# Liveness Checking as Safety Checking for Infinite State Spaces

Viktor Schuppan [1]

*ETH Zürich, Computer Systems Institute, CH-8092 Zürich, Switzerland*

Armin Biere [2]

*Johannes Kepler University, Institute for Formal Models and Verification*
*Altenbergerstrasse 69, A-4040 Linz, Austria*

**Abstract**

In previous work we have developed a syntactic reduction of repeated reachability to reachability for finite state systems. This may lead to simpler and more uniform proofs for model checking of liveness properties, help to find shortest counterexamples, and overcome limitations of closed-source model-checking tools. In this paper we show that a similar reduction can be applied to a number of infinite state systems, namely, ($\omega-$)regular model checking, push-down systems, and timed automata.

*Key words:* liveness, safety, linear temporal logic, model checking, infinite state space

## 1 Introduction

While model checking of safety properties can be reduced to computing the set of reachable states of a system [22], verification of general LTL properties is typically performed by searching for (infinite) fair paths in the product of the system and an automaton representing the property [33].

In previous work [7,26] we developed a syntactic reduction from computing fair repeated reachability to computing reachability for finite state systems. This reduction has been used to develop a BDD-based method to find shortest counterexamples to linear time properties [27]. On selected examples a significant speed up compared to traditional liveness checking can be observed [26,29]. It may also discourage tool vendors from charging separately for liveness-enabled versions of their verification tools. From a theoretical point

[1] Email: vschuppan@acm.org
[2] Email: biere@jku.at

of view, the reduction gives a “quick and dirty” algorithm to verify liveness properties. It can help to simplify proofs if a proof for safety properties is easier than the corresponding proof for general LTL properties. It may finally be seen as a continuation of existing work to explore the limits of reachability checking [22,3].

In this paper we develop similar reductions for a number of infinite state systems. Classes of infinite state systems, which have received considerable attention in the past and for which verification tools are available (e.g., [1,17,23]), are ($\omega-$)regular model checking [20,34,12,9], pushdown systems [10,18,16], and timed automata [4].

Bouajjani et al. independently used the same reduction to verify liveness properties in regular model checking [11]. They only sketch the reduction. No complexity results are given and timed automata are not discussed.

Shilov et al. present a game-theoretic reduction from their Second Order Elementary Propositional Dynamic Logic (SOEPDL) [30] to reachability for classes of models which include all finite models and which are closed under Cartesian product and power set [31,30]. SOEPDL is more expressive than Stirling’s second order propositional model logic 2M [32] (i.e., it subsumes LTL, CTL, and the propositional $\mu$-calculus [21]). While the reduction by Shilov et al. is more powerful than our reduction if their prerequisites are satisfied, in terms of number of configurations, [31] is doubly exponential where ours is typically quadratic. In the words of [31], this renders it “totally non-efficient, impractical”. The reduction [31] applies in principle also to infinite states systems but no concrete examples are given.

Aceto et al. developed a specification language for timed systems and proved for a subset that it can express precisely those properties that can be checked by reachability in the timed system composed with a test automaton (basically, a timed automaton with designated bad locations) [3].

Early work on liveness for regular model checking includes [12,24]. Pnueli and Shahar, too, use a copy of a current state to detect bad cycles in parameterized systems [24]. Rather than transforming the model syntactically they use the copy as part of a dedicated liveness checking algorithm. A variant of LTL geared towards parameterized systems is proposed in [2]. [13] gives details on how to encode a broader set of properties than [2] for ($\omega-$)regular model checking, which can be used in conjunction with our reduction. Algorithms to compute repeated reachability, on which we also base our reductions, can be found for pushdown systems, e.g., in [10], and for timed automata in [4].

For related work using finite state systems see [25].

After some notation common to all classes of systems in Sect. 2, Sect. 3 presents the basic idea of our reduction using finite state systems as an example. It is extended to ($\omega-$)regular model checking in Sect. 4 and to pushdown system in Sect. 5. The construction for timed automata is technically involved and only sketched in Sect. 6. Details can be found in [25]. The last section concludes.

## 2 Common Notation

The set of Booleans is denoted by $\mathbb{B} = \{0, 1\}$; $\mathbb{N}$ and $\mathbb{R}$ are naturals and reals, respectively. Elements of a tuple are separated by commas. Elements of a sequence typically have no operator between them, $\circ$ is used only if ambiguity might arise. For a sequence $\rho$, $\rho(i)$ denotes the $i$-th element of the sequence (starting with $\rho(0)$). The length of a sequence, $|\rho|$, is defined as the number of its elements. If $S$ is a set, $S^*$ and $S^\omega$ are the sets of finite and infinite sequences of elements of $S$.

We introduce an operator $\mu$, which forms a sequence of tuples from a tuple of sequences. Given two words $v, w \in \Sigma^*$ with $|v| \leq |w|$ we define $\mu(v, w) = (v(0), w(0)) \dots (v(|v| - 1), w(|v| - 1)) \in (\Sigma \times \Sigma)^*$.

## 3 Liveness Checking as Safety Checking – Finite Case

In this section we briefly restate the main results from [26] to explain the basic idea and notation of our reduction.

### *3.1 Preliminaries*

Let $AP$ be a finite set of atomic propositions. A *fair Kripke structure*, see, e.g., [14], is a five tuple $M = (S, S_0, R, L, F)$ where $S$ is a finite set of *states*, $S_0 \subseteq S$ is the set of *initial states*, $R \subseteq S \times S$ is a *transition relation*, $L : S \mapsto 2^{AP}$ is a *labeling* of the states, and $F \subseteq S$ is a set of *fair states*.

A *run* is a (finite or infinite) sequence of states $\rho = \rho(0)\rho(1)\dots$ where $\forall 0 < i < |\rho| \, . \, (\rho(i-1), \rho(i)) \in R$. $\rho$ is *initialized* if $\rho(0) \in S_0$; it is *fair* if $\mathit{inf}(\rho) \cap F \neq \emptyset$, where $\mathit{inf}(\rho)$ is the set of states occuring infinitely often in $\rho$. $\mathit{Runs}(M)$ denotes the set of runs of $M$.

### *3.2 Reduction*

A liveness property $\mathbf{F}p$, where $p$ is propositional, is violated in a finite state system iff there exists a lasso-shaped run such that $p$ never holds on that run. Finding such loop is a key ingredient of many model checking algorithms for LTL, e.g., [33,8]. Our reduction integrates the detection of a *fair* loop into the model to be verified. It first nondeterministically saves the current state, i.e., it guesses a potential loop start. Next, it watches out for a fair state. Once that has been seen, it waits for a second occurrence of the saved state to conclude that a fair loop has been closed.

The reduction extends a state $s$ in the original model with a component $\hat{s}$ to store a previously seen state, a flag $f$ to remember the occurence of a $f$air state, and a flag $lo$ ($l$ass$o$) to indicate the position on the presumed lasso. $lo$ has the value $st$ for *st*em on the stem. When $lo$ nondeterministically changes its value to $lb$ ($l$oop $b$ody) the value of $s$ is stored in $\hat{s}$. Note, that once a value has been saved in $\hat{s}$, it cannot be overwritten. When $lo = lb$, occurence

