On relative and probabilistic finite counterability

Orna Kupferman · Gal Vardi

Published online: 12 April 2017
© Springer Science+Business Media New York 2017

Abstract A counterexample to the satisfaction of a linear property \( \psi \) in a system \( S \) is an infinite computation of \( S \) that violates \( \psi \). When \( \psi \) is a safety property, a counterexample to its satisfaction need not be infinite. Rather, it is a bad-prefix for \( \psi \): a finite word all whose extensions violate \( \psi \). The existence of finite counterexamples is very helpful in practice. Liveness properties do not have bad-prefixes and thus do not have finite counterexamples. We extend the notion of finite counterexamples to non-safety properties. We study counterable languages—ones that have at least one bad-prefix. Thus, a language is counterable iff it is not liveness. Three natural problems arise: (1) given a language, decide whether it is counterable, (2) study the length of minimal bad-prefixes for counterable languages, and (3) develop algorithms for detecting bad-prefixes for counterable languages. We solve the problems for languages given by means of LTL formulas or nondeterministic Büchi automata. In particular, our EXPSPACE-completeness proof for the problem of deciding whether a given LTL formula is counterable, and hence also for deciding liveness, settles a long-standing open problem. In addition, we make finite counterexamples more relevant and helpful by introducing two variants of the traditional definition of bad-prefixes. The first adds a probabilistic component to the definition. There, a prefix is bad if almost all its extensions violate the property. The second makes it relative to the system. There, a prefix is bad if all its extensions in the system violate the property. We also study the combination of the probabilistic and relative variants. Our framework suggests new variants also for safety and liveness languages. We solve the above three problems for the different variants. Interestingly, the probabilistic variant not

A preliminary version has appeared in the 24th EACSL Annual Conference on Computer Science Logic.

The research leading to these results has received funding from the European Research Council under the European Union’s Seventh Framework Programme (FP7/2007–2013)/ERC Grant Agreement No. 278410.

Gal Vardi
gal.vardi@mail.huji.ac.il

Orna Kupferman
orna@cs.huji.ac.il

1 School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel
only increases the chances to return finite counterexamples, but also makes the solution of the three problems exponentially easier.

**Keywords** Model checking · Counterexamples · Safety · Liveness · Probability

1 Introduction

In model checking, we verify that a system meets a desired property by checking that a mathematical model of the system meets a formal specification that describes the property. Safety and liveness [2] are two classes of system properties. Essentially, a safety property states that something “bad” never happens and a liveness property states that something “good” eventually happens. Formally, consider a language $L$ of infinite words over an alphabet $\Sigma$. A finite word $x \in \Sigma^*$ is a bad-prefix for $L$ if for all infinite words $y \in \Sigma^\omega$, the concatenation $x \cdot y$ is not in $L$. Thus, a bad-prefix for $L$ is a finite word that cannot be extended to a word in $L$. A language $L$ is safety if every infinite word not in $L$ has a prefix that is a bad-prefix for $L$. A language $L$ is liveness if there are no bad-prefixes for $L$, that is, every word in $\Sigma^*$ can be extended to a word in $L$.

The classes of safety and liveness properties have been extensively studied. From a theoretical point of view, their importance stems from their topological characteristics. Consider the natural topology on $\Sigma^\omega$, where similarity between words corresponds to the length of the prefix they share. Formally, the distance between $w$ and $w'$ is $2^{-i}$, where $i \geq 0$ is the position of the first letter in which $w$ and $w'$ differ. In this topology, safety and liveness properties are exactly the closed and dense sets, respectively [2]. This, for example, implies that every linear property can be expressed as a conjunction of a safety and a liveness property [2,3].

From a practical point of view, reasoning about safety and liveness properties require different methodologies, and the distinction between them pinpoints different challenges of formal methods. In liveness, one has to demonstrate progress towards fulfilling eventualities. Thus, liveness is the concept behind fairness [16], and behind the need for rich $\omega$-regular acceptance conditions [41], progress measures [29,32], and many more. On the other hand, in safety one can reason about finite computations of the system [37]. The latter has significant advantages in symbolic methods [25], bounded model checking [7], run-time verification [18], synthesis [27], and many more.

An important advantage of model-checking tools is their ability to accompany a negative answer to the correctness query by a counterexample to the satisfaction of the specification in the system. Thus, together with a negative answer, the model checker returns some erroneous execution of the system. These counterexamples are not only essential in detecting subtle errors in complex systems [9], but also in improving the modeling of systems. For example, in CEGAR, counterexamples are used in order to guide the refinement of abstract models [8]. In the general case, the erroneous execution of the system is infinite. It is known, however, that for linear temporal logic (LTL) properties, there is always a lasso-shaped counterexample—one of the form $uv^\omega$, for finite computations $u$ and $v$. Clearly, the simpler the counterexample is, the more helpful it is for the user, and indeed there have been efforts for designing algorithms that return short counterexamples [19,24,35]. The analysis of counterexamples makes safety properties even more appealing: rather than a lasso-shaped counterexample, it is possible to return to the user a bad-prefix. This enables the user to find errors as soon as they appear. In addition, the analysis of bad-prefixes is often more helpful, as they point the user not just to one erroneous execution, but rather to a finite execution all whose continuations are erroneous.
We extend the notion of finite counterexamples to non-safety specifications. We also make finite counterexamples more relevant and helpful by introducing two variants of the traditional definition of bad-prefixes. The first adds a probabilistic component to the definition. The second makes it relative to the system. We also consider the combination of the probabilistic and relative variants. Before we describe our contribution in detail, let us demonstrate the idea with the following example. Consider a system $S$ and a specification $\psi$ stating that every request is eventually followed by a response. There might be some input sequence that leads $S$ to an error state in which it stops sending responses. While $\psi$ is not safety, the system $S$ has a computation with a prefix that is bad with respect to $S$: all its extensions in $S$ do not satisfy $\psi$. Returning this prefix to the user, with its identification as bad with respect to $S$, is more helpful than returning a lasso-shaped counterexample. Consider now a specification $\varphi$ stating that the system eventually stops allocating memory. There might be some input sequence that leads $S$ to a state in which every request is followed by a memory allocation. A computation that reaches this state almost surely violates the specification. Indeed, it is possible that requests eventually stop arriving and the specification would be satisfied, but the probability of this behavior of the input is 0. Thus, the system $S$ has a computation with a prefix that is bad with respect to $S$ in a probabilistic sense: almost all of its extensions in $S$ do not satisfy $\varphi$. Again, we want to return this prefix to the user, identified as bad with high probability.

Recall that a language $L$ is liveness if every finite word can be extended to an infinite word in $L$. Equivalently, $L$ has no bad-prefixes. We say that $L$ is counterable if it has a bad-prefix. That is, $L$ is counterable iff it is not liveness. Note that a language, for example $a^n \cdot b \cdot (a + b + c)^\omega$, may be counterable and not safety. When a system does not satisfy a counterable specification $\psi$, it may contain a bad-prefix for $\psi$, which we would like to return to the user. Three natural problems arise: (1) Given a language, decide whether it is counterable, (2) study the length of minimal bad-prefixes for counterable languages, and (3) develop algorithms for detecting bad-prefixes for counterable languages.

In fact, the last two problems are open also for safety languages. Deciding whether a given language is safety is known to be PSPACE-complete for languages given by LTL formulas or nondeterministic Büchi word automata (NBWs, for short) [37]. For the problem of deciding whether a language is counterable, an EXPSPACE upper-bound for languages given by LTL formulas is not difficult [37], yet the tight complexity is open. This is surprising, as recall that a language is counterable iff it is not liveness, and one could expect the complexity of deciding liveness to be settled by now. As it turns out, the problem was studied in [31], where it is stated to be PSPACE-complete. The proof in [31], however, is not convincing, and indeed efforts to solve the problem have continued, and the problem was declared open in [4] (see also [28]). Our first contribution is an EXPSPACE lower bound, implying that the long-standing open problem of deciding liveness (and hence, also counterability) of a given LTL formula is EXPSPACE-complete. In a recent communication with Diekert, Muscholl, and Walukiewicz, we have learned that they recently came up with an independent EXPSPACE lower bound, in the context of monitoring of infinite computations [14]. For languages given by means of an NBW, the problem is PSPACE-complete [31,37]. Thus, interestingly, while in deciding safety the exponential succinctness of LTL with respect to NBWs does not make the problem more complex, in deciding liveness it makes the problem exponentially more complex. This phenomenon is reflected also in the solutions to the problems about the length and the detection of bad-prefixes, as detailed in the table in Fig. 1, which summarizes our results. We also show that when a language given by an LTL formula is safety, the solutions for the three problems become exponentially easier.
Let us return to our primary interest, of finding finite counterexamples. Consider a system modeled by a Kripke structure $K$ over a set $AP$ of atomic propositions. Let $\Sigma = 2^{AP}$, and consider an $\omega$-regular language $L \subseteq \Sigma^\omega$. We say that a finite computation $x \in \Sigma^*$ of $K$ is a $K$-bad-prefix for $L$, if $x$ cannot be extended to an infinite computation of $K$ that is in $L$. Formally, for all $y \in \Sigma^\omega$, if $x \cdot y$ is a computation of $K$, then it is not in $L$. Once we define $K$-bad-prefixes, the definitions of safety and counterability are naturally extended to the relative setting: A language $L$ is $K$-counterable if it has a $K$-bad-prefix and is $K$-safety if every computation of $K$ that is not in $L$ has a $K$-bad-prefix for $L$. Using a product of $K$ with an NBW for $L$, we are able to show that the solutions we suggest for the three problems in the non-relative setting apply also to the relative one, with an additional NLOGSPACE or linear-time dependency in the size of $K$. We also study $K$-safety, and the case $L$ is $K$-safety.

We note that relative safety and liveness properties have already been considered in the literature, with different motivation and results. In [20] the notion of relative safety and liveness is defined. The goal there is to lift the practical advantages of safety to liveness properties. The idea is that a liveness property may be safety relative to some other property, that states a reasonable assumption about the behavior of the system. For example, when the behavior of a system is described with a sequence of pairs $\langle \sigma_i, \tau_i \rangle$ where $\sigma_i$ is a state of the system and $\tau_i$ is a time, the bounded-response property that states that every request is followed by a response within some time $\delta$ is a liveness property. However, the bounded-response property is safety relative to some timing assumptions. The notion of lifting the practical advantages of safety to liveness properties is also studied in [5,36], where the finiteness of the system is taken into account. In [31], the notion of liveness relative to a system is re-interpreted as satisfaction within fairness. The idea there is that if a property is liveness relative to a system, then every prefix of a behavior of the system can be extended to an infinite behavior that satisfies the property, and therefore in crude terms, the system almost satisfies the property—it just needs the “help of some fairness”. They also show that satisfaction within fairness is preserved by some abstraction mappings, and therefore can be used also in a setting where an abstract model of the system is used.

We continue to the probabilistic view. A random word over $\Sigma$ is a word in which all letters are drawn from $\Sigma$ uniformly at random. In particular, when $\Sigma = 2^{AP}$, then the
probability of each atomic proposition to hold in each position is $\frac{1}{2}$. Consider a language $L \subseteq \Sigma^\omega$. We say that a finite word $x \in \Sigma^*$ is a \textit{prob-bad-prefix} for $L$ if the probability of an infinite word with prefix $x$ to be in $L$ is 0. Formally, $Pr(\{y \in \Sigma^\omega : x \cdot y \in L\}) = 0$. Then, $L$ is \textit{prob-counterable} if it has a prob-bad-prefix. Now, given a Kripke structure $K$, we combine the relative and probabilistic views in the expected way; a finite computation $x \in (2^{AP})^*$ of $K$ is a $K$-prob-bad-prefix for $L$ if a computation of $K$ obtained by continuing $x$ with some random walk on $K$, is almost surely not in $L$. Thus, a computation of $K$ that starts with $x$ and continues according to some random walk on $K$ is in $L$ with probability 0. We show that this definition is independent of the probabilities of the transitions in the random walk on $K$. Again, $L$ is \textit{$K$-prob-counterable} if it has a $K$-prob-bad-prefix. We note that a different approach to probabilistic counterexamples is taken in [1]. There, the focus is on reachability properties, namely properties of the form “the probability of reaching a set $T$ of states starting from state $s$ is at most $p$”. Accordingly, a counterexample is a set of paths from $s$ to $T$, such that the probability of the event of taking some path in the set is greater than $p$. We, on the other hand, cover all $\omega$-regular languages, and a counterexample is a single finite path—one whose extensions result in a counterexample with high probability.

We study the theoretical properties of the probabilistic setting and show that an $\omega$-regular language $L$ is prob-counterable iff the probability of a random word to be in $L$ is less than 1. We also show that $\omega$-regular languages have a “safety-like” behavior in the sense that the probability of a word not to be in $L$ and not to have a prob-bad-prefix is 0. Similar properties hold in the relative setting and suggest that attempts to return to the user prob-bad-prefixes and $K$-prob-bad-prefixes are likely to succeed.

From a practical point of view, we show that the probabilistic setting not only increases our chances to return finite counterexamples, but also makes the solution of our three basic problems easier: As specified in the table in Fig. 1, deciding prob-counterability and $K$-prob-counterability for LTL formulas is exponentially easier than deciding counterability and $K$-counterability! Moreover, the length of bad-prefixes is exponentially smaller, and finding them is exponentially easier. Our results involve a careful analysis of the product of $K$ with an automaton for $L$. Now, the product is defined as a Markov chain, and we also need the automaton to be deterministic. Our construction also suggest a simpler proof to the known probabilistic NBW model-checking result of [11]. While the blow-up determinization involves is legitimate in the case $L$ is given by an NBW, it leads to a doubly-exponential blow-up in the case $L$ is given by an LTL formula $\psi$. We show that in this case, we can avoid the construction of a product Markov chain and, adopting an idea from [11], generate instead a sequence of Markov chains, each obtained from its predecessor by refining the states according to the probability of the innermost temporal subformula of $\psi$.

