# On Relative and Probabilistic Finite Counterability

CSL 2015

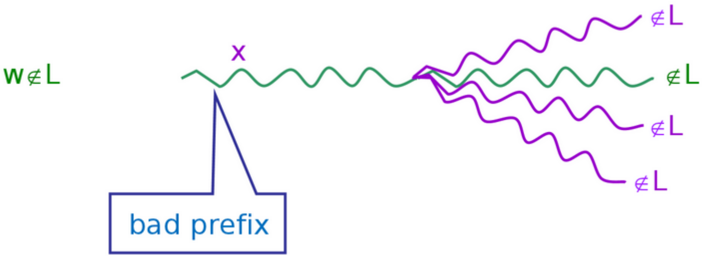
Orna Kupferman and Gal Vardi

Hebrew University

September 7, 2015

# Basic definitions

- A finite word $x \in \Sigma^*$ is a *bad-prefix* for a language $L \subseteq \Sigma^\omega$ if for all infinite words $y \in \Sigma^\omega$, the concatenation $x \cdot y$ is not in $L$.
- The language $L$ is *safety* if every word not in $L$ has a bad-prefix [AS85].




- The language $L$ is *liveness* if it has no bad-prefixes [AS85].

# Examples

- Let $\Sigma = \{a, b\}$.
- $L = \{a^\omega\}$ is safety
  - Every word not in $L$ has a bad-prefix – one that contains the letter $b$
- $L = (a + b)^* \cdot a^\omega$ is liveness
  - By concatenating $a^\omega$ to every word in $\Sigma^*$, we end up with a word in the language

# Counterexamples in model-checking

- An important advantage of model-checking tools is their ability to accompany a negative answer by a *counterexample*.
- *lasso-shaped* counterexample: $uv^\omega$, for finite computations $u$ and $v$.

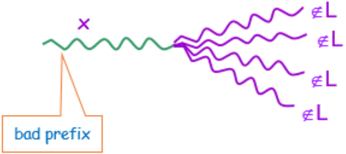
# Counterexamples in model-checking

- An important advantage of model-checking tools is their ability to accompany a negative answer by a *counterexample*.
- *lasso-shaped* counterexample: $uv^\omega$, for finite computations $u$ and $v$.


- The simpler the counterexample is, the more helpful it is for the user.
- Efforts for designing algorithms that return short counterexamples [SB05, KS06].

# Counterexamples in model-checking

- The analysis of counterexamples makes safety properties appealing: rather than a lasso-shaped counterexample, it is possible to return to the user a bad-prefix.





- This is simpler and points 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.

# Example 1

- Consider a system $\mathcal{S}$ and a specification $\psi = G(req \to Fres)$
- Note that $\psi$ is not safety and it does not have bad-prefixes.
- There might be some input sequence that leads $\mathcal{S}$ to an error state in which it stops sending responses.
- Thus, there is a computation prefix that is "bad with respect to $\mathcal{S}$": all its extensions in $\mathcal{S}$ do not satisfy $\psi$.
- Returning this prefix to the user is more helpful than returning a lasso-shaped infinite counterexample.

# Example 2

- $\varphi = FG\neg allocate$
  The system eventually stops allocating memory.
- There might be some input sequence that leads $\mathcal{S}$ to an error state in which every request is followed by a memory allocation.
- A computation that reaches this state almost-surely violates the specification.
- 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.

# Example 2

- $\varphi = FG\neg allocate$
  The system eventually stops allocating memory.
- There might be some input sequence that leads $\mathcal{S}$ to an error state in which every request is followed by a memory allocation.
- A computation that reaches this state almost-surely violates the specification.
- 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, there is a prefix that is "bad with respect to $\mathcal{S}$ in a probabilistic sense": almost all of its extensions in $\mathcal{S}$ do not satisfy $\varphi$.
- We want to return this prefix to the user.

