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 [16], verification of general LTL properties is typically performed by searching for infinite paths in the product of the system and an automaton representing the property [22].

In [19] we have developed a syntactic reduction from computing repeated reachability to computing reachability for finite state systems. This reduction has been used to develop a BDD-based method to find shortest counterexamples [20]. On selected examples a significant speed up compared to traditional liveness checking can be observed [19,21]. It can also help to simplify proofs if a proof for safety properties is easier than the corresponding proof for general LTL properties. It may finally discourage tool vendors from charging

[1] Email: Viktor.Schuppan@inf.ethz.ch
[2] Email: biere@jku.at

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

separately for liveness-enabled versions of their verification tools. For further motivation and related work on finite state systems see [19].

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., [1,12,17]), are ($\omega-$)regular model checking [15,23,7,5], pushdown systems [6,13,11], and timed automata [3].

Early work on liveness for regular model checking includes [7,18]. Pnueli and Shahar [18] also use a copy of a current state to detect bad cycles in parameterized systems. However, this is not performed as syntactic transformation of a model but as part of a dedicated liveness checking algorithm. A variant of LTL geared towards parameterized systems is proposed in [2]. [8] 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 [6], and for timed automata in [3].

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 systems in Sect. 5. Due to space constraints the construction for timed automata can only be sketched in Sect. 6. 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 ρ, $\rho(i)$ 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^ω are the sets of finite and infinite sequences of elements of S.

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

3 Liveness Checking as Safety Checking – Finite Case

In this section we briefly restate the main result from [19] to explain the basic idea and notation of our reduction.

3.1 Preliminaries

Let AP be a finite set of atomic propositions. A *Kripke structure*, see, e.g., [9], is a four tuple $M = (S, S_0, R, L)$ where S is a finite set of *states*, $S_0 \subseteq S$

Definition 3.1 Let $M = (S, S_0, R, L)$ be a Kripke structure. Then $M^{\mathbf{S}} = (S^{\mathbf{S}}, {S_0}^{\mathbf{S}}, R^{\mathbf{S}}, L^{\mathbf{S}})$ is defined as:

$$
\begin{aligned}
S^{\mathbf{S}} &= S \times S \times \mathbb{B} \times \mathbb{B} \\
{S_0}^{\mathbf{S}} &= \{(s_0, \hat{s}_0, 0, 0) \mid s_0 \in S_0\} \cup \{(s_0, s_0, 1, 0) \mid s_0 \in S_0\} \\
R^{\mathbf{S}} &= \{((s, \hat{s}, lb, lc), (s', \hat{s}', lb', lc')) \mid (s, s') \in R \wedge \\
&\quad ((\neg lb \wedge \neg lb' \wedge \neg lc \wedge \neg lc' \wedge \hat{s} = \hat{s}') \vee && (1) \\
&\quad\ (\neg lb \wedge lb' \wedge \neg lc \wedge \neg lc' \wedge s' = \hat{s}') \vee && (2) \\
&\quad\ (lb \wedge lb' \wedge \neg lc \wedge \neg lc' \wedge \hat{s} = \hat{s}') \vee && (3) \\
&\quad\ (lb \wedge lb' \wedge \neg lc \wedge lc' \wedge \hat{s} = s' = \hat{s}') \vee && (4) \\
&\quad\ (lb \wedge lb' \wedge lc \wedge lc' \wedge \hat{s} = \hat{s}'))\} && (5) \\
L^{\mathbf{S}}((s, \hat{s}, lb, lc)) &= L(s)
\end{aligned}
$$

is the set of *initial states*, $R \subseteq S \times S$ is a *transition relation*, and $L : S \mapsto 2^{AP}$ is a *labeling* of the states.

A *run* is a (finite or infinite) sequence of states $\rho = \rho(0)\rho(1)\ldots$ where $\forall 0 \leq i < |\rho| \ . \ (\rho(i), \rho(i+1)) \in R$. ρ is initialized if $\rho(0) \in S_0$, $Runs(M)$ 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 path where p never holds on that path. Finding such loop is a key ingredient of many model checking algorithms for LTL, e.g., [22,4]. Our reduction integrates the detection of a loop into the model to be verified by nondeterministically saving the current state (i.e., guessing a potential loop start) and then watching for a second occurrence of that state (i.e., detecting closure of the loop). For this purpose, the reduction extends a state s in the original model with a component to store a previously seen state, $\hat{s}$, and two flags lb ($l$oop $b$ody) and lc ($l$oop $c$losed). lb is set to true when a state is saved and prevents future overwriting of the stored state. lc indicates that a second occurrence of $\hat{s}$ has been found.