**Definition 3.1** Let $M = (S, S_0, R, L, F)$ be a fair Kripke structure with $\hat{s}_0 \in S$ arbitrary but fixed. Then $M^{\mathbf{S}} = (S^{\mathbf{S}}, {S_0}^{\mathbf{S}}, R^{\mathbf{S}}, L^{\mathbf{S}}, F^{\mathbf{S}})$ is defined as:

$$
\begin{array}{lll}
S^{\mathbf{S}} & = S \times S \times \{st, lb, lc\} \times \mathbb{B} & \\
{S_0}^{\mathbf{S}} & = \{(s_0, \hat{s}_0, st, 0) \mid s_0 \in S_0\} \cup & \\
 & \quad \{(s_0, s_0, lb, f) \mid s_0 \in S_0 \wedge (f \rightarrow s_0 \in F)\} & \\
R^{\mathbf{S}} & = \{((s, \hat{s}, lo, f), (s', \hat{s}', lo', f')) \mid (s, s') \in R \wedge & \\
 & \qquad ((lo = st \wedge lo' = st \wedge \neg f \wedge \neg f' \wedge \hat{s} = \hat{s}' = \hat{s}_0) \vee & (1) \\
 & \qquad\ (lo = st \wedge lo' = lb \wedge \neg f \wedge (f' \rightarrow s' \in F) \wedge \hat{s} = \hat{s}_0 \wedge s' = \hat{s}') \vee & (2) \\
 & \qquad\ (lo = lb \wedge lo' = lb \wedge (f \rightarrow f') \wedge (f' \rightarrow f \vee s' \in F) \wedge \hat{s} = \hat{s}') \vee & (3) \\
 & \qquad\ (lo = lb \wedge lo' = lc \wedge f \wedge f' \wedge \hat{s} = s' = \hat{s}') \vee & (4) \\
 & \qquad\ (lo = lc \wedge lo' = lc \wedge f \wedge f' \wedge \hat{s} = \hat{s}'))\} & (5) \\
L^{\mathbf{S}}(s^{\mathbf{S}}) & = L(s) \text{ where } s^{\mathbf{S}} = (s, \hat{s}, lo, f) & \\
F^{\mathbf{S}} & = \emptyset &
\end{array}
$$

of a fair state may be recorded in $f$. The value of $lo$ may finally change to $lc$ (*l*oop *c*losed) when $f$ is true and a second occurence of the stored state is detected.

Definition 3.1 shows the construction. The transitions of $R^{\mathbf{S}}$ are partitioned into five subsets. Subset (1) covers the case when no state has been saved so far. Saving happens either at the initial state or via a transition from set (2). Transitions from the third set (3) are taken as long as no second occurrence of the stored state has been seen. A second occurrence is finally detected by a transition in (4). After that only transitions from the last set (5) can be taken.

**Theorem 3.2** *Let $M = (S, S_0, R, L, F)$ be a fair Kripke structure, let $M^{\mathbf{S}}$ be defined as above. Then there is a lasso-shaped initialized fair run $\rho = (s_0 \dots s_{l-1}) \circ (s_l \dots s_m \dots s_{k-1})^\omega$ with $k > m \geq l \geq 0$ and $s_m \in F$ in $Runs(M)$ iff there is a reachable state $s^{\mathbf{S}} = (s, \hat{s}, lc, f)$ in $M^{\mathbf{S}}$.*

**Proof.** We prove the following biimplications from top to bottom:

$$
\begin{gathered}
\exists \rho = (s_0 \dots s_{l-1})(s_l \dots s_m \dots s_{k-1})^\omega \in Runs(M) \\
\text{with } k > m \geq l \geq 0 \wedge s_m \in F \\
\Leftrightarrow \\
\exists \rho^{\mathbf{S}} = (s_0, \hat{s}_0, st, 0) \dots (s_{l-1}, \hat{s}_0, st, 0)(s_l, s_l, lb, 0) \dots (s_{m-1}, s_l, lb, 0) \circ \\
(s_m, s_l, lb, 1) \dots (s_{k-1}, s_l, lb, 1)(s_k, s_l, lc, 1) \in Runs(M^{\mathbf{S}}) \text{ with } k > m \geq l \geq 0 \\
\Leftrightarrow \\
\exists s^{\mathbf{S}} = (s, \hat{s}, lc, f) \in S^{\mathbf{S}} \text{ that is reachable in } M^{\mathbf{S}}
\end{gathered}
$$

(i) "$\Rightarrow$": Let $\rho = (s_0 \dots s_{l-1})(s_l \dots s_m \dots s_{k-1})^\omega$ be an initialized fair run in

$M$ with $k > m \geq l \geq 0$ and $s_m \in F$. We construct $\rho^{\mathbf{S}}$ as follows.

If $l > 0$ choose $\rho^{\mathbf{S}}(0) = (s_0, \hat{s}_0, st, 0)$. Construct $(s_0, \hat{s}_0, st, 0) \dots (s_{l-1}, \hat{s}_0, st, 0)$ by taking transitions from subset (1). Assume first that $m > l$. Proceed to $(s_l, s_l, lb, 0)$ via a transition from (2), continue via $(s_{m-1}, s_l, lb, 0)$ to $(s_m, s_l, lb, 1)$ and then to $(s_{k-1}, s_l, lb, 1)$ with $k - l - 1$ transitions from (3). As $k > l$ there is a transition from (4) to $(s_k, s_l, lc, 1)$ with $s_k = s_l$. If $m = l$, modify the target state of the transition from (2) to be $(s_l, s_l, lb, 1)$ and continue to $(s_{k-1}, s_l, lb, 1)$ and from there to $(s_k, s_l, lc, 1)$, again with $s_k = s_l$.

Otherwise, if $l = 0$, start with $(s_0, s_0, lb, 0)$ if $m > l$ and $(s_0, s_0, lb, 1)$ if $m = l$ and continue with $k - 1$ transitions from (3) and one from (4) as before.

"$\Leftarrow$": Let

$$\rho^{\mathbf{S}} = (s_0, \hat{s}_0, st, 0) \dots (s_{l-1}, \hat{s}_0, st, 0)(s_l, s_l, lb, 0) \dots (s_{m-1}, s_l, lb, 0) \circ \\ (s_m, s_l, lb, 1) \dots (s_{k-1}, s_l, lb, 1)(s_k, s_l, lc, 1)$$

be a run in $M^{\mathbf{S}}$ such that $k > m \geq l \geq 0$. From the construction of $M^{\mathbf{S}}$, $\rho' = s_0 \dots s_{l-1} s_l \dots s_{m-1} s_m \dots s_{k-1} s_k$ is a finite initialized run in $M$ with $s_k = s_l$ and $s_m \in F$. Hence, $\rho = (s_0 \dots s_{l-1})(s_l \dots s_m \dots s_{k-1})^\omega$ is an initialized fair run in $M$ as desired.

(ii) "$\Rightarrow$": Obvious.

"$\Leftarrow$": Let $s^{\mathbf{S}} = (s, \hat{s}, lc, f) \in S^{\mathbf{S}}$ be a reachable state in $M^{\mathbf{S}}$, By definition of $M^{\mathbf{S}}$, $f$ is 1. Further, there is an initialized run $\rho^{\mathbf{S}\prime}$ ending in $s^{\mathbf{S}}$. According to the definition of $R^{\mathbf{S}}$, $\rho^{\mathbf{S}\prime}$ takes precisely one transition from subset (4). Let $\rho^{\mathbf{S}}$ be the prefix of $\rho^{\mathbf{S}\prime}$ up to the target state of that transition. Let $k = |\rho^{\mathbf{S}}| - 1$. Clearly, $k > 0$. Let $\rho^{\mathbf{S}}(k) = (s_k, s_k, lc, 1)$. By definition of $R^{\mathbf{S}}$ there exists $0 \leq m' < k$ such that $\forall m' \leq i < k \ . \ \rho^{\mathbf{S}}(i) = (s_i, s_k, lb, 1)$ with $s_{m'} \in F$. Choose $m$ to be the smallest such $m'$.

**Case 1** $m = 0$: With $l = 0$ and the definition of ${S_0}^{\mathbf{S}}$ and $R^{\mathbf{S}}$ we have that $\rho^{\mathbf{S}}(0) = (s_0, s_k, lb, 1) = (s_0, s_0, lb, 1)$.

**Case 2** $m > 0 \wedge \rho^{\mathbf{S}}(m-1) = (s_{m-1}, s_k, st, 0)$: Set $l = m$. By definition of $R^{\mathbf{S}}$, $\forall 0 \leq i < l \ . \ \rho^{\mathbf{S}}(i) = (s_i, \hat{s}_0, st, 0)$ and $\rho^{\mathbf{S}}(l) = (s_l, s_k, lb, 1) = (s_l, s_l, lb, 1)$.

**Case 3** $m > 0 \wedge \rho^{\mathbf{S}}(m-1) = (s_{m-1}, s_k, lb, 0)$: By definition of $R^{\mathbf{S}}$ there is $0 \leq l' < m$ such that $\forall l' \leq i < m \ . \ \rho^{\mathbf{S}}(i) = (s_i, s_k, lb, 0)$. Set $l$ to the smallest such $l'$.