It is easy to see that there is a trade-off between the length of a counterexample and its “precision”, in the sense that the longer a finite prefix of an erroneous computation is, the larger is the probability in which it is a $K$-prob-bad-prefix. We allow the user to play with this trade-off and study both the problem in which the user provides, in addition to $K$ and $\psi$, also a probability $0 < \gamma < 1$, and gets back a shortest finite computation $x$ of $K$ such that the probability of a computation of $K$ that starts with $x$ to satisfy $\psi$ is less than $\gamma$, and the problem in which the user provides a length $m \geq 1$ and gets back a finite computation $x$ of $K$ of length at most $m$ such that the probability of a computation of $K$ that starts with $x$ to satisfy $\psi$ is minimal.
2 Preliminaries

2.1 Automata and LTL

A nondeterministic automaton on infinite words is a tuple $A = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, where $\Sigma$ is a finite alphabet, $Q$ is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, and $\alpha$ is an acceptance condition whose type depends on the class of $A$. A run of $A$ on a word $w = \sigma_0 \cdot \sigma_1 \cdots \in \Sigma^\omega$ is a sequence of states $r = q_0, q_1, \ldots$ such that $q_0 \in Q_0$ and $q_{i+1} \in \delta(q_i, \sigma_i)$ for all $i \geq 0$. The run is accepting if it satisfies the condition $\alpha$. We consider here Büchi and parity automata. In a Büchi automaton, $\alpha \subseteq Q$ and the run $r$ satisfies $\alpha$ if it visits some state in $\alpha$ infinitely often. Formally, let $inf(r) = \{ q : q = q_i \text{ for infinitely many } i \text{’s} \}$ be the set of states that $r$ visits infinitely often. Then, $r$ satisfies $\alpha$ iff $inf(r) \cap \alpha \neq \emptyset$ [6]. In a parity automaton, $\alpha : Q \to \{ 0, \ldots, k \}$ maps each state to a color in $\{ 0, \ldots, k \}$. We refer to $k$ as the index of $A$. A run $r$ satisfies $\alpha$ if the minimal color $c$ such that $inf(r) \cap \alpha^{-1}(c) \neq \emptyset$ is even. A word $w \in \Sigma^\omega$ is accepted by $A$ if there is an accepting run of $A$ on $w$. The language of $A$, denoted $L(A)$, is the set of words that $A$ accepts. When $|Q_0| = 1$ and $|\delta(q, \sigma)| = 1$ for all $q \in Q$ and $\sigma \in \Sigma$, then $A$ is deterministic. When a state $q \in Q$ is such that no word is accepted from $q$ (equivalently, the language of $A$ with initial state $q$ is empty), we say that $q$ is empty. We use the acronyms NBW, DBW, NPW, and DPW to denote nondeterministic and deterministic Büchi and parity word automata, respectively. We also refer to the standard nondeterministic and deterministic automaton on finite words, abbreviated NFW and DFW, respectively. We define the size of $A$, denoted $|A|$, as the size of $\delta$, namely, $\sum_{q \in Q, \sigma \in \Sigma} |\delta(q, \sigma)|$. For every NBW there is an equivalent DPW, and the translation from an NBW to a DPW involves an exponential blow-up [33].

An automaton $A$ induces a graph $G_A = \langle Q, E \rangle$ where $(q, q') \in E$ iff there is $\sigma \in \Sigma$ such that $q' \in \delta(q, \sigma)$. When we refer to the strongly connected sets (SCSs) of $A$, we refer to the SCSs of this graph. Formally, a set $C \subseteq Q$ of states is an SCS of $A$ if for all $q, q' \in C$, there is a path from $q$ to $q'$ in $G_A$. An SCS $C$ is maximal if for all sets $C'$ such that $C' \not\subseteq C$, the set $C \cup C'$ is no longer an SCS. A maximal SCS is termed a strongly connected component (SCC). An SCC $C$ is accepting if a run that visits exactly all the states in $C$ infinitely often satisfies $\alpha$. For example, when $\alpha$ is a parity condition, then $C$ is accepting if the minimal color $c$ such that $C \cap \alpha^{-1}(c) \neq \emptyset$ is even. An SCC $C$ is ergodic iff for all $(q, q') \in E$, if $q \in C$ then $q' \in C$. That is, an SCC is ergodic if no edge leaves it.

The logic LTL is a linear temporal logic [34]. Formulas of LTL are constructed from a set $AP$ of atomic propositions using the usual Boolean operators and the temporal operators $G$ (“always”), $F$ (“eventually”), $X$ (“next time”), and $U$ (“until”). The semantics of LTL is defined with respect to infinite computations over $AP$. We use $w \models \psi$ to indicate that the computation $w \in (2^{AP})^\omega$ satisfies the LTL formula $\psi$. The language of an LTL formula $\psi$, denoted $L(\psi)$, is the set of infinite computations that satisfy $\psi$. For the full syntax and semantics of LTL see [34]. We define the size of an LTL formula $\psi$, denoted $|\psi|$, to be the number of its Boolean and temporal operators. Given an LTL formula $\psi$, one can construct an NBW $A_\psi$ that accepts exactly all the computations that satisfy $\psi$. The size of $A_\psi$ is, in the worst case, exponential in $|\psi|$ [41].

We model systems by Kripke structures. A Kripke structure is a tuple of the form $K = \langle A_P, W, W_0, R, l \rangle$, where $W$ is the set of states, $R \subseteq W \times W$ is a total transition relation (that is, for every $w \in W$, there is at least one state $w'$ such that $R(w, w')$), $W_0 \subseteq W$ is a set of initial states, and $l : W \to 2^{AP}$ maps each state to the set of atomic propositions that hold in
it. A path in $K$ is a (finite or infinite) sequence $w_0, w_1, \ldots$ of states in $W$ such that $w_0 \in W_0$ and for all $i \geq 0$ we have $R(w_i, w_{i+1})$. A computation of $K$ is a (finite or infinite) sequence $l(w_0), l(w_1), \ldots$ of assignments in $2^{AP}$ for a path $w_0, w_1, \ldots$ in $K$. We assume that different states of $K$ are labeled differently. That is, for all states $w, w' \in W$ such that $w \neq w'$, we have $l(w) \neq l(w')$. The assumption makes our setting cleaner, as it amounts to working with deterministic systems, so all the nondeterminism and probabilistic choices are linked to the specification and the distribution of the inputs, which is our focus. The simplest way to adjust nondeterministic systems to our setting is to add atomic propositions that resolve nondeterminism. The language of $K$, denoted $L(K)$, is the set of its infinite computations. We say that $K$ satisfies an LTL formula $\psi$, denoted $K \models \psi$, if all the computations of $K$ satisfy $\psi$, thus $L(K) \subseteq L(\psi)$. We define the size of a Kripke structure $K$, denoted $|K|$, as $|W| + |R|$.

For a set $AP$ of atomic propositions, we define the following Kripke structure: $K_{AP} = \langle AP, 2^{AP}, 2^{AP} \times 2^{AP}, I \rangle$, where $l(w) = w$ for all $w \in 2^{AP}$. Thus, $K_{AP}$ is a $2^{AP}$-clique satisfying $L(K_{AP}) = (2^{AP})^\omega$.

### 2.2 Safety, liveness, and counterable languages

Consider an alphabet $\Sigma$, a language $L \subseteq \Sigma^\omega$, and a finite word $u \in \Sigma^*$. We say that $u$ is a prefix for $L$ if it can be extended to an infinite word in $L$, thus there is $v \in \Sigma^\omega$ such that $uv \in L$. Then, $u$ is a bad-prefix for $L$ if it cannot be extended to an infinite word in $L$, thus for every $v \in \Sigma^\omega$, we have that $uv \notin L$. Note that if $u$ is a bad-prefix, so are all its finite extensions. We denote by $\text{pref}(L)$ the set of all prefixes for $L$. Hence, a finite word $u$ is a bad-prefix for $L$ iff $u \notin \text{pref}(L)$.

The following classes of languages have been extensively studied (c.f., [2, 3]). A language $L \subseteq \Sigma^\omega$ is a safety language if every infinite word not in $L$ has a bad-prefix. For example, $\{a^\omega\}$ over $\Sigma = \{a, b, c\}$ is safety, as every word not in $L$ has a bad-prefix—one that contains the letter $b$ or $c$. A language $L$ is a liveness language if every finite word can be extended to a word in $L$. Thus, $\text{pref}(L) = \Sigma^*$. For example, the language $(a + b + c)^* \cdot a^\omega$ is a liveness language: by concatenating $a^\omega$ to every word in $\Sigma^*$, we end up with a word in the language. When $L$ is not liveness, namely $\text{pref}(L) \neq \Sigma^*$, we say that $L$ is counterable. Note that while a liveness language has no bad-prefix, a counterable language has at least one bad-prefix. For example, $L = a^* \cdot b \cdot (a + b + c)^\omega$ is a counterable language. Indeed, $c$ is a bad-prefix for $L$. It is not hard to see that if $L$ is safety and $L \neq \Sigma^\omega$, then $L$ is counterable. The other direction does not hold. For example, $L$ above is not safety, as the word $a^\omega$ has no bad-prefix.

We extend the definitions and classes above to specifications given by LTL formulas or by NBWs. For example, an LTL formula $\psi$ is counterable iff $L(\psi)$ is counterable.

### 3 Probabilistic and relative counterability

In this section we introduce and make some observations on two variants to counterability. The first variant adds a probabilistic component to the definitions. The second makes them relative to a Kripke structure. We also consider the combination of the probabilistic and relative variants.

#### 3.1 Probabilistic counterability

For a finite or countable set $X$, a probability distribution on $X$ is a function $Pr : X \to [0, 1]$ assigning a probability to each element in $X$. Accordingly, $\sum_{x \in X} Pr(x) = 1$. A finite Markov
chain is a tuple $\mathcal{M} = \langle V, p_{in}, p \rangle$, where $V$ is a finite set of states, $p_{in} : V \rightarrow [0, 1]$ is a probabilistic distribution on $V$ that describes the probability of a path to start in the state, and $p : V \times V \rightarrow [0, 1]$ is a function describing a distribution over the transitions. Formally, for $v \in V$, let $p_v : V \rightarrow [0, 1]$ be such that $p_v(u) = p(v, u)$ for all $u \in V$. For all $v \in V$, the function $p_v$ is a probabilistic distribution on $V$. The Markov chain $\mathcal{M}$ induces the directed graph $G = (V, E)$ in which for all $u, v \in V$, we have that $(u, v) \in E$ iff $p(u, v) > 0$. Thus, $G$ includes transitions that have a positive probability in $\mathcal{M}$. When we talk about the SCCs of $\mathcal{M}$, we refer to these of $G$.

A random walk on $\mathcal{M}$ is an infinite path $v_0, v_1, \ldots$ in $G$ such that $v_0$ is drawn at random according to $p_{in}$ and the $i$th state $v_i$ is drawn at random according to $p_{v_{i-1}}$. More formally, there is a probability space $(V^\omega, \mathcal{F}, P_{\mathcal{M}})$ defined on the set $V^\omega$ of infinite sequences of states. The family of measurable sets $\mathcal{F}$ is the $\sigma$-algebra (also called Borel field) generated by $C = \{C(x) : x \in V^\omega\}$ where $C(x)$ is the set of infinite sequences with prefix $x$, namely, $C(x) = \{y : y \in V^\omega \}$. The measure $P_{\mathcal{M}}$ is defined on $C$ (and can be extended uniquely to the rest of $\mathcal{F}$) as follows: $P_{\mathcal{M}}(C(x_0 \ldots x_n)) = p_{in}(x_0) \cdot p(x_0, x_1) \cdots p(x_{n-1}, x_n)$. For more background on the construction of this probability space, see, for example, [22]. A random walk on $\mathcal{M}$ from a state $v \in V$ is a random walk on the Markov chain $\mathcal{M}^v = \langle V, p_{in}^v, p \rangle$, where we have $p_{in}^v(v) = 1$.

The following lemma states two fundamental properties of Markov chains (see, for example, [22]).

**Lemma 1** Consider a Markov chain $\mathcal{M} = \langle V, p_{in}, p \rangle$ and a state $v \in V$.

1. An infinite random walk on $\mathcal{M}^v$ reaches some ergodic SCC with probability 1.
2. Once a random walk on $\mathcal{M}^v$ reaches an ergodic SCC, it visits all its states infinitely often with probability 1.

A labeled finite Markov chain is a tuple $\mathcal{S} = \langle \Sigma, V, p_{in}, p, \tau \rangle$, where $\Sigma$ is a finite alphabet, $\mathcal{M} = \langle V, p_{in}, p \rangle$ is a finite Markov chain, and $\tau : V \rightarrow \Sigma$ maps each state in $V$ to a letter in $\Sigma$. We extend $\tau$ to paths in the expected way, thus for $\pi = v_0, v_1, \ldots \in V^\omega$, we define $\tau(\pi) = \tau(v_0), \tau(v_1), \ldots$. A random walk on $\mathcal{S}$ is a random walk on $\mathcal{M}$. The chain $\mathcal{S}$ induces a probability space on $\Sigma^\omega$, induced from $\mathcal{M}$. That is, for $L \subseteq \Sigma^\omega$, we have $Pr_{\mathcal{S}}(L) = Pr_{\mathcal{M}}[\{\pi \in V^\omega : \tau(\pi) \in L\}]$. It is known that $\omega$-regular languages are measurable in the probability space induced by $\mathcal{S}$ (c.f., [40]).

Consider an alphabet $\Sigma$. A random word over $\Sigma$ is a word in which for which all indices $i$, the $i$th letter is drawn from $\Sigma$ uniformly at random. We denote by $Pr[L]$ the probability of a measurable language $L \subseteq \Sigma^\omega$ in this uniform distribution. For a finite word $w \in \Sigma^*$, we denote by $Pr^w[L]$ the probability that a word obtained by concatenating an infinite random word to $w$ is in $L$. Formally, $Pr^w[L] = Pr[\{v \in \Sigma^\omega : uv \in L\}]$. For example, let $\Sigma = \{a, b, c\}$ and $L = a^* \cdot b \cdot (a + b + c)^\omega$. Then, $Pr[L] = \sum_{i=1}^{\infty} \frac{1}{3^i} = \frac{1}{2}$, $Pr^a[L] = Pr[L] = \frac{1}{2}$, $Pr^b[L] = 1$, and $Pr^c[L] = 0$.

Consider a language $L \subseteq \Sigma^\omega$. We say that a finite word $w \in \Sigma^*$ is a prob-bad-prefix for $L$ if $Pr^w[L] = 0$. That is, $w$ is a prob-bad-prefix if an infinite word obtained by continuing $w$ randomly is almost surely not in $L$. We say that $L$ is prob-counterable if it has a prob-bad-prefix. Consider for example the language $L = a \cdot (a + b)^\omega + b^\omega$ over $\Sigma = \{a, b\}$. All the words $u \in b^+$ are not bad-prefixes for $L$, but are prob-bad-prefixes, as $Pr^w[L] = Pr[b^\omega] = 0$. As another example, consider the LTL formula $\psi = (\neg req \land GFgrant) \lor (\neg \neg req \land FG\neg grant)$. The formula $\psi$ is a liveness formula and does not have a bad-prefix. Thus, $\psi$ is not counterable. All finite computations in which a request is not sent in the beginning.
of the computation are, however, prob-bad-prefixes for $\psi$, as the probability of satisfying $FG^{\neg grant}$ is 0. Hence, $\psi$ is prob-counterable.

Prob-counterability talks about prefixes after which the probability of being in $L$ is 0. We can relate such prefixes to words that lead to rejecting ergodic SCCs in DPWs that recognize $L$, giving rise to the following alternative definitions:

**Theorem 1** Consider an $\omega$-regular language $L$. The following are equivalent:

1. The language $L$ is prob-counterable.
2. $Pr[L] < 1$. That is, the probability of an infinite random word to be in $L$ is strictly smaller than 1.
3. Every DPW that recognizes $L$ has a rejecting ergodic SCC.

**Proof** For 1 $\Rightarrow$ 2, assume that $L$ is prob-counterable, we show that $Pr[L] < 1$. Let $v$ be a prob-bad-prefix. Let $L'$ be the language $\{v' : vv' \notin L\}$. Note that $Pr[L'] = 1$. Let $u$ be a random word and let $u^i$ be the suffix of $u$ obtained by removing the first $i$ letters. Then, we have $Pr[comp(L)] \geq Pr[\{u : u$ starts with $v\} \cap \{u : |u^i| \in L'\}] = Pr[u$ starts with $v] \cdot Pr[|u^i| \in L'] > 0$.