Definition 3.1 shows the construction. The transitions of $R^{\mathbf{S}}$ are partitioned into 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) are taken.

Theorem 3.2 *Let $M = (S, S_0, R, L)$ be a Kripke structure, let $M^{\mathbf{S}}$ be defined*

as above. Assume $k > l \geq 0$.

$$(s_0 \dots s_{l-1})(s_l \dots s_{k-1})^\omega \in Runs(M)$$

$$\Leftrightarrow$$

$$(s_0, \hat{s}_0, 0, 0) \dots (s_{l-1}, \hat{s}_0, 0, 0)(s_l, s_l, 1, 0) \dots (s_{k-1}, s_l, 1, 0)(s_k, s_l, 1, 1) \in Runs(M^{\mathbf{S}})$$

Proof. "⇒": Let $\rho = (s_0 \dots s_{l-1})(s_l \dots s_{k-1})^\omega$ be a run in M. We construct $\rho^{\mathbf{S}}$ as follows. If $l > 0$ choose $\rho^{\mathbf{S}}(0) = (s_0, \hat{s}_0, 0, 0)$ with arbitrary $\hat{s}_0$. Construct $(s_0, \hat{s}_0, 0, 0) \dots (s_{l-1}, \hat{s}_0, 0, 0)$ by taking transitions from subset (1). Proceed to $(s_l, s_l, 1, 0)$ via a transition from (2). Continue to $(s_{k-1}, s_l, 1, 0)$ with $k - l - 1$ transitions from (3). Finally, as $k > l$, there exists $s_k = s_l$, so take a transition from (4) to $(s_k, s_l, 1, 1)$. Otherwise, if $l = 0$, start with $(s_0, s_0, 1, 0)$ and continue with $k - 1$ transitions from (3) and one from (4) as before.

"⇐": Let $\rho^{\mathbf{S}} = (s_0, \hat{s}_0, 0, 0) \dots (s_{l-1}, \hat{s}_0, 0, 0)(s_l, s_l, 1, 0) \dots (s_{k-1}, s_l, 1, 0) \circ (s_k, s_l, 1, 1)$ be a run in $M^{\mathbf{S}}$ such that $k > l$. From the construction of $M^{\mathbf{S}}$, $\rho' = s_0 \dots s_{l-1} s_l \dots s_{k-1} s_k$ is a finite run in M with $s_k = s_l$. Hence, $\rho = (s_0 \dots s_{l-1})(s_l \dots s_{k-1})^\omega$ is a run in M as desired. □

Remark 3.3 Checking properties given as Büchi automata requires finding *fair loops*. This can be achieved by adding a flag for each fairness constraint, for details see [19]. The infinite cases can be handled similarly.

3.3 Complexity

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

Proposition 3.4 *Let* $M = (S, S_0, R, L)$ *be a 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 flags *lb*, *lc*. $R^{\mathbf{S}}$ contains $\mathbf{O}(|S|)$ transitions $t^{\mathbf{S}}$ per transition $t \in R$ in subsets (1), (3), and (5), and $\mathbf{O}(1)$ $t^{\mathbf{S}}$ per $t \in R$ in subsets (2) and (4). □

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 [19].

4 Regular Model Checking

4.1 Preliminaries

The notation in this section is mostly borrowed from [7]. Let Σ 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, δ, F)