# Counterability

- We say that a language $L$ is *counterable* if it has a bad-prefix.
- That is, $L$ is counterable iff it is not liveness.
- A language may be counterable and not safety:
  - For example, $a^* \cdot b \cdot (a + b + c)^\omega$:

# Counterability

- We say that a language $L$ is *counterable* if it has a bad-prefix.
- That is, $L$ is counterable iff it is not liveness.
- A language may be counterable and not safety:
  - For example, $a^* \cdot b \cdot (a + b + c)^\omega$:
    - $c$ is a bad-prefix

# Counterability

- We say that a language $L$ is *counterable* if it has a bad-prefix.
- That is, $L$ is counterable iff it is not liveness.
- A language may be counterable and not safety:
  - For example, $a^* \cdot b \cdot (a + b + c)^\omega$:
    - $c$ is a bad-prefix
    - $a^\omega$ has no bad-prefixes

# Counterability

- We say that a language $L$ is *counterable* if it has a bad-prefix.
- That is, $L$ is counterable iff it is not liveness.
- A language may be counterable and not safety:
  - For example, $a^* \cdot b \cdot (a + b + c)^\omega$:
    - $c$ is a bad-prefix
    - $a^\omega$ has no bad-prefixes
- Three natural problems arise:
  1. Given a language, decide whether it is counterable.
  2. Study the length of minimal bad-prefixes for counterable languages.
  3. Develop algorithms for detecting bad-prefixes for counterable languages.

# Deciding counterability

- Deciding whether a given language is safety: PSPACE-complete for both LTL formulas and nondeterministic Büchi word automata (NBWs) [Sis94].
- Deciding whether an NBW is liveness: PSPACE-complete.

# Deciding counterability

- Deciding whether a given language is safety: PSPACE-complete for both LTL formulas and nondeterministic Büchi word automata (NBWs) [Sis94].
- Deciding whether an NBW is liveness: PSPACE-complete.
- Deciding whether an LTL formula is liveness:
  - PSPACE-hard
  - In EXPSPACE [Sis94]
  - In PSPACE? [NW97] – wrong upper bound
- The problem was declared open in [BJKZ14, Lip14].

# Deciding counterability

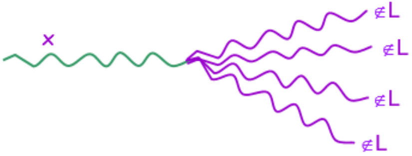
- Deciding whether a given language is safety: PSPACE-complete for both LTL formulas and nondeterministic Büchi word automata (NBWs) [Sis94].
- Deciding whether an NBW is liveness: PSPACE-complete.
- Deciding whether an LTL formula is liveness:
  - PSPACE-hard
  - In EXPSPACE [Sis94]
  - In PSPACE? [NW97] – wrong upper bound
- The problem was declared open in [BJKZ14, Lip14].
- We show: EXPSPACE-hard.
- Therefore, for LTL, deciding liveness is exponentially more complex than deciding safety.

# Deciding counterability

- Deciding whether a given language is safety: PSPACE-complete for both LTL formulas and nondeterministic Büchi word automata (NBWs) [Sis94].
- Deciding whether an NBW is liveness: PSPACE-complete.
- Deciding whether an LTL formula is liveness:
  - PSPACE-hard
  - In EXPSPACE [Sis94]
  - In PSPACE? [NW97] – wrong upper bound
- The problem was declared open in [BJKZ14, Lip14].
- We show: EXPSPACE-hard.
- Therefore, for LTL, deciding liveness is exponentially more complex than deciding safety.
- Independent EXPSPACE lower bound was found by Diekert, Muscholl and Walukiewicz [DMW15] - to be published soon.

# Length and the detection of bad-prefixes

- The length of a shortest bad-prefix for an NBW is tightly exponential and it can be found in PSPACE or in EXPTIME.
- The length of a shortest bad-prefix for an LTL formula is tightly doubly-exponential. A bad-prefix can be found in EXPSPACE or in 2EXPTIME.
- The length of a shortest bad-prefix for a safety LTL formula is tightly exponential and it can be found in PSPACE or in EXPTIME.

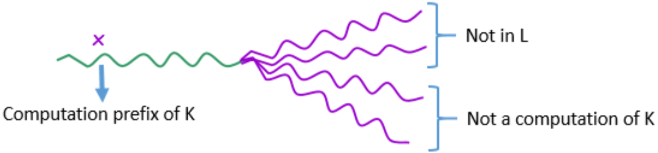


# Finite counterexamples

- Our primary interest – finding finite counterexamples.
- Recall Example 1: A system $\mathcal{S}$ and a specification $\psi = G(req \rightarrow Fres)$.
- The specification $\psi$ is not counterable.
- However, there might be some input sequence that leads $\mathcal{S}$ to an error state in which it stops sending responses.
- Therefore, there is a computation prefix that is "bad with respect to $\mathcal{S}$".
- This prefix can be used as a finite counterexample.

