Shortest Counterexamples for Symbolic Model Checking of LTL with Past

Viktor Schuppan[1] and Armin Biere[2]

[1] ETH Zürich, Computer Systems Institute
CH-8092 Zürich, Switzerland, `Viktor.Schuppan@inf.ethz.ch`
[2] Johannes Kepler University, Institute for Formal Models and Verification
Altenbergerstrasse 69, A-4040 Linz, Austria, `biere@jku.at`

Abstract. Shorter counterexamples are typically easier to understand. The length of a counterexample, as reported by a model checker, depends on both the algorithm used for state space exploration and the way the property is encoded. We provide necessary and sufficient criteria for a Büchi automaton to accept shortest counterexamples. We prove that Büchi automata constructed using the approach of Clarke, Grumberg, and Hamaguchi accept shortest counterexamples of future time LTL formulae, while an automaton generated with the algorithm of Gerth et al. (GPVW) may lead to unnecessary long counterexamples. Optimality is lost in the first case as soon as past time operators are included. Adapting a recently proposed encoding for bounded model checking of LTL with past, we construct a Büchi automaton that accepts shortest counterexamples for full LTL. We use our method of translating liveness into safety to find shortest counterexamples with a BDD-based symbolic model checker without modifying the model checker itself. Though our method involves a quadratic blowup of the state space, it outperforms SAT-based bounded model checking on a number of examples.

1 Introduction

Counterexamples are a salient feature of model checking that help developers to understand the problem in a faulty design. Most counterexamples still need to be interpreted by humans, and shorter counterexamples will, in general, be easier to understand.

As LTL is defined over infinite paths counterexamples are, in principle, infinitely long. In a finite state system every failing LTL property also has a lasso-shaped counterexample $\beta\gamma^\omega$ [27]. Such a counterexample can be finitely represented, where its length is defined as the sum of the lengths of the stem β and loop γ [7]. Counterexamples to safety properties also have finite bad prefixes that are more useful for a developer than a corresponding infinite path. In [17] Kupferman and Vardi showed how to recognize the shortest bad prefix using an automaton of size doubly exponential in the size of the corresponding formula. In this paper we concentrate on shortest lasso-shaped counterexamples for general LTL properties.

BDD-based symbolic techniques usually proceed breadth first and can find shortest bad prefixes for many safety properties [17]. For more general specifications, finding a shortest counterexample amounts to finding a shortest fair cycle, which is an NP-complete problem [7]. Most BDD-based model checkers offer only heuristics to minimize the length of counterexamples to such properties. For a comparative study on their performance and the length of the generated counterexamples see [20]. In explicit state model checking a double DFS [8] is typically used to search the state space. It does not find shortest counterexamples. Gastin et al. propose an algorithm [11] to minimize the length of counterexamples, which may visit a state an exponential number of times.

The first technique in widespread use that can produce shortest counterexamples for general LTL properties is SAT-based bounded model checking [3]. While [3] was restricted to future time LTL, more recent implementations cover full LTL [2], [5], [19]. Whether shortest counterexamples can be reported depends also on the encoding of the property. Both, [2] and [19] find shortest counterexamples. [5] achieves higher performance than [2] but sacrifices shortest counterexamples. A detailed experimental comparison of [5] and [19] is not yet available. As SAT-based model checking does not perform equally well on all examples as the BDD-based variant and vice versa [1], an efficient BDD-based technique that produces shortest counterexamples is desirable.

We recently proposed a method to translate liveness into safety [22], which finds shortest lassos and performs well on a number of examples in a BDD-based model checker. The automaton-based approach to model checking [27] employs such loop detection but requires translation of an LTL property into a Büchi automaton. Hence, not only must the shortest lasso be found, but the property automaton must also accept a shortest counterexample [11, 1]. So far, size of Büchi automata was a more important criterion than length of the resulting counterexamples, and little is known about the latter.

In this paper we establish necessary and sufficient criteria for Büchi automata to accept shortest counterexamples. We prove that the approach by Clarke et al. [6] generates Büchi automata that satisfy these criteria for future time LTL. This is not the case if past time is included, and we establish a quadratic bound on the excess length. We give an example that the algorithm by Gerth et al. [12] and many of its descendants do not generate shortest counterexamples even for future time LTL.

Past time operators do not add expressive power to future time LTL [15]. Still, a specification that includes past time operators may be more natural than the pure future variant, and it can be exponentially more succinct [18]. We are not aware of an efficient, easy-to-implement algorithm to translate a past time LTL formula into its future time equivalent. We instead construct a Büchi automaton that accepts shortest counterexamples for full LTL by adapting a recent, simple and efficient encoding for bounded model checking with past [19]. We then use our transformation from liveness to safety to find shortest counterexamples with a BDD-based symbolic model checker. The transformation itself does not require modifications to the model checker but is purely on the model and the specification to be checked. The only requirement is a breadth-first reachability check. Our experiments show that finding shortest counterexamples in the transformed model with the BDD-based algorithm of NuSMV [4] can be significantly faster than SAT-based bounded model checking of the original model.

In the following section we introduce our notation. In Sect. 3 we define shortest counterexamples and investigate which Büchi automata can accept them. We present our construction of a Büchi automaton that accepts shortest counterexamples in Sect. 4 and give some hints on our implementation in Sect. 5. Experimental results are reported in Sect. 6. The last section concludes.

2 Preliminaries

Let Σ be a finite set, let α be a finite or infinite sequence over Σ. The *length* of a sequence α is defined as $|\alpha| = n+1$ if $\alpha = \sigma_0\sigma_1\ldots\sigma_n$ is finite, ∞ otherwise. $\alpha(i)$ denotes the element at index i, α_i is the suffix $\alpha(i)\alpha(i+1)\ldots$ of α with its first i states chopped off. We also call sequences over Σ *words* over Σ. The crossproduct of two sequences $\alpha \times \beta$ is defined componentwise.

Let β, γ be finite sequences. A sequence α is a $\langle\beta,\gamma\rangle$*-lasso* with *stem* β and *loop* γ iff $\alpha = \beta\gamma^\omega$. We sometimes write $\langle\alpha,\beta\rangle$ instead of $\alpha\beta^\omega$. The *length* of a lasso is defined as $|\langle\beta,\gamma\rangle| = |\beta| + |\gamma|$. A lasso $\langle\beta,\gamma\rangle$ is *minimal* for α iff $\alpha = \beta\gamma^\omega$ and $\forall\beta',\gamma' \,.\, \alpha = \beta'\gamma'^\omega \Rightarrow |\langle\beta,\gamma\rangle| \leq |\langle\beta',\gamma'\rangle|$. The *type* [18] of a $\langle\beta,\gamma\rangle$-lasso is defined as $type(\langle\beta,\gamma\rangle) = (|\beta|,|\gamma|)$. A sequence α can be mapped to a set of types: $type(\alpha) = \{type(\langle\beta,\gamma\rangle) \mid \alpha = \beta\gamma^\omega\}$. We state the following fact about sequences (proved in the full version of this paper [23]).

Lemma 1. *Let $\langle\beta,\gamma\rangle$ be a minimal lasso for α, $\langle\beta',\gamma'\rangle$ a minimal lasso for α', and $\alpha'' = \alpha \times \alpha'$. Then there are finite sequences β'',γ'' such that $\langle\beta'',\gamma''\rangle$ is a minimal lasso for α'', $|\beta''| = max(|\beta|,|\beta'|)$, and $|\gamma''| = lcm(|\gamma|,|\gamma'|)$*[3].

2.1 Kripke Structures

Following [16] we define a fair *Kripke structure* as tuple $K = (V,I,T,F)$. V is a finite set of *state variables* v_i, each ranging over a finite set V_i. A *state* s is a valuation of the variables in V, the set of all states is S. I is the *initial condition* that defines the set of initial states of K. The *transition relation* T is also given as a predicate, referring to valuations of the variables in the current state, s, and in the successor state, s'. $F = \{F_1,\ldots,F_n\}$ is a set of (weak) fairness constraints. The value of v in s is denoted by $v(s)$. If s is clear from the context, v also denotes the value of v in the current state, and v' that in the successor state. We assume a set of atomic propositions AP that relates variables to their potential valuations, each of the form $v_i = c_j$ with $c_j \in V_i$. A mapping L is implicitly given that maps a state s to the set of atomic propositions true in s.