Definition 4.1 Let $\mathcal{P} = (\Sigma, \Phi_I, R)$ be a program. Then $\mathcal{P}^{\mathbf{S}} = (\Sigma^{\mathbf{S}}, {\Phi_I}^{\mathbf{S}}, R^{\mathbf{S}})$ is defined as

$$
\begin{aligned}
\Sigma^{\mathbf{S}} &= \mathbb{B} \cup (\Sigma \times \Sigma) \\
{\Phi_I}^{\mathbf{S}} &= \{0\} \circ \{0\} \circ \{\mu(w, \hat{w}) \in (\Sigma \times \Sigma)^* \mid |w| = |\hat{w}| \wedge w \in \Phi_I\} \cup \\
&\quad \{1\} \circ \{0\} \circ \{\mu(w, w) \in (\Sigma \times \Sigma)^* \mid w \in \Phi_I\} \\
R^{\mathbf{S}} &= \{((lb\ lc\ \mu(w, \hat{w})), (lb'\ lc'\ \mu(w', \hat{w}'))) \subseteq (\mathbb{B} \circ \mathbb{B} \circ (\Sigma \times \Sigma)^*)^2 \mid \\
&\quad |w| = |\hat{w}| = |w'| = |\hat{w}'| \wedge (w, w') \in R \wedge \\
&\quad ((\neg lb \wedge \neg lb' \wedge \neg lc \wedge \neg lc' \wedge \hat{w} = \hat{w}') \vee && (1) \\
&\quad\ (\neg lb \wedge lb' \wedge \neg lc \wedge \neg lc' \wedge w' = \hat{w}') \vee && (2) \\
&\quad\ (lb \wedge lb' \wedge \neg lc \wedge \neg lc' \wedge \hat{w} = \hat{w}') \vee && (3) \\
&\quad\ (lb \wedge lb' \wedge \neg lc \wedge lc' \wedge \hat{w} = w' = \hat{w}') \vee && (4) \\
&\quad\ (lb \wedge lb' \wedge lc \wedge lc' \wedge \hat{w} = \hat{w}'))\} && (5)
\end{aligned}
$$

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.

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*.

A *configuration* of a program $\mathcal{P}$ is a word w over Σ. *Runs* are finite or infinite sequences of configurations $\rho = \rho(0)\rho(1)\ldots$, such that $\forall 0 \leq i < |\rho|\ .\ (\rho(i), \rho(i+1)) \in R$. 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 exactly the same as in the finite case.

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 Φ_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 $(Q_I, q_{0I}, \delta_I, F_I)$, $(Q_I^{\neq}, q_{0I}^{\neq}, \delta_I^{\neq}, F_I^{\neq})$ and $(Q_I^{=}, q_{0I}^{=}, \delta_I^{=}, f_I^{=})$. Then $({Q_I}^{\mathbf{S}}, {q_{0I}}^{\mathbf{S}}, {\delta_I}^{\mathbf{S}}, {F_I}^{\mathbf{S}})$ with

$$
\begin{aligned}
{Q_I}^{\mathbf{S}} &= Q_I^{\neq} \cup Q_I^{=} \cup \{q_{lb}, q_{lc}^{\neq}, q_{lc}^{=}\},\\
{q_{0I}}^{\mathbf{S}} &= q_{lb},\\
{\delta_I}^{\mathbf{S}} &= \{(q_{lb}, 0, q_{lc}^{\neq}), (q_{lc}^{\neq}, 0, q_{0I}^{\neq})\} \cup \{(q^{\neq}, (a, \hat{a}), q^{\neq\prime}) \mid (q^{\neq}, a, q^{\neq\prime}) \in \delta_I^{\neq}\} \cup\\
&\quad \{(q_{lb}, 1, q_{lc}^{=}), (q_{lc}^{=}, 0, q_{0I}^{=})\} \cup \{(q^{=}, (a, a), q^{=\prime}) \mid (q^{=}, a, q^{=\prime}) \in \delta_I^{=}\}, \text{ and}\\
{F_I}^{\mathbf{S}} &= F_I^{\neq} \cup F_I^{=},
\end{aligned}
$$

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

Similarly, if R is given by $(Q_R, q_{0R}, \delta_R, F_R)$, we construct a finite transducer $({Q_R}^{\mathbf{S}}, {q_{0R}}^{\mathbf{S}}, {\delta_R}^{\mathbf{S}}, {F_R}^{\mathbf{S}})$ to accept $R^{\mathbf{S}}$. We use separate copies of $(Q_R, q_{0R}, \delta_R, F_R)$ to leave the saved word unchanged (superscript ^{135}, corresponding to disjuncts 1, 3, and 5 in Def. 4.1), save a word (sup. ^{2}, corr. to subset (2)), and compare current and stored word (sup. ^{4}, corr. to subset (4)).

