk}(\mathsf{mat}(\overline{p},t))$ to $\mathsf{mark}(u)$ or $\mathsf{active}(u)$, first some reductions may take place inside $t$. Hence, $t$ is reduced to $t'$ with $t \to^*_{L'(R,p)} t'$. As shown in the proof of Lemma A.2, this implies $\mathsf{fil}(t) \to^*_{R_2} \mathsf{fil}(t')$ and hence, $t \gtrsim t'$. Then, the redex $\mathsf{mat}(\overline{p},t')$ is reduced.

Hence, $t' = f(t_1, \ldots, t_n)$ and $f \in \Sigma$ (otherwise no $\mathsf{mat}$-rule is applicable).

If $f \neq \mathrm{root}(\overline{p})$ and $f$ is a constant (i.e., $n = 0$), then we have

$$\begin{aligned}\mathsf{chk}(\mathsf{mat}(\overline{p}, t)) &\to^*_{L'(R,p)} \mathsf{chk}(\mathsf{mat}(\overline{p}, t'))\\ &= \quad \mathsf{chk}(\mathsf{mat}(\overline{p}, f))\\ &\to_{L'(R,p)} \mathsf{chk}(\mathsf{no}(f))\\ &\to_{L'(R,p)} \mathsf{active}(f)\end{aligned}$$

and $\mathsf{active}(f) \to_{L'(R,p)} \mathsf{mark}(r)$ if $f \to r \in R$, which proves the lemma.

Now we regard the case where $f \neq \mathrm{root}(\overline{p})$ and $n > 0$. Here we have

$$\begin{aligned}\mathsf{chk}(\mathsf{mat}(\overline{p}, t)) &\to^*_{L'(R,p)} \mathsf{chk}(\mathsf{mat}(\overline{p}, t'))\\ &= \quad \mathsf{chk}(\mathsf{mat}(\overline{p}, f(t_1, \ldots, t_n)))\\ &\to_{L'(R,p)} \mathsf{chk}(\mathsf{no}(f(t_1, \ldots, t_n)))\\ &\to^+_{L'(R,p)} f(\mathsf{chk}(\mathsf{mat}(\overline{p}, t'_1)), \ldots, \mathsf{chk}(\mathsf{mat}(\overline{p}, t'_n)))\end{aligned}$$

with $t_i \to^*_{L'(R,p)} t'_i$. This term can only rewrite to $\mathsf{mark}(u)$ if there is a $j$ such that $\mathsf{chk}(\mathsf{mat}(\overline{p}, t'_j))$ rewrites to some $\mathsf{mark}(u_j)$ and $\mathsf{chk}(\mathsf{mat}(\overline{p}, t'_i))$ rewrites to a term of the form $\mathsf{active}(u_i)$ for all $i \neq j$. Similarly, $f(\mathsf{chk}(\mathsf{mat}(\overline{p}, t'_1)), \ldots, \mathsf{chk}(\mathsf{mat}(\overline{p}, t'_n)))$ can only rewrite to $\mathsf{active}(u)$ if all $\mathsf{chk}(\mathsf{mat}(\overline{p}, t'_i))$ rewrite to terms of the form $\mathsf{active}(u_i)$. Since $t \succsim t' \succ t_i \succsim t'_i$ for all $i$, we can apply the induction hypothesis which implies the lemma.

Finally, if $f = \mathrm{root}(\overline{p})$ then we have

$$\begin{aligned}\mathsf{chk}(\mathsf{mat}(\overline{p}, t')) = &\quad \mathsf{chk}(\mathsf{mat}(\overline{p}, f(t_1, \ldots, t_n)))\\ \to_{L'(R,p)} &\ \mathsf{chk}(f(t_1, \ldots, \mathsf{mat}(\overline{p}, t_i), \ldots, t_n)).\end{aligned}$$

Reducing a $t_j$ with $j \neq i$ to a term $\mathsf{no}(u_j)$ and then lifting this $\mathsf{no}$ on top of $f$ would lead to a term $\mathsf{chk}(\mathsf{no}(f(\ldots, t'_i, \ldots))) \to_{L'(R,p)} f(\ldots, \mathsf{chk}(\mathsf{mat}(\overline{p}, t'_i)), \ldots)$ with $t'_i \notin \mathcal{T}(\Sigma)$. Note that $t \succsim t' \succ t_i \succsim t'_i$. By the induction hypothesis, the term $\mathsf{chk}(\mathsf{mat}(\overline{p}, t'_i))$ cannot reduce to a term of the form $\mathsf{mark}(u_i)$ or $\mathsf{active}(u_i)$ and thus, the whole term cannot reduce to $\mathsf{mark}(u)$ or $\mathsf{active}(u)$.