A non-empty sequence of states is a *path* in K if $\forall 0 \leq i < |\pi| \,.\, (s_i,s_{i+1}) \models T$. If $s_0 \models I$, π is *initialized*. An infinite path π is *fair* if $\forall F_i \in F \,.\, \forall j \,.\, \exists k > j \,.\, \pi(k) \models F_i$. Π is the set of paths in K. Via L a path implicitly defines a sequence over 2^{AP}.

The synchronous product of two Kripke structures $K_1 = (V_1,I_1,T_1,F_1)$ and $K_2 = (V_2,I_2,T_2,F_2)$ is a Kripke structure $K_1 \times K_2 = (V_1 \cup V_2, I_1 \wedge I_2, T_1 \wedge T_2, F_1 \cup F_2)$. The projection of a state s onto a set of variables V' is denoted $s|_{V'}$.

2.2 PLTL

We consider specifications given in Propositional LTL with both future and past time operators (PLTLB) [9]. The syntax of PLTLB is defined over a set of atomic propositions AP. If ϕ and ψ are PLTLB formulae, so are $\neg\phi$, $\phi \vee \psi$, $\mathbf{X}\phi$, $\phi\,\mathbf{U}\,\psi$, $\mathbf{Y}\phi$, $\phi\,\mathbf{S}\,\psi$. The semantics of PLTLB is defined recursively on infinite sequences over 2^{AP} in Fig. 1.

If the past time operators **Y** and **S** are excluded, we obtain future time LTL formulae (PLTLF). Similarly, a past time formula (PLTLP) has no occurrences of **X** and **U**. For

[3] $lcm(a,b)$ denotes the *least common multiple* of a and b.

$\pi_i \models p$	iff $p \in \pi_i$ for $p \in AP$	$\pi_i \models \mathbf{X}\phi$	iff $\pi_{i+1} \models \phi$
$\pi_i \models \neg\phi$	iff $\pi_i \not\models \phi$	$\pi_i \models \phi\,\mathbf{U}\,\psi$	iff $\exists j \geq i\;.\;(\pi_j \models \psi \wedge \forall i \leq k < j\;.\;\pi_k \models \phi)$
$\pi_i \models \phi \vee \psi$	iff $\pi_i \models \phi$ or $\pi_i \models \psi$	$\pi_i \models \mathbf{Y}\phi$	iff $i > 0$ and $\pi_{i-1} \models \phi$
		$\pi_i \models \phi\,\mathbf{S}\,\psi$	iff $\exists 0 \leq j \leq i\;.\;(\pi_j \models \psi \wedge \forall j < k \leq i\;.\;\pi_k \models \phi)$

Fig. 1. The semantics of PLTLB

this reason, when we speak about future or past we include present. We have the following usual abbreviations: $\top \equiv p \vee \neg p$, $\bot \equiv \neg\top$, $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$, $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$, $\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$, $\phi\;\mathbf{R}\;\psi \equiv \neg(\neg\phi\;\mathbf{U}\;\neg\psi)$, $\mathbf{F}\phi \equiv \top\;\mathbf{U}\;\phi$, $\mathbf{G}\phi \equiv \neg\mathbf{F}\neg\phi$, $\mathbf{Z}\phi \equiv \neg\mathbf{Y}\neg\phi$, $\phi\;\mathbf{T}\;\psi \equiv \neg(\neg\phi\;\mathbf{S}\;\neg\psi)$, $\mathbf{O}\phi \equiv \top\;\mathbf{S}\;\phi$, and $\mathbf{H}\phi \equiv \neg\mathbf{O}\neg\phi$.

A PLTLB property ϕ *holds universally* in a Kripke structure K, denoted $K \models_\forall \phi$, iff it holds for every initialized fair path. If $K \not\models_\forall \phi$, each initialized fair path π in K with $\pi \models \neg\phi$ is a *counterexample* for ϕ. ϕ *holds existentially*, $K \models_\exists \phi$, iff there exists an initialized fair path that fulfills ϕ. Each such path is a *witness* for ϕ. For every finite K, if $K \not\models_\forall \phi$, then there exists a fair $\langle\beta,\gamma\rangle$-lasso α in K such that $\alpha \not\models \phi$ [27]. A finite path π_{pre} is a *bad prefix* for ϕ iff $\forall\pi_{inf}\;.\;(|\pi_{inf}| = \infty \Rightarrow \pi_{pre}\pi_{inf} \not\models \phi)$ [17].

For **U** and **S** there exist recursive expansion formulae (e.g. [16]):

$$\phi = \psi_1\;\mathbf{U}\;\psi_2 : \pi_i \models \phi \text{ iff } (\pi_i \models \psi_2) \vee (\pi_i \models \psi_1) \wedge (\pi_{i+1} \models \phi)$$
$$\phi = \psi_1\;\mathbf{S}\;\psi_2 : \pi_i \models \phi \text{ iff } (\pi_i \models \psi_2) \vee (i > 0) \wedge (\pi_i \models \psi_1) \wedge (\pi_{i-1} \models \phi)$$

The expansion of **U** is not sufficient to guarantee proper semantics: additional measures must be taken to select the desired fixed point, e.g., by adding fairness constraints.

Finally, the *past operator depth* [2] of a formula ϕ, $h(\phi)$, is the maximal number of nested past operators in ϕ:

$$h(\phi) = \begin{cases} 0 & \text{iff } \phi \in AP \\ h(\psi) & \text{iff } \phi = \circ\psi \text{, where } \circ \in \{\neg, \mathbf{X}\} \\ max(h(\psi_1), h(\psi_2)) & \text{iff } \phi = \psi_1 \circ \psi_2 \text{, where } \circ \in \{\vee, \mathbf{U}\} \\ 1 + h(\psi) & \text{iff } \phi = \mathbf{Y}\psi \\ 1 + max(h(\psi_1), h(\psi_2)) & \text{iff } \phi = \psi_1\;\mathbf{S}\;\psi_2 \end{cases}$$

The authors of [18, 2] proved independently that a PLTLB property ϕ can distinguish at most $h(\phi)$ loop iterations of a lasso. We restate Lemma 5.2 of [18] for PLTLB:

Lemma 2. *For any lasso π of type (l_s, l_l), for any PLTLB property ϕ with at most $h(\phi)$ nested past-time modalities, and any $i \geq l_s + l_l h(\phi)$, $\pi_i \models \phi \Leftrightarrow \pi_{i+l_l} \models \phi$.*

2.3 Büchi Automata

A *Büchi automaton* over a set of variables V^K with a corresponding set of states S^K is a Kripke structure $B = (V, I, T, F)$, where $V = V^K \cup \hat{V}$. A *run* ρ of a Büchi automaton B on an infinite word α over S^K, denoted $\rho \models \alpha$, is an initialized fair path in B such that $\forall i\;.\;\alpha(i) = \rho(i)|_{V^K}$. The set of all runs of B is $Runs(B)$. A word is *accepted* by B iff B has a run on α. The set of words accepted by B defines its *language* $Lang(B)$.