**Case 3.1** $l = 0$: By def. of ${S_0}^{\mathbf{S}}$, $\rho^{\mathbf{S}}(0) = (s_0, s_k, lb, 0) = (s_0, s_0, lb, 0)$.

**Case 3.2** $l > 0$: From the definition of $R^{\mathbf{S}}$, $\forall 0 \leq i < l \ . \ \rho^{\mathbf{S}}(i) = (s_i, \hat{s}_0, st, 0)$ and $\rho^{\mathbf{S}}(l) = (s_l, s_k, lb, 0) = (s_l, s_l, lb, 0)$.

In all cases $s_k = s_l$ and the $\rho^{\mathbf{S}}$ has the desired shape.

□

### *3.3 Complexity*

Intuitively, $M^{\mathbf{S}}$ consists of $|S|$ parallel copies of $M$. Hence, we immediately have the following result.

**Proposition 3.3** *Let $M = (S, S_0, R, L, F)$ be a fair Kripke structure. $M^{\mathbf{S}}$ has $\mathbf{O}(|S|^2)$ states and $\mathbf{O}(|S||R|)$ transitions.*

**Proof.** Each state in $S^{\mathbf{S}}$ stores, in addition to the original state $s$, another state $\hat{s}$ and the flags $f$ and $lo$. $R^{\mathbf{S}}$ contains $\mathbf{O}(|S|)$ transitions $t^{\mathbf{S}}$ per transition $t \in R$ in subsets (3) and (5), and $\mathbf{O}(1)$ $t^{\mathbf{S}}$ per $t \in R$ in subsets (1), (2) and (4). $\square$

As reachability in a Kripke structure can be determined in $\mathbf{O}(|S| + |R|)$ time and $\mathbf{O}(|S|)$ space, $S^{\mathbf{S}}$ can be checked in $\mathbf{O}(|S|^2 + |S||R|)$ time and $\mathbf{O}(|S|^2)$ space. For results on other parameters that are important when using BDD-based symbolic model checking, e.g., radius, diameter, or BDD size, see [26,25].

### *3.4 Shortest Lasso-Shaped Counterexamples*

When performing reachability analysis on $M^{\mathbf{S}}$, the algorithm will either reach a fixed point or find a counterexample from which a lasso-shaped counterexample in $M$ can be derived. Moreover, if the property under consideration is false and if breadth-first search is used for reachability analysis in $M^{\mathbf{S}}$, the proof of Thm. 3.2 implies that a shortest lasso-shaped counterexample in $M$ (i.e., with respect to the product of the automaton for the property and that for the original system to be verified) can be derived. If a tight Büchi automaton [27] is used to encode the property and multiple fairness constraints are encoded in a suitable way (e.g., [25]), this implies that the counterexample is a shortest one with respect to the property in the original model to be verified. Note, that the translated system needs one step to detect a loop. Hence, when lasso-shaped counterexamples and violating prefixes [22] are searched for in parallel, and search is stopped after finding the first counterexample, a reported shortest violating prefix may be one state longer than the length of a potential shortest lasso-shaped counterexample.

## 4 Regular Model Checking

### *4.1 Preliminaries*

The notation in this section is mostly borrowed from [12]. Let $\Sigma$ be a finite alphabet. Regular sets (respectively relations) can be represented as finite-state automata (resp. transducers). These are given as four tuple $(Q, q_0, \delta, F)$ where $Q$ is a finite set of states, $q_0$ is the initial state, $\delta : (Q \times \Sigma) \mapsto 2^Q$ (resp. $\delta : (Q \times (\Sigma \times \Sigma)) \mapsto 2^Q$) is the transition function, and $F \subseteq Q$ is the set of accepting states.

**Definition 4.1** Let $\mathcal{P} = (\Sigma, \Phi_I, R)$ be a program with $\hat{a}_0 \in \Sigma$ arbitrary but fixed. Then $\mathcal{P}^{\mathbf{S}} = (\Sigma^{\mathbf{S}}, {\Phi_I}^{\mathbf{S}}, R^{\mathbf{S}})$ is defined as

$$\begin{aligned}
\Sigma^{\mathbf{S}} &= \{st, lb, lc\} \cup (\Sigma \times \Sigma) \\
{\Phi_I}^{\mathbf{S}} &= st \circ \{\mu(w, \hat{w}) \in (\Sigma \times \Sigma)^* \mid |w| = |\hat{w}| \wedge w \in \Phi_I \wedge \hat{w} = {\hat{a}_0}^*\} \cup \\
&\quad\; lb \circ \{\mu(w, w) \in (\Sigma \times \Sigma)^* \mid w \in \Phi_I\} \\
R^{\mathbf{S}} &= \{((lo\ \mu(w, \hat{w})), (lo'\ \mu(w', \hat{w}'))) \subseteq (\{st, lb, lc\} \circ (\Sigma \times \Sigma)^*)^2 \mid \\
&\qquad |w| = |\hat{w}| = |w'| = |\hat{w}'| \wedge (w, w') \in R \wedge \\
&\qquad ((lo = st \wedge lo' = st \wedge \hat{w} = \hat{w}' = {\hat{a}_0}^*) \vee && (1) \\
&\qquad\; (lo = st \wedge lo' = lb \wedge \hat{w} = {\hat{a}_0}^* \wedge w' = \hat{w}') \vee && (2) \\
&\qquad\; (lo = lb \wedge lo' = lb \wedge \hat{w} = \hat{w}') \vee && (3) \\
&\qquad\; (lo = lb \wedge lo' = lc \wedge \hat{w} = w' = \hat{w}') \vee && (4) \\
&\qquad\; (lo = lc \wedge lo' = lc \wedge \hat{w} = \hat{w}'))\} && (5)
\end{aligned}$$

A relation $R \subseteq \Sigma^* \times \Sigma^*$ is *length-preserving* iff $\forall (w, w') \in R \,.\, |w| = |w'|$. A *program* is a triple $\mathcal{P} = (\Sigma, \Phi_I, R)$ where $\Phi_I \subseteq \Sigma^*$ is a regular set of *initial configurations* and $R \subseteq \Sigma^* \times \Sigma^*$ is a regular, length-preserving *transition relation*. From now on we do not consider fairness. It can be incorporated into the reductions for infinite state systems as in the finite case.

A *configuration* of a program $\mathcal{P}$ is a word $w$ over $\Sigma$. *Runs* are finite or infinite sequences of configurations $\rho = \rho(0)\rho(1)\ldots$, such that $\forall 0 < i < |\rho| \,.\, (\rho(i-1), \rho(i)) \in R$. A run is *initialized* if $\rho(0) \in \Phi_I$. *Runs*$(\mathcal{P})$ is the set of runs of $\mathcal{P}$.

## *4.2 Reduction*

In the finite case the state to be saved was simply added as a separate component to the state of the transformed system. A finite automaton can only remember a finite amount of information. Hence, in order to apply the reduction to regular model checking it is not possible to construct an automaton that first reads a state of the original program and compares that with a saved copy. Instead, we extend the alphabet of the program to tuples of letters to store and compare states position by position of a word. Other than that, the construction in Def. 4.1 is the same as in the finite case. The following Lemma 4.2 shows that the reduced program is still a program. Theorem 4.3 then establishes correctness of the reduction.