# $K$-bad-prefixes

- A finite computation $x \in (2^{AP})^*$ of a Kripke structure $K$ is a *$K$-bad-prefix* for a language $L \subseteq (2^{AP})^\omega$, if $x$ cannot be extended to an infinite computation of $K$ that is in $L$.
- We wish to return to the user a $K$-bad-prefix as a finite counterexample.



# $K$-counterability

- Consider a language $L$ and a Kripke structure $K$.
- The language $L$ is *$K$-counterable* if it has a $K$-bad-prefix.

# $K$-counterability

- Consider a language $L$ and a Kripke structure $K$.
- The language $L$ is *$K$-counterable* if it has a $K$-bad-prefix.
- Three natural problems arise:
  1. Decide whether a language is $K$-counterable.
  2. Study the length of minimal $K$-bad-prefixes for $K$-counterable languages.
  3. Develop algorithms for detecting $K$-bad-prefixes for $K$-counterable languages.

# $K$-counterability problems

- The solutions for the three problems in the non-relative setting apply also to the relative one, with an additional NLOGSPACE or linear-time dependency in $|K|$:

# $K$-counterability problems

- The solutions for the three problems in the non-relative setting apply also to the relative one, with an additional NLOGSPACE or linear-time dependency in $|K|$:
  - Deciding $K$-counterability and finding a shortest $K$-bad-prefix:
    - PSPACE-complete for NBW
    - EXPSPACE-complete for LTL
    - In both cases: time linear or space polylogarithmic in $|K|$

# $K$-counterability problems

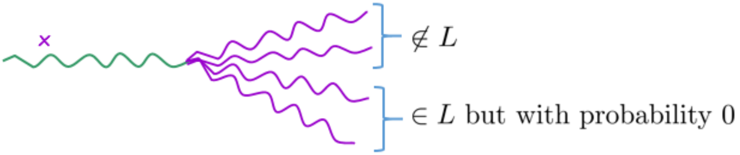
- The solutions for the three problems in the non-relative setting apply also to the relative one, with an additional NLOGSPACE or linear-time dependency in $|K|$:
  - Deciding $K$-counterability and finding a shortest $K$-bad-prefix:
    - PSPACE-complete for NBW
    - EXPSPACE-complete for LTL
    - In both cases: time linear or space polylogarithmic in $|K|$
  - The length of a shortest $K$-bad-prefix:
    - Tightly exponential for NBW
    - Tightly doubly-exponential for LTL
    - In both cases: tightly linear in $|K|$

# A 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 $\frac{1}{2}$.

# A 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 $\frac{1}{2}$.
- A finite word $x \in \Sigma^*$ is a *prob-bad-prefix* for a language $L \subseteq \Sigma^\omega$ if the probability of an infinite word with prefix $x$ to be in $L$ is 0.
- That is, $Pr(\{y \in \Sigma^\omega : x \cdot y \in L\}) = 0$.
- $L$ is *prob-counterable* if it has a prob-bad-prefix.



# Example of a prob-counterable formula

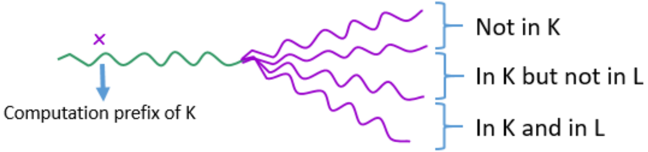
- Counterable $\Rightarrow$ prob-counterable. The other direction does not hold.
- Consider the LTL formula
  $\psi = (req \wedge GF grant) \vee (\neg req \wedge FG \neg grant)$.
- $\psi$ does not have a bad-prefix – it is not counterable.
- All finite computations in which a request is not sent in the beginning are prob-bad-prefixes for $\psi$ – it is prob-counterable.