So $\mathsf{mat}(\overline{p}, t_i) \to^+_{L'(R,p)} \mathsf{no}(u_i)$ for some $u_i$ and the reduction continues with:

$$\begin{aligned}&\mathsf{chk}(f(t_1, \ldots, \mathsf{mat}(\overline{p}, t_i), \ldots, t_n)) \to^+_{L'(R,p)}\\ &f(\mathsf{chk}(\mathsf{mat}(\overline{p}, t'_1)), \ldots, \mathsf{chk}(\mathsf{mat}(\overline{p}, u'_i)), \ldots \mathsf{chk}(\mathsf{mat}(\overline{p}, t'_n)))\end{aligned}$$

where $t_j \to^*_{L'(R,p)} t'_j$ and $u_i \to^*_{L'(R,p)} u'_i$. Note that $t \succsim t' \succ t_j \succsim t'_j$ for $j \neq i$ and $t \succsim t' \succ t_i \succsim u'_i$. The reason for $t_i \succsim u'_i$ is that $\mathsf{mat}(\overline{p}, t_i) \to^*_{L'(R,p)} \mathsf{no}(u_i)$ implies $\mathsf{fil}(t_i) = \mathsf{fil}(\mathsf{mat}(\overline{p}, t_i)) \to^*_{R_2} \mathsf{fil}(\mathsf{no}(u_i)) = \mathsf{fil}(u_i)$. So similar to the case where $f \neq \mathrm{root}(\overline{p})$, now the conjecture follows from the induction hypothesis.□

Now we can formulate a lemma on the relationship between $L(R, p)$ and $\to_G$.

**Lemma A.4 (Relating $L(R, p)$ and $\to_G$)** *Let $p \in \mathcal{T}(\Sigma, \mathcal{V})$ be linear. For*

*ground terms $t$ over $\Sigma$ we have $t \to_G^+ u$ iff $\mathsf{tp}(\mathsf{mark}(t)) \to_{L(R,p)}^+ \mathsf{tp}(\mathsf{mark}(u))$.*

**Proof.** For "only if", we must show that $\mathsf{tp}(\mathsf{mark}(t)) \to_{L(R,p)}^+ \mathsf{tp}(\mathsf{mark}(u))$ if $t \to_G u$. By the assumption $t \to_G u$ we know $t \notin G$, hence $\mathsf{mat}(\overline{p}, t') \to_{L(R,p)}^+ \mathsf{no}(t')$ for subterms $t'$ of $t$ (Lemma A.1). Now the reduction $\mathsf{tp}(\mathsf{mark}(t)) \to_{L(R,p)}^+ \mathsf{tp}(\mathsf{mark}(u))$ can easily be constructed by propagating the $\mathsf{chk}$-symbols to the leaves until all leaves $c$ are replaced by $\mathsf{active}(c)$. Then these $\mathsf{active}$-symbols can be propagated back to the root. During this propagation the redex from the reduction $t \to_G u$ is reduced and the corresponding $\mathsf{active}$-symbol is replaced by a $\mathsf{mark}$-symbol. Hence, we end up in $\mathsf{tp}(\mathsf{mark}(u))$.

For the converse, since $t \in \mathcal{T}(\Sigma)$, we must have $\mathsf{tp}(\mathsf{mark}(t)) \to_{L(R,p)} \mathsf{tp}(\mathsf{chk}(\mathsf{mat}(\overline{p}, t))) \to_{L(R,p)}^+ \mathsf{tp}(\mathsf{mark}(u))$. Hence, Lemma A.3 implies $t \to_R u$.

We still need to prove that $t \notin G$. Note that $\mathsf{chk}(\mathsf{mat}(\overline{p}, t)) \to_{L(R,p)}^+ \mathsf{mark}(u)$ implies that $\mathsf{chk}$ is propagated downwards to the leaves of $t$. By Lemma A.1 this implies that no subterm of $t$ is matched by $p$. Thus, the $\mathsf{active}(\ldots) \to \mathsf{mark}(\ldots)$ step done in the reduction $\mathsf{chk}(\mathsf{mat}(\overline{p}, t)) \to_{L(R,p)}^+ \mathsf{mark}(u)$ can also be done with $\to_G$. Note that in the reduction $\mathsf{chk}(\mathsf{mat}(\overline{p}, t)) \to_{L(R,p)}^+ \mathsf{mark}(u)$ we perform exactly one such step (if one would perform another $\mathsf{active}(\ldots) \to \mathsf{mark}(\ldots)$ step, then $\mathsf{mark}$ could no longer be propagated to the top since the rules

$$f(\mathsf{active}(x_1), \ldots, \mathsf{mark}(x_i), \ldots, \mathsf{active}(x_n)) \to \mathsf{mark}(f(x_1, \ldots, x_n))$$

can only be applied if there is exactly one occurrence of $\mathsf{mark}$). □

Now we prove Thm. 5.1: $\to_G$ is terminating iff the TRS $L(R, p)$ is terminating.