**Lemma 4.2** *If $\mathcal{P} = (\Sigma, \Phi_I, R)$ is a program, so is $\mathcal{P}^{\mathbf{S}} = (\Sigma^{\mathbf{S}}, {\Phi_I}^{\mathbf{S}}, R^{\mathbf{S}})$.*

**Proof.** Assume that $\Phi_I$ is given by $(Q_I, q_{0I}, \delta_I, F_I)$. To represent an automaton (not) saving the initial state we use separate copies of $(Q_I, q_{0I}, \delta_I, F_I)$, $(Q_I^{\neq}, {q_0}_I^{\neq}, \delta_I^{\neq}, F_I^{\neq})$ and $(Q_I^{=}, {q_0}_I^{=}, \delta_I^{=}, f_I^{=})$. Then $({Q_I}^{\mathbf{S}}, {q_{0I}}^{\mathbf{S}}, {\delta_I}^{\mathbf{S}}, {F_I}^{\mathbf{S}})$ with

$$\begin{array}{rcl}
{Q_I}^{\mathbf{S}} & = & Q_I^{\neq} \cup Q_I^{=} \cup \{q_{lo}\}, \\
{q_0}_I^{\mathbf{S}} & = & q_{lo}, \\
{\delta_I}^{\mathbf{S}} & = & \{(q_{lo}, st, {q_0}_I^{\neq})\} \cup \{(q^{\neq}, (a, \hat{a}_0), q^{\neq\prime}) \mid (q^{\neq}, a, q^{\neq\prime}) \in \delta_I^{\neq}\} \cup \\
 & & \{(q_{lo}, lb, {q_0}_I^{=})\} \cup \{(q^{=}, (a, a), q^{=\prime}) \mid (q^{=}, a, q^{=\prime}) \in \delta_I^{=}\}, \text{ and} \\
{F_I}^{\mathbf{S}} & = & F_I^{\neq} \cup F_I^{=},
\end{array}$$

is a finite automaton accepting ${\Phi_I}^{\mathbf{S}}$.

Similarly, if $R$ is given by $(Q_R, {q_0}_R, \delta_R, F_R)$, we construct a finite transducer $({Q_R}^{\mathbf{S}}, {q_0}_R^{\mathbf{S}}, {\delta_R}^{\mathbf{S}}, {F_R}^{\mathbf{S}})$ to accept $R^{\mathbf{S}}$. We use separate copies of $(Q_R, {q_0}_R, \delta_R, F_R)$ to leave the saved word unchanged and check for it being ${\hat{a}_0}^*$ (superscript $^1$, corresponding to disjunct 1 in Def. 4.1), save a word (sup. $^2$, corr. to subset (2)), leave the saved word unchanged (sup. $^{35}$, corr. to subsets (3) and (5)), and compare current and stored word (sup. $^4$, corr. to subset (4)).

$$\begin{array}{rcl}
{Q_R}^{\mathbf{S}} & = & Q_R^1 \cup Q_R^2 \cup Q_R^{35} \cup Q_R^4 \cup \{q_{lo}\}, \\
{q_0}_R^{\mathbf{S}} & = & q_{lo}, \\
{\delta_R}^{\mathbf{S}} & = & \{(q_{lo}, (st, st), q_0^1), (q_{lo}, (st, lb), q_0^2), (q_{lo}, (lb, lb), q_0^{35}), \\
 & & \quad (q_{lo}, (lb, lc), q_0^4), (q_{lo}, (lc, lc), q_0^{35})\} \cup \\
 & & \{(q^1, ((a, \hat{a}_0), (a', \hat{a}_0)), q^{1\prime}) \mid (q^1, (a, a'), q^{1\prime}) \in \delta_R^1\} \cup \\
 & & \{(q^2, ((a, \hat{a}_0), (a', a')), q^{2\prime}) \mid (q^2, (a, a'), q^{2\prime}) \in \delta_R^2\} \cup \\
 & & \{(q^{35}, ((a, \hat{a}), (a', \hat{a})), q^{35\prime}) \mid (q^{35}, (a, a'), q^{35\prime}) \in \delta_R^{35}\} \cup \\
 & & \{(q^4, ((a, a'), (a', a')), q^{4\prime}) \mid (q^4, (a, a'), q^{4\prime}) \in \delta_R^4\}, \text{ and} \\
{F_R}^{\mathbf{S}} & = & F_R^1 \cup F_R^2 \cup F_R^{35} \cup F_R^4
\end{array}$$

□

**Theorem 4.3** *Let $\mathcal{P} = (\Sigma, \Phi_I, R)$ be a program, $\mathcal{P}^{\mathbf{S}}$ be defined as above, and $\hat{w}_I \in {\hat{a}_0}^*$ with $|\hat{w}_I| = |w_0|$. Assume $k > l \geq 0$.*

$$(w_0 \ldots w_{l-1})(w_l \ldots w_{k-1})^\omega \in \mathit{Runs}(\mathcal{P})$$

$$\Leftrightarrow$$

$$(st\ \mu(w_0, \hat{w}_I)) \ldots (st\ \mu(w_{l-1}, \hat{w}_I))(lb\ \mu(w_l, w_l)) \ldots (lb\ \mu(w_{k-1}, w_l))(lc\ \mu(w_k, w_l))$$

$$\in \mathit{Runs}(\mathcal{P}^{\mathbf{S}})$$

**Proof.** Analogous to the proof of Thm. 3.2. □

## *4.3 Complexity*

We trivially have

**Proposition 4.4** *Let $\mathcal{P} = (\Sigma, \Phi_I, R)$ be a program, and $\mathcal{P}^{\mathbf{S}}$ be defined as* *above. Then*

$$|Q_I{}^{\mathbf{S}}| = 2|Q_I| + 1 \qquad |\delta_I{}^{\mathbf{S}}| = 2|\delta_I| + 2$$
$$|Q_R{}^{\mathbf{S}}| = 4|Q_R| + 1 \qquad |\delta_R{}^{\mathbf{S}}| = (|\Sigma| + 3)|\delta_R| + 5$$

□

## *4.4 Example*

As an example of a parameterized system, consider token passing as used, e.g., in [12]. An array of processes passes a single token from left to right. Initially, the leftmost process has the token. Each transition either leaves the token where it is, or passes it on to the right neighbour of the current owner. Processes can be in states $t$ or $n$ depending on whether they do ($t$) or don't have ($n$) the token. Hence, $\Sigma = \{n, t\}$. An automaton and a transducer representing the initial states $\Phi_I$ and the transition relation $R$ are shown in Fig. 1 (a) and (b).

According to Def. 4.1, $\Sigma^{\mathbf{S}} = \{st, lb, lc\} \cup \{(n, n), (n, t), (t, n), (t, t)\}$. $\Phi_I{}^{\mathbf{S}}$ and $R^{\mathbf{S}}$ are given in Fig. 1 (c) and (d). From top to bottom, the two (four) main branches of the automaton (transducer) correspond to the state sets $Q_I^{\neq}$ and $Q_I^{=}$ ($Q_R^1$, $Q_R^2$, $Q_R^{35}$, and $Q_R^4$), respectively.

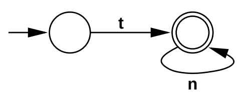
## *4.5 Discussion*

Bouajjani et al. developed a technique to compute the transitive closure of a regular relation $R$ [12,19]. A sufficient criterion for termination of that computation is *bounded local depth* [12,19] of $R$. Our construction preserves that property. Intuitively, a relation has local depth $k$ if for any $(w, w') \in R^+$ each position in $w$ needs to be rewritten no more than $k$ times. Note that in any run $\rho^{\mathbf{S}}$ of $\mathcal{P}^{\mathbf{S}}$ the projection of $\rho^{\mathbf{S}}$ onto $lo$ will be a prefix of $st^* \; lb^+ \; lc^+$. Furthermore, $\hat{w}$ changes its value in $\rho^{\mathbf{S}}$ at most once at the transition of $lo$ from $st$ to $lb$. Hence, we can infer that if $R$ has local depth $k$, $R^{\mathbf{S}}$ has local depth $\leq 3k + 2$[3]. For an illustration see Fig. 2.

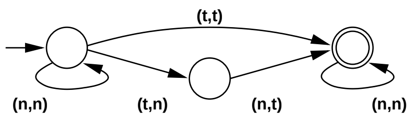
As is, the transitive closure construction of [12,19] does not preserve sufficient information to find a shortest counterexample. One could therefore determine truth or falsity of a given specification using the transitive closure [12,19] to reach a fixpoint also in the case of an infinite radius. If the specification turns out to be false, standard reachability checking (i.e., without acceleration) can be used to determine a shortest counterexample, which has necessarily finite distance from the set of initial configurations.

