The Complexity of Counting Models of Linear-time Temporal Logic

Joint work with Hazem Torfah (Saarland University)

Martin Zimmermann

Saarland University

January 22nd, 2015
Research Training Group AlgoSyn
RWTH Aachen University

Why Model Counting

How many models does a boolean formula φ have?

Why Model Counting

How many models does a boolean formula φ have?

- Generalization of satisfiability: does φ have a model?
- Applications:
  - probabilistic inference problems
  - planning problems
  - combinatorial designs
  - etc.

Why LTL Model Counting

LTL model counting comes in two flavors:

Why LTL Model Counting

LTL model counting comes in two flavors: for fixed φ and $k \in \mathbb{N}$..

Why LTL Model Counting

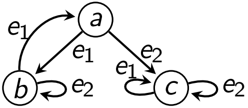
LTL model counting comes in two flavors: for fixed φ and $k \in \mathbb{N}$..

- .. count (ultimately periodic) word models $u \cdot v^\omega$ with $|u| + |v| = k$:
  - Analogue to model checking: count the number of error traces of a given system.

Why LTL Model Counting

LTL model counting comes in two flavors: for fixed φ and $k \in \mathbb{N}$..

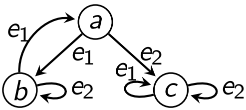
- .. count (ultimately periodic) word models $u \cdot v^\omega$ with $|u| + |v| = k$:
  - Analogue to model checking: count the number of error traces of a given system.
- .. count tree models of depth k with back-edges at leaves:
  - Analogue to synthesis: count the number of implementations (implementation freedom).



Why LTL Model Counting

LTL model counting comes in two flavors: for fixed φ and $k \in \mathbb{N}$..

- .. count (ultimately periodic) word models $u \cdot v^\omega$ with $|u| + |v| = k$:
  - Analogue to model checking: count the number of error traces of a given system.
- .. count tree models of depth k with back-edges at leaves:
  - Analogue to synthesis: count the number of implementations (implementation freedom).



Theorem (Finkbeiner and Torfah '14)

1. *Word models can be counted in time* $\mathcal{O}(k \cdot 2^{2^{|\varphi|}})$.
2. *Tree models can be counted in time* $\mathcal{O}(k \cdot 2^{2^{2^{|\varphi|}}})$.

Outline

1. **Counting Complexity**
2. Counting Word Models
3. Counting Tree Models
4. Conclusion

Counting Complexity

- $f\colon \Sigma^* \to \mathbb{N}$

Counting Complexity

- $f : \Sigma^* \to \mathbb{N}$ is in $\#\mathrm{P}$ if there is an NP machine $\mathcal{M}$ such that $f(w)$ is equal to the number of accepting runs of $\mathcal{M}$ on w.

Counting Complexity

- $f : \Sigma^* \to \mathbb{N}$ is in $\#\mathrm{P}$ if there is an NP machine $\mathcal{M}$ such that $f(w)$ is equal to the number of accepting runs of $\mathcal{M}$ on w.

Examples:

- #SAT is in $\#\mathrm{P}$.
- #CLIQUE is in $\#\mathrm{P}$.

Counting Complexity

- $f : \Sigma^* \rightarrow \mathbb{N}$ is in $\#\mathrm{P}$ if there is an NP machine $\mathcal{M}$ such that $f(w)$ is equal to the number of accepting runs of $\mathcal{M}$ on w.

Examples:

- $\#$SAT is in $\#\mathrm{P}$.
- $\#$CLIQUE is in $\#\mathrm{P}$.

(Parsimonious) Reductions:

- f $\#\mathrm{P}$-hard: for all $f' \in \#\mathrm{P}$ there is a polynomial time computable function r such that $f'(x) = f(r(x))$ for all inputs x.

Counting Complexity

- $f : \Sigma^* \to \mathbb{N}$ is in $\#\mathrm{P}$ if there is an NP machine $\mathcal{M}$ such that $f(w)$ is equal to the number of accepting runs of $\mathcal{M}$ on w.

Examples:

- $\#$SAT is in $\#\mathrm{P}$.
- $\#$CLIQUE is in $\#\mathrm{P}$.

(Parsimonious) Reductions:

- f $\#\mathrm{P}$-hard: for all $f' \in \#\mathrm{P}$ there is a polynomial time computable function r such that $f'(x) = f(r(x))$ for all inputs x.
- If f' is *computed* by $\mathcal{M}$, then r may depend on $\mathcal{M}$ and its time-bound $p(n)$.

Counting Complexity

- $f : \Sigma^* \to \mathbb{N}$ is in $\#\mathrm{P}$ if there is an NP machine $\mathcal{M}$ such that $f(w)$ is equal to the number of accepting runs of $\mathcal{M}$ on w.