For 2 $\Rightarrow$ 3, we assume, by way of contradiction, that there is a DPW that recognizes $L$ and its ergodic SCCs are all accepting. We denote the DPW by $D$ and its initial state by $s_0$. Note that a random word induces a random walk on a Markov chain on $D$ from $s_0$. According to Lemma 1, this random walk visits all the states in some ergodic SCC of $D$ infinitely often with probability 1. Therefore, if all of the ergodic SCCs are accepting, then the random walk is an accepting run in $D$ with probability 1. Thus, $Pr[L] = 1$, and we have reached a contradiction.

Finally, for 3 $\Rightarrow$ 1, if a DPW that recognizes $L$ has a rejecting ergodic SCC, we can choose a word $u \in \Sigma^*$ whose traversal in the DPW reaches a rejecting ergodic SCC. According to Lemma 1, if we start a random walk from there, it visits every state in the rejecting ergodic SCC infinitely often with probability 1. Therefore $Pr^u[L] = 0$, that is, the word $u$ is a prob-bad-prefix for $L$. Thus $L$ is prob-counterable and we are done. 

Analyzing the SCCs of DPWs for $L$ also implies that $\omega$-regular languages have a “safety-like” behavior in the following probabilistic sense:

**Theorem 2** For every $\omega$-regular language $L$, we have $Pr[\{v \in \Sigma^\omega : v \notin L$ and $v$ does not have a prob-bad-prefix$\}] = 0$.

**Proof** If a word $v \in \Sigma^\omega$ is not in $L$ but does not have a prob-bad-prefix, then the run of $v$ in the DPW that recognizes $L$ either (1) does not reach an ergodic SCC or (2) reaches an ergodic SCC but does not visit all its states infinitely often. This is because a word whose run visits all the states of an ergodic SCC infinitely often is either in $L$ (if the ergodic SCC is accepting) or has a prob-bad-prefix (which is a prefix that reaches the rejecting ergodic SCC). A random word induces a random walk on a Markov chain on the DPW that recognizes $L$, and therefore according to Lemma 1 we have $Pr[\{v \in \Sigma^\omega : v \notin L$ and $v$ does not have a prob-bad-prefix$\}] = 0$. 

Note that if $L$ is prob-counterable, thus $Pr[v \notin L] > 0$, then $Pr[v$ has a prob-bad-prefix $| v \notin L] = 1$.

### 3.2 Relative counterability

Recall that the standard definitions of bad-prefixes consider extensions in $\Sigma^\omega$. When $\Sigma = 2^{AP}$ and the language $L$ is examined with respect to a Kripke structure $K$ over $AP$, it is interesting
to restrict attention to extensions that are feasible in $K$. Consider a finite word $u \in \Sigma^*$. We say that $u$ is a bad-prefix for $L$ with respect to $K$ ($K$-bad-prefix, for short) if $u$ is a finite computation of $K$ that cannot be extended to a computation of $K$ that is in $L$. Thus, $u \in \text{pref}(L(K)) \setminus \text{pref}(L(K) \cap L)$. We say that $L$ is safety with respect to $K$ ($K$-safety, for short) if every computation of $K$ that is not in $L$ has a $K$-bad-prefix. We say that $L$ is counterable with respect to $K$ ($K$-counterable, for short) if the language $L$ has a $K$-bad-prefix. Thus, $\text{pref}(L(K)) \not\subseteq \text{pref}(L(K) \cap L)$.

**Theorem 3** Consider an $\omega$-regular language $L \subseteq (2^{AP})^\omega$.

1. $L$ is safety iff $L$ is $K$-safety for every Kripke structure $K$ over $AP$.
2. For every Kripke structure $K$ over $AP$, we have that $L$ is $K$-safety iff $L(K) \cap L$ is safety.

**Proof** We start with the first claim. Clearly, if $L$ is safety, then $L$ is $K$-safety for every $K$. Indeed, every computation of $K$ that is not in $L$ has a bad-prefix for $L$, which is a $K$-bad-prefix. For the other direction, since $L$ is $K$-safety for every Kripke structure $K$, it is $K_{AP}$-safety. Thus, every $w \notin L$ has a bad-prefix for $L$, which implies that $L$ is safety.

We proceed to the second claim. Consider a Kripke structure $K$. Assume first that $L$ is $K$-safety. Let $w$ be an infinite computation over $2^{AP}$ that is not in $L(K) \cap L$. That is, either $w$ is not in $L(K)$, or $w$ is in $L(K)$ but not in $L$. If $w$ is not in $L(K)$, then $w$ has a prefix that is not a finite computation of $K$, and therefore it is a bad-prefix for $L(K)$. This prefix is a bad-prefix for $L(K) \cap L$. If $w$ is in $L(K)$ but not in $L$, then since $L$ is $K$-safety, the computation $w$ has a $K$-bad-prefix for $L(K)$ and this prefix is a bad-prefix for $L(K) \cap L$.

Assume now that $L(K) \cap L$ is safety. Let $w$ be a computation of $K$ that is not in $L$, and therefore $w \notin L(K) \cap L$. Thus, the computation $w$ has a bad-prefix $w'$ for $L(K) \cap L$. Every computation of $K$ that extends $w'$ is in $L(K)$ but not in $L(K) \cap L$, and therefore it is not in $L$. Thus, $w'$ is a $K$-bad-prefix for $L$.

Recall that if $L \subseteq \Sigma^\omega$ is safety and $L \neq \Sigma^\omega$, then $L$ is counterable. Also, if $L$ is $K$-safety and $L(K) \notin L$ then $L$ is $K$-counterable. Note that it is possible that $L(K) \cap L$ is counterable but $L$ is not $K$-counterable. For example, we can choose $K$ and $L$ such that $L(K) \subseteq L \neq (2^{AP})^\omega$. Then, $L$ is not $K$-counterable, but a word $u$ that is not a computation of $K$ is a bad-prefix for $L(K)$, making it also a bad-prefix for $L(K) \cap L$. Hence, $L(K) \cap L$ is counterable.

### 3.3 Probabilistic relative counterability

We now combine the probabilistic and relative counterability. Consider a Kripke structure $K = \langle AP, W, W_0, R, l \rangle$. A $K$-walk-distribution is a tuple $P = \langle p_{in}, p \rangle$ such that the labeled Markov chain $M_{K, p} = \langle 2^{AP}, W, p_{in}, p, l \rangle$ induces a graph that agrees with $K$. Thus, $p_{in}(w) > 0$ iff $w \in W_0$, and $p(w, w') > 0$ iff $R(w, w')$. A random walk on $K$ with respect to $P$ is a random walk on the Markov chain $\langle W, p_{in}, p \rangle$. We define the probability of an $\omega$-regular language $L \subseteq (2^{AP})^\omega$ with respect to $K$ and $P$ as $Pr_{K, P}[L] = Pr_{MK, P}[L]$. Namely, $Pr_{K, P}[L]$ is the probability that a computation obtained by a random walk on $K$ with respect to $P$ is in $L$. Let $u$ be a finite computation of $K$ and let $w_0, \ldots, w_k \in W^*$ be such that $u = l(w_0), \ldots, l(w_k)$. We say that an infinite computation $u'$ is a continuation of $u$ with a random walk on $K$ with respect to $P$ if $u' = l(w_0), \ldots, l(w_{k-1}), l(w'_0), l(w'_1), \ldots$, where $w'_0, w'_1, \ldots$ is obtained by a random walk on $K$ from $w_k$ with respect to $P$ (recall that we assume that different states in $K$ are labeled differently). We define $Pr_{K, P}^u[L]$ as the probability that a computation obtained by continuing $u$ with a random walk on $K$ with
Fig. 2 A Kripke structure $K$ such that $\psi = G(req \rightarrow \text{Ack}) \land FG(\text{open})$ is $K$-prob-counterable and not $K$-counterable.

respect to $P$ is in $L$. Formally, we define $Pr_{K,P}^u$ using a conditional probability: $Pr_{K,P}^u[L] = Pr_{K,P}[L]$ for the word starts with $u$. We extend the definition to every labeled Markov chain $M$; that is $Pr_{M}^u[L] = Pr_{M}[L]$ for the word starts with $u$. Thus, $Pr_{K,P}^u[L] = Pr_{M}^u[L]$. Note that $Pr_{M}^u[L]$ is well-defined only when $Pr_{M}[\text{the word starts with } u] > 0$.

Consider a Kripke structure $K$ over $AP$ and an $\omega$-regular language $L \subseteq (2^AP)^\omega$. We say that $u \in (2^AP)^*$ is a prob-bad-prefix for $L$ with respect to $K$ ($K$-prob-bad-prefix, for short) if $u$ is a finite computation of $K$ such that $Pr_{K,P}^u[L] = 0$, for some $K$-walk-distribution $P$. Thus, a computation obtained by continuing $u$ with some random walk on $K$ is almost surely not in $L$. As we show in Lemma 2 below, the existential quantification on the $K$-walk-distribution $P$ can be replaced by a universal one, or by the specific $K$-walk-distribution that traverses $K$ uniformly at random. We say that $L$ is prob-counterable with respect to $K$ ($K$-prob-counterable, for short) if it has a $K$-prob-bad-prefix.

Note that if $L$ is counterable, then $L$ is prob-counterable. Also, if $L$ is $K$-counterable then $L$ is $K$-prob-counterable. As we have seen above, a language may be prob-counterable but not counterable. Taking $K = K_{AP}$, this implies that a language may be $K$-prob-counterable but not $K$-counterable. As an example with an explicit dependency in $K$, consider the counterable LTL formula $\psi = G(req \rightarrow \text{Ack}) \land FG(\text{open})$. Let $K$ be a Kripke structure over $AP = \{req, \text{ack, open}\}$ as shown in Fig. 2, such that the atomic propositions in $AP$ are mutually exclusive and $L(K)$ contains exactly the computations that start in $req$ and in which every $req$ is immediately followed by $\text{ack}$, or computations in which $\text{open}$ is always valid. Note that while $\psi$ is not $K$-counterable, it is $K$-prob-counterable, as every finite computation of $K$ that starts with $req$ is a $K$-prob-bad-prefix for $\psi$.

Consider an NBW $A$. By [11,40], deciding whether $Pr_{K,P}[L(A)] = 0$ or whether $Pr_{K,P}[L(A)] = 1$ is independent of the $K$-walk-distribution $P$. Consequently, we have the following.

**Lemma 2** [11,40] Let $u$ be a finite computation of a Kripke structure $K$ over $AP$, and let $L \subseteq (2^AP)^\omega$ be an $\omega$-regular language. For all pairs $P$ and $P'$ of $K$-walk-distributions, we have $Pr_{K,P}^u[L] = 0$ iff $Pr_{K,P'}^u[L] = 0$ and $Pr_{K,P}^u[L] = 1$ iff $Pr_{K,P'}^u[L] = 1$.

Recall that the analysis of probabilistic counterability involved reasoning about the SCCs of a DPW that recognizes the language. For probabilistic relative counterability, we need to consider the product of this DPW with the system, and define it as a labeled Markov chain.

Let $D = \langle 2^AP, S, s_0, \delta_D, \alpha \rangle$ be a DPW for a language $L$ and let $K = \langle AP, W, W_0, R, l \rangle$ be a Kripke structure. We define $D_{K \times D} = \langle 2^AP, W \times S, W_0 \times \{s_0\}, \rho, \alpha' \rangle$ as the product NPW of $K$ and $D$. Formally, we have $\alpha'(\langle w, s \rangle) = \alpha(s)$, and $\langle w', s' \rangle \in \rho(\langle w, s \rangle, \sigma)$ iff $l(w) = \sigma, R(w, w')$ and $s' = \delta_D(s, \sigma)$. Note that $L(D_{K \times D}) = L(K) \cap L(D)$. Also, note that all of the successors of a state in $D_{K \times D}$ share the same second component.

We now define the product as a labeled Markov chain. Let $D$ and $K$ be as above and let $P = \langle p_{in}, p \rangle$ be a $K$-walk-distribution. We define a labeled Markov chain $M' = \langle 2^AP, W \times S, p_{in}^i, p^l, l \rangle$, where $p^l : (W \times S) \times (W \times S) \rightarrow [0, 1]$ is such that $p^l(\langle w, s \rangle, \langle w', s' \rangle) = p((w, w'))$ if $\delta_D(s, l(w)) = s'$, and otherwise $p^l(\langle w, s \rangle, \langle w', s' \rangle) = 0$. Also, $p_{in}^i : W \times
\[ S \rightarrow [0, 1] \text{ is such that } p'_l((w, s)) = p_{in}(w) \text{ if } s = s_0, \text{ and otherwise } p'_l((w, s)) = 0. \]

Finally, \( l' : W \times S \rightarrow 2^{\mathbb{Z}_D} \) is such that \( l'(w, s) = l(w) \). Thus, \( p'_{in} \) and \( p' \) attribute the states of \( M = M_K, p \) by the deterministic behavior of \( D \). In particular, note that for every \( \langle w, s \rangle \in W \times S \), we have \( \sum_{\langle w', s' \rangle \in W \times S} p'_{\langle (w, s), (w', s') \rangle} = \sum_{w' \in W} p(w, w') = 1. \)

Let \( g : W \times S \rightarrow W \) be the function that projects pairs in \( W \times S \) on their \( W \)-component, namely \( g((w, s)) = w \). Consider random walks \( X = X_1, X_2, \ldots \) and \( X' = X'_1, X'_2, \ldots \) on \( M \) and \( M' \), respectively. Let \( Y = Y_1, Y_2, \ldots \) be the projection of \( X' \) on \( W \), thus \( Y_i = g(X'_i) \) for \( i = 1, 2, \ldots \). Note that \( X \) and \( Y \) both take values in \( W^\omega \) and that they have the same distribution. Therefore, we have \( Pr_M[L] = Pr_{M'}[L] \). Also, for a finite computation \( u = l(w_0), l(w_1), \ldots, l(w_n) \) of \( K \), we have \( Pr_{M'}^u[L] = Pr_{M'}^u[L] \). Note that \( w_0, \ldots, w_n \) induces a single finite path \( \langle w_0, s_0 \rangle, \ldots, \langle w_n, s_n \rangle \) in \( D_{K \times D} \), and that every infinite path in \( D \) induces a single infinite path in \( D_{K \times D} \). For a finite computation \( u \), let \( reach(u) \) be the state reached in \( D_{K \times D} \) after traversing \( u \). Thus, \( reach(u) = \langle w_n, s_n \rangle \). By the above, \( Pr_M^u[L] \) is the probability that a random walk in \( M' \) from \( reach(u) \) is an accepting run in \( D_{K \times D} \). For a state \( x \) of \( M' \), we denote by \( \gamma_x \) the probability that a random walk from \( x \) in \( M' \) is an accepting run in \( D_{K \times D} \). Note that a finite computation \( u \) is a \( K \)-prob-bad-prefix iff \( \gamma_{reach(u)} = 0. \)

We can now point to equivalent definitions of \( K \)-prob-counterability.