The ideas of regular model checking have been extended to infinite words by regarding the finite automata used to represent sets of states and the transition relation as Büchi automata on infinite words [9]. The techniques of

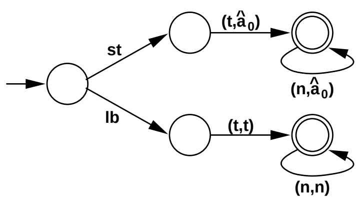
[3] The factor of 3 increases if fairness constraints are added.



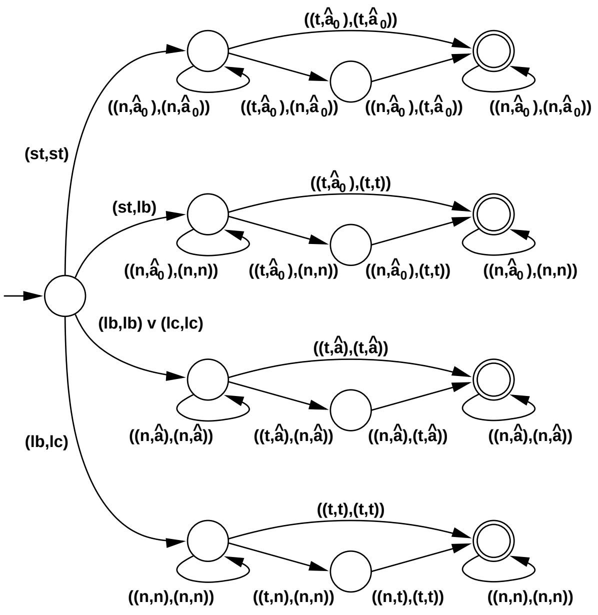
(a) initial configurations $\Phi_I$



(b) transition relation $R$



(c) reduced initial configurations $\Phi_I{}^{\mathbf{S}}$



(d) reduced transition relation $R^{\mathbf{S}}$

Fig. 1. Example: token passing [12]

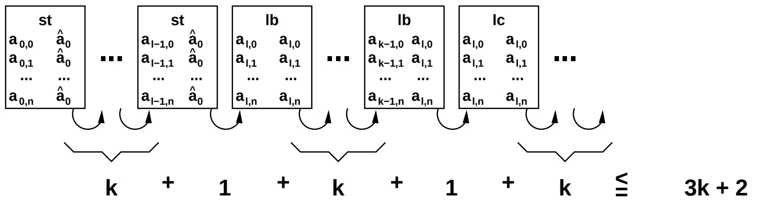


Fig. 2. The reduction preserves boundedness of local depth.

[9] require the Büchi automata to be *weakly deterministic*. A Büchi automaton is weak (1) if each of its strongly connected components contains either only accepting or only non-accepting states and (2) if the set of states can be partitioned into an ordered set of subsets such that each path in the automaton progresses in descending order through these subsets. From the proof of Lemma 4.2 it's easy to see that, if $B$ is a weakly deterministic Büchi automaton (for the set of initial configurations) or transducer (for the transition relation), so is $B^{\mathbf{S}}$. Clearly, repeated reachability may not be sufficient to verify general LTL properties for $\omega-$regular programs.

## 5 Pushdown Systems

### 5.1 Preliminaries

Notation in this section is along the lines of [16]. A *pushdown system* $M$ is a four tuple $M = (P, \Gamma, \Delta, C_I)$ where $P$ is a finite set of *control locations*, $\Gamma$ is a finite *stack alphabet*, $\Delta \subseteq (P \times \Gamma) \times (P \times \Gamma^*)$ is a finite set of *transition rules*, and $C_I \subseteq P \times \Gamma$ is a finite set of *initial configurations.*

A *configuration* is a pair $\langle p, w\rangle$ with $p \in P$ and $w \in \Gamma^*$. A *run* is a (finite or infinite) sequence of configurations $\rho = \rho(0)\rho(1)\ldots$, where $\rho(i) = \langle p_i, w_i\rangle$, such that $\forall i < |\rho| - 1 \,.\, \exists \gamma_i \in \Gamma, \exists u_i, v_i \in \Gamma^* \,.\, w_i = \gamma_i v_i \wedge w_{i+1} = u_i v_i \wedge ((p_i, \gamma_i), (p_{i+1}, u_i)) \in \Delta$. A run is *initialized* if $\rho(0) \in C_I$. *Runs*$(M)$ is the set of runs of $M$.

A *head* is a pair $\langle p, \gamma\rangle$ with $p \in P$ and $\gamma \in \Gamma$. If $c = \langle p, \gamma w\rangle$ is a configuration, *head*$(c) = \langle p, \gamma\rangle$. A head $\langle p, \gamma\rangle$ is *repeating* if there exist a run $\rho$ in $M$ and $w \in \Gamma^*$ such that $|\rho| > 1$, $\rho(0) = \langle p, \gamma\rangle$, and $\rho(|\rho| - 1) = \langle p, \gamma w\rangle$. *heads*$(\rho)$ denotes the sequence of heads derived from a run $\rho$.

Bouajjani et al. proved [10] that (1) every run that ends in a configuration with a repeating head can be extended to an infinite run, and (2) from every infinite run $\rho$ a run $\sigma\tau$ can be derived such that $|\sigma| < \infty$ and *heads*$(\tau) = (\langle p_0, \gamma_0\rangle \ldots \langle p_{l-1}, \gamma_{l-1}\rangle)^\omega$. I.e., if there exists an infinite run in $M$, then there also exists one whose sequence of heads forms a lasso.

### 5.2 Reduction

Based on the results of [10] it is sufficient to find repeating heads when checking LTL formulae on pushdown systems. Hence, a reduction of repeated reachability to reachability need only store and watch out for a second occurrence of a repeating head $\langle p, \gamma\rangle$ rather than an entire configuration. However, to infer from the second occurrence of a head that this head is indeed repeating, one has to ensure that the stack height between the first and the second occurrence never fell below the stack height at the first occurrence. To this end the stack alphabet is extended such that each stack symbol has an additional flag $bs$ (*b*ottom of *s*tack) to remember a given stack height. When saving a head, this flag is set for the bottom element pushed on the stack in the post-configuration. Whenever an element with $bs = 1$ is removed from the stack without being replaced in the same transition, a *l*oop *e*rror flag $le$ is set.

In the previous examples, $lo = lc$ signals a second occurrence of a configuration immediately at that occurrence. However, the definition of the transition rules for pushdown systems may not give access to the topmost element of the stack in the post-configuration. If no new element is pushed on the stack a comparison with a stored stack element cannot be performed. For this reason we introduce a one-state delay in the case of pushdown systems for $lo$ and the stored head. Hence, there is no need for an initial configuration with that configuration already saved.

**Definition 5.1** Let $M = (P, \Gamma, \Delta, C_I)$ be a pushdown system, let $(\hat{p}_I, \hat{\gamma}_I) \in P \times \Gamma$ be arbitrary but fixed. Then, $M^{\mathbf{S}} = (P^{\mathbf{S}}, \Gamma^{\mathbf{S}}, \Delta^{\mathbf{S}}, {C_I}^{\mathbf{S}})$ is defined as