Examples:

- $\#$SAT is in $\#\mathrm{P}$.
- $\#$CLIQUE is in $\#\mathrm{P}$.

(Parsimonious) Reductions:

- f $\#\mathrm{P}$-hard: for all $f' \in \#\mathrm{P}$ there is a polynomial time computable function r such that $f'(x) = f(r(x))$ for all inputs x.
- If f' is *computed* by $\mathcal{M}$, then r may depend on $\mathcal{M}$ and its time-bound $p(n)$.
- Completeness: hardness and membership.

Counting Complexity

- $\#$SAT is $\#\mathrm{P}$-complete.
- $\#$CLIQUE is $\#\mathrm{P}$-complete.

Counting Complexity

- $\#$SAT is $\#\mathrm{P}$-complete.
- $\#$CLIQUE is $\#\mathrm{P}$-complete.
- $\#$2SAT is $\#\mathrm{P}$-complete.
- $\#$DNF-SAT is $\#\mathrm{P}$-complete.
- $\#$PERFECT-MATCHING is $\#\mathrm{P}$-complete.

Note:
Decision problems 2SAT, DNF-SAT, and PERFECT-MATCHING are in P:

Counting Complexity

- $\#$SAT is $\#\mathrm{P}$-complete.
- $\#$CLIQUE is $\#\mathrm{P}$-complete.
- $\#$2SAT is $\#\mathrm{P}$-complete.
- $\#$DNF-SAT is $\#\mathrm{P}$-complete.
- $\#$PERFECT-MATCHING is $\#\mathrm{P}$-complete.

Note:
Decision problems 2SAT, DNF-SAT, and PERFECT-MATCHING are in P:

Counting versions of easy problems can be hard!

Beyond $\#\mathrm{P}$

Remark: $f \in \#\mathrm{P}$ implies $f(w) \in \mathcal{O}(2^{p(|w|)})$ for some polynomial p.

Beyond $\#\mathrm{P}$

Remark: $f \in \#\mathrm{P}$ implies $f(w) \in \mathcal{O}(2^{p(|w|)})$ for some polynomial p.

We need *larger* counting classes.

- $f : \Sigma^* \to \mathbb{N}$ is in $\#\mathrm{PSPACE}$, if there is a nondeterministic polynomial-space Turing machine $\mathcal{M}$ such that $f(w)$ is equal to the number of accepting runs of $\mathcal{M}$ on w.

Beyond #P

Remark: $f \in \#\mathrm{P}$ implies $f(w) \in \mathcal{O}(2^{p(|w|)})$ for some polynomial p.

We need *larger* counting classes.

- $f\colon \Sigma^* \to \mathbb{N}$ is in $\#\textsc{Pspace}$, if there is a nondeterministic polynomial-space Turing machine $\mathcal{M}$ such that $f(w)$ is equal to the number of accepting runs of $\mathcal{M}$ on w.
- Analogously: $\#\textsc{Exptime}$, $\#\textsc{Expspace}$, and $\#2\textsc{Exptime}$.

Beyond $\#\mathrm{P}$

Remark: $f \in \#\mathrm{P}$ implies $f(w) \in \mathcal{O}(2^{p(|w|)})$ for some polynomial p.

We need *larger* counting classes.

- $f : \Sigma^* \to \mathbb{N}$ is in $\#\textsc{Pspace}$, if there is a nondeterministic polynomial-space Turing machine $\mathcal{M}$ such that $f(w)$ is equal to the number of accepting runs of $\mathcal{M}$ on w.
- Analogously: $\#\textsc{Exptime}$, $\#\textsc{Expspace}$, and $\#2\textsc{Exptime}$.

Remark:

- $f \in \#\textsc{Exptime}$ implies $f(w) \in \mathcal{O}(2^{2^{p(|w|)}})$ for a polynomial p.
- $f \in \#2\textsc{Exptime}$ implies $f(w) \in \mathcal{O}(2^{2^{2^{p(|w|)}}})$ for a polynomial p.
- $w \mapsto 2^{2^{|w|}}$ is in $\#\textsc{Pspace}$.
- $w \mapsto 2^{2^{2^{|w|}}}$ is in $\#\textsc{Expspace}$.

Outline

Counting Word-Models for Binary Bounds

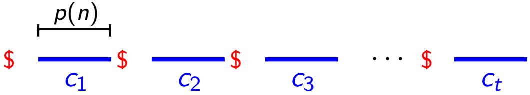
Theorem

The following problem is $\#\textsc{Pspace}$*-complete: Given an LTL formula* φ *and a bound k (in binary), how many k-word-models does* φ *have?*

Counting Word-Models for Binary Bounds

Theorem
The following problem is #PSPACE-complete: Given an LTL formula φ and a bound k (in binary), how many k-word-models does φ have?