**Theorem 4** Consider an \( \omega \)-regular language \( L \subseteq (2^A)^\omega \) and a Kripke structure \( K \) over \( A \). The following are equivalent:

1. The language \( L \) is \( K \)-prob-counterable.
2. There is a finite computation \( u \) of \( K \) and a \( K \)-walk-distribution \( P \) s.t. \( Pr_K^u[P[L] < 1]. \)
3. There is a finite computation \( u \) of \( K \) such that for all \( K \)-walk-distribution \( P \), we have \( Pr_K^u[P[L] < 1]. \)
4. There is a \( K \)-walk-distribution \( P \) s.t. \( Pr_K[P[L] < 1]. \)
5. For all \( K \)-walk-distributions \( P \), we have \( Pr_K[P[L] < 1]. \)

**Proof** The proof for 2 \( \Leftrightarrow \) 3 and 4 \( \Leftrightarrow \) 5 follows from Lemma 2.

For 1 \( \Rightarrow \) 4, assume that \( L \) is \( K \)-prob-counterable. Let \( u \) be a \( K \)-prob-bad-prefix. Then, for a \( K \)-walk-distribution \( P \) we have \( Pr_K[P[comp(L)] \geq Pr_K[P[the computation starts with \( u \) and is not in \( L \)] = Pr_K[P[the computation starts with \( u \)] > 0]. \)

For 4 \( \Rightarrow \) 2, assume that \( Pr_K[P[L] < 1 \). We choose \( u = \varepsilon \). Clearly, \( Pr_K^u[P[L] < 1]. \)

For 2 \( \Rightarrow \) 1, assume that \( Pr_K^u[P[L] < 1 \). According to the definitions of \( D_{K \times D} \) and \( \gamma_{reach(u)} \), we have \( \gamma_{reach(u)} = Pr_K^u[P[L] < 1 \). Therefore, by Lemma 1, \( D_{K \times D} \) has a rejecting ergodic SCC. Let \( \nu \) be a finite computation of \( K \) that reaches this ergodic SCC in \( D_{K \times D} \), then \( \nu \) is a \( K \)-prob-bad-prefix. \( \Box \)

We can also generalize Theorem 2, and show that \( \omega \)-regular languages have “safety-like” behaviors also with respect to Kripke structures, in the following probabilistic sense:

**Theorem 5** Consider an \( \omega \)-regular language \( L \subseteq (2^A)^\omega \), a Kripke structure \( K \) over \( A \), and a \( K \)-walk-distribution \( P \). Then, \( Pr_K[P[[u \in (2^A)^\omega : u \notin L \text{ and } u \text{ does not have a } K \text{-prob-bad-prefix for } L]] = 0. \)

**Proof** According to the construction of \( M' \), a computation obtained by a random walk on \( M = M_K, p \) has the same distribution as a computation obtained by a random walk on \( M' \). Therefore \( Pr_M\l[ the computation is not in \( L \) and does not have a \( K \)-prob-bad-prefix] = Pr_M\l[ the computation is not in \( L \) and does not have a \( K \)-prob-bad-prefix]. This equals to the probability that a random walk on \( M' \) is a rejecting run of \( D_{K \times D} \) and its computation does not have a \( K \)-prob-bad-prefix (and therefore it does not reach a rejecting ergodic SCC of \( D_{K \times D} \)). According to Lemma 1 this probability is 0. \( \Box \)
If $L$ is also $K$-prob-counterable then we also have $Pr_{K,P}[u \text{ has a } K\text{-prob-bad-prefix for } L | u \notin L] = 1$ for every $K$-walk-distribution $P$.

Conceptually, Theorem 4 implies that if an error has a positive probability to occur in a random execution of the system, then the specification is prob-counterable with respect to the system. Theorem 5 then suggests that in this case, a computation of the system that does not satisfy the specification, almost surely has a prob-bad-prefix with respect to the system. Thus, almost all the computations that violate the specification start with a prob-bad-prefix with respect to the system. Hence, attempts to find and return to the user such bad-prefixes are very likely to succeed.

4 Deciding liveness

Recall that a language $L$ is counterable iff $L$ is not liveness. As discussed in Sect. 1, the complexity of the problem of deciding whether a given LTL formula is liveness is open [4]. In this section we solve this problem and prove that it is EXPSPACE-complete. The result will be handy also for our study of the probabilistic and relative variants.

**Theorem 6** The problem of deciding whether a given LTL formula is liveness is EXPSPACE-complete.

*Proof* The upper-bound is known [37], and follows from the fact that every LTL formula $\psi$ can be translated to an NBW $A_\psi$ with an exponential blow-up [41]. By removing empty states from $A_\psi$ and making all other states accepting, we get an NFW for $\text{pref}(L(\psi))$, which is universal iff $\psi$ is liveness. Checking NFW universality can be done in PSPACE by complementing the NFW on-the-fly and checking emptiness, implying the EXPSPACE upper bound.

For the lower bound, we show a reduction from an exponent version of the tiling problem, defined as follows. We are given a finite set $T$, two relations $V \subseteq T \times T$ and $H \subseteq T \times T$ of tiles, an initial tile $t_0$, a final tile $t_f$, and a bound $n > 0$. We have to decide whether there is some $m > 0$ and a tiling of a $2^n \times m$-grid such that (1) The tile $t_0$ is in the bottom left corner and the tile $t_f$ is in the top left corner, (2) A horizontal condition: every pair of horizontal neighbors is in $H$, and (3) A vertical condition: every pair of vertical neighbors is in $V$. Formally, we have to decide whether there is a function $f : \{0, \ldots, 2^n - 1\} \times \{0, \ldots, m - 1\} \to T$ such that (1) $f(0, 0) = t_0$ and $f(0, m - 1) = t_f$, (2) For every $0 \leq i \leq 2^n - 2$ and $0 \leq j \leq m - 1$, we have that $(f(i, j), f(i + 1, j)) \in H$, and (3) For every $0 \leq i \leq 2^n - 1$ and $0 \leq j \leq m - 2$, we have that $(f(i, j), f(i, j + 1)) \in V$. When $n$ is given in unary, the problem is known to be EXPSPACE-complete [39].

We reduce this problem to the problem of deciding whether an LTL formula is not liveness.

Given a tiling problem $\tau = \langle T, H, V, t_0, t_f, n \rangle$, we construct a formula $\varphi$ such that $\tau$ admits tiling iff $\varphi$ has a good-prefix—one all whose extensions satisfy $\varphi$. Formally, $x \in \Sigma^*$ is a good prefix for $\varphi$ iff for all $y \in \Sigma^\omega$, we have that $x \cdot y$ satisfies $\varphi$. Therefore, for $\psi = \neg \varphi$, we have $\tau$ admits tiling iff $\psi$ is not liveness. The idea is to encode a tiling as a word over $T$, consisting of a sequence of rows (each row is of length $2^n$). Such a word represents a proper tiling if it starts with $t_0$, has a last row that starts with $t_f$, ever pair of adjacent tiles in a row are in $H$, and every pair of tiles that are $2^n$ tiles apart are in $V$. The difficulty is in relating tiles that are far apart. To do that we represent every tile by a block of length $n$, which encodes the tile’s position in the row. Even with such an encoding, we have to specify a property of the form “for every $i$, if we meet a block with position counter $i$, then the next time we meet a block...
with position counter \(i\), the tiles in the two blocks are in \(V^n\). Such a property can be expressed in an LTL formula of polynomial length, but there are exponentially many \(i\)'s to check. The way to use liveness in order to mimic the universal quantification on all \(i\)'s is essentially the following: the good prefix for \(\varphi\) encodes the tiling. The set of atomic propositions in \(\varphi\) includes a proposition \(\$\) that is not restricted beyond this prefix. The property that checks \(V\) then has to hold in blocks whose counter equals the counter of the block that starts at the last \(\$\) in the computation. Thus, universal quantification in \(i\) is replaced by the explicit universal quantification on suffixes in the definition of good prefixes.

We proceed to describe the reduction in detail. A block of length \(n\) that represents a single tile contains the tile itself in the first letter (each tile has an atomic proposition \(t_i \in T\)). The block also contains the position of the tile in the row. This position is a number between 0 and \(2^n - 1\) and we use an atomic proposition \(c_1\) to encode it as an \(n\)-bit vector. For simplicity we denote \(\neg c_1\) by \(c_0\) (that is, \(c_0\) is not an atomic proposition). The vector is stored in the letters of the block with the least significant bit stored at the first letter. The position is increased by one from one block to the next. We require that the first letter of each block is marked with the atomic proposition \(\#\), that the first block is the last block that must be marked with this pattern: the first block of the first \(\$\) is the last block that must be marked with this pattern:

\[
\varphi_1 = \# \land \left( \bigwedge_{1 \leq i < n} X^i (\neg \#) \right) \land \left( \bigwedge_{0 \leq i < n} X^i c_0 \right).
\]

– Until the first \(\$\), the atomic proposition \# marks the beginning of a block of length \(n\) (the block of the first \(\$\) is the last block that must be marked with this pattern):

\[
\varphi_2 = ( (\# \rightarrow X^n \#) \land ( (\neg \#) \rightarrow X^n (\neg \#))) U \$.
\]

– The following four formulas make sure that the position (that is encoded by \(c_1\)) is increased by one every \#. We use an additional atomic proposition \(z\) that represents the carry. Thus, we add 1 to the least significant bit and then propagate the carry to the other bits. Note that the requirements hold until \(\$\) (the block of the first \(\$\) still has a position counter):

\[
\varphi_3 = (((\# \lor z) \land c_0) \rightarrow (X (\neg z) \land X^n c_1)) U \$.
\]

\[
\varphi_4 = (((\neg (\# \lor z) \land c_0) \rightarrow (X (\neg z) \land X^n c_0)) U \$.
\]

\[
\varphi_5 = (((\# \lor z) \land c_1) \rightarrow (X z \land X^n c_0)) U \$.
\]

\[
\varphi_6 = (((\neg (\# \lor z) \land c_1) \rightarrow (X (\neg z) \land X^n c_1)) U \$.
\]

– Each letter has at most one tile. This requirement holds until \(\$\):

\[
\varphi_7 = \left( \bigwedge_{t_i \in T} \left( t_i \rightarrow \bigwedge_{t_j \in T, t_j \neq t_i} \neg t_j \right) \right) U \$.
\]

– The tiling starts with \(t_0\), thus \(f(0, 0) = t_0\): \(\varphi_8 = t_0\)

– The requirement that \(f(0, m) = t_f\) is translated to the requirement that there is a row that starts with \(t_f\) and the letter right after this row is marked with the first \(\$\). We use \(\@\) to mark a \(t_f\) that appears at the beginning of a row (that is, a block with a position counter of 0). By using the formula \(\theta_{row \rightarrow start} = \# \land (\bigwedge_{0 \leq i < n} X^i c_0)\) we require that the first \(\@\) appears on a tile \(t_f\) at the beginning of a row: \(\varphi_9 = (\neg \@) U (@ \land t_f \land \theta_{row \rightarrow start})\).
– The following requires that the first $ appears right after the row that starts with the first
@: $10 = (¬S ∧ ¬@) U (@ ∧ ¬S ∧ X ((¬θ_{row-start} ∧ ¬S) U (θ_{row-start} ∧ S))).
– For the horizontal condition, namely that for every $0 ≤ i ≤ 2^n - 2$ and $0 ≤ j ≤ m - 1$ we
have $(f(i, j), f(i + 1, j)) ∈ H$, we first define $θ_H = \bigwedge_{t_i ∈ T} ((# ∧ t_i) → X ((¬#) U (# ∧ \bigvee_{t_j: (i, t_j) ∈ H} t_j))).$ Since this should hold in every tile except for the last tile in every row
and only until $, we have $θ_{row-end} = # ∧ (\bigwedge_{0 ≤ i < n} X^i c_1).$ Now,

$φ_{11} = ((¬θ_{row-end} → θ_H) U S).
– The vertical condition, namely that for every $0 ≤ i ≤ 2^n - 1$ and $0 ≤ j ≤ m - 2$ we have
$(f(i, j), f(i, j + 1)) ∈ V,$ is the challenging one. We want to relate tiles with the same
position counter value in consequent rows. We first define a formula $θ_{eq}$ that holds iff the
atomic proposition $ appears finitely many times and the current position counter agrees
with the position counter in the block of length $n$ that begins at the last appearance of $:

$θ_{eq} = # ∧ \bigwedge_{0 ≤ i < n} ((X^i c_0 \land F (S \land \bigwedge_{0 ≤ i} X^i c_0 \land X G S)) \lor (X^i c_1 \land F (S \land X^i c_1 \land X G S))).$

Then, the formula $θ_{next-t_i}$ requires that a block that satisfies $θ_{eq}$ in the next row must
have a tile $t_i$:

$θ_{next-t_i} = X ((¬θ_{row-start}) U (θ_{row-start} ∧ (θ_{eq} → t_i) U Xθ_{row-start})).$

Using $θ_{next-t_i}$ we can define the formula $θ_V$ that requires that if $θ_{eq}$ is satisfied, then the
tile with the same position counter in the next row satisfies the vertical condition:

$θ_V = θ_{eq} → \bigwedge_{t_i ∈ T} (t_i → \bigvee_{t_j: (i, t_j) ∈ V} θ_{next-t_j}).$

By requiring $θ_V$ to hold until $@,$ we specify the vertical condition. Thus, $φ_{12} = θ_V U @.$
Note that this formula checks the vertical condition only for positions that agree with the
position counter of the block of the last $ (when $ appears finitely often).
– Finally, the LTL formula $φ$ requires all the above requirements to be satisfied. Thus,

$φ = \bigwedge_{1 ≤ i ≤ 12} φ_i.$

We now prove that indeed $τ$ admits tiling iff $φ$ has a good-prefix. Assume first that $τ$
admits tiling. We claim that a finite word of length $(2^n \cdot m + 1) \cdot n$ that represents the tiling
as explained above is a good-prefix for $φ$. This prefix includes the block of length $n$ right
after the $m$th row (which has $ in its first letter and a position counter of 0. . . 0. The atomic
propositions $t_i$ for this block can be defined arbitrarily). Note that $φ$ does not restrict $S$ beyond
requiring that its first appearance is in the letter that is right after the $m$th row. The formulas
$φ_1, . . . , φ_{11}$ depend only on this prefix and therefore are satisfied for every extension of this
prefix with an infinite suffix. If the atomic proposition $S$ appears infinitely often in the
suffix then $φ_{12}$ is also satisfied. Otherwise, there is a position that is encoded by the atomic
proposition $c_1$ in the block of length $n$ that starts at the last $. In this case, the formula $φ_{12}$
checks the vertical condition for tiles with this position. Since the prefix represents a valid
tiling, the vertical condition holds and therefore the formula $φ_{12}$ holds. Note that although $φ$
has a good-prefix, it is not necessarily co-safety.

Now, assume that $φ$ has a good-prefix. By the formulas $φ_1, . . . , φ_{11},$ this prefix must start
with the tile $t_0,$ have $m > 0$ such that the $m$th row starts with $t_f$ and satisfy the horizontal
condition in the first $m$ rows. Note that $m$ is the index of the first row that starts with $@,$ and
that since the prefix is a good-prefix for $φ$ then it must contain at least $2^n \cdot m + 1$ blocks.
Indeed, $\varphi$ requires that such $@$ appears, and that the first block in the $m + 1$th row starts with $\$$ and has a position counter of $0 \ldots 0$. We now show that the vertical condition also holds for every position in the tiling that this prefix represents. Since $\varphi_{12}$ holds for every extension of the prefix with an infinite suffix, then in particular it holds for suffixes in which the atomic proposition $\$$ appears finitely often. In this case, the block of length $n$ that starts at the last $\$$ encodes a position, and by $\varphi_{12}$ the vertical condition must hold for this position. Since for every possible position there is a suffix such that this position is encoded by the block of the last $\$$, then the vertical condition must hold for every position.