ψ	definition			
	$V^{\psi} =$	$I^{\psi} =$	$T^{\psi} =$	$F^{\psi} =$
p	$\{x_p\}$	$\top$	$x_p \leftrightarrow p$	$\emptyset$
$\neg\psi_1$	$V^{\psi_1} \cup \{x_\psi\}$	I^{ψ_1}	$T^{\psi_1} \wedge (x_\psi \leftrightarrow \neg x_{\psi_1})$	F^{ψ_1}
$\psi_1 \vee \psi_2$	$V^{\psi_1} \cup V^{\psi_2} \cup \{x_\psi\}$	$I^{\psi_1} \wedge I^{\psi_2}$	$T^{\psi_1} \wedge T^{\psi_2} \wedge$ $(x_\psi \leftrightarrow x_{\psi_1} \vee x_{\psi_2})$	$F^{\psi_1} \cup F^{\psi_2}$
$\mathbf{X}\psi_1$	$V^{\psi_1} \cup \{x_\psi\}$	I^{ψ_1}	$T^{\psi_1} \wedge (x_\psi \leftrightarrow x'_{\psi_1})$	F^{ψ_1}
$\psi_1 \mathbf{U} \psi_2$	$V^{\psi_1} \cup V^{\psi_2} \cup \{x_\psi\}$	$I^{\psi_1} \wedge I^{\psi_2}$	$T^{\psi_1} \wedge T^{\psi_2} \wedge$ $(x_\psi \leftrightarrow x_{\psi_2} \vee x_{\psi_1} \wedge x'_\psi)$	$F^{\psi_1} \cup F^{\psi_2} \cup$ $\{\{\neg x_\psi \vee x_{\psi_2}\}\}$
$\mathbf{Y}\psi_1$	$V^{\psi_1} \cup \{x_\psi\}$	$I^{\psi_1} \wedge (x_\psi \leftrightarrow \bot)$	$T^{\psi_1} \wedge (x'_\psi \leftrightarrow x_{\psi_1})$	F^{ψ_1}
$\psi_1 \mathbf{S} \psi_2$	$V^{\psi_1} \cup V^{\psi_2} \cup \{x_\psi\}$	$I^{\psi_1} \wedge I^{\psi_2} \wedge$ $(x_\psi \leftrightarrow x_{\psi_2})$	$T^{\psi_1} \wedge T^{\psi_2} \wedge$ $(x'_\psi \leftrightarrow x'_{\psi_2} \vee x'_{\psi_1} \wedge x_\psi)$	$F^{\psi_1} \cup F^{\psi_2}$

Table 1. Property-dependent part of a Büchi automaton constructed with CGH+

In the automaton-based approach to model checking [27] a Büchi automaton that recognizes counterexamples to the specification is constructed. In other words, the language of the automaton is precisely the set of witnesses for the negation of the specification. Then, an initialized fair path in the synchronous product of the model and that automaton indicates failure of the specification. Formally, to check whether $K \models_\forall \phi$ holds for some model K and LTL formula ϕ, we negate ϕ and construct a Büchi automaton $B^{\neg\phi}$ with $Lang(B^{\neg\phi}) = \{\alpha \mid \alpha \models \neg\phi\}$. Any initialized fair path in $K \times B^{\neg\phi}$ is a counterexample for ϕ.

In this scenario V^K corresponds to the set of atomic propositions in $\neg\phi$, whereas $\hat{V}$ depends on the specific algorithm used to obtain B. Our definition of a Büchi automaton is similar to a state-labeled, generalized Büchi automaton but splits states according to the variables in V^K. This is more convenient in a symbolic setting, where this split happens anyway when the synchronous product with the model automaton is formed. It does not restrict the generality of the results in Sect. 3 and 4.

An approach to construct a Büchi automaton tailored to symbolic model checking (used, e.g., in NuSMV [4]) is by Clarke, Grumberg, and Hamaguchi [6]. The original version deals only with future time formulae, but extensions to PLTLB are available, see. e.g., [16, 21]. We refer to this extended version as *CGH+* below. An automaton B^ϕ_{CGH+} is constructed as $B^\phi_{CGH+} = (V^\phi, I^\phi \wedge x_\phi, T^\phi, F^\phi)$ where V^ϕ, I^ϕ, T^ϕ, and F^ϕ are defined recursively in Tab. 1. All x_ψ are Boolean. On every run ρ on a word α the valuation of a state variable x_ψ of B^ϕ_{CGH+} reflects the validity of the corresponding subformula ψ of ϕ, i.e., $x_\psi(\rho(i)) \leftrightarrow \alpha_i \models \psi$. By [6, 16, 21] we have $Lang(B^\phi_{CGH+}) = \{\alpha \mid \alpha \models \phi\}$. Note that, for a uniform explanation, Tab. 1 uses state variables also for Boolean connectives. In [6, 16, 21] these are replaced by macros.

3 Büchi Automata to Detect Shortest Counterexamples

3.1 Shortest Counterexamples for PLTLB

We have defined PLTLB over infinite paths, hence we need to specify what should be considered a shortest counterexample. Given that we are only interested in finite representations, and a failing PLTLB property in a finite state system always has a lasso-shaped counterexample [27], we adopt the following definition from [7]: a shortest counterexample is one that has a most compact representation as a lasso.

Definition 1. *Let $K = (V, I, T, F)$ be a Kripke structure, let ϕ be a PLTLB property. A path α in K is a* shortest counterexample *for ϕ in K iff*

1. $\alpha \not\models \phi$
2. $\exists \beta, \gamma \,.\, (\alpha = \beta\gamma^\omega \wedge \forall \beta', \gamma' \,.\, (\beta'\gamma'^\omega \in \Pi \wedge \beta'\gamma'^\omega \not\models \phi \Rightarrow |\langle \beta, \gamma \rangle| \leq |\langle \beta', \gamma' \rangle|))$

This definition is not optimal. First, an early position of the violation (if that can be clearly attributed) need not coincide with the least number of states required to close a loop. Second, apart from length, ease of understanding is not a criterion either.

The first problem is most relevant for properties that also have finite bad prefixes, i.e., properties that are a subset of a safety property [17]. Finding the shortest bad prefix for safety formulae can be done in parallel, using the (doubly exponential) method proposed in [17]. The solution to the second problem is left as future work; for approaches and more references see [13].

3.2 Tight Büchi Automata

In the automaton-based approach to model checking, a PLTLB property is verified by searching for loops in the synchronous product of a Kripke structure K, representing the model, and a Büchi automaton B, accepting counterexamples for the property. Hence, if shortest counterexamples are desired, the product of the model and the Büchi automaton must have an initialized fair path $\lambda = \langle \mu, \nu \rangle$ that can be represented as lasso of the same length as the shortest counterexample $\alpha = \langle \beta, \gamma \rangle$. Kupferman and Vardi [17] call an automaton on finite words *tight* if it accepts shortest prefixes for violations of safety formulae. We extend that notion to Büchi automata on infinite words.

Definition 2. *Let B be a Büchi automaton. B is* tight *iff*

$$\forall \alpha \in \mathit{Lang}(B) \,.\, \forall \beta, \gamma \,.\, (\langle \beta, \gamma \rangle \text{ is minimal for } \alpha \Rightarrow$$
$$\exists \rho \in \mathit{Runs}(B) \,.\, \exists \lambda, \mu, \nu \,.\, (\rho \models \alpha \;\wedge\; \lambda = \alpha \times \rho = \mu\nu^\omega \;\wedge\; |\langle \mu, \nu \rangle| = |\langle \beta, \gamma \rangle|))$$

Consider the scenarios in Fig. 2. The automaton B in the left scenario has a run $\sigma\tau^\omega$ of the same structure as the counterexample $\beta\gamma^\omega$ in K, leading to an equally short counterexample $(\beta \times \sigma)(\gamma \times \tau)^\omega$ in the product $K \times B$. The run of the Büchi automaton in the right scenario has an unnecessarily long stem and loop.

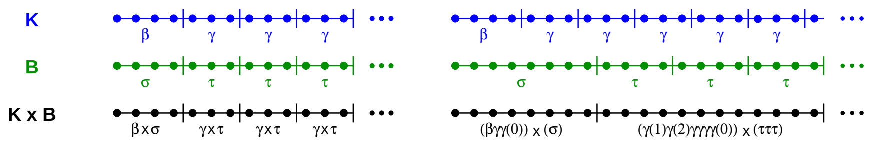


Fig. 2. Scenarios with shortest and non-optimal counterexample

From Lemma 1 it can be inferred that a path of the same length in $K \times B$ as the counterexample in K implies that the corresponding run $\rho = \sigma\tau^\omega$ in B can be represented as the same type as $\langle\beta,\gamma\rangle$. The left scenario in Fig. 2 suggests another, alternative formulation, which may be more intuitive and is easier to prove for some automata: the subsequences of α starting at indices $4,7,10,\ldots$ are the same, as are those beginning at $5,7,11,\ldots$, and $6,9,12,\ldots$. On the other hand, the subsequences starting at the respective indices in a single iteration are all different — otherwise a part of the loop could be cut out, contradicting minimality. Hence, if B is tight, there must be a run ρ on α with the following property: for each pair of indices i,j, if the subsequences of α starting at i and j have the same future ($\alpha_i = \alpha_j$), then ρ maps i and j to the same state in B ($\rho(i) = \rho(j)$). Theorem 1 establishes the equivalence of the criteria.