- **Lower bound:** PSPACE-hardness of LTL satisfiability **[Sistla & Clarke '85]** made parsimonious.



Counting Word-Models for Binary Bounds

Theorem

The following problem is #PSPACE-complete: Given an LTL formula φ and a bound k (in binary), how many k-word-models does φ have?

- **Lower bound:** PSPACE-hardness of LTL satisfiability **[Sistla & Clarke '85]** made parsimonious.

Counting Word-Models for Binary Bounds

Theorem

The following problem is #PSPACE-complete: Given an LTL formula φ and a bound k (in binary), how many k-word-models does φ have?

- **Lower bound:** PSPACE-hardness of LTL satisfiability **[Sistla & Clarke '85]** made parsimonious.

Length of prefix is exponential, but k can be encoded in binary.

Counting Word-Models for Binary Bounds

Theorem

The following problem is $\#\textsc{Pspace}$*-complete: Given an LTL formula* φ *and a bound* k *(in binary), how many* k*-word-models does* φ *have?*

- **Lower bound:** $\textsc{Pspace}$-hardness of LTL satisfiability **[Sistla & Clarke '85]** made parsimonious.

  Length of prefix is exponential, but k can be encoded in binary.
- **Upper bound:** guess word of length k letter-by-letter (starting at the end) and model-check it on the fly (using unambiguous non-determinism). Then: one accepting run per model.

Counting Word-Models

Theorem

The following problem is $\#\mathrm{P}$*-complete: Given an LTL formula* φ *and a bound* k *(in unary), how many* k*-word-models does* φ *have?*

Counting Word-Models

Theorem

The following problem is $\#\mathrm{P}$-complete: Given an LTL formula φ and a bound k (in unary), how many k-word-models does φ have?

- **Lower bound:** Same as before, but we have to encode k in unary. Thus, k has to be polynomial.

Counting Word-Models

Theorem

The following problem is $\#\mathrm{P}$-complete: Given an LTL formula φ and a bound k (in unary), how many k-word-models does φ have?

- **Lower bound:** Same as before, but we have to encode k in unary. Thus, k has to be polynomial.
- **Upper bound:** Guess word of length k and model-check it.

Outline

Counting Tree-Models with Unary Bounds

Theorem

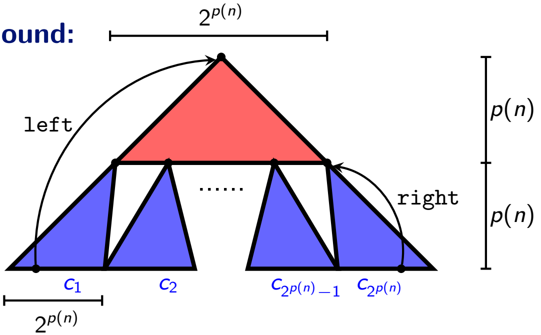
The following problem is #EXPTIME*-complete: Given an LTL formula* φ *and a bound* k *(in unary), how many* k*-tree-models does* φ *have?*

Counting Tree-Models with Unary Bounds

Theorem

*The following problem is #*Exptime*-complete: Given an LTL formula φ and a bound k (in unary), how many k-tree-models does φ have?*

- **Lower bound:**

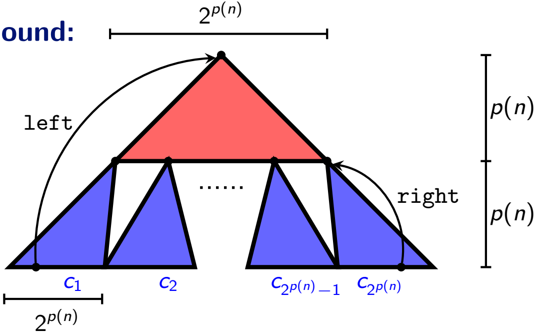


Counting Tree-Models with Unary Bounds

Theorem

The following problem is #EXPTIME-complete: Given an LTL formula φ and a bound k (in unary), how many k-tree-models does φ have?

- **Lower bound:**



- **Upper bound:** Guess tree of height k and model-check it.