Therefore, we have: $\tau$ admits tiling iff $\varphi$ has a good-prefix iff $\psi$ is not liveness. Since the size of $\psi$ is polynomial in $n$, then our reduction is done.

When a language is given by means of an NBW, the complexity of deciding its liveness is much simpler:

**Theorem 7** The problem of deciding whether a given NBW is liveness is PSPACE-complete.

**Proof** The upper bound is described in [37]. For the lower bound we describe a reduction from a PSPACE Turing machine. As shown in [30], we can generate, given a PSPACE Turing machine $T$ and an input $x$ to it, an NBW $A$ that rejects a word $w$ iff $w$ starts with an encoding of a legal and accepting computation of $T$ over the input $x$. Therefore, $T$ accepts the input $x$ iff $L(A)$ is not liveness, and we are done.

5 Counterability

In this section we study the problem of deciding whether a given language is counterable as well as the length of short bad-prefixes and their detection. In order to complete the picture, we also compare the results to those of safety languages.

We start with the complexity of deciding safety and counterability. The results for safety are from [37]. These for counterability follow from Theorems 6 and 7 and the fact that $L$ is counterable iff it is not liveness.

**Theorem 8** Consider a language $L$.

1. [37] The problem of deciding whether $L$ is safety is PSPACE-complete for $L$ given by an LTL formula or by an NBW.
2. The problem of deciding whether $L$ is counterable is EXPSPACE-complete for $L$ given by an LTL formula and is PSPACE-complete for $L$ given by an NBW.

We find Theorem 8 surprising: both safety and counterability ask about the existence of bad-prefixes. In safety, a bad-prefix should exist to all bad words. In counterability, not all bad words have a bad-prefix, but at least some should have. Theorem 8 implies that there is something in LTL, yet not in NBWs, that makes the second type of existence condition much more complex.

We now turn to study the length of shortest bad-prefixes. Both (non-valid) safety and counterable languages have bad-prefixes. As we show, however, the complexity of counterability continues, and a tight bound on shortest bad-prefixes for counterable languages is exponentially bigger than that of safety languages. As detailed in the proof, the gap follows from our ability to construct a fine automaton $A_{\psi}^{fine}$ for all safety LTL formulas $\psi$ [23]. The NFW $A_{\psi}^{fine}$ is exponential in $|\psi|$, it accepts only bad-prefixes for $\psi$, and each computation
that does not satisfy $\psi$ has at least one bad-prefix accepted by $A^\text{fine}_{\psi}$. A shortest witness to the nonemptiness of $A^\text{fine}_{\psi}$ can serve as a bad-prefix. On the other hand, nothing is guaranteed about the behavior of $A^\text{fine}_{\psi}$ when constructed for a non-safety formula $\psi$, thus it is of no help in the case of counterable languages that are not safety. In particular, the LTL formula used in the proof of Theorem 6 is neither safety nor its complement is safety, thus its doubly-exponential shortest bad-prefix does not contradict upper bounds known for these classes of languages.

**Theorem 9** The length of shortest bad-prefixes for a language given by an LTL formula $\psi$ is tightly exponential in $|\psi|$ in case $\psi$ is safety, and is tightly doubly-exponential in $|\psi|$ in case $\psi$ is counterable.

**Proof** We start with the case of safety formulas. In [23], the authors associated with each safety LTL formula $\psi$ an NFW $A^\text{fine}_{\psi}$ that accepts a trap for $\psi$. That is, it is guaranteed that $A^\text{fine}_{\psi}$ accepts only bad-prefixes for $\psi$ and that each computation that does not satisfy $\psi$ has at least one bad-prefix accepted by $A^\text{fine}_{\psi}$. Unlike an NFW that accepts exactly all the bad-prefixes for $\psi$ and which is doubly-exponential in $|\psi|$, it is possible to define $A^\text{fine}_{\psi}$ so that it is only exponential in $|\psi|$. Now, a shortest witness to the nonemptiness of $A^\text{fine}_{\psi}$ is linear in its size, and it bounds the length of a shortest bad-prefix for $\psi$. Hence the exponential upper bound. For the lower bound, consider the family $L_1, L_2, \ldots$ of safety languages where $L_n$ contains exactly all words over the alphabet $\{0, 1, \#\}$ that do not start with a description of an $n$-bit counter that counts from 0 to $2^n - 1$. For example, $L_2$ contains all words that do not start with 00011011. It is not hard to see that $L_n$ is a safety language, that it can be specified by an LTL formula of length polynomial in $n$, and that the shortest bad-prefix for $L_n$ is the description of an $n$-bit counter that counts from 0 to $2^n - 1$, which is of length exponential in $n$. Formally, the LTL formula is over the atomic propositions $AP = \{b, c, d\}$ where $b$ represents the value of a bit, $c$ represents a carry and $d$ represents a start of an $n$-bits block.

We proceed to counterable languages. In [25], the authors associated with each LTL formula $\psi$ an NFW $A^\text{tight}_{\psi}$ that accepts exactly all the bad-prefixes for $\psi$. The size of $A^\text{tight}_{\psi}$ is doubly-exponential in $|\psi|$. A shortest witness to the nonemptiness of $A^\text{tight}_{\psi}$ is linear in its size, and it bounds from above the length of a shortest bad-prefix for $\psi$. Hence the doubly-exponential upper bound. For the lower bound, we consider the formula $\psi$ generated in the EXPSPACE hardness proof in Theorem 6. It is known that there are inputs to the tiling problem described there for which the shortest legal tiling has doubly-exponentially many rows. Indeed, such tilings can encode all problems in EXPSPACE, including those whose running time in doubly-exponential. A shortest bad-prefix for the LTL formula $\psi$ constructed in the proof describes the tiling with its doubly-exponentially many rows, and is thus doubly-exponential in $|\psi|$.

When the specification formalism is automata, the difference between safety and counterable languages disappears.

**Theorem 10** The length of shortest bad-prefixes for a safety or a counterable language given by an NBW $A$ is tightly exponential in $|A|$.

**Proof** In [25], the authors associated with each NBW $A$ an NFW $A^\text{tight}$ that accepts exactly all the bad-prefixes for $A$. The size of $A^\text{tight}$ is exponential in $|A|$. A shortest witness to the
nonemptiness of \( A^{\text{right}} \) is linear in its size, and it bounds from above the length of a shortest bad-prefix for \( A \). Hence the exponential upper bound.

For the lower bound, let \( \Sigma = \{a, b\} \), and let \( p_1, \ldots, p_n \) be the first \( n \) prime numbers. For \( k = 1, \ldots, n \) let \( A_k \) be an NBW of size \( O(p_k) \) such that \( L(A_k) = \{w : w = a^m b \text{ or } w \text{ starts with } a^m b, \text{ for } m = 0 \text{ or } m \not\equiv 0 \text{ mod } p_k\} \). Let \( A \) be an NBW of size \( O(p_1 + \ldots + p_n) \) such that \( L(A) = \bigcup_{k=1}^{n} L(A_k) \). Note that \( L(A) \) is safety. Since \( p_n = O(n \log n) \), the size of \( A \) is polynomial in \( n \). On the other hand, a bad-prefix for \( A \) is of length at least \( 1 + \prod_{k=1}^{n} p_k \), which is greater than \( n! \), and hence exponential in \( n \).

\[ \square \]

### 6 Relative counterability

In this section we add a Kripke structure \( K \) to the setting and study \( K \)-counterability and shortest \( K \)-bad-prefixes. Our results use variants of the product of automata for \( K \) and \( L \), and we first define this product below. Consider a Kripke structure \( K = (AP, W, W_0, R, l) \) and an NBW \( A = (2^{AP}, Q, Q_0, \delta, \alpha) \). Essentially, the states of the product \( A_{K \times A} \) are pairs in \( W \times Q \). Recall that the states in \( K \) are differently labeled. Thus, we can define the product so that whenever it reads a letter in \( 2^{AP} \), its next \( W \)-component is determined. Formally, we define the NBW \( A_{K \times A} = (2^{AP}, (W \cup \{s_0\}) \times Q, \{s_0\} \times Q_0, \rho, W \times \alpha) \), where for all \( \sigma \in 2^{AP} \), we have \( \langle w', q' \rangle \in \rho((s_0, q), \sigma) \) iff \( w' \in W_0 \) and \( l(w') = \sigma \) and \( q' \in \delta(q, \sigma) \), and for \( w \in W \) we have \( \langle w', q' \rangle \in \rho((w, q), \sigma) \) iff \( R(w, w') \) and \( l(w') = \sigma \) and \( q' \in \delta(q, \sigma) \). Thus, when the product \( A_{K \times A} \) proceeds from state \( \langle w, q \rangle \) with \( \sigma \), its new \( W \)-component is the single successor of \( w \) that is labeled \( \sigma \), paired with the \( \sigma \)-successors of \( q \). It is easy to see that \( L(A_{K \times A}) = L(K) \cap L(A) \). When \( A \) is an NBW that corresponds to an LTL formula \( \psi \) (that is, \( L(A) = L(\psi) \)), we denote the product by \( A_{K \times \psi} \).

We start with the problem of deciding relative safety. By Theorem 3, a language \( L \) is \( K \)-safety iff \( L(K) \cap L \) is safety. Thus, the check can be reduced to checking the safety of \( A_{K \times A} \) (respectively \( A_{K \times \psi} \)). This check, however, if done naively, is PSPACE in \( |A_{K \times A}| \) (respectively \( |A_{K \times \psi}| \)), which is PSPACE in \( |K| \). The technical challenge is to find a more efficient way to do the check, and the one we describe in the proof is based on decomposing \( A_{K \times A} \) so that the complementation that its safety check involves is circumvented. As for the lower bound, note that using the Kripke structure \( K_{AP} \), one can reduce traditional safety to relative safety.\(^3\) Our reduction, however, shows that the complexity of deciding \( K \)-safety coincides with that of model checking in both its parameters.

**Theorem 11** Consider a Kripke structure \( K \) over \( AP \) and a language \( L \subseteq (2^{AP})^\omega \). The problem of deciding whether \( L \) is \( K \)-safety is PSPACE-complete for \( L \) given by an NBW or by an LTL formula. In both cases it can be done in time linear and space polylogarithmic in \( |K| \).

**Proof** For the upper bound, we describe an efficient way for checking whether \( A_{K \times A} \) (or \( A_{K \times \psi} \)) is safety. By [37], an NBW \( S \) is safety iff \( L(S^\text{loop}) \subseteq L(S) \), where \( S^\text{loop} \) is the NBW obtained from \( S \) by removing all empty states and making all other states accepting. For the case of an NBW, \( L(A) \) is \( K \)-safety iff \( L(A_{loop}) \subseteq L(A_{K \times A}) = L(K) \cap L(A) \). However, note that every word in \( L(A_{loop}) \) has an infinite path in \( K \) and therefore \( L(A_{loop}) \subseteq L(K) \). Thus, \( L(A) \) is \( K \)-safety iff \( L(A_{loop}) \subseteq L(A) \). The required upper bound is then obtained by

\(^3\) Indeed \( K_{AP} \) is exponential in \( |AP| \), but safety is known to be PSPACE-hard also when the number of atomic propositions is fixed.
checking the emptiness of $L(A_{K \times A}^{\text{loop}}) \cap \text{comp}(L(A))$. The case of an LTL formula is similar: The required upper bound is obtained by checking the emptiness of $L(A_{K \times A}^{\text{loop}}) \cap L(\neg \psi)$.

For the lower bound, we describe a reduction from the model-checking problem. Given a Kripke structure $K$ and an LTL formula $\psi$, let $q \notin AP$ be a fresh atomic proposition, and let $K'$ be a Kripke structure obtained from $K$ by duplicating each state so that $w \in 2^{AP \cup \{q\}}$ is a computation of $K'$ iff the projection of $w$ on $AP$ is a computation of $K$. Note that $K'$ is twice as large as $|K|$. We claim that $K \models \psi$ iff the LTL formula $\psi \lor Fq$ is $K'$-safety. First, if $K \models \psi$, then every computation of $K$ satisfies $\psi$, implying that all the computations of $K'$ satisfy $\psi$, and thus also satisfy $\psi \lor Fq$. Hence, $\psi \lor Fq$ is $K'$-safety. For the other direction, observe that every finite computation of $K'$ can be continued with a suffix that reaches a state in which $q$ holds. Hence, $\psi \lor Fq$ does not have a $K'$-bad-prefix. Therefore $\psi \lor Fq$ is $K'$-safety only when all the computations of $K'$ satisfy $\psi \lor Fq$. By the definition of $K'$, the latter holds only when $K \models \psi$. The reduction for the case of an NBW is similar.  

We continue to relative counterability. We first show that the complexity of deciding counterability is carried over to the relative setting. For the upper bound, note that a language $L$ is $K$-counterable iff $\text{comp}(\text{pref}(L(K)) \cap \text{comp}(\text{pref}(L(K) \cap L))) \neq \emptyset$. Again, this check, if done naively, is PSPACE in $|K|$, and the challenge in the proof is to use the deterministic behavior of $A_{K \times A}$ with respect to the $W$-component of its states in order to avoid a blow-up in $K$ in its complementation.

**Theorem 12.** Consider a Kripke structure $K$ over $AP$ and a language $L \subseteq (2^{AP})^\omega$. The problem of deciding whether $L$ is $K$-counterable and finding a shortest $K$-bad-prefix is EXPSPACE-complete for $L$ given by an NBW and is EXPSPACE-complete for $L$ given by an LTL formula. In both cases it can be done in time linear in the size of $K$.

**Proof.** We start with the upper bounds. By definition, $L$ is $K$-counterable iff $\text{pref}(L(K)) \subseteq \text{pref}(L(K) \cap L)$. Equivalently, $\text{pref}(L(K)) \cap \text{comp}(\text{pref}(L(K) \cap L)) \neq \emptyset$. We construct an NFW $U'$ for $\text{comp}(\text{pref}(L(K) \cap L))$ as follows. Let $A$ be an NBW such that $L = L(A)$. First, we remove from $A_{K \times A}$ empty states and make all its states accepting. This results in an NFW $U'$ for $\text{pref}(L(K) \cap L)$. The NFW $U'$ is obtained by determinizing and dualizing $U'$. As explained above, $A_{K \times A}$ has a deterministic behavior with respect to the $W$-component of its states. Hence, determinizing it with the subset construction results in state space $W \times 2^Q$. It follows that $\mathcal{U}$ is linear in $\|K\|$ and exponential in $|A|$. Also, an NFW $S$ for $\text{pref}(L(K))$ is clearly linear in $\|K\|$. Now, checking whether $L$ is $K$-counterable is reduced to checking the nonemptiness of $S \times \mathcal{U}$. The states of $S \times \mathcal{U}$ are of the form $W \times (W \times 2^Q)$. Since both $S$ and the $W$-component of $\mathcal{U}$ proceed according to the transition relation of $K$, they coincide, until possibly one of them gets stuck. Hence, the size of the product $S \times \mathcal{U}$ is linear in $\|K\|$ and exponential in $|A|$. The nonemptiness check can be done on-the-fly and in NLOGSPACE. Thus, it can be done in PSPACE for $L$ given by an NBW and in EXPSPACE for $L$ given by an LTL formula. Also it can be done in time linear and space polylogarithmic in $\|K\|$. A shortest witness to the nonemptiness is a shortest $K$-bad-prefix.