Theorem 1. *Let B be a Büchi automaton. The following statements are equivalent:*

1. *B is tight.*
2. $\forall\alpha \in \mathit{Lang}(B)\ .\ \forall\beta,\gamma\ .\ (\langle\beta,\gamma\rangle \text{ is minimal for } \alpha \Rightarrow$
   $\exists\rho \in \mathit{Runs}(B)\ .\ (\rho \models \alpha \wedge \mathit{type}(\langle\beta,\gamma\rangle) \in \mathit{type}(\rho)))$
3. $\forall\alpha \in \mathit{Lang}(B)\ .\ ((\exists\beta,\gamma\ .\ \alpha = \beta\gamma^\omega) \Rightarrow$
   $(\exists\rho \in \mathit{Runs}(B)\ .\ (\rho \models \alpha \wedge (\forall i,j\ .\ \alpha_i = \alpha_j \Rightarrow \rho(i) = \rho(j)))))$

Proof. $1 \Rightarrow 2$: Assume a run $\rho = \sigma\tau^\omega$ such that $\lambda = \alpha \times \rho = \mu\nu^\omega$ with $|\langle\mu,\nu\rangle| = |\langle\beta,\gamma\rangle|$. Let $\langle\sigma',\tau'\rangle$ be minimal for ρ. Lemma 1 gives $|\sigma'| \leq |\beta|$ and $|\tau'|$ divides $|\gamma|$. Now it's easy to find σ'',τ'' with $\sigma''\tau''^\omega = \sigma\tau^\omega$, and $\mathit{type}(\langle\sigma'',\tau''\rangle) = \mathit{type}(\langle\beta,\gamma\rangle)$.

$2 \Rightarrow 1$: Assume a run ρ with $\mathit{type}(\langle\beta,\gamma\rangle) \in \mathit{type}(\rho)$. By definition of type, there exist σ,τ such that $\rho = \sigma\tau^\omega$, $|\beta| = |\sigma|$, and $|\gamma| = |\tau|$. Hence, with $\mu = \beta \times \sigma$ and $\nu = \gamma \times \tau$, we have $\lambda = \alpha \times \rho = \mu\nu^\omega$ and $|\langle\mu,\nu\rangle| = |\langle\beta,\gamma\rangle|$.

$2 \Rightarrow 3$: Let $\alpha \in \mathit{Lang}(B)$, assume $\langle\beta,\gamma\rangle$ minimal for α, and let $\rho = \sigma\tau^\omega$ be a run on α such that $|\beta| = |\sigma|$ and $|\gamma| = |\tau|$. Let i,j with $\alpha_i = \alpha_j$. It remains to show that $\rho(i) = \rho(j)$. This is done by case distinction according to the positions of i and j w.r.t. to β and γ in α. The case $i = j$ is obvious, the other cases either contradict the minimality of $\langle\beta,\gamma\rangle$ for α or can be reduced to a previous case. Details are given in the full version [23].

$3 \Rightarrow 2$: Let $\alpha = \beta\gamma^\omega \in \mathit{Lang}(B)$ and ρ a run on α with $\forall i,j\ .\ \alpha_i = \alpha_j \Rightarrow \rho(i) = \rho(j)$. Let $\langle\beta,\gamma\rangle$ be minimal for α.

$$\begin{aligned}\alpha = \beta\gamma^\omega &\Rightarrow \forall i < |\gamma|, \forall k\ .\ \alpha_{|\beta|+i} = \gamma_i = \alpha_{|\beta|+i+|\gamma|k}\\ &\Rightarrow \forall i < |\gamma|, \forall k\ .\ \rho(|\beta|+i) = \rho(|\beta|+i+|\gamma|k)\end{aligned}$$

Let $\sigma = \rho(0),\ldots,\rho(|\beta|-1)$ and $\tau = \rho(|\beta|),\ldots,\rho(|\beta|+|\gamma|-1)$. Hence, $\rho = \sigma\tau^\omega$ such that $|\sigma| = |\beta|$ and $|\tau| = |\gamma|$. □

3.3 (Non-) Optimality of Specific Approaches

The approach by Gerth et al. (GPVW) [12] for future time LTL forms the basis of many algorithms to construct small Büchi automata, which benefits explicit state model checking but is also used, e.g., for symbolic model checking in VIS [14]. Figure 3 shows an example that GPVW does not, in general, lead to tight automata. Subsequences starting from the initial state of the Büchi automaton fulfill $p \wedge \mathbf{XG}q$, those starting from the other state satisfy $\mathbf{G}q$. The model has a single, infinite path satisfying $\mathbf{G}(p \wedge q)$ — a counterexample of length 1 to the specification $\neg(p \wedge \mathbf{XG}q)$. Note that adding transitions or designating more initial states is not enough to make the automaton in Fig. 3 tight: an additional state is required. Non-optimality of GPVW is shared by many of its descendants, e.g., [26].

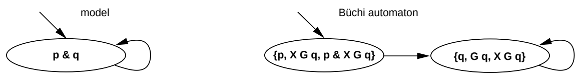


Fig. 3. Model and Büchi automaton to recognize counterexamples for $\neg(p \wedge \mathbf{XG}q)$ resulting in non-optimal counterexample

In a Büchi automaton B^{ϕ}_{CGH+} each state variable corresponds to a subformula ψ of ϕ (see Tab. 1). This directly proves tightness of B^{ϕ}_{CGH+} for a PLTLF formula ϕ.

Proposition 1. *Let ϕ be a future time LTL formula, let B^{ϕ}_{CGH+} be defined as above. Then B^{ϕ}_{CGH+} is tight.*

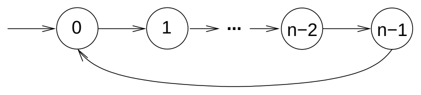
Proof. Every two states in B^{ϕ}_{CGH+} differ in the valuation of at least one state variable, and therefore specify a different, non-overlapping future. According to Thm. 1, a Büchi automaton B is tight iff for each accepted word α there exists a run ρ on α in B with $\forall i, j \,.\, (\alpha_i = \alpha_j \Rightarrow \rho(i) = \rho(j))$. Clearly, $\alpha_i = \alpha_j$ have the same future, hence, on each run in B we have $\alpha_i = \alpha_j \Rightarrow \rho(i) = \rho(j)$. □

What is useful for future time hurts tightness as soon as past operators are included: B^{ϕ}_{CGH+} may also distinguish states of an accepted word that have different past but same future. Lemma 2 states that a past time formula can distinguish only finitely many iterations of a loop. This can be used to establish an upper bound on the excess length of a counterexample produced by CGH+ for a PLTLB formula:

Proposition 2. *Let K be a Kripke structure, ϕ a PLTLB property with $K \not\models_{\forall} \phi$, and $B^{\neg\phi}_{CGH+}$ a Büchi automaton constructed with CGH+. Let $\alpha = \langle \beta, \gamma \rangle$ be a shortest counterexample in K. Then, there is an initialized fair lasso $\lambda = \langle \mu, \nu \rangle$ in $K \times B^{\neg\phi}_{CGH+}$ with $|\mu| \leq |\beta| + (h(\neg\phi) + 1)|\gamma|$ and $|\nu| = |\gamma|$.*

The proof is given in the full version [23]. For an example that exhibits excess length, which is quadratic in the length of the shortest counterexample, consider the simple modulo-n counter and property in Fig. 4 (adapted from [2]). The innermost formula $\mathbf{O}(c = n-1)$ remains true from the end of the first loop iteration in the counter, $\mathbf{O}((c = n-2) \wedge (\mathbf{O}(c = n-1)))$ becomes and remains true $n-1$ steps later, etc. Hence, a loop in $B^{\neg\phi}_{CGH+}$ is only reached after $\mathbf{O}(n^2)$ steps of the counter have been performed. Clearly, the shortest counterexample is a single iteration of the loop with $\mathbf{O}(n)$ steps.