$$
\begin{aligned}
{Q_R}^{\mathbf{S}} &= Q_R^{135} \cup Q_R^{2} \cup Q_R^{4} \cup \{q_{lb}, q_{lc}^{1}, q_{lc}^{2}, q_{lc}^{345}\},\\
{q_{0R}}^{\mathbf{S}} &= q_{lb},\\
{\delta_R}^{\mathbf{S}} &= \{(q_{lb}, (0,0), q_{lc}^{1}), (q_{lb}, (0,1), q_{lc}^{2}), (q_{lb}, (1,1), q_{lc}^{345})\} \cup\\
&\quad \{(q_{lc}^{1}, (0,0), q_0^{135}), (q_{lc}^{2}, (0,0), q_0^{2}), (q_{lc}^{345}, (0,0), q_0^{135}),\\
&\quad\ (q_{lc}^{345}, (0,1), q_0^{4}), (q_{lc}^{345}, (1,1), q_0^{135})\} \cup\\
&\quad \{(q^{135}, ((a, \hat{a}), (a', \hat{a})), q^{135\prime}) \mid (q^{135}, (a, a'), q^{135\prime}) \in \delta_R^{135}\} \cup\\
&\quad \{(q^{2}, ((a, \hat{a}), (a', a')), q^{2\prime}) \mid (q^{2}, (a, a'), q^{2\prime}) \in \delta_R^{2}\} \cup\\
&\quad \{(q^{4}, ((a, a'), (a', a')), q^{4\prime}) \mid (q^{4}, (a, a'), q^{4\prime}) \in \delta_R^{4}\}, \text{ and}\\
{F_R}^{\mathbf{S}} &= F_R^{135} \cup F_R^{2} \cup F_R^{4}
\end{aligned}
$$

□

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

$$
\begin{gathered}
(w_0 \dots w_{l-1})(w_l \dots w_{k-1})^{\omega} \in Runs(\mathcal{P})\\
\Leftrightarrow\\
(0\ 0\ \mu(w_0, \hat{w}_I)) \dots (0\ 0\ \mu(w_{l-1}, \hat{w}_I))(1\ 0\ \mu(w_l, w_l)) \dots\\
\dots (1\ 0\ \mu(w_{k-1}, w_l))(1\ 1\ \mu(w_k, w_l)) \ \in Runs(\mathcal{P}^{\mathbf{S}})
\end{gathered}
$$

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

Remark 4.4 Bouajjani et al. developed a technique to compute the transitive closure of a regular relation R [7,14]. A sufficient criterion for termination of that computation is *bounded local depth* [7,14] 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 (lb, lc) will be a prefix of $(0,0)^* (1,0)^+ (1,1)^+$. Furthermore, $\hat{w}$ changes its value in $\rho^{\mathbf{S}}$ at most once

at the transition of (lb, lc) from $(0, 0)$ to $(1, 0)$. Hence, with similar reasoning as for radius and diameter in [19] we can infer that, if R has local depth k, $R^{\mathbf{S}}$ has local depth $\leq 3k + 2$.

Remark 4.5 The ideas of regular model checking have been extended to infinite words [5] by regarding the finite automata used to represent sets of states and the transition relation as Büchi automata on infinite words. The techniques of [5] 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 [11]. A *pushdown system* M is a four tuple $M = (P, \Gamma, \Delta, C_I)$ where P is a finite set of *control locations*, Γ 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 ρ 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 ρ.

Bouajjani et al. proved [6] 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 ρ 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 [6] 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, *lc* signals a second occurrence of a state 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 *lb*, *lc*, and the stored head. Hence, there is no need for an initial configuration with that configuration already saved.