We proceed to the lower bounds. According to the reduction of Theorem 6, the problem of deciding whether an LTL formula $\psi$ over a fixed set of atomic propositions $AP$ is liveness, is EXPSPACE-hard. Note that $\psi$ is liveness if and only if $K_{AP}$-counterable. Since $|AP|$ is independent of $|\psi|$, it gives a polynomial reduction that shows that the problem of deciding whether $\psi$ is $K$-counterable is EXPSPACE-hard in $|\psi|$. Similarly, according to Theorem 7, deciding whether $L(A)$ is $K$-counterable is PSPACE-hard in $|A|$.  

We now study the length of $K$-bad-prefixes.
Theorem 13 The length of a shortest $K$-bad-prefix for a $K$-counterable language $L$ is tightly doubly-exponential in $|\psi|$ in case $L$ is given by means of an LTL formula $\psi$, and is tightly exponential in $|A|$, in case $L$ is given by an NBW $A$. In both cases, it is also tightly linear in $|K|$.

Proof The upper bounds follow from the proof of Theorem 12.

For the lower bounds with respect to $|\psi|$ and $|A|$, note that when we use the Kripke structure $K_{AP}$, the constructions described in the proofs of Theorems 9 and 10 also apply here.

Finally, for the lower bound with respect to $|K|$, let $AP = \{0, \ldots, n\}$, fix $\psi$ to the safety formula $G\neg 0$, and consider the family of Kripke structures $K_1, K_2, \ldots$, with $K_n = \langle AP, AP, \{n\}, R, l \rangle$, where $l(w) = \{w\}$, and the transitions are such that $R(k, k-1)$ for $k \geq 2$, $R(2, 0)$, $R(1, 1)$ and $R(0, 0)$. It is easy to see that $\psi$ is $K_n$-counterable and that the length of the shortest $K_n$-bad-prefix is $n$. Indeed, every computation of $K_n$ starts with the word $n \cdot (n-1) \ldots 2$ and then it either continues with $1^\omega$ or with $0^\omega$. Therefore, the shortest $K_n$-bad-prefix is $n \cdot (n-1) \ldots 2 \cdot 0$.

Interestingly, when the LTL formula is $K$-safety, deciding its $K$-counterability and finding a $K$-bad-prefix can be done more efficiently, and its $K$-bad-prefixes are shorter. Formally, we have the following.

Theorem 14 Let $K$ be a Kripke structure and let $\psi$ be a $K$-safety LTL formula. Deciding whether $\psi$ is $K$-counterable and finding a $K$-bad-prefix is PSPACE-complete in $|\psi|$. The length of shortest $K$-bad-prefixes is tightly exponential in $|\psi|$.

Proof Since $\psi$ is $K$-safety, we have that $\psi$ is $K$-counterable iff $K \not\models \psi$. From the standard reduction for LTL model checking PSPACE-hardness [38] it follows that LTL model checking is PSPACE-complete already for the safety fragment of LTL.

We now show an algorithm for finding a $K$-bad-prefix and study its length. In [23] it is shown that if $A$ is a safety NBW with $n$ states and $\overline{A}$ is an NBW with $\overline{n}$ states, $\overline{m}$ of them are accepting, such that $L(\overline{A}) = \text{comp}(L(A))$, then there exists an NFW $A'$ with $\overline{n}$, $(\overline{m} \cdot n + 1)$ states such that $A'$ is fine for $L(\overline{A})$. That is, $A'$ recognizes only bad-prefixes for $L(A)$ and every word that is not in $L(A)$ has at least one bad-prefix that is recognized by $A'$. In Theorem 3 we showed that if $K$ is $K$-safety then $A_{K \times \psi}$ is safety. Let $\tilde{A}_{K \times \psi}$ be an NFW of size exponential in $|\psi|$ and linear in $|K|$, such that $L(\tilde{A}_{K \times \psi}) = \text{comp}(L(K)) \cup \text{comp}(L(\psi)) = \text{comp}(L(A_{K \times \psi}))$. Therefore, according to [23], there exists an NFW $A$ of size exponential in $|\psi|$ which is fine for $L(A_{K \times \psi})$. Let $A' = A_{K \times \psi}^{pref} \times A$ be a product NFW, where $A_{K \times \psi}^{pref}$ is an NFW that recognizes $\text{pref}(L(K))$. If $L(A')$ is not empty then every word in $L(A')$ is a $K$-bad-prefix. If there is a $K$-bad-prefix for $\psi$ then there is a computation $\pi$ of $K$ (that starts with this $K$-bad-prefix) that does not satisfy $\psi$. Since $\pi$ is not in $L(A_{K \times \psi})$, then $\pi$ has a bad-prefix for $L(A_{K \times \psi})$ that is recognized by $A$. This prefix is recognized also by $A_{K \times \psi}^{pref}$ and therefore by $A'$, and it is a $K$-bad-prefix. The size of $A'$ is exponential in $|\psi|$, and therefore the task of finding a word in $L(A')$ can be done on-the-fly in space polynomial in $|\psi|$, and its size is exponential in $|\psi|$.

The lower bound for the length of a shortest $K$-bad-prefix follows from the construction described in the proof of Theorem 9 by using the Kripke structure $K_{AP}$.

Note that the space complexity of the algorithm described in the proof of Theorem 14 in also polylogarithmic in $|K|$. Finally, when the specification formalism is automata, the
difference between $K$-safety and $K$-counterable languages disappears, and a short $K$-bad-prefix for a $K$-safety or $K$-counterable language given by an NBW $A$ is tightly exponential in $|A|$ and linear in $|K|$.

7 Probabilistic relative counterability

In this section we study $K$-prob-counterability. We start with the corresponding decision problem:

**Theorem 15** Consider a language $L \subseteq (2^{AP})^\omega$ and a Kripke structure $K$ over $AP$. Deciding whether $L$ is $K$-prob-counterable can be done in time $O(|K| \cdot 2^{O(|L|)})$ or in space polynomial in $|L|$ and polylogarithmic in $|K|$, for $L$ given by an LTL formula $\psi$, in which case $|L| = |A|$. In both cases, the problem is PSPACE-complete.

**Proof** Let $P$ be a $K$-walk-distribution. By Theorem 4, the language $L$ is $K$-prob-counterable iff $Pr_{K,P}[L] < 1$. By Theorems 3.1.2.1 and 4.1.7 in [11], the latter can be checked in the required complexities. According to [40], the problem is PSPACE-hard. $\square$

Thus, deciding whether an LTL formula is $K$-prob-counterable is exponentially easier than in the non-probabilistic case.

We turn to the problem of finding a $K$-prob-bad-prefix for an $\omega$-regular language. Handling a language given by an NBW can proceed by an exponential translation to a DPW [33]. For languages given by LTL formulas, going to a DPW involves a doubly-exponential blow-up. We show that in order to find a $K$-prob-bad-prefix for an LTL formula, we can carefully proceed according to the syntax of the formula and do exponentially better than an algorithm that translates the formula to automata. We note that the PSPACE-hardness for NBWs in Theorem 15 implies that we cannot hope to obtain a PSPACE algorithm for LTL by translating LTL formulas to NBWs, unless the structure of the latter is analyzed to a level in which it essentially follows the structure of the LTL formula (see, for example [12] for probabilistic LTL model checking).

7.1 Probabilistic relative counterability of NBWs

Recall that in Sect. 3.3 we constructed an NPW $D_{K \times D}$ and a labeled Markov chain $M'$. We also defined $\gamma_x$ as the probability that a random walk from a state $x$ in $M'$ is an accepting run in $D_{K \times D}$, and showed that for a finite computation $u$ we have $\gamma_{\text{reach}(u)} = Pr_{M'}[L] = Pr_{K,P}[L]$. That construction, together with the following lemma, is useful for finding $K$-prob-bad-prefixes.

**Lemma 3** Deciding whether $\gamma_x$ is 0, 1 or in $(0, 1)$, for all states $x$ in $M'$, can be done in time linear in $|D|$ and in $|K|$ or in space polylogarithmic in $|D|$ and in $|K|$. Furthermore, the probability $\gamma_x$ can be calculated in time polynomial in $|D|$ and in $|K|$.

**Proof** First we show that determining for every $x$ whether $\gamma_x$ is 0, 1 or in $(0, 1)$ can be done in time linear in $|D_{K \times D}|$ or in space polylogarithmic in $|D_{K \times D}|$.

We first find the ergodic SCCs in $D_{K \times D}$. The partition of $D_{K \times D}$ to SCCs can be done in time linear in $|D_{K \times D}|$ [10]. Then, we partition the ergodic SCCs of $D_{K \times D}$ to accepting and rejecting. Note that we can find in time linear in $|D_{K \times D}|$ for each ergodic SCC $C$ the minimal $i$ such that $C \cap a'^{-1}(i) \neq \emptyset$, and therefore the partition of the ergodic SCCs can be done in
time linear in $|D_K \times D|$. By Lemma 1, for each state $x$ in an ergodic SCC of $D_K \times D$, we have $\gamma_x = 1$ iff the ergodic SCC is accepting and $\gamma_x = 0$ iff the ergodic SCC is rejecting. Also, for each state $x$ in $D_K \times D$, we have $\gamma_x = 1$ iff there is no path in $D_K \times D$ from $x$ to a rejecting ergodic SCC, and $\gamma_x = 0$ iff there is no path in $D_K \times D$ from $x$ to an accepting ergodic SCC. We can check for each SCC whether it can reach an accepting ergodic SCC and whether it can reach a rejecting ergodic SCC (reverse reachability) in time linear in $|D_K \times D|$.

We now show how to determine for every $x$ whether $\gamma_x$ is 0, 1 or in $(0, 1)$ in space polylogarithmic in $|D_K \times D|$. We need to check whether $x$ can reach a state in an accepting ergodic SCC and whether $x$ can reach a state in a rejecting ergodic SCC. Therefore, we check for every state $y$ in $D_K \times D$ the following conditions: first, we check whether $x$ can reach $y$, if so, we check whether $y$ is in an ergodic SCC and whether the ergodic SCC is accepting or rejecting. In order to check if a state $y$ is in an ergodic SCC we cycle over all possible states $z$ and for each $z$ we check whether there is a path from $y$ to $z$ and a path from $z$ to $y$. The state $y$ is not in an ergodic SCC iff for some $z$ there is a path from $y$ to $z$ but not from $z$ to $y$. In order to check whether the ergodic SCC of a state $y$ is accepting or rejecting we cycle over all possible states $z$ and for each $z$ we check whether there is a path from $y$ to $z$ (thus they are in the same ergodic SCC) and check the value of $\alpha'(z)$. All of the above reachability problems can be solved in space polylogarithmic in $|D_K \times D|$.

For calculating $\gamma_x$ we first find the ergodic SCCs in $D_K \times D$, and check for every ergodic SCC whether it is accepting or rejecting. By Lemma 1, the probability $\gamma_x$ of a state $x$ in $M'$ is the probability of a random walk in $M'$ from $x$ to reach an accepting ergodic SCC. This probability can be calculated for every $x$ in $M'$ in time polynomial in $|D_K \times D|$ (see, for example, Chapter 11.2 in [17]).

We now show an algorithm for finding a shortest $K$-prob-bad-prefix for a language given by an NBW $A$.

Theorem 16 Consider an NBW $A$ over the alphabet $2^{AP}$ and a Kripke structure $K$ over $AP$. Finding a shortest $K$-prob-bad-prefix for $L(A)$ can be done in space polynomial in $|A|$ and polylogarithmic in $|K|$ or in time exponential in $|A|$ and linear in $|K|$. Furthermore, the length of the shortest $K$-prob-bad-prefix for $L(A)$ is tightly exponential in $|A|$ and tightly linear in $|K|$.

Proof Let $D$ be a DPW equivalent to $A$ and let $M'$ be the labeled Markov chain induced by $D$ and $K$ as described in Sect. 3.3. We need to find a shortest word whose path in $M'$ reaches a state $x$ for which $\gamma_x = 0$. By Lemma 3, we can decide in space polynomial in $|A|$ and polylogarithmic in $|K|$ or in time exponential in $|A|$ and linear in $|K|$, for every state $x$, whether $\gamma_x = 0$. Thus, the problem can be reduced to the problem of finding shortest paths in a graph of size exponential in $|A|$ and linear in $|K|$. The required complexities then follow from the fact a shortest path is simple, and the complexity of known shortest-path algorithms [10]. Finally, the lower bounds with respect to $|A|$ and $|K|$ for the length of a shortest $K$-prob-bad-prefix follow from the constructions described in the proofs of Theorem 10 (by using the Kripke structure $K_{AP}$) and Theorem 13, respectively. □

Consider a Kripke structure $K$ over $AP$ and a language $L \subseteq (2^{AP})^\omega$ that is $K$-prob-counterable. In practice, the user of the model-checking tool often has some estimation of the likelihood of every transition in the system. That is, we assume that the user knows what the typical $K$-walk-distribution $P$ in a typical behavior of the system is. Clearly, there is a trade-off between the length of a counterexample and its “precision”, in the sense that the longer a finite prefix of an erroneous computation is, the larger is the probability in which it...
Lemma 3, the probability $\Pr_{K, \gamma}^u$ we need to find a shortest finite computation whose path in $M'$ reaches a state $s$ such that $\Pr_{M}^u[L(A)] < \gamma$. We therefore need to find a shortest finite computation whose path in $M'$ reaches a state $s$ such that $\gamma_s < \gamma$. For a length $m > 0$ we need to find a shortest finite computation of length at most $m$ such that $\Pr_{M}^u[L(A)]$ is minimal. Therefore we need to find a shortest finite computation of length at most $m$ whose path in $M'$ reaches a state $s$ such that $\gamma_s$ is minimal. Both tasks can be done using BFS in $D_{K \times D}$ from its initial states as required.

We note that using our construction of $M'$, together with Lemma 3, we can reduce the calculation of $\Pr_{K, \gamma}^u[L(A)]$ or the problem of its classification to 1, 0, or (0, 1) to a sequence of calculations in $M'$, simplifying the known result of [11] (Theorem 4.1.7 there). Formally, we have the following.

Theorem 17 The shortest bounded-prob-$K$-bad-prefix and the bounded-length prob-$K$-bad-prefix problems, for a language given by an NBW $A$, can be solved in time exponential in $|A|$ and polynomial in $|K|$.

Proof Let $\mathcal{D}$ be a DPW equivalent to $A$. Recall that $|\mathcal{D}|$ is exponential in $|A|$. According to Lemma 3, the probability $\gamma_s$ for every state $s$ in $M'$ can be calculated in time exponential in $|A|$ and polynomial in $|K|$. For a probability $0 < \gamma < 1$ we need to find a shortest finite computation $u$ of $K$ such that $\Pr_{M}^u[L(A)] < \gamma$. Therefore, we need to find a shortest finite computation whose path in $M'$ reaches a state $s$ such that $\gamma_s < \gamma$. For a length $m > 0$ we need to find a finite computation $u'$ of $K$ of length at most $m$ such that $\Pr_{M}^u[L(A)]$ is minimal. Therefore we need to find a finite computation of length at most $m$ whose path in $M'$ reaches a state $x'$ such that $\gamma_{x'}$ is minimal. Both tasks can be done using BFS in $D_{K \times D}$ from its initial states as required.