Every PLTLB formula can be transformed into a future time LTL formula equivalent at the beginning of a sequence [10]. Due to [18] we can expect an at least exponential worst-case increase in the size of the formula. Rather than translating an LTL formula with past into a pure future version, we follow a different path in the next section.



$$\begin{array}{l}
\neg(\mathbf{F}(\mathbf{G}(\mathbf{O}((c=0)\wedge \\
\qquad\quad \mathbf{O}((c=1)\wedge \\
\qquad\qquad \dots \\
\qquad\qquad\quad \mathbf{O}(c=n-1) \\
\qquad\qquad \dots \\
\qquad\qquad) \\
\qquad\quad))))
\end{array}$$

Fig. 4. Simple modulo-*n* counter with property

4 A Tight Look at LTL Model Checking

Proposition 2 states that a Büchi automaton constructed with CGH+ accepts a shortest counterexample with a run that may have an overly long stem but a loop of the same length as that of the counterexample. Bounded model checking [3] has been extended recently to include past time operators [2, 5, 19]. Of these, [2, 19] use *virtual unrolling* of the transition relation to find shortest counterexamples if past time operators are present. Inspired by [19], we adapt this approach to construct a tight Büchi automaton for PLTLB based on CGH+.

4.1 Virtual Unrolling for Bounded Model Checking of PLTLB

In bounded model checking, the model checking problem, which asks whether $K \models_{\forall} \phi$ holds, is translated into a sequence of propositional formulae of the form $|[M,\phi,k]|$ in the following way: $|[M,\phi,k]|$ is satisfiable iff a finite informative bad prefix [17] or lasso-shaped counterexample π of length k exists. In the case of a lasso-shaped counterexample, a loop is assumed to be closed between the last state $\pi(k)$ and some successor $\pi(l+1)$ of a previous occurrence of that last state $\pi(l) = \pi(k)$. The resulting formulae are then handed to a SAT solver for increasing bounds k until either a counterexample is found, absence of a counterexample is proved, or a user defined resource threshold is reached. Typically, one fresh Boolean variable $x_{j,\psi}$ is introduced for each pair of relative position in the path ($0 \leq j \leq k$) and subformula ψ of ϕ, such that $x_{j,\psi}$ is true iff ψ holds at position j.

On a lasso-shaped path, the truth of a future time formula ϕ at position k may depend on the truth of some of its subformulae ψ at positions $> k$. While those are not available directly, the truth of a future time formula at a given position within the loop does not change between different iterations of the loop. Hence, the truth value of ψ at position $0 \leq m < k-l$ in any iteration $i \geq 0$ of the loop can be substituted with the truth value of ψ at position m in the first iteration: $\pi_{l+i(k-l)+m} \models \psi \Leftrightarrow \pi_{l+m} \models \psi$. A single unrolling of the loop is therefore sufficient, resulting in a shortest counterexample.

When past time operators are admitted, this is no longer true. By Lemma 2, the truth of a subformula ψ may change between the first $h(\psi)+1$ iterations of the loop before it stabilizes. Hence, only after $h(\psi)+1$ iterations can the truth value of ψ in some iteration $i \geq h(\psi)+1$ of the loop be replaced by the truth value of ψ in iteration $h(\psi)+1$: $\pi_{l+i(k-l)+m} \models \psi \Leftrightarrow \pi_{l+(h(\psi)+1)(k-l)+m} \models \psi$. A naive approach for checking a past time formula ϕ would still have one Boolean variable per pair of relative position in the path and subformula. However, the approach would have to ensure that the path ends with $h(\phi)+1$ copies of the loop. This would lead to a more complicated formulation of loop detection and would not allow to find shortest counterexamples. A less naive, but still suboptimal solution might not guarantee a high enough number of loop unrollings directly but could include the variables representing the truth of properties in the loop detection. That approach could not ensure shortest counterexamples either.

Benedetti and Cimatti [2] showed how to do better: note, that some subformulae ψ of ϕ have lower past operator depth, and, therefore, require fewer loop iterations to stabilize. In particular, atomic propositions remain stable from the first iteration onward. It is sufficient to perform a single unrolling of the loop. Rather than having only one Boolean variable $x_{j,\psi}$ per pair of relative position j in the path and subformula ψ, there are now as many variables per pair (j,ψ) as iterations of the loop are required for that subformula to stabilize. Each variable corresponds to the truth value of ψ at the same relative position j but in a different iteration i of the loop: $x_{j,\psi,i} \Leftrightarrow \pi_{j+i(k-l)} \models \psi$ with $0 \leq j \leq k \wedge 0 \leq i \leq h(\psi)$ (the value of $x_{j,\psi,i}$ may not be well-defined if $i > 0 \wedge j < l$). This *virtual unrolling* of the loop leads to shortest counterexamples.

4.2 A Tight Büchi Automaton for PLTLB

A Büchi automaton constructed with CGH+ suffers from similar problems as the naive approaches to bounded model checking of PLTLB. The automaton has a single variable representing the truth of a subformula in a given state. For a loop in the product of the model and the automaton to occur, the truth of all subformulae must have stabilized. Hence, we can adopt the same idea as outlined above to obtain a tight Büchi automaton.

We construct a Büchi automaton $B^{\phi}_{SB} = (V^{\phi}_{SB}, I^{\phi}_{SB}, T^{\phi}_{SB}, F^{\phi}_{SB})$ for a PLTLB formula ϕ as follows: $V^{\phi}_{SB} = V^{\phi} \cup \{lb, le\}$ with $LB = LE = \{\perp, \top\}$, $I^{\phi}_{SB} = I^{\phi} \wedge x_{\phi,0}$, $T^{\phi}_{SB} = T^{\phi} \wedge (lb \rightarrow lb')$, and $F^{\phi}_{SB} = F^{\phi} \cup \{\{lb \wedge le\}\}$, where V^{ϕ}, I^{ϕ}, T^{ϕ}, and F^{ϕ} are defined recursively in Tab. 2.

Each subformula ψ of ϕ is represented by $h(\psi)+1$ state variables $x_{\psi,i}$. We refer to the i in $x_{\psi,i}$ as *generation* below. Two more state variables *lb* (for *l*oop *b*ody) and *le* (for *l*oop *e*nd) are added. As long as *lb* is false (on the stem), only variables in generation 0 are constrained according to the recursive definition of PLTLB. When *lb* becomes true (on the loop), the definitions apply to all generations. While *le* is false (the end of a loop iteration is not yet reached), $x_{\psi,i}$ is defined in terms of current and next-state values of variables in the same generation. When *le* is true (at the end of a loop iteration), the next-state values are obtained from the next generation of variables if the present generation is not already the last. The fairness constraints, which guarantee the correct fixed point for **U** formulae, are only applied to the last generation of the corresponding variables.

The intuition is as follows. Starting with generation 0 on the stem and the first iteration of the loop, each generation i of $x_{\psi,i}$ represents the truth of ψ in one loop iteration, the end of which is signaled by $lb \wedge le$ being true. Formally, for $i < h(\psi)$, $x_{\psi,i}(j)$ holds the truth of ψ at position j of a word iff $lb \wedge le$ has been true on that path i times prior to the current state. From the $h(\psi)$-th occurrence of $lb \wedge le$, $x_{\psi,h(\psi)}$ continues to represent the truth of ψ.

Note that *lb* and *le* are oracles. The valuation of these variables on an arbitrary run may not correspond to the situations they are named after. However, for B^{ϕ}_{SB} to