**Proof.** We first show the "if"-direction. If $\to_G$ were not terminating, then there would be an infinite reduction

$$t_0 \to_G t_1 \to_G \ldots$$

of ground terms over $\Sigma$. (We extended the signature such that ground termination already implies termination of $\to_G$.) By Lemma A.4 this would imply

$$\mathsf{tp}(\mathsf{mark}(t_0)) \to_{L(R,p)}^+ \mathsf{tp}(\mathsf{mark}(t_1)) \to_{L(R,p)}^+ \ldots$$

in contradiction to the termination of $L(R, p)$.

For "only if", assume that $L(R, p)$ is not terminating. Then by type introduction [12,15] one can show that there is an infinite $L(R, p)$-reduction where every term has $\mathsf{tp}$ as root, whereas $\mathsf{tp}$ does not occur below the root. Due to Lemma A.2 the reduction contains infinitely many applications of the rule $\mathsf{tp}(\mathsf{mark}(x)) \to \mathsf{tp}(\mathsf{chk}(\ldots))$. Hence, the reduction has the form

$$\mathsf{tp}(\mathsf{mark}(t_0)) \to_{L(R,p)} \mathsf{tp}(\mathsf{chk}(\mathsf{mat}(\overline{p}, t_0))) \to_{L(R,p)}^+$$
$$\mathsf{tp}(\mathsf{mark}(t_1)) \to_{L(R,p)} \mathsf{tp}(\mathsf{chk}(\mathsf{mat}(\overline{p}, t_1))) \to_{L(R,p)}^+ \ldots$$

By Lemma A.3 this implies $t_0, t_1, \ldots \in \mathcal{T}(\Sigma)$. Thus, Lemma A.4 implies

$$t_0 \to_G^+ t_1 \to_G^+ \ldots$$

in contradiction to the termination of $\rightarrow_G$. □

## References


[1] Alpern, B. and F. B. Schneider, *Defining liveness*, Information Processing Letters **21** (1985), pp. 181–185.

[2] Arts, T. and J. Giesl, *Termination of term rewriting using dependency pairs*, Theoretical Computer Science **236** (2000), pp. 133–178.

[3] Baader, F. and T. Nipkow, "Term Rewriting and All That," Cambridge University Press, 1998.

[4] Bouajjani, A., *Languages, rewriting systems, and verification of infinite-state systems*, in: *Proc. ICALP '01*, LNCS 2076, 2001, pp. 24–39.

[5] Caucal, D., *On the regular structure of prefix rewriting*, Theoretical Computer Science **106** (1992), pp. 61–86.

[6] Dershowitz, N., *Termination of rewriting*, Journal of Symbolic Computation **3** (1987), pp. 69–116.

[7] Fribourg, L. and H. Olsén, *Reachability sets of parametrized rings as regular languages*, in: *Proc. INFINITY '97*, ENTCS 9, 1997.

[8] Giesl, J. and T. Arts, *Verification of Erlang processes by dependency pairs*, Appl. Algebra in Engineering, Communication and Computing **12** (2001), pp. 39–72.

[9] Giesl, J. and A. Middeldorp, *Transforming context-sensitive rewrite systems*, in: *Proc. 10th RTA*, LNCS 1631, 1999, pp. 271–285.

[10] Giesl, J. and A. Middeldorp, *Transformation techniques for context-sensitive rewrite systems*, Journal of Functional Programming (2003), to appear. Preliminary extended version appeared as Technical Report AIB-2002-02, RWTH Aachen, Germany, `http://aib.informatik.rwth-aachen.de`.

[11] Giesl, J. and H. Zantema, *Liveness in rewriting*, in: *Proc. 14th RTA*, LNCS 2706, 2003, pp. 321–336.

[12] Middeldorp, A. and H. Ohsaki, *Type introduction for equational rewriting*, Acta Informatica **36** (2000), pp. 1007–1029.

[13] Moller, F., *Infinite results*, in: *Proc. CONCUR '96*, LNCS 1119, 1996, pp. 195–216.

[14] van Gasteren, A. and G. Tel, *Comments on "On the proof of a distributed algorithm": always-true is not invariant*, Information Processing Letters **35** (1990), pp. 277–279.

[15] Zantema, H., *Termination of term rewriting: Interpretation and type elimination*, Journal of Symbolic Computation **17** (1994), pp. 23–50.