We now show an algorithm for finding a $K$-prob-bad-prefix for an LTL formula $\theta$. Since the algorithm is based on the construction of [11], we start with an overview of that construction.

Let $K = \langle A, P, W, W_0, R, l \rangle$ be a Kripke structure and let $\theta$ be an LTL formula. We assume that the only temporal operators in $\theta$ are $X$ and $U$. Since $F$ and $G$ can be expressed with $U$, this does not restrict attention. Let $P = \langle p_{in}, p \rangle$ be a $K$-walk-distribution and let

7.2 Probabilistic relative counterability of LTL formulas

We now show an algorithm for finding a $K$-prob-bad-prefix for an LTL formula $\theta$. Since the algorithm is based on the construction of [11], we start with an overview of that construction.

Let $K = \langle A, P, W, W_0, R, l \rangle$ be a Kripke structure and let $\theta$ be an LTL formula. We assume that the only temporal operators in $\theta$ are $X$ and $U$. Since $F$ and $G$ can be expressed with $U$, this does not restrict attention. Let $P = \langle p_{in}, p \rangle$ be a $K$-walk-distribution and let
$M = M_{K, p}$ be the labeled Markov chain induced by $K$ and $P$. The algorithm of [11] checks whether $Pr_{M}(L(\theta)) = 0$ (or whether $Pr_{M}(L(\theta)) = 1$) in time $O(|K| \cdot 2^{[\theta]}$). It proceeds by iteratively replacing the innermost temporal sub-formula of $\theta$ by a fresh atomic proposition, and adjusting $M$ so that the probability of satisfying $\theta$ is maintained. More formally, in each iteration we transform $(M, \theta)$ to $(M', \theta')$ such that $\theta'$ replaces an innermost temporal subformula of $\theta$ by a new atomic proposition $\xi$, and $Pr_{M}(L(\theta)) = Pr_{M}(L(\theta'))$. There are two transformations, denoted $CU$ and $CX$, corresponding to the two temporal operators $U$ and $X$. We now describe these transformations.

We start with the transformation $CU$. Let $\varphi U \psi$ be an innermost temporal subformula of $\theta$; that is, $\varphi$ and $\psi$ are Boolean assertions of atomic propositions and can be evaluated on each state of $K$. The new labeled Markov chain $M' = M_{K, p'}$ is defined over a Kripke structure $K'$ with a new atomic proposition $\xi$, which is going to replace $\varphi U \psi$ in $\theta'$. Formally, $K' = \langle AP \cup \{\xi\}, W', O', \pi' \rangle$, where $W' \subseteq W \times \xi, \neg \xi \rangle$ is defined as follows. First, we partition $W$ into $W^{YES} \cup W^{NO} \cup W^{?}$ such that if a random walk starts from a state in $W^{YES}$ (respectively, $W^{NO}$), then with probability 1 it satisfies the formula $\varphi U \psi$ (respectively, $\neg (\varphi U \psi))$, and if it starts from a state in $W^{?}$ then both events have nonzero probability. Every state in $W^{?}$ satisfies $\varphi \land \neg \psi$. Also, from every state in $W^{?}$, there is a finite path to a state in $W^{YES}$ that visits only states in $W^{?}$, and a finite path to a state in $W^{NO}$ that visits only states in $W^{?}$. Now, $W' = W^{YES} \times \{\xi\} \cup W^{NO} \times \{\neg \xi\} \cup W^{?} \times \{\xi, \neg \xi\}$. The function $l'$ is defined as: $l'(w, \xi) = l(w) \cup \{\xi\}$ and $l'(w, \neg \xi) = l(w)$. The set of initial states is $W_0' = W_0 \cap (W_0 \times \xi, \neg \xi\}, and the transition relation $R'$ is contained in $\{\langle w, \xi_1 \rangle, \langle w', \xi_2 \rangle: \xi_1, \xi_2 \in \xi, \neg \xi\} \cup R(w, w') \cap W^{?} \times W^{?}$.

The transformation $CX$ follows similar lines. Let $X \varphi$ be an innermost temporal subformula of $\theta$. The partition of the states in $W$ is now such that a state in $W^{YES}$ (respectively $W^{NO}$) has transitions only to states satisfying $\varphi$ (respectively $\neg \varphi$), and a state in $W^{?}$ has transitions to states satisfying $\varphi$ and to states satisfying $\neg \varphi$.

Formally, the set of transitions $R'$ for transformation $CX$ is defined as follows: $R' = \{((u, \xi_1), (v, \xi_2)): R(u, v), \xi_1, \xi_2 \in \xi, \neg \xi\}, u \in W^{YES} \cup W^{NO} \cup \langle u, \xi_1 \rangle, \langle v, \xi_2 \rangle \in W'\} \cup \{((u, \xi'), (v, \xi'')): R(u, v), \xi', \xi'' \in \xi, \neg \xi\}, u \in W^{?} \cup \langle u, \xi' \rangle, \langle v, \xi'' \rangle \in W'\}$. The set of transitions $R'$ for transformation $CX$ is defined by $R' = \{((u, \xi_1), (v, \xi_2)): R(u, v), \xi_1 = \xi$ if $u$ satisfies $\varphi$ and otherwise $\xi_1 = \neg \xi, \xi_2 \in \xi, \neg \xi\} \cup \langle u, \xi_1 \rangle, \langle v, \xi_2 \rangle \in W'\}$.

The $K'$-walk-distribution $P'$ is defined so that the transition probability of moving from state $\langle w, \beta \rangle$ to state $\langle w', \beta' \rangle$, for $\beta, \beta' \in \{\xi, \neg \xi\}$, is the probability that $M$, starting from state $w$, moves to state $w'$ and from state $w'$ onward satisfies $\beta'$, conditioned on the event that in state $w$ it satisfies $\beta$. The detailed definitions of $P'$ can be found in [11].

Let $g$ be the mapping of states in $K'$ into states in $K$ by projecting on the first component of the states of $K'$. The properties of $M'$ that are important for our algorithm are described below.

1. Let $X = X_1, X_2, \ldots$ be a random walk on $M$, and let $X' = X'_1, X'_2, \ldots$ be a random walk on $M'$. Let $Y = Y_1, Y_2, \ldots$ be defined by $Y_i = g(X'_i)$ for $i = 1, 2, \ldots$. Note that $X$ and $Y$ both take values in $W^{\omega}$ and that they have the same distribution.
2. Let $x'_1, \ldots, x'_m$ be a finite path in $K'$ and let $x_1, \ldots, x_m$ be its projection on the first component (that is, $x_j = g(x'_j)$ for $1 \leq j \leq m$). Then, $x'_1, \ldots, x'_m$ is the only path in $K'$ that ends in $x'_m$ and has projection $x_1, \ldots, x_m$.
3. A random walk $X' = X'_1, X'_2, \ldots$ on $M'$ satisfies the following property with probability 1: for every $j \geq 0$, the atomic proposition $\xi$ is satisfied in state $X'_j$ iff $X'_j, X'_{j+1}, \ldots$ satisfies the formula $\varphi U \psi$ in the case of transformation $CU$ and the formula $X \psi$ in the case of transformation $CX$. 

© Springer
4. \( P_{RM}(L(\theta)) = P_{RM^i}(L(\theta')) \).

If \( \theta \) has \( k \) temporal operators then, by Property 4, we can compute \( P_{RM}(L(\theta)) \) as follows: We apply \( k \) times the appropriate transformations \( C_U \) and \( C_X \) in order to get the sequence \((M^1, \theta^1), \ldots, (M^k, \theta^k)\), where \( \theta^k \) does not contain temporal operators. Then, \( P_{RM}(L(\theta)) = P_{RM^k}(L(\theta^k)) \), which is simply the sum of the initial probabilities in \( M^k \) over all states satisfying \( \theta^k \). In order to check whether \( P_{RM}(L(\theta)) > 0 \) (or whether \( P_{RM}(L(\theta)) < 1 \)), it is enough to construct the Kripke structures \( K^1, \ldots, K^k \) corresponding to \( M^1, \ldots, M^k \) and check whether \( K^k \) has an initial state satisfying \( \theta^k \) (respectively, \( \neg \theta^k \)). Thus, according to Theorem 4, the LTL formula \( \theta \) is \( K \)-prob-counterable iff \( K^k \) has an initial state that does not satisfy \( \theta^k \). For every \( i \), the construction of the Kripke structure \( K^{i+1} \) is based only on \( K^i \) and is independent of the probabilities in \( M^i \). Thus, we can construct \( K^1, \ldots, K^k \) from \( K \) without constructing \( M^1, \ldots, M^k \). The construction of \((K^1, \theta^1), \ldots, (K^k, \theta^k)\) can be done in time \( O(|K| \cdot 2^{\|\theta^i\|}) \) or in space polynomial in \(|\theta|\) and polylogarithmic in \(|K|\).

We now use a variant of this idea in order to find \( K \)-prob-bad-prefixes:

**Theorem 19** Finding a \( K \)-prob-bad-prefix for a \( K \)-prob-counterable LTL formula \( \theta \) can be done in time \( O(|K| \cdot 2^{\|\theta^i\|}) \) or in space polynomial in \(|\theta|\) and polylogarithmic in \(|K|\). Furthermore, the \( K \)-prob-bad-prefix that is found is of length \( O(|K| \cdot 2^{\|\theta^i\|}) \).

**Proof** Let \( M = M_{K,p} \) be a labeled Markov chain over \( K \) with respect to some \( K \)-walk-distribution \( P \), and let \((M^1, \theta^1), \ldots, (M^k, \theta^k)\) be the construction described above. Let \( K^1, \ldots, K^k \) be the Kripke structures corresponding to \( M^1, \ldots, M^k \), respectively. Recall that the construction of \((K^1, \theta^1), \ldots, (K^k, \theta^k)\) is independent of the \( K \)-walk-distribution \( P \). We first show how to find a \( K^k \)-prob-bad-prefix for \( \theta^k \). Then, we show how to find a \( K^i \)-prob-bad-prefix for \( \theta^i \) given a \( K^{i+1} \)-prob-bad-prefix for \( \theta^{i+1} \), with \((K^0, \theta^0) = (K, \theta)\). Thus, we end up with the required \( K \)-prob-bad-prefix for \( \theta \).

Let \( K^k = \langle AP^k, W^k, W'_k, R^k, l^k \rangle \). Recall that \( \theta^k \) does not contain temporal operators, and thus the formula \( \theta \) is \( K \)-prob-counterable iff \( K^k \) has an initial state that does not satisfy \( \theta^k \). Therefore, if \( \theta \) is \( K \)-prob-counterable, then there is a \( K^k \)-prob-bad-prefix for \( \theta^k \) of length 1, obtained by a state \( x_1 \in W^0 \) such that \( l^k(x_1) \) does not satisfy \( \theta^k \).

Let \( K^i = \langle AP, W, W_0, R, l \rangle \) and \( K^{i+1} = \langle AP', W', W'_0, R', l' \rangle \) be the Kripke structures corresponding to \( M^i \) and \( M^{i+1} \) respectively. Let \( x_1', \ldots, x_m' \in W^{m'} \) be such that \( l'(x_1'), \ldots, l'(x_m') \) is a \( K^{i+1} \)-prob-bad-prefix for \( \theta^{i+1} \) and let \( x_1, \ldots, x_m \in W^m \) be its projection on the \( W \)-element. That is, \( x_j' = g(x_j) \), for \( 1 \leq j \leq m \). We construct a \( K^i \)-prob-bad-prefix for \( \theta^i \).

We define an LTL formula \( \tau \) as follows. If \( M^{i+1} \) is constructed with transformation \( C_U \), we proceed as follows. If \( x_m' = (x_m, \xi) \), then \( \tau = \varphi U \psi \) where \( \varphi U \psi \) is the subformula that is replaced in the transformation, and if \( x_m' = (x_m, \neg \xi) \), then \( \tau = \neg(\varphi U \psi) \). Likewise, if \( M^{i+1} \) is constructed with transformation \( C_X \), then \( x_m' = (x_m, \xi) \) then \( \tau = X \varphi \) where \( \varphi \) is the subformula that is replaced in the transformation, and otherwise \( \tau = \neg X \varphi \).

Since \( l'(x_1'), \ldots, l'(x_m') \) is a \( K^{i+1} \)-prob-bad-prefix for \( \theta^{i+1} \), then according to its definition as a conditional probability, we have that \( P_{RM^{i+1}}(l'(y_1), l'(y_2), \ldots : l'(y_1), l'(y_2), \ldots \mid \theta^{i+1}, x_j = x'_j, \) for \( 1 \leq j \leq m \)) = 0. Therefore, by Properties 2 and 3, we have \( P_{RM^{i+1}}(l'(y_1), l'(y_2), \ldots : l'(y_1), l'(y_2), \ldots \mid \theta^{i+1}, y_j = y_j \) for all \( 1 \leq j \leq m \), and \( l'(y_m), l'(y_{m+1}), \ldots \mid \tau) = 0. \) By Property 3, the following property holds with probability 1: the computation \( l'(y_1), l'(y_2), \ldots \mid \theta^{i+1} \) if \( l'(y_1), l'(y_2), \ldots \mid \theta^i \). Therefore, \( P_{RM^{i+1}}(l'(y_1), l'(y_2), \ldots : l'(y_1), l'(y_2), \ldots \mid \theta^i, y_j = y_j \) for all \( 1 \leq j \leq m \), and \( l'(y_m), l'(y_{m+1}), \ldots \mid \tau) = 0. \) Note that \( \theta^i \) and \( \tau \) do not contain the atomic proposition \( \xi \).
Therefore, a computation of $K_i^{i+1}$ satisfies $\theta^i$ (respectively $\tau$) iff its projection in $K_i$ satisfies $\theta^i$ (respectively $\tau$). Therefore, by Property 1, we conclude that $Pr_{M^i}(l(z_1),l(z_2),\ldots:l(z_j),l(z_j+1),\ldots:l(\tau)) = 0$. It follows that we need to find in $K_i$ a finite computation of the form $l(x_1),\ldots,l(x_m)$ such that it is guaranteed that $\tau$ is satisfied with probability 1 in a computation that starts with $l(x_m),\ldots,l(x_n)$ and continues with a random walk on $M^i$. For such $x_1,\ldots,x_n$, we have $Pr_{M^i}(l(z_1),l(z_2),\ldots:l(z_j),l(z_j+1),\ldots:l(\tau) \land x_j = x_j$ for $1 \leq j \leq n$) = 0. Hence, the finite computation $l(x_1),\ldots,l(x_n)$ is a $K^i$-prob-bad-prefix for $\theta^i$.