ψ	definition
p	$V^{\psi} = \{x_{p,0}\}$, where $X_{p,0} = \{\bot, \top\}$ $T^{\psi} = x_{p,0} \leftrightarrow p$ $I^{\psi} = \top$ $\quad F^{\psi} = \emptyset$
$\neg\psi_1$	$V^{\psi} = V^{\psi_1} \cup \bigcup_{i=0}^{h(\psi)} \{x_{\psi,i}\}$, where $X_{\psi,i} = \{\bot, \top\}$ $T^{\psi} = T^{\psi_1} \wedge \bigwedge_{i=0}^{h(\psi)} (x_{\psi,i} \leftrightarrow \neg x_{\psi_1,i})$ $I^{\psi} = I^{\psi_1}$ $\quad F^{\psi} = F^{\psi_1}$
$\psi_1 \vee \psi_2$	$V^{\psi} = V^{\psi_1} \cup V^{\psi_2} \cup \bigcup_{i=0}^{h(\psi)} \{x_{\psi,i}\}$, where $X_{\psi,i} = \{\bot, \top\}$ $T^{\psi} = T^{\psi_1} \wedge T^{\psi_2} \wedge \bigwedge_{i=0}^{h(\psi)} (x_{\psi,i} \leftrightarrow x_{\psi_1,min(i,h(\psi_1))} \vee x_{\psi_2,min(i,h(\psi_2))})$ $I^{\psi} = I^{\psi_1} \wedge I^{\psi_2}$ $\quad F^{\psi} = F^{\psi_1} \cup F^{\psi_2}$
$\mathbf{X}\psi_1$	$V^{\psi} = V^{\psi_1} \cup \bigcup_{i=0}^{h(\psi)} \{x_{\psi,i}\}$, where $X_{\psi,i} = \{\bot, \top\}$ $T^{\psi} = T^{\psi_1} \wedge (\neg lb \rightarrow (x_{\psi,0} \leftrightarrow x'_{\psi_1,0}))$ $\wedge((lb \wedge \neg le) \rightarrow \bigwedge_{i=0}^{h(\psi)-1}(x_{\psi,i} \leftrightarrow x'_{\psi_1,i}))$ $\wedge((lb \wedge le) \rightarrow \bigwedge_{i=0}^{h(\psi)-1}(x_{\psi,i} \leftrightarrow x'_{\psi_1,i+1}))$ $\wedge(lb \rightarrow (x_{\psi,h(\psi)} \leftrightarrow x'_{\psi_1,h(\psi_1)}))$ $I^{\psi} = I^{\psi_1}$ $\quad F^{\psi} = F^{\psi_1}$
$\psi_1 \ \mathbf{U} \ \psi_2$	$V^{\psi} = V^{\psi_1} \cup V^{\psi_2} \cup \bigcup_{i=0}^{h(\psi)} \{x_{\psi,i}\}$, where $X_{\psi,i} = \{\bot, \top\}$ $T^{\psi} = T^{\psi_1} \wedge T^{\psi_2} \wedge (\neg lb \rightarrow (x_{\psi,0} \leftrightarrow x_{\psi_2,0} \vee (x_{\psi_1,0} \wedge x'_{\psi,0})))$ $\wedge((lb \wedge \neg le) \rightarrow \bigwedge_{i=0}^{h(\psi)-1}(x_{\psi,i} \leftrightarrow x_{\psi_2,min(i,h(\psi_2))} \vee (x_{\psi_1,min(i,h(\psi_1))} \wedge x'_{\psi,i})))$ $\wedge((lb \wedge le) \rightarrow \bigwedge_{i=0}^{h(\psi)-1}(x_{\psi,i} \leftrightarrow x_{\psi_2,min(i,h(\psi_2))} \vee (x_{\psi_1,min(i,h(\psi_1))} \wedge x'_{\psi,i+1})))$ $\wedge(lb \rightarrow (x_{\psi,h(\psi)} \leftrightarrow x_{\psi_2,h(\psi_2)} \vee (x_{\psi_1,h(\psi_1)} \wedge x'_{\psi,h(\psi)})))$ $I^{\psi} = I^{\psi_1} \wedge I^{\psi_2}$ $\quad F^{\psi} = F^{\psi_1} \cup F^{\psi_2} \cup \{\{\neg x_{\psi,h(\psi)} \vee x_{\psi_2,h(\psi_2)}\}\}$
$\mathbf{Y}\psi_1$	$V^{\psi} = V^{\psi_1} \cup \bigcup_{i=0}^{h(\psi)} \{x_{\psi,i}\}$, where $X_{\psi,i} = \{\bot, \top\}$ $T^{\psi} = T^{\psi_1} \wedge (\neg lb \rightarrow (x'_{\psi,0} \leftrightarrow x_{\psi_1,0}))$ $\wedge((lb \wedge \neg le) \rightarrow \bigwedge_{i=0}^{h(\psi)-1}(x'_{\psi,i} \leftrightarrow x_{\psi_1,i}))$ $\wedge((lb \wedge le) \rightarrow \bigwedge_{i=0}^{h(\psi)-2}(x'_{\psi,i+1} \leftrightarrow x_{\psi_1,i}))$ $\wedge(lb \rightarrow (x'_{\psi,h(\psi)} \leftrightarrow x_{\psi_1,h(\psi_1)}))$ $I^{\psi} = I^{\psi_1} \wedge (x_{\psi,0} \leftrightarrow \bot)$ $\quad F^{\psi} = F^{\psi_1}$
$\psi_1 \ \mathbf{S} \ \psi_2$	$V^{\psi} = V^{\psi_1} \cup V^{\psi_2} \cup \bigcup_{i=0}^{h(\psi)} \{x_{\psi,i}\}$, where $X_{\psi,i} = \{\bot, \top\}$ $T^{\psi} = T^{\psi_1} \wedge T^{\psi_2} \wedge (\neg lb \rightarrow (x'_{\psi,0} \leftrightarrow x'_{\psi_2,0} \vee (x'_{\psi_1,0} \wedge x_{\psi,0})))$ $\wedge((lb \wedge \neg le) \rightarrow \bigwedge_{i=0}^{h(\psi)-1}(x'_{\psi,i} \leftrightarrow x'_{\psi_2,min(i,h(\psi_2))} \vee (x'_{\psi_1,min(i,h(\psi_1))} \wedge x_{\psi,i})))$ $\wedge((lb \wedge le) \rightarrow \bigwedge_{i=0}^{h(\psi)-1}(x'_{\psi,i+1} \leftrightarrow x'_{\psi_2,min(i+1,h(\psi_2))} \vee (x'_{\psi_1,min(i+1,h(\psi_1))} \wedge x_{\psi,i})))$ $\wedge(lb \rightarrow (x'_{\psi,h(\psi)} \leftrightarrow x'_{\psi_2,h(\psi_2)} \vee (x'_{\psi_1,h(\psi_1)} \wedge x_{\psi,h(\psi)})))$ $I^{\psi} = I^{\psi_1} \wedge I^{\psi_2} \wedge (x_{\psi,0} \leftrightarrow x_{\psi_2,0})$ $\quad F^{\psi} = F^{\psi_1} \cup F^{\psi_2}$

Table 2. Property-dependent part of a tight Büchi automaton

correctly recognize $\{\alpha \mid \alpha \models \phi\}$, it is not relevant which generation holds the truth at a given position. It is only required that at each position some generation represents truth correctly, each generation passes on to the next at some point, and ultimately, depending on ψ, the last generation $h(\psi)$ continues to hold the proper values.

For tightness, the variables of a given generation need to be able to take on the same values in every iteration of the loop, regardless of whether they currently hold the truth or not. This requires breaking the links to previous iterations for variables of generation 0 representing **Y** and **S** formulae at each start of a loop iteration after the first. In addition, **Y**- and **S**-variables of generations > 0 may not be constrained by past values at the beginning of the loop body. On a shortest run on some lasso-shaped word α, *lb* and *le* will correctly signal loop body and loop end.

Theorem 2. *Let ϕ be a PLTLB formula, let B_{SB}^{ϕ} be defined as above. Then, $Lang(B_{SB}^{\phi}) = \{\alpha \mid \alpha \models \phi\}$ and B_{SB}^{ϕ} is tight.*

Proof. By Lemma 3 and 4. □

Lemma 3. $Lang(B_{SB}^{\phi}) = \{\alpha \mid \alpha \models \phi\}$

Proof. (Correctness) We show that on every fair path in $(V_{SB}^{\phi}, I^{\phi}, T_{SB}^{\phi}, F_{SB}^{\phi})$ the values of $x_{\psi,i_j}(j)$ represent the validity of the subformula ψ at position j, where i_j is either the number of *le*'s seen so far or $h(\psi)$, whichever is smaller. Formally, let ρ be a run on α in $(V_{SB}^{\phi}, I^{\phi}, T_{SB}^{\phi}, F_{SB}^{\phi})$. For each position j in α, let $i_j = min(|\{k \mid (k \leq j-1) \wedge lb(\rho(k)) \wedge le(\rho(k))\}|, h(\psi))$. Inspection of Tab. 2 shows that the constraints on the $x_{\psi,i_j}(j)$ are the same as the constraints on the corresponding $x_{\psi}(j)$ in Tab. 1. Hence, $\alpha_j \models \psi \Leftrightarrow x_{\psi,i_j}(\rho(j))$.