$$
\begin{aligned}
P^{\mathbf{S}} &= P \times P \times \Gamma \times \{st, lb, lc\} \times \mathbb{B} \\
\Gamma^{\mathbf{S}} &= \Gamma \times \mathbb{B} \\
\Delta^{\mathbf{S}} &= \{(((p, \hat{p}, \hat{\gamma}, lo, le), (\gamma, bs)), ((p', \hat{p}', \hat{\gamma}', lo', le'), \mu(w', bs'_h \dots bs'_0))) \mid \\
&\qquad (((p, \gamma), (p', w')) \in \Delta) \wedge (|w'| > 1 \rightarrow \neg bs'_h \wedge \dots \wedge \neg bs'_1) \wedge \\
&\qquad ((lo = st \wedge lo' = st \wedge \neg le \wedge \neg le' \wedge \hat{p} = \hat{p}' = \hat{p}_I \wedge \hat{\gamma} = \hat{\gamma}' = \hat{\gamma}_I \wedge (|w'| > 0 \rightarrow \neg bs'_0)) \vee && (1) \\
&\qquad (lo = st \wedge lo' = lb \wedge \neg le \wedge \neg le' \wedge p = \hat{p}' \wedge \hat{p} = \hat{p}_I \wedge \gamma = \hat{\gamma}' \wedge \hat{\gamma} = \hat{\gamma}_I \wedge |w'| > 0 \wedge bs'_0) \vee && (2) \\
&\qquad (lo = lb \wedge lo' = lb \wedge ((|w'| = 0 \wedge bs \vee le) \leftrightarrow le') \wedge \\
&\qquad \hat{p} = \hat{p}' \wedge \hat{\gamma} = \hat{\gamma}' \wedge (|w'| > 0 \rightarrow (bs \leftrightarrow bs'_0))) \vee && (3) \\
&\qquad (lo = lb \wedge lo' = lc \wedge \neg le \wedge \neg le' \wedge p = \hat{p} = \hat{p}' \wedge \gamma = \hat{\gamma} = \hat{\gamma}' \wedge (|w'| > 0 \rightarrow \neg bs'_0)) \vee && (4) \\
&\qquad (lo = lc \wedge lo' = lc \wedge \neg le \wedge \neg le' \wedge \hat{p} = \hat{p}' \wedge \hat{\gamma} = \hat{\gamma}' \wedge (|w'| > 0 \rightarrow \neg bs'_0)))\} && (5) \\
{C_I}^{\mathbf{S}} &= \{\langle (p_I, \hat{p}_I, \hat{\gamma}_I, st, 0), (\gamma_I, 0) \rangle \mid \langle p_I, \gamma_I \rangle \in C_I\}
\end{aligned}
$$

Definition 5.1 shows the entire reduction. The transition relation is partitioned into 5 sets again. While no state has been saved (subset (1)), $lo = st$ and $\neg le$ remain constant, the initial values for $\hat{p}$ and $\hat{\gamma}$ are just copied, and no stack height need be remembered ($bs_0$ is false). Saving a state (subset (2)) can only occur if a non-empty word is pushed back on the stack — otherwise, the next transition would immediately violate the above-mentioned condition for the stack height of a repeating head. Taking a transition from subset (2) saves the head $\langle p, \gamma \rangle$ (in the pre-configuration) in $\hat{p}$ and $\hat{\gamma}$ (in the post-configuration), sets $lo$ to $lb$, and marks the current stack height by setting $bs$ to true for the bottom element pushed on the stack. Transitions from subset (3) are taken while a second occurrence of the stored head has not been seen, hence, $lo$ as well as $\hat{p}$ and $\hat{\gamma}$ keep their values. In addition, the condition not to fall below the stack height at the time of saving is checked. When this is the case, i.e., when an element with $bs$ true is popped from the stack and only an empty word is pushed back, the loop error flag $le$ is set to true. This prevents signalling a repeating head in the future by restricting subsequent transitions to subset (3). When the stack height remains above the required level, $le$ keeps its value and the flag $bs$ is set in the bottom element of the word pushed onto the stack iff it was set in the symbol popped from the stack. A second occurrence of $\langle p, \gamma \rangle$ is signalled by setting $lo = lc$ when taking a transition from subset (4). $le$, $\hat{p}$, and $\hat{\gamma}$ keep their values. Any remembered stack height is discarded. Transitions of the last subset (5) keep all additional location components constant.

In the following we prove correctness of the reduction.

**Theorem 5.2** *Let $M = (P, \Gamma, \Delta, c_I)$ be a pushdown system and $M^{\mathbf{S}}$ be defined as above. There exists an initialized run $\rho$ to a repeating head $\langle p_0, \gamma \rangle$ in $M$ if and only if there exists an initialized run $\rho^{\mathbf{S}}$ in $M^{\mathbf{S}}$ with $\rho^{\mathbf{S}}(|\rho^{\mathbf{S}}| - 2) = \langle (p_0, p_0, \gamma, lb, 0), w_{|\rho^{\mathbf{S}}|-2} \rangle$, where $w_{|\rho^{\mathbf{S}}|-2}(0) = \gamma$, and $\rho^{\mathbf{S}}(|\rho^{\mathbf{S}}| - 1) = \langle (p, p_0, \gamma, lc, 0), w_{|\rho^{\mathbf{S}}|-1} \rangle$.*

**Proof.** "$\Rightarrow$": Assume a run $\rho$ to a repeatable head $\langle p_0, \gamma \rangle$. Hence, there exist $l \geq 0$, $q_0, \dots, q_{l-1} \in P$, $w_0, \dots, w_{l-1} \in \Gamma^*$, $v \in \Gamma^*$ where $\forall i < l \,.\, \rho(i) = \langle q_i, w_i \rangle$

and $\rho(l) = \langle p_0, \gamma v \rangle$.

By the definition of a repeating head there are $k > l$, $p_1, \ldots, p_{k-l-1} \in P$, $u_0, \ldots, u_{k-l} \in \Gamma^+$, where $u_0 = u_{k-l}(0) = \gamma$, such that $\rho$ can be extended to an infinite run $\rho^\infty \in \mathit{Runs}(M)$:

$$\begin{aligned}
&\forall i < l \;.\; \rho^\infty(i) = \rho(i) \\
&\forall i \geq l \;.\; \rho^\infty(i) = \langle p_{(i-l)\ \mathit{mod}\ (k-l)}, \\
&\qquad\qquad u_{(i-l)\ \mathit{mod}\ (k-l)}(u_{k-l}(1) \ldots u_{k-l}(|u_{k-l}|-1))^{(i-l)\ \mathit{div}\ (k-l)} v \rangle
\end{aligned}$$

From that we construct a finite run $\rho^{\mathbf{S}}$ as follows:

$$\begin{aligned}
&\forall i < l \;.\; \rho^{\mathbf{S}}(i) = \langle (q_i, \hat{p}_I, \hat{\gamma}_I, st, 0), \mu(w_i, 0^{|w_i|}) \rangle \\
&\rho^{\mathbf{S}}(l) = \langle (p_0, \hat{p}_I, \hat{\gamma}_I, st, 0), (\gamma, 0)\ \mu(v, 0^{|v|}) \rangle \\
&\rho^{\mathbf{S}}(l+1) = \langle (p_1, p_0, \gamma, lb, 0), \mu(u_1, 0^{|u_1|-1}1)\ \mu(v, 0^{|v|}) \rangle \\
&\forall l+1 < i < l+k \;.\; \rho^{\mathbf{S}}(i) = \langle (p_{i-l}, p_0, \gamma, lb, 0), \mu(u_{i-l}, 0^{|u_{i-l}|-1}1)\ \mu(v, 0^{|v|}) \rangle
\end{aligned}$$

if $|u_{k-l}| > 1$

$$\begin{aligned}
&\rho^{\mathbf{S}}(k) = \langle (p_0, p_0, \gamma, lb, 0), (\gamma, 0)\ \mu(u_{k-l}(1) \ldots u_{k-l}(|u_{k-l}|-1), 0^{|u_{k-l}|-2}\ 1)\ \mu(v, 0^{|v|}) \rangle \\
&\rho^{\mathbf{S}}(k+1) = \langle (p_1, p_0, \gamma, lc, 0), \\
&\qquad\qquad \mu(u_1, 0^{|u_1|})\ \mu(u_{k-l}(1) \ldots u_{k-l}(|u_{k-l}|-1), 0^{|u_{k-l}|-2}\ 1)\ \mu(v, 0^{|v|}) \rangle
\end{aligned}$$

otherwise

$$\begin{aligned}
&\rho^{\mathbf{S}}(k) = \langle (p_0, p_0, \gamma, lb, 0), (\gamma, 1) \mu(v, 0^{|v|}) \rangle \\
&\rho^{\mathbf{S}}(k+1) = \langle (p_1, p_0, \gamma, lc, 0), \mu(u_1, 0^{|u_1|})\ \mu(v, 0^{|v|}) \rangle
\end{aligned}$$