Case (a): Transformation $C_U$. Assume that $\tau = \varphi U \psi$ (respectively $\tau = \neg(\varphi \psi)$). Note that $x_m \not\in W^{NO}$ (respectively $x_m \not\in W^{YES}$), because otherwise the state $x_m' = (x_m,\xi)$ (respectively $x_m' = (x_m,\neg\xi)$) would not have been existed in $M_i^{i+1}$. If $x_m \in W^{YES}$ (respectively $x_m \in W^{NO}$) then a random walk starting from $x_m$ satisfies $\varphi U \psi$ (respectively $\neg (\varphi \psi)$) with probability 1 and therefore $l(x_1),\ldots,l(x_m)$ is a $K^i$-prob-bad-prefix for $\theta^i$. If $x_m \in W^2$ then there is a finite path from $x_m$ to some $x_n \in W^{YES}$ (respectively $x_n \in W^{NO}$) that visits only states in $W^2$ (these states satisfy $\varphi \land \neg \psi$). The length of this finite path $x_m,\ldots,x_n$ is at most $|W|$ and it can be found in time $O(|K^i|)$ or in space polylogarithmic in $O(|K^i|)$. We add $l(x_{m+1}),\ldots,l(x_n)$ to $l(x_1),\ldots,l(x_m)$ and get a $K^i$-prob-bad-prefix for $\theta^i$.

Case (b): Transformation $C_X$. Assume that $\tau = X \varphi$ (respectively $\tau = \neg X \varphi$). Note that $x_m \not\in W^{NO}$ (respectively $x_m \not\in W^{YES}$). If $x_m \in W^{YES}$ (respectively $x_m \in W^{NO}$) then $l(x_1),\ldots,l(x_m)$ is a $K^i$-prob-bad-prefix for $\theta^i$. If $x_m \in W^2$, then $x_m$ has a successor $x_{m+1}$ that satisfies $\varphi$ (respectively $\neg \varphi$). We add $l(x_{m+1})$ to $l(x_1),\ldots,l(x_m)$ and get a $K^i$-prob-bad-prefix for $\theta^i$.

We turn to analyze the complexity of the algorithm. For the construction of the $K$-prob-bad-prefix for $\theta$ described above, we need to construct $(K^1,\theta^1),\ldots,(K^k,\theta^k)$ (the partition of the states to $W^{YES} \cup W^{NO} \cup W^2$ in each step is a part of the construction). For every $i$, the construction of the Kripke structure $K_i^{i+1}$ is done based only on $K_i$ and not on the probabilities in $M^i$. Therefore, in order to construct $K^1,\ldots,K^k$ we do not have to construct $M^1,\ldots,M^k$. The construction of $(K^1,\theta^1),\ldots,(K^k,\theta^k)$ can be done in time $O(|K| \cdot 2^{\theta i})$, or in space polynomial in $|\theta|$ and polylogarithmic in $|K|$. Recall that $k = O(|\theta|)$ and that for every $i$ we have $|K^i| = O(|K| \cdot 2^{\theta i})$. In each of the steps in the construction of the $K$-prob-bad-prefix we add a sequence of length $O(|K^i|)$. This sequence can be found in time $O(|K^i|)$. It can also be found on-the-fly in space polylogarithmic in $O(|K^i|)$. Therefore both the time complexity and the length of the $K$-prob-bad-prefix are $O(|K| \cdot 2^{\theta i})$. The space complexity is polynomial in $|\theta|$ and polylogarithmic in $|K|$.

We now study the length of shortest $K$-prob-bad-prefixes.

**Theorem 20** The length of a shortest $K$-prob-bad-prefix for a $K$-prob-counterable language given by an LTL formula $\psi$ is tightly exponential in $|\psi|$ and tightly linear in $|K|$.

**Proof** The upper bounds follow from Theorem 19. The lower bound with respect to $|\psi|$ follows from the construction described in the proof of Theorem 9 by using the Kripke structure $K_{AP}$. The lower bound with respect to $|K|$ follows from the construction described in the proof of Theorem 13.

Note that the $K$-prob-bad-prefix that our algorithm finds is not necessarily the shortest, however it matches the lower bound from Theorem 20.
Thus, we showed that the probabilistic approach for relative bad prefixes for LTL formulas is exponentially better than the non-probabilistic approach both in its complexity and in the length of the prefixes.

8 Probabilistic counterability

In this section we study prob-counterability. The solutions to the three basic problems are specified in Theorems 21, 22, and 23 below.

Theorem 21 The problem of deciding whether a language $L$ is prob-counterable is PSPACE-complete for $L$ given by an LTL formula or by an NBW.

Proof According to Theorem 1, an $\omega$-regular language $L$ is prob-counterable iff $Pr[L] < 1$. Let $\psi$ be an LTL formula and let $A$ be an NBW. By [11], deciding whether $Pr[L(\psi)] < 1$ and whether $Pr[L(A)] < 1$ can be done in PSPACE. A matching lower bound is described in [13] for LTL. The proof in [13] is by a generic reduction and uses the ability of LTL to encode the computations of a PSPACE Turing machine. The same idea works also for NBW, thus deciding whether $Pr[L] < 1$ is PSPACE-hard for $L$ given by an LTL formula or by an NBW.

Theorem 22 Let $A$ be an NBW. Finding a shortest prob-bad-prefix for $L(A)$ can be done in time exponential in $|A|$, or in space polynomial in $|A|$. Furthermore, the length of the shortest prob-bad-prefix is tightly exponential in $|A|$.

Proof A computation is a prob-bad-prefix iff it is a $KAP$-prob-bad-prefix. Hence, a shortest prob-bad-prefix can be found by using the algorithm from Theorem 16 with $KAP$. The lower bound for the length of a shortest prob-bad-prefix follows from the same construction as in the proof of Theorem 9.

Theorem 23 Finding a prob-bad-prefix for a prob-counterable LTL formula $\psi$ can be done in time $2^{O(|\psi|)}$ or in space polynomial in $|\psi|$. Furthermore, the prob-bad-prefix that is found is of length $2^{O(|\psi|)}$. The shortest prob-bad-prefix for $\psi$ is tightly exponential in $|\psi|$.

Proof A prob-bad-prefix can be found by using the algorithm from Theorem 19 with the Kripke structure $KAP$. The lower bound for the length of a shortest prob-bad-prefix follows from the same construction as in the proof of Theorem 9.

Thus, the exponential advantage of the probabilistic approach in the case the language is given by an LTL formula is carried over to the non-relative setting. When the specification is given by means of an NBW, the complexities of the probabilistic and non-probabilistic approaches coincide. The probabilistic approach, however, may return more bad prefixes.

9 Discussion

We extended the applicability of finite counterexamples by introducing relative and probabilistic bad-prefixes. This lifts the advantage of safety properties, which always have bad-prefixes, to $\omega$-regular languages that are not safety. We believe that $K$-bad-prefixes and $K$-prob-bad-prefixes may be very helpful in practice, as they describe a finite execution that leads the system to an error state. From a computational point of view, finding a
-bad-prefix for an LTL formula $\psi$ is unfortunately EXPSPACE-complete in $|\psi|$. Experience shows that even highly complex algorithms often run surprisingly well in practice. Also here, the complexity originates from the blow-up in the translation of LTL to automata, which rarely happens in practice. In cases the complexity is too high, we suggest the following two alternatives, which do not go beyond the PSPACE complexity of LTL model checking (and, like model checking, are NLOGSPACE in $K$): (1) Recall that when $\psi$ is $K$-safety and $K \not\models\psi$, then finding a $K$-bad-prefix can be done in PSPACE. Thus, we suggest to check $\psi$ for $K$-safety with the algorithm from Theorem 11, and then apply the algorithm from Theorem 14. (2) Recall that finding a $K$-prob-bad-prefix is only PSPACE-complete. Thus, we suggest to apply the algorithm from Theorem 19. Note that the probabilistic approach is not only exponentially less complex, but may be essential when $\psi$ is $K$-prob-countable and not $K$-counterable.

When a user gets a lasso-shaped counterexample, he can verify that indeed it does not satisfy the specification. For finite bad-prefixes, the user knows that they lead the system to an error state, and it is desirable to accompany the prefix with information explaining why these states are erroneous. We suggest the following three types of explanations. (1) A $K$-bad-prefix leads the product $K \times A_\psi$ to states $\langle w, S \rangle$ that are empty. Recall that the states of $A_\psi$ consist of subsets of subformulas of $\psi$, and that $\langle w, S \rangle$ being empty means that $w$ does not satisfy the conjunction of the formulas in $S$ [41]. Returning $S$ to the user explains what makes $w$ an error state. (2) Researchers have studied certified model checking [26], where a positive answer of model checking (that is, $K \models \psi$) is accompanied by a certificate—a compact explanation as to why $K \times A_{\neg\psi}$ is empty. In our setting, certificates can provide a compact explanation as to why $K \times A_\psi$ with initial state $\langle w, S \rangle$ is empty. (3) When a $K$-prob-bad-prefix $u$ that is not a $K$-bad-prefix is returned, it may be helpful to accompany $u$ with an infinite lasso-shaped computation $\tau$ of the system that starts with $u$ and does satisfy the specification. Thus, the user would get an exception: he would know that almost all computations that start in $u$ except for $\tau$ (and possibly more computations, whose probability is 0) violate $\psi$. The exceptional correct behavior would help the user understand why almost all other behaviors are incorrect.

Finally, our probabilistic approach has been extended in [15], which introduces a quantitative measure for safety. Essentially, the safety level of a language $L$ measures the fraction of words not in $L$ that have a bad prefix. In particular, a safety language has safety level 1 and a liveness language has safety level 0, thus, the study in [15] uses our probabilistic approach in order to span the spectrum between safety and liveness.

Acknowledgements We thank Moshe Y. Vardi for helpful discussions.

References

1. Ábrahám E, Becker B, Dehnert C, Jansen N, Katoen JP, Wimmer R (2014) Counterexample generation for discrete-time markov models: an introductory survey. In: Bernardo M, Damiani F, Hähnle R, Johnsen EB, Schaefer I (eds) Formal methods for executable software models. Springer, Berlin, pp 65–121
2. Alpern B, Schneider FB (1985) Defining liveness. Inf Process Lett 21:181–185
3. Alpern B, Schneider FB (1987) Recognizing safety and liveness. Distrib Comput 2:117–126
4. Basin DA, Jiménez CC, Klaedtke F, Zalinescu E (2014) Deciding safety and liveness in TPTL. Inf Process Lett 114(12):680–688
5. Biere A, Artho C, Schuppan V (2002) Liveness checking as safety checking. In: Proceedings of 7th international workshop on formal methods for industrial critical systems, volume 66:2 of Electronic Notes in Theoretical Computer Science
6. Büchi JR (1962) On a decision method in restricted second order arithmetic. In: Proceedings of international congress on logic, method, and philosophy of science, 1960. Stanford University Press, pp 1–12
7. Clarke EM, Biere A, Raimi R, Zhu Y (2001) Bounded model checking using satisfiability solving. Form Methods Syst Des 19(1):7–34
8. Clarke EM, Grumberg O, Jha S, Lu Y, Veith H (2000) Counterexample-guided abstraction refinement. In: Proceedings of 12th international conference on computer aided verification, volume 1855 of Lecture Notes in Computer Science. Springer, Berlin, pp 154–169
9. Clarke EM, Grumberg O, McMillan KL, Zhao X (1995) Efficient generation of counterexamples and witnesses in symbolic model checking. In: Proceedings of 32nd design automation conference. IEEE Computer Society, pp 427–432
10. Cormen TH, Leiserson CE (1990) Introduction to algorithms. MIT Press and McGraw-Hill, Cambridge
11. Courcoubetis C, Yannakakis M (1995) The complexity of probabilistic verification. J ACM 42:857–907
12. Couvreur JM, Saheb N, Sutre G (2003) An optimal automata approach to LTL model checking of probabilistic systems. In: Proceedings 10th international conference on logic for programming artificial intelligence and reasoning. Springer, Berlin, pp 361–375
13. David BS, Kupferman O (2013) A framework for ranking vacuity results. 11th international symposium on automated technology for verification and analysis, volume 8172 of Lecture Notes in Computer Science. Springer, Berlin, pp 148–162
14. Hariharan S, Pnueli A (1981) Temporal logic and the synthesis of asynchronous sequential circuits. In: Proceedings of 2nd annual ACM SIGACT-SIGPLAN symposium on principles of programming languages. ACM, New York, pp 228–240
15. Havelund K, Rosu G (2004) Efficient monitoring of safety properties. Softw Tools Technol Transf 6(2):18–173
16. Henzinger TA, Kupferman O, Qadeer S (1998) From pre-historic to post-modern symbolic model checking. In: Proceedings of 10th international conference on computer aided verification, volume 1427 of Lecture Notes in Computer Science. Springer, Berlin
17. Henzinger TA, Kupferman O, Vardi MY (2000) Model checking of safety properties. In: Proceedings of 12th international conference on computer aided verification, volume 1855 of Lecture Notes in Computer Science. Springer, Berlin, pp 361–375
18. Kemeny JG, Snell JL, Knapp AW (1976) Denumerable Markov chains. Springer, Berlin
19. Kupferman O, Lampert R (2006) On the construction of finite automata from safety properties. In: 4th international symposium on software verification and analysis, volume 4218 of Lecture Notes in Computer Science. Springer, Berlin, pp 110–124
20. Kupferman O, Vardi MY (2001) Model checking of safety properties. Form Methods Syst Des 19(3):291–314
21. Kupferman O, Vardi MY (2004) From complementation to certification. In: Proceedings of 10th international conference on tools and algorithms for the construction and analysis of systems, volume 2988 of Lecture Notes in Computer Science. Springer, Berlin, pp 591–606
22. Lipton RJ (1976) The role of heuristics in the development of algorithms. J Assoc Comput Mach 23(3):368–375
23. Pnueli A (1977) The temporal logic of programs. In: Proceedings of 1977 ACM symposium on theory of computing. ACM, New York, pp 160–165
24. Pnueli A (1977) The temporal logic of programs. In: Proceedings of 1977 ACM symposium on theory of computing. ACM, New York, pp 160–165
25. Pnueli A, Rosner R (1989) On the temporal logic of programs. J Assoc Comput Mach 36(4):981–1029
26. Pnueli A, Rosner R (1989) On the temporal logic of programs. J Assoc Comput Mach 36(4):981–1029
27. Pnueli A (1981) The temporal semantics of concurrent programs. Theor Comput Sci 13:45–60
35. Schuppan V, Biere A (2005) Shortest counterexamples for symbolic model checking of LTL with past. In: Proceedings of 11th international conference on tools and algorithms for the construction and analysis of systems, volume 3440 of Lecture Notes in Computer Science. Springer, Berlin, pp 493–509
36. Schuppan V, Biere A (2006) Liveness checking as safety checking for infinite state spaces. Electron Notes Theor Comput Sci 149(1):79–96
37. Sistla AP (1994) Safety, liveness and fairness in temporal logic. Form Aspects Comput 6:495–511
38. Sistla AP, Clarke EM (1985) The complexity of propositional linear temporal logic. J ACM 32:733–749
39. van Emde Boas P (1997) The convenience of tilings. Lecture Notes on Pure and Applied Mathematics, pp 331–363
40. Vardi MY (1985) Automatic verification of probabilistic concurrent finite-state programs. In: Proceedings of 26th IEEE symposium on foundations of computer science, pp 327–338
41. Vardi MY, Wolper P (1994) Reasoning about infinite computations. Inf Comput 115(1):1–37