(Completeness) We show that there is a run in $(V_{SB}^{\phi}, I^{\phi}, T_{SB}^{\phi}, F_{SB}^{\phi})$ for each word α. Choose a set of indices $U = \{j_0, j_1, \ldots\}$ such that $le(j) \leftrightarrow j \in U$. Further, choose $ls \leq j_0$ and set $lb(j) \leftrightarrow j \geq ls$. We inductively construct a valuation for $x_{\psi,i}(j)$ for each subformula ψ of ϕ, $i \leq h(\psi)$, and $j \geq 0$. If ψ is an atomic proposition p, set $x_{p,0}(j) \leftrightarrow (\alpha_j \models p)$. If the top level operator of ψ is Boolean, the valuation follows directly from the semantics of the operator. For **X**, each $x_{\psi,i}(j)$ is defined exactly once in Tab. 2. $\psi = \mathbf{Y}\psi_1$ is similar. Note that $h(\psi) = h(\psi_1) + 1$. Therefore, i runs only up to $h(\psi) - 2$ if $lb \wedge le$; $i = h(\psi) - 1$ is covered by the case for lb in the line below. $x_{\psi,i}(j)$ is unconstrained if $i = 0$ and $j - 1 \in U$ as well as if $i \geq 1$ and $j \leq ls$. For $\psi = \psi_1 \mathbf{U} \psi_2$, start with generation $h(\psi)$. If $x_{\psi_2,h(\psi_2)}$ remains false from some j_m on, assign $\forall j \geq j_m \,.\, x_{\psi,h(\psi)}(j) \leftrightarrow \bot$. Now work towards decreasing j from each j_n with $x_{\psi_2,h(\psi_2)}(j_n) \leftrightarrow \top$, using line 4 in the definition of T for **U**. Continue with generation $h(\psi) - 1$. Start at each $j \in U$ by obtaining $x_{\psi,h(\psi)-1}(j)$ from the previously assigned $x_{\psi,h(\psi)}(j+1)$ via line 3. Then work towards decreasing j again, using lines 1 or 2 in the definition of T until $x_{\psi,h(\psi)-1}$ is assigned for all j. This is repeated in decreasing order for each generation $0 \leq i < h(\psi) - 1$. For **S**, start with $x_{\psi,0}(0)$ and proceed towards increasing j, also increasing i when $j \in U$ (lines 1 – 3 in the definition of T for **S**). When $i = h(\psi)$ is reached, assign $x_{\psi,h(\psi)}$ for all j using the fourth line in the definition of T. Then, similar to **U**, work towards decreasing i and j from each $j \in U$. Fairness follows from the definition of U, ls, and the valuation chosen for **U**.

The claim is now immediate by the definition of I_{SB}^{ϕ}. □

Lemma 4. *B_{SB}^{ϕ} is tight.*

Proof. We show inductively that the valuations of the variables $x_{\psi,i}(j)$ can be chosen such that the valuation at a given relative position in a loop iteration is the same for each iteration in a generation i. Formally, let $\alpha = \beta\gamma^{\omega}$ with $\alpha \models \phi$. There exists a run ρ such that for all subformulae ψ of ϕ

$$\forall i \leq h(\psi) \,.\, \forall j_1, j_2 \geq |\beta| \,.\, ((\exists k \geq 0 \,.\, j_2 - j_1 = k|\gamma|) \Rightarrow (x_{\psi,i}(\rho(j_1)) \leftrightarrow x_{\psi,i}(\rho(j_2))))$$

Atomic propositions, Boolean connectives, and **X** are clear. **Y** is also easy, we only have to assign the appropriate value from other iterations when $x_{\psi,i}(j)$ is unconstrained. For $\psi = \psi_1 \mathbf{U} \psi_2$, by the induction hypothesis, $x_{\psi_2,h(\psi_2)}$ is either always false (in which case we assigned $x_{\psi,h(\psi)}(j)$ to false according to the proof of Lemma 3) or becomes true at the same time in each loop iteration. Hence, the claim holds for generation $h(\psi)$. From there we can proceed to previous generations in the same manner as in the proof of Lemma 3. For **S** we follow the order of assignments from the proof of Lemma 3. By induction, the claim holds for generation $h(\psi)$. From there, we proceed towards decreasing j and i. We use, by induction, the same valuations of subformulae and the same equations (though in reverse direction) as we used to get from $x_{\psi,0}(0)$ to generation $h(\psi)$. □

B_{SB}^{ϕ} has $\mathbf{O}(2^{|\phi|^2})$ states. A symbolic representation can be constructed in $\mathbf{O}(|\phi|^2)$ time and space. Note, that the size of a Büchi automaton that is tight in the original sense of [17] (i.e., it recognizes shortest violating prefixes of safety properties) is doubly exponential in $|\phi|$ [17].

The same optimization as used in Sect. 2 for CGH+ can be applied. It replaces state variables for Boolean connectives with macros in order to reduce the number of BDD variables in the context of symbolic model checking with BDDs.

5 Finding Shortest Counterexamples with Symbolic Model Checking

We implemented the Büchi automaton described in the previous section for NuSMV [4]. We use our reduction of finite state model checking to reachability analysis [22] to find a shortest counterexample. For efficiency reasons, the encoding of the automaton is tightly integrated with the symbolic loop detection, which is at the heart of [22]. As an example, the signals for loop body and loop end are provided directly by the reduction rather than being separate input variables.

In fact, our implementation started as an adaptation of the very elegant encoding of PLTLB in [19] to our reduction. Only then we extracted a tight Büchi automaton from the construction. We kept our original implementation for its superior performance but chose to provide the more abstract view in the previous section, as, in our opinion, it provides better understanding and is also more widely applicable.

6 Experimental Results

In this section we compare our implementation to find shortest counterexamples with symbolic model checking from Sect. 5 with bounded model checking using the encoding of [19] and the standard LTL model checking algorithm of NuSMV [4]. For our translation, we performed invariant checking with NuSMV 2.2.2. For standard LTL, also in NuSMV 2.2.2, forward search on the reachable state space was applied. Bounded model checking was performed with the implementation of Timo Latvala in a modified NuSMV 2.1.2. If cone of influence reduction is to be used with our translation, the reduction must be applied before the translation. However, NuSMV 2.2.2 doesn't seem to provide a direct way to output the reduced model. Therefore, cone of influence reduction was disabled in all experiments. Otherwise, NuSMV 2.2.2 would find shorter loops, involving only the variables in the cone of the property, in the reduced model. Platform was an Intel Pentium IV at 2.8 GHz with 2 GB RAM running Linux 2.4.18. Timeout for each experiment was set to 1 hour, memory usage was limited to 1.5 GByte.

As the focus of this paper is on producing lasso-shaped counterexamples, only properties were chosen that proved false with such a counterexample. Results are shown in Tab. 3. The experiments include all real-world models used in [19]: abp4, brp, dme?, pci, and srg5. If the property checked in [19] has a lasso-shaped counterexample, it was used unmodified in our experiments ("L"). We also used the negated version of that property if that yields such a counterexample ("$\neg$ L"). Some of the properties were made a liveness property by prefixing them with **F** (requiring a loop to prove false) or were enhanced to make part of the property non-volatile (yielding a more interesting counterexample), marked "nv". In addition, we chose some of the models from our previous work [22], with some properties already verified there and with new, more complicated properties. Templates of the properties are shown in the full version of this paper [23].

Columns 3 – 5 give the results for standard LTL model checking ("LTL"): l is the length of the counterexample, time is in seconds, and memory usage in thousand BDD nodes allocated. The 6th col. gives the length of a shortest counterexample as reported by our translation and bounded model checking. Columns 7 and 8 give run time and memory usage for our algorithm ("L2S"). The last three columns indicate run time for bounded model checking ("BMC"). The first of these is the time for the last unsuccessful iteration of the bounded model checker alone (not yet producing a counterexample), the second is the time for the first successful iteration alone (giving the shortest counterexample), and the last column is the time for all bounds from 1 until a counterexample is found. The implementation of [19] is not incremental [25], i.e., the SAT solver cannot benefit from results of previous iterations. We use the time required for the last unsuccessful iteration ("Time $l-1$") to estimate the amount of work that an incremental implementation would at least have to do. If our algorithm needs less time than that, we conclude that our algorithm is faster. "t.o." or "m.o." indicate time- or memory-out.