"$\Leftarrow$": Assume an initialized run $\rho^{\mathbf{S}}$ to $\rho^{\mathbf{S}}(|\rho^{\mathbf{S}}|-2) = \langle (p_0, p_0, \gamma, lb, 0), w_{|\rho^{\mathbf{S}}|-2} \rangle$, where $w_{|\rho^{\mathbf{S}}|-2}(0) = \gamma$, and $\rho^{\mathbf{S}}(|\rho^{\mathbf{S}}|-1) = \langle (p_1, p_0, \gamma, lc, 0), w_{|\rho^{\mathbf{S}}|-1} \rangle$. By Def. 5.1, $\exists 0 < l < |\rho^{\mathbf{S}}| - 2$ such that $\rho^{\mathbf{S}}(l) = \langle (p_0, \hat{p}_I, \hat{\gamma}_I, st, 0), \mu(w_l, 0^{|w_l|}) \rangle$ and $w_l(0) = \gamma$. Clearly, the projection of $\rho^{\mathbf{S}}(0 \ldots l)$ on the first components of its state and stack is a run in $M$ to a repeatable head. □

## *5.3 Complexity*

**Proposition 5.3** *Let $M = (P, \Gamma, \Delta, C_I)$ be a pushdown system. $M^{\mathbf{S}}$ has $\mathbf{O}(|P||\Gamma||P|)$ locations and $\mathbf{O}(|P||\Gamma||\Delta|)$ transition rules.*

**Proof.** The locations of $M$ are extended in $M^{\mathbf{S}}$ to store another location, a stack symbol, and a small constant amount of additional state information. For $\Delta^{\mathbf{S}}$, there are $\mathbf{O}(|\Delta|)$ transition rules in subsets (1), (2), and (4), and $\mathbf{O}(|P||\Gamma||\Delta|)$ in (3) and (5). □

A number of algorithms has been proposed that can be used to check reachability in a pushdown system (e.g., [10,18,16]). [16] improves on previous results, the algorithms (one for forward and one for backward reachability) as well as their analysis are clearly formulated, and an implementation [17] is available. We therefore chose [16] as the basis for a more detailed complexity analysis of our reduction.

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | $\gamma_2$ | $\gamma_2$ | | | | | | | | | | | $\gamma_2$ | $\gamma_2$ | | | | | |
| | | | | | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | | | | | | | | | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | | | | |
| w(i) | | | | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | | | | | | | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | | | |
| | | | $\gamma_2$ | $\gamma_2$ | $\gamma_2$ | $\gamma_2$ | $\gamma_2$ | $\gamma_2$ | $\gamma_2$ | $\gamma_2$ | | | | | $\gamma_2$ | $\gamma_2$ | $\gamma_2$ | $\gamma_2$ | $\gamma_2$ | $\gamma_2$ | $\gamma_2$ | $\gamma_2$ | | |
| | | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | | | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | $\gamma_1$ | |
| | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ | $\gamma_0$ |
| $p(i)$ | $p_0$ | $p_0$ | $p_0$ | $p_1$ | $p_1$ | $p_1$ | $p_2$ | $p_2$ | $p_2$ | $p_3$ | $p_3$ | $p_3$ | $p_4$ | $p_4$ | $p_4$ | $p_5$ | $p_5$ | $p_5$ | $p_2$ | $p_2$ | $p_2$ | $p_3$ | $p_3$ | $p_3$ |
| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

Fig. 3. The soonest second occurrence of a repeating head might not indicate the shortest counterexample.

Algorithm 3 in [16] can be used to check forward reachability for a pushdown system $M = (P, \Gamma, \Delta, C_I)$ where $(p, \gamma, p', w') \in \Delta \Rightarrow |w'| \leq 2$. It computes the set of configurations reachable from $C_I$ in $\mathbf{O}(|P||\Delta|^2 + |P||\Gamma|)$ time and space. When applied to $M^{\mathbf{S}}$ a blow-up of $\mathbf{O}(|P||\Gamma|)$ is sufficient. Similarly, algorithm 1 can be used to compute the sets of configurations from which some "bad" configuration $\langle(p, \hat{p}, \hat{\gamma}, lc, 0), w\rangle$ is reachable.

**Proposition 5.4** *Let $M = (P, \Gamma, \Delta, C_I)$ be a pushdown system such that $(p, \gamma, p', w') \in \Delta \Rightarrow |w'| \leq 2$. Algorithm 3 in [16] computes the set of configurations reachable from ${C_I}^{\mathbf{S}}$ in $M^{\mathbf{S}}$ (forward reachability) in time and space $\mathbf{O}(|P||\Gamma|(|P||\Delta|^2))$. Algorithm 1 in [16] computes the set of configurations from which a configuration in $\{\langle(p, \hat{p}, \hat{\gamma}, lc, 0), w\rangle \mid p, \hat{p} \in P \wedge \hat{\gamma} \in \Gamma \wedge w \in (\Sigma, \mathbb{B})^*\}$ is reachable in $M^{\mathbf{S}}$ (backward reachability) in time $\mathbf{O}(|P||\Gamma|(|P|^2|\Delta| + |P||\Gamma|))$ and space $\mathbf{O}(|P||\Gamma|(|P||\Delta| + |P||\Gamma|))$.*

**Proof.** See [25]. □

## *5.4 Shortest Lasso-Shaped Counterexamples*

Assume again that $M = (P, \Gamma, \Delta, C_I)$ is a pushdown system such that $(p, \gamma, p', w') \in \Delta \Rightarrow |w'| \leq 2$. In his thesis [28], Schwoon shows how to construct a shortest path to a reachable configuration. If applied to a pushdown system obtained by the transformation in Def. 5.1 the soonest second occurrence of a repeating head can be found. However, this is not sufficient to find shortest counterexamples.

Finding a shortest counterexample requires to extend the definition of a repeating head to a repeating prefix: any configuration $\langle p, w\rangle$ with $|w| > 0$ is a prefix. It is repeating iff there exist a run $\rho$ and a word $v$ with $\rho(0) = \langle p, w\rangle$ and $\rho(|\rho| - 1) = \langle p, wv\rangle$. For an example see the run in Fig. 3.

The second occurrence of the repeating head $(p_3, \gamma_0)$ can only be detected at $i = 23$ while the repeating prefix $\langle p_2, \gamma_2\gamma_1\gamma_0\gamma_2\gamma_1\gamma_0\rangle$ indicates a path whose heads form a lasso at $i = 18$.

The example also shows that the length of the prefix to be considered is $\mathbf{O}(|P||\Gamma|)$. On the other hand, once a run reaches a stack height of $|P||\Gamma| + 1$ there must have been a second occurrence of a repeating head: consider an initialized run $\rho = \langle p_0, w_0 \rangle \dots \langle p_k, w_k \rangle$ such that $|w_k| = |P||\Gamma| + 1$. Remember that the stack height grows or shrinks by at most one per transition. For each $0 \leq h \leq |P||\Gamma| + 1$ there exists $0 \leq i_h \leq k$ such that all $\rho(i)$ with $i > i_h$ have stack height larger than $h$, i.e., $\forall i > i_h \,.\, |w_i| > |w_{i_h}| = h$. Clearly, there must be $h_1 \neq h_2$ such that $\mathit{head}(\rho(i_{h_1})) = \mathit{head}(\rho(i_{h_2}))$. From the construction of the $i_h$, $\rho(0) \dots \rho(i_{h_1}) \dots \rho(i_{h_2})$ provides evidence that $\mathit{head}(\rho(i_{h_1}))$ is a reachable repeatable head. As a final remark, it is clear that the length of any counterexample known to be present can be used to bound the length of a repeating prefix.

## 6 Timed Automata

The reduction for timed automata is technically involved and provides few new insights. We only give the idea of how to apply our reduction to timed automata [4] and refer the interested reader to [25] for details.

In addition to a finite set of control locations, timed automata have a finite set of real-valued clocks. Transitions are labeled with integer clock constraints of the form $c \sim n$ where $c$ is a clock variable, $\sim \in \{<, \leq, =, \geq, >\}$, and $n \in \mathbb{N}$.

Alur and Dill showed [4] that for model checking of LTL the precise value of the clocks is not relevant. Rather, clock valuations fall into a finite number of equivalence classes called *regions*. Model checking is then performed on the abstract *region automaton*.