# Finite counterexamples - a probabilistic view

- Recall Example 2: A system $\mathcal{S}$ and a specification $\varphi$ stating that the system eventually stops allocating memory.
- There might be some input sequence that leads $\mathcal{S}$ to a state in which every request is followed by a memory allocation.
- A computation that reaches this state almost-surely violates the specification.
- Thus, there is a computation prefix that is "bad with respect to $\mathcal{S}$ in a probabilistic sense" : almost all of its extensions in $\mathcal{S}$ do not satisfy $\varphi$.
- This prefix can be used as a finite counterexample.

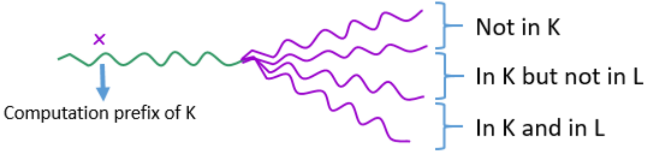
# $K$-prob-counterability

- A finite computation $x \in (2^{AP})^*$ of a Kripke structure $K$ is a *$K$-prob-bad-prefix* for a language $L \subseteq (2^{AP})^\omega$ if a computation of $K$ obtained by continuing $x$ with some random walk on $K$, is almost surely not in $L$.
- The definition is independent of the probabilities of the transitions in the random walk on $K$.



# $K$-prob-counterability

- A finite computation $x \in (2^{AP})^*$ of a Kripke structure $K$ is a *$K$-prob-bad-prefix* for a language $L \subseteq (2^{AP})^\omega$ if a computation of $K$ obtained by continuing $x$ with some random walk on $K$, is almost surely not in $L$.
- The definition is independent of the probabilities of the transitions in the random walk on $K$.




- $L$ is *$K$-prob-counterable* if it has a $K$-prob-bad-prefix.
- $K$-counterable $\Rightarrow$ $K$-prob-counterable. The other direction does not hold.

# Probabilistic counterability advantages

- The probabilistic setting increases our chances to return finite counterexamples.
- It also makes the solution of our three basic problems exponentially easier for LTL formulas:
  - Deciding prob-counterability and $K$-prob-counterability and finding the prefixes are exponentially easier than deciding counterability and $K$-counterability.
  - The length of the prefixes is exponentially smaller.

# Probabilistic counterability basic problems

- Formally (we show here only the relative variant):

# Probabilistic counterability basic problems

- Formally (we show here only the relative variant):
- Deciding $K$-prob-counterability and finding a $K$-prob-bad-prefix:
  - PSPACE-complete for NBW
  - PSPACE-complete for LTL
  - In both cases: time linear or space polylogarithmic in $|K|$

# Probabilistic counterability basic problems

- Formally (we show here only the relative variant):
- Deciding $K$-prob-counterability and finding a $K$-prob-bad-prefix:
  - PSPACE-complete for NBW
  - PSPACE-complete for LTL
  - In both cases: time linear or space polylogarithmic in $|K|$
- The length of a shortest $K$-prob-bad-prefix:
  - Tightly exponential for NBW
  - Tightly exponential for LTL
  - In both cases: tightly linear in $|K|$

Questions?

# References I


- B. Alpern and F.B. Schneider, *Defining liveness*, Information Processing Letters **21** (1985), 181–185.
- D.A. Basin, C.C. Jiménez, F. Klaedtke, and E. Zalinescu, *Deciding safety and liveness in TPTL*, Information Processing Letters **114** (2014), no. 12, 680–688.
- V. Diekert, A. Muscholl, and I. Walukiewicz, *A note on monitors and büchi automata*, *Proc. 12th ICTAC, to appear, 2015.*
- *O. Kupferman and S. Sheinvald-Faragy, Finding shortest witnesses to the nonemptiness of automata on infinite words, Proc. 17th Int. Conf. on Concurrency Theory, Lecture Notes in Computer Science, vol. 4137, Springer, 2006, pp. 492–508.*
- *Marcel Lippmann, Temporalised description logics for monitoring partially observable events.*

# References II


- *U. Nitsche and P. Wolper, Relative liveness and behavior abstraction, Proc. 24th ACM Symp. on Principles of Programming Languages, ACM, 1997, pp. 45–52.*
- *V. Schuppan and A. Biere, Shortest counterexamples for symbolic model checking of LTL with past, Proc. 11th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science, vol. 3440, Springer, 2005, pp. 493–509.*
- *A.P. Sistla, Safety, liveness and fairness in temporal logic, Formal Aspects of Computing* **6** *(1994), 495–511.*