Both, L2S and BMC, find significantly shorter counterexamples than LTL. Our algorithm often outperforms BMC with respect to time. On the other hand, L2S needs more memory than standard LTL in most cases. L2S may even give a speed up when compared to the standard algorithm on some examples.

7 Conclusions

We have presented a method to find shortest lasso-shaped counterexamples for full LTL. Experimental results show competitive performance with bounded model checking. We

		LTL				L2S		BMC		
model	property	l'	time	memory	l	time	memory	time $l-1$	time l	time $1 \ldots l$
1394-3-2	0	16	72.8	119	11	**7.9**	1267	9.3	3.1	54.3
	1	12	17.0	157	11	**6.8**	1556	9.4	3.7	60.0
1394-4-2	0	t.o.	t.o.	t.o.	16	462.1	34695	**219.7**	13.6	1233.6
	1	20	812.6	2356	16	429.0	44177	**314.6**	14.9	1499.9
abp4	L	37	< 1	234	16	**16.3**	844	78.8	8.4	340.2
brp	$\neg$ L	6	4.8	46	1	< 1	192	< 1	< 1	< 1
	$\neg$ L, nv	68	15.0	122	24	**104.9**	1560	1005.0	260.8	3171.0
dme2	$\neg$ L	1	< 1	123	1	< 1	128	< 1	< 1	< 1
	$\neg$ L, nv	40	2.3	408	39	**1.2**	52	97.4	7.9	502.9
dme5	$\neg$ L	1	11.3	112	1	1.1	186	$<$ **1**	< 1	< 1
	$\neg$ L, nv	344	1533.1	330	99	**384.8**	1396	t.o.	t.o.	t.o.
dme6	$\neg$ L	1	29.1	183	1	1.6	362	$<$ **1**	< 1	< 1
	$\neg$ L, nv	t.o.	t.o.	t.o.	119	**926.4**	2093	t.o.	t.o.	t.o.
pci	**F** L	22	231.4	341	18	t.o.	t.o.	**771.2**	965.4	1879.6
prod-cons	0	69	3.1	311	26	**16.5**	722	442.4	41.7	551.8
	1	33	2.0	250	21	**1.8**	162	25.0	11.1	126.2
	2	58	71.0	216	24	**3.1**	221	7.6	10.9	178.8
	3	42	7.9	241	24	**2.6**	224	28.0	8.93	361.6
production-cell	0	85	< 1	300	81	**9.8**	220	59.1	107.8	t.o.
	1	146	1.4	241	81	t.o.	t.o.	**23.4**	30.0	t.o.
bc57-sensors	0	112	141.3	213	103	**194.1**	4382	1143.1	201.9	t.o.
srg5	$\neg$ L	16	< 1	120	1	< 1	74	< 1	< 1	< 1
	$\neg$ L, nv	15	< 1	31	6	1.5	217	$<$ **1**	< 1	< 1

Table 3. Real-world examples

have established general criteria for Büchi automata to accept shortest lasso-shaped counterexamples, extending the notion of a tight automaton from [17]. We have presented a construction of a Büchi automaton that is tight for full LTL.

Our construction generates Büchi automata with a high number of states. In ongoing work we apply virtual unrolling to obtain tight Büchi automata from the subclass of automata that, like automata constructed with CGH+, accepts counterexamples with an overly long stem but shortest loop. This should result in tight automata with fewer states and may help to facilitate application also in explicit state model checking employing, e.g., the algorithm of [11]. Further options include using transition-labeled instead of state-labeled automata [1] as well as more deterministic automata [24].

Acknowledgements. We thank Timo Latvala for providing us with his modified variant of NuSMV 2.1.2 including a very timely bug fix and Roderick Bloem for pointing us to the problem of a shortest informative vs. any shortest counterexample.

References

1. M. Awedh and F. Somenzi. Proving more properties with bounded model checking. In *CAV'04*, volume 3114 of *LNCS*, pages 96–108. Springer, 2004.
2. M. Benedetti and A. Cimatti. Bounded model checking for past LTL. In *TACAS'03*, volume 2619 of *LNCS*, pages 18–33. Springer, 2003.
3. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *TACAS'99*, volume 1579 of *LNCS*, pages 193–207. Springer, 1999.
4. A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An opensource tool for symbolic model checking. In *CAV'02*, volume 2404 of *LNCS*, pages 359–364. Springer, 2002.

5. A. Cimatti, M. Roveri, and D. Sheridan. Bounded verification of past LTL. In *FMCAD'04*, volume 3312 of *LNCS*. Springer, 2004.
6. E. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. *FMSD*, 10(1):47–71, 1997.
7. E. Clarke, O. Grumberg, K. McMillan, and X. Zhao. Efficient generation of counterexamples and witnesses in symbolic model checking. In *DAC'95*, pages 427–432. ACM, 1995.
8. C. Courcoubetis, M. Vardi, P. Wolper, and M. Yannakakis. Memory-efficient algorithms for the verification of temporal properties. *FMSD*, 1(2/3):275–288, 1992.
9. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Volume B, Formal Methods and Semantics*, pages 995–1072. North-Holland Pub. Co., 1990.
10. D. Gabbay. The declarative past and imperative future. In *Temporal Logic in Specification*, volume 398 of *LNCS*, pages 409–448. Springer, 1989.
11. P. Gastin, P. Moro, and M. Zeitoun. Minimization of counterexamples in SPIN. In *SPIN'04*, volume 2989 of *LNCS*, pages 92–108. Springer, 2004.
12. R. Gerth, D. Peled, M. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *PSTV'95*, volume 38 of *IFIP Conference Proceedings*, pages 3–18. Chapman & Hall, 1996.
13. A. Groce and D. Kröning. Making the most of BMC counterexamples. In A. Biere and O. Strichman, editors, *BMC'04*, pages 71–84, 2004.
14. The VIS Group. VIS: A system for verification and synthesis. In *CAV'96*, volume 1102 of *LNCS*, pages 428–432. Springer, 1996.
15. J. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California at Los Angeles, 1968.
16. Y. Kesten, A. Pnueli, and L. Raviv. Algorithmic verification of linear temporal logic specifications. In *ICALP'98*, volume 1443 of *LNCS*, pages 1–16. Springer, 1998.
17. O. Kupferman and M. Vardi. Model checking of safety properties. In *CAV'99*, volume 1633 of *LNCS*, pages 172–183. Springer, 1999.
18. F. Laroussinie, N. Markey, and P. Schnoebelen. Temporal logic with forgettable past. In *LICS'02*, pages 383–392. IEEE Computer Society, 2002.
19. T. Latvala, A. Biere, K. Heljanko, and T. Junttila. Simple is better: Efficient bounded model checking for past LTL. In *VMCAI'05*, volume 3385 of *LNCS*. Springer, 2005.
20. K. Ravi, R. Bloem, and F. Somenzi. A comparative study of symbolic algorithms for the computation of fair cycles. In *FMCAD'00*, volume 1954 of *LNCS*, pages 143–160. Springer, 2000.
21. K. Schneider. Improving automata generation for linear temporal logic by considering the automaton hierarchy. In *LPAR'01*, volume 2250 of *LNCS*, pages 39–54. Springer, 2001.
22. V. Schuppan and A. Biere. Efficient reduction of finite state model checking to reachability analysis. *International Journal on Software Tools for Technology Transfer (STTT)*, 5(2–3):185–204, 2004.
23. V. Schuppan and A. Biere. Shortest counterexamples for symbolic model checking of LTL with past. Technical Reports 470, ETH Zürich, Computer Systems Institute, 01 2005.
24. R. Sebastiani and S. Tonetta. "More deterministic" vs. "smaller" Büchi automata for efficient LTL model checking. In *CHARME'03*, volume 2860 of *LNCS*, pages 126–140. Springer, 2003.
25. O. Shtrichman. Pruning techniques for the SAT-based bounded model checking problem. In *CHARME'01*, volume 2144 of *LNCS*, pages 58–70. Springer, 2001.
26. F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *CAV'00*, volume 1855 of *LNCS*, pages 248–263. Springer, 2000.
27. M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *LICS'86*, pages 332–344. IEEE Computer Society, 1986.