We use this fact in our reduction as follows. We do not store the precise valuation of the clocks but the clock region. This requires a variable in the range $\{0, \dots, c_x\}$ and a flag for each clock $x$, where $c_x$ is the maximal integer $x$ is compared with in a clock constraint. Furthermore, we store the order of the fractional parts of the clocks. This requires $k$ variables of range $0 \dots k-1$ if there are $k$ clocks and $k-1$ flags to indicate equality between each pair of successors in the order.

As in the finite case the reduction can be used to find lasso-shaped counterexamples with a minimal number of transitions for LTL properties if breadth first search is used to determine reachability. Alternatively, using a priority queue instead of a queue in the reachability algorithm, the lasso-shaped path which spends least time until the closure of the loop can be found [6]. Uppaal [23] offers both possibilities.

## 7 Conclusion

We have extended our reduction of repeated reachability to reachability to some popular classes of infinite state systems. For these classes the reductions "pull the original algorithm into the model". To explore the limits of our method we are looking for systems where liveness can still be reduced to repeated reachability, but where our method might not seem applicable. It is clear that the construction for the finite case can not always be lifted to infinite state systems. In general, counterexamples to liveness properties in infinite state systems can not necessarily be restricted to have lasso shape. In some cases, abstractions [24] or simulations [13] might help. Maybe our method can also provide additional insight why liveness is undecidable for some classes of systems. Finally, experiments need to prove the viability of our approach.

## References

[1] Abdulla, P., B. Jonsson, M. Nilsson and J. d'Orso, *Algorithmic improvements in regular model checking*, in: W. Hunt and F. Somenzi, editors, *CAV'03*, LNCS **2725** (2003), pp. 236–248.

[2] Abdulla, P., B. Jonsson, M. Nilsson, J. d'Orso and M. Saksena, *Regular model checking for LTL(MSO)*, in: Alur and Peled [5], pp. 348–360.

[3] Aceto, L., P. Bouyer, A. Burgueño and K. Larsen, *The power of reachability testing for timed automata*, Theor. Comput. Sci. **300** (2003), pp. 411–475.

[4] Alur, R. and D. Dill, *A theory of timed automata*, Theor. Comput. Sci. **126** (1994), pp. 183–235.

[5] Alur, R. and D. Peled, editors, "Computer Aided Verification, 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004, Proceedings," LNCS **3114**, Springer, 2004.

[6] Behrmann, G., A. Fehnker, T. Hune, K. Larsen, P. Pettersson and J. Romijn, *Efficient guiding towards cost-optimality in UPPAAL*, in: T. Margaria and W. Yi, editors, *TACAS'01*, LNCS **2031** (2001), pp. 174–188.

[7] Biere, A., C. Artho and V. Schuppan, *Liveness checking as safety checking*, in: R. Cleaveland and H. Garavel, editors, *Formal Methods for Industrial Critical Systems, Proceedings of the 7th International ERCIM Workshop, FMICS'02, Málaga, Spain, July 12–13, 2002*, Electronic Notes in Theoretical Computer Science, 66(2) (2002).

[8] Biere, A., A. Cimatti, E. Clarke and Y. Zhu, *Symbolic model checking without BDDs*, in: R. Cleaveland, editor, *TACAS'99*, LNCS **1579** (1999), pp. 193–207.

[9] Boigelot, B., A. Legay and P. Wolper, *Omega-regular model checking*, in: K. Jensen and A. Podelski, editors, *TACAS'04*, LNCS **2988** (2004), pp. 561–575.

[10] Bouajjani, A., J. Esparza and O. Maler, *Reachability analysis of pushdown automata: Application to model-checking*, in: A. W. Mazurkiewicz and J. Winkowski, editors, *CONCUR'97*, LNCS **1243** (1997), pp. 135–150.

[11] Bouajjani, A., P. Habermehl and T. Vojnar, *Abstract regular model checking*, in: Alur and Peled [5], pp. 372–386.

[12] Bouajjani, A., B. Jonsson, M. Nilsson and T. Touili, *Regular model checking*, in: Emerson and Sistla [15], pp. 403–418.

[13] Bouajjani, A., A. Legay and P. Wolper, *Handling liveness properties in ($\omega-$)regular model checking*, in: *INFINITY'04*, 2004.

[14] Clarke, E., O. Grumberg and D. Peled, "Model Checking," MIT Press, 1999.

[15] Emerson, E. and A. Sistla, editors, "Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings," LNCS **1855**, Springer, 2000.

[16] Esparza, J., D. Hansel, P. Rossmanith and S. Schwoon, *Efficient algorithms for model checking pushdown systems*, in: Emerson and Sistla [15], pp. 232–247.

[17] Esparza, J. and S. Schwoon, *A BDD-based model checker for recursive programs*, in: G. Berry, H. Comon and A. Finkel, editors, *CAV'01*, LNCS **2102** (2001), pp. 324–336.

[18] Finkel, A., B. Willems and P. Wolper, *A direct symbolic approach to model checking pushdown systems (extended abstract)*, in: F. Moller, editor, *INFINITY'97*, ENTCS **9** (1997).

[19] Jonsson, B. and M. Nilsson, *Transitive closures of regular relations for verifying infinite-state systems*, in: S. Graf and M. Schwartzbach, editors, *TACAS'00*, LNCS **1785** (2000), pp. 220–234.

[20] Kesten, Y., O. Maler, M. Marcus, A. Pnueli and E. Shahar, *Symbolic model checking with rich assertional languages.*, Theor. Comput. Sci. **256** (2001), pp. 93–112.

[21] Kozen, D., *Results on the propositional $\mu$-calculus*, Theor. Comput. Sci. **27** (1983), pp. 333–354.

[22] Kupferman, O. and M. Vardi, *Model checking of safety properties*, Formal Methods in System Design **19** (2001), pp. 291–314.

[23] Larsen, K., P. Pettersson and W. Yi, Uppaal *in a Nutshell*, International Journal on Software Tools for Technology Transfer (STTT) **1** (1997), pp. 134–152.

[24] Pnueli, A. and E. Shahar, *Liveness and acceleration in parameterized verification*, in: Emerson and Sistla [15], pp. 328–343.

[25] Schuppan, V., "Liveness Checking as Safety Checking to Find Shortest Counterexamples to Linear Time Properties," Ph.D. thesis, ETH Zürich (2005).

[26] Schuppan, V. and A. Biere, *Efficient reduction of finite state model checking to reachability analysis*, International Journal on Software Tools for Technology Transfer (STTT) **5** (2004), pp. 185–204.

[27] Schuppan, V. and A. Biere, *Shortest counterexamples for symbolic model checking of LTL with past*, in: N. Halbwachs and L. Zuck, editors, *TACAS'05*, LNCS **3440** (2005), pp. 493–509.

[28] Schwoon, S., "Model-Checking Pushdown Systems," Ph.D. thesis, Technische Universität München (2002).

[29] Sebastiani, R., S. Tonetta and M. Vardi, *Symbolic systems, explicit properties: on hybrid approaches for LTL symbolic model checking*, in: K. Etessami and S. Rajamani, editors, *CAV'05*, LNCS **3576** (2005), pp. 350–363.

[30] Shilov, N. and K. Yi, *On expressive and model checking power of propositional program logics*, in: D. Bjørner, M. Broy and A. Zamulin, editors, *PSI'01*, LNCS **2244** (2001), pp. 39–46.

[31] Shilov, N., K. Yi, H. Eo, S. O and K.-M. Choe, *Proofs about folklore: why model checking = reachability?* (2005), submitted.

[32] Stirling, C., *Games and modal mu-calculus*, in: T. Margaria and B. Steffen, editors, *TACAS'96*, LNCS **1055** (1996), pp. 298–312.

[33] Vardi, M. and P. Wolper, *An automata-theoretic approach to automatic program verification*, in: *LICS'86* (1986), pp. 332–344.

[34] Wolper, P. and B. Boigelot, *Verifying systems with infinite but regular state spaces*, in: A. Hu and M. Vardi, editors, *CAV'98*, LNCS **1427** (1998), pp. 88–97.