Definition 5.1 shows the entire reduction. The transition relation is partitioned into 5 sets again. While no state has been saved (subset (1)), flags *lb*, *lc*, and *le* remain false, 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 (p, γ) (in the pre-configuration) in $\hat{p}$ and $\hat{\gamma}$ (in the post-configuration), sets *lb* to true, 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, the flags *lb*, *lc*, 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 when a second occurrence of the stored head could be detected 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 (p, γ) is signalled by setting *lc* to true when taking a transition from subset (4). *lb*, *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 ρ to a repeating head $\langle p_0, \gamma \rangle$*

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

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

in M if and only if there exists an initialized run $\rho^{\mathbf{S}}$ in $M^{\mathbf{S}}$ with $\rho^{\mathbf{S}}(|\rho^{\mathbf{S}}| - 2) = \langle (p_0, p_0, \gamma, 1, 0, 0), w_{|\rho^{\mathbf{S}}|-2} \rangle$, where $w_{|\rho^{\mathbf{S}}|-2}(0) = \gamma$, and $\rho^{\mathbf{S}}(|\rho^{\mathbf{S}}| - 1) = \langle (p, p_0, \gamma, 1, 1, 0), w_{|\rho^{\mathbf{S}}|-1} \rangle$.

Proof. "⇒": Assume a run ρ 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 $k > l$, $p_1, \dots, p_{k-l-1} \in P$, $u_0, \dots, u_{k-l} \in \Gamma^+$, where $u_0 = u_{k-l}(0) = \gamma$, such that ρ can be extended to an infinite run $\rho^\infty \in Runs(M)$:

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

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

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

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

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

otherwise

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

"⇐": Assume an initialized run $\rho^{\mathbf{S}}$ to $\rho^{\mathbf{S}}(|\rho^{\mathbf{S}}|-2) = \langle (p_0, p_0, \gamma, 1, 0, 0), w_{|\rho^{\mathbf{S}}|-2} \rangle$, where $w_{|\rho^{\mathbf{S}}|-2}(0) = \gamma$, and $\rho^{\mathbf{S}}(|\rho^{\mathbf{S}}| - 1) = \langle (p_1, p_0, \gamma, 1, 1, 0), w_{|\rho^{\mathbf{S}}|-1} \rangle$. By Def. 5.1, $\exists 0 < l < |\rho^{\mathbf{S}}| - 2$ such that $\rho^{\mathbf{S}}(l) = \langle (p_0, \hat{p}_I, \hat{\gamma}_I, 0, 0, 0), \mu(w_l, 0^{|w_l|}) \rangle$ 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 three flags. 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). □

Algorithm 3 in [11] can be used to check 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 reachable configurations in $\mathbf{O}(|P||\Delta|^2 + |\delta|)$ time and space.

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 [11] runs on $M^{\mathbf{S}}$, with $A_{M^{\mathbf{S}}}$ accepting ${C_I}^{\mathbf{S}}$, in time and space*

$$\mathbf{O}(|P||\Gamma|(|P||\Delta|^2) + |\delta|)$$

Proof. See the full version of this paper. □

6 Timed Automata

In this section we briefly give the idea of how to apply our reduction to timed automata [3]. Details can be found in the full version of this paper. 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 [3] 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 $\{0, \ldots, c_x\}$ 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 $0 \ldots k-1$ if there are k clocks and $k-1$ flags to indicate equality between each pair of successors in the order.

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 [18] or simulations [8] 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: R. Alur and D. Peled, editors, *CAV'04*, LNCS **3114** (2004), pp. 348–360.

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

[4] 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.

[5] 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.

[6] 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.

[7] Bouajjani, A., B. Jonsson, M. Nilsson and T. Touili, *Regular model checking*, in: Emerson and Sistla [10], pp. 403–418.

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

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

[10] 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.

[11] Esparza, J., D. Hansel, P. Rossmanith and S. Schwoon, *Efficient algorithms for model checking pushdown systems*, in: Emerson and Sistla [10], pp. 232–247.

[12] 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.

[13] 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).

[14] 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.

[15] 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.

[16] Kupferman, O. and M. Vardi, *Model checking of safety properties*, in: N. Halbwachs and D. Peled, editors, *CAV'99*, LNCS **1633** (1999), pp. 172–183.

[17] 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.

[18] Pnueli, A. and E. Shahar, *Liveness and acceleration in parameterized verification*, in: Emerson and Sistla [10], pp. 328–343.

[19] 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.

[20] 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.

[21] Sebastiani, R., S. Tonetta and M. Vardi, *Symbolic systems, explicit properties: on hybrid approaches for LTL symbolic model checking*, in: *CAV'05*, 2005, to appear.

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

[23] 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.