Counting Tree-Models with Binary Bounds

Theorem

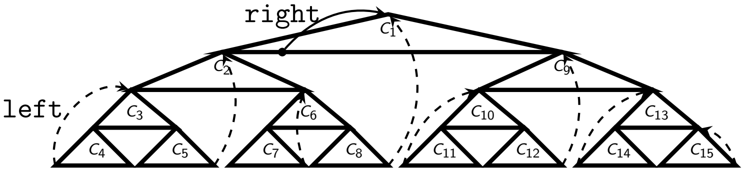
The following problem is $\#\textsc{Expspace}$*-hard and in* $\#2\textsc{Exptime}$*: Given an LTL formula* φ *and a bound* k *(in binary), how many* k*-tree-models does* φ *have?*

Counting Tree-Models with Binary Bounds

Theorem

*The following problem is #*Expspace*-hard and in #2*Exptime*: Given an LTL formula* φ *and a bound k (in binary), how many k-tree-models does* φ *have?*

- **Lower bound:**



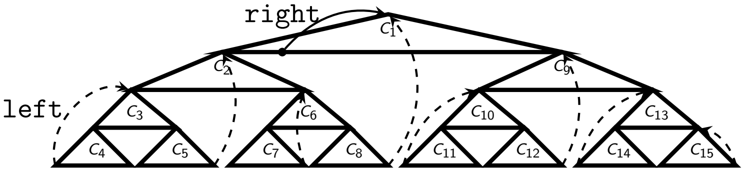
  - each inner tree has exponentially many leaves.
  - tree has exponential height (thus, doubly-exponentially many inner trees).

Counting Tree-Models with Binary Bounds

Theorem

*The following problem is #*EXPSPACE*-hard and in #2*EXPTIME*:*
Given an LTL formula φ and a bound k (in binary), how many k-tree-models does φ have?

- **Lower bound:**



  - each inner tree has exponentially many leaves.
  - tree has exponential height (thus, doubly-exponentially many inner trees).

- **Upper bound:** Guess tree of height k and model-check it.

Outline

1. Counting Complexity
2. Counting Word Models
3. Counting Tree Models
4. **Conclusion**

Conclusion

Overview of results:

	unary	binary
words	#P-compl.	#PSPACE-compl.
trees	#EXPTIME-compl.	#EXPSPACE-hard/#2EXPTIME

Conclusion

Overview of results:

	unary	binary
words	#P-compl.	#PSPACE-compl.
trees	#EXPTIME-compl.	#EXPSPACE-hard/#2EXPTIME

Lower bounds: safety LTL, upper bounds: full LTL

Conclusion

Overview of results:

	unary	binary
words	#P-compl.	#PSPACE-compl.
trees	#EXPTIME-compl.	#EXPSPACE-hard/#2EXPTIME

Lower bounds: safety LTL, upper bounds: full LTL

Open problems:

- Close the gap!

Conclusion

Overview of results:

	unary	binary
words	#P-compl.	#PSPACE-compl.
trees	#EXPTIME-compl.	#EXPSPACE-hard/#2EXPTIME

Lower bounds: safety LTL, upper bounds: full LTL

Open problems:

- Close the gap!
  - Lowering the upper bound: how to guess and model-check doubly-exponentially sized trees in exponential space?

Conclusion

Overview of results:

	unary	binary
words	#P-compl.	#PSPACE-compl.
trees	#EXPTIME-compl.	#EXPSPACE-hard/#2EXPTIME

Lower bounds: safety LTL, upper bounds: full LTL

Open problems:

- Close the gap!
  - Lowering the upper bound: how to guess and model-check doubly-exponentially sized trees in exponential space?
  - Raising the lower bound: how to encode doubly-exponentially sized configurations using polynomially sized formulas? Do games help?

Conclusion

Overview of results:

	unary	binary
words	#P-compl.	#PSPACE-compl.
trees	#EXPTIME-compl.	#EXPSPACE-hard/#2EXPTIME
graphs	#P-hard/$\#_o$PSPACE.	#EXPTIME-compl.

Lower bounds: safety LTL, upper bounds: full LTL

Open problems:

- Close the gap!
  - Lowering the upper bound: how to guess and model-check doubly-exponentially sized trees in exponential space?
  - Raising the lower bound: how to encode doubly-exponentially sized configurations using polynomially sized formulas? Do games help?

Conclusion

Overview of results:

	unary	binary
words	#P-compl.	#PSPACE-compl.
trees	#EXPTIME-compl.	#EXPSPACE-hard/#2EXPTIME
graphs	#P-hard/$\#_o$PSPACE.	#EXPTIME-compl.

Lower bounds: safety LTL, upper bounds: full LTL

Open problems:

- Close the gap!
  - Lowering the upper bound: how to guess and model-check doubly-exponentially sized trees in exponential space?
  - Raising the lower bound: how to encode doubly-exponentially sized configurations using polynomially sized formulas? Do games help?
- Close the gap for graph models, too.