The Complexity of Counting Models of Linear-time Temporal Logic

Joint work with Hazem Torfah

Martin Zimmermann

Saarland University

September 4th, 2014

Highlights 2014, Paris, France
Counting Complexity

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

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

We need larger counting classes.

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

Analogously: $\#dExptime$, $\#dExpspace$, and $\#d2Exptime$. 

Counting Complexity

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

For complexity class $C$:

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

Remark: $f \in \#C$ implies $f(w) \in O(2^{p(|w|)})$ for some polynomial $p$. We need larger counting classes.

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

Analogously: $\#dExptime$, $\#dExpspace$, and $\#d2Exptime$. 

Counting Complexity

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

For complexity class \( C \):

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

**Remark:** \( f \in \#C \) implies \( f(w) \in O(2^p(|w|)) \) for some polynomial \( p \).
Counting Complexity

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

For complexity class \( C \):

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

**Remark:** \( f \in \#C \) implies \( f(w) \in O(2^p(|w|)) \) for some polynomial \( p \).

We need *larger* counting classes.

- \( f : \Sigma^* \rightarrow \mathbb{N} \) is in \( \#dPSPACE \), 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 \( w \).
Counting Complexity

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

For complexity class \(C\):

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

**Remark:** \(f \in #C\) implies \(f(w) \in O(2^{p(|w|)})\) for some polynomial \(p\).

We need *larger* counting classes.

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

Analogously: \(#dEXPTIME\), \(#dEXPSPACE\), and \(#d2EXPTIME\).
Lemma

\#P
Lemma

$\#P \subseteq \#PSPACE$
Lemma

\[ \#P \subseteq \#P_{\text{SPACE}} \subseteq \#\text{EXPTIME} \]
Lemma

\[ \#P \subseteq \#PSPACE \subseteq \#EXPSPACE \subseteq \#NEXPSPACE \]
Lemma

\[ \#P \subseteq \#PSPACE \subseteq \#EXPSPACE \subseteq \#\text{NEXPSPACE} \subseteq \#\text{EXPSPACE} \]
Lemma

\[ \#P \subseteq \#PSPACE \subseteq \#EXPTIME \subseteq \#NEXPTIME \subseteq \#EXPSPACE \subseteq \#2EXPTIME \]
Lemma

$\#_d \text{PSPACE}$

$\#P \subseteq \#\text{PSPACE} \subseteq \#\text{EXPTIME} \subseteq \#\text{NEEXPTIME} \subseteq \#\text{EXPSPACE} \subseteq \#\text{2EXPSPACE}$
Lemma

\[ \#_d PSPACE \subseteq \#_d \text{EXPTIME} \]

\[ \#P \subseteq \#PSPACE \subseteq \# \text{EXPTIME} \subseteq \# \text{NEXPTIME} \subseteq \# \text{EXPSPACE} \subseteq \#2 \text{EXPTIME} \]
Lemma

\[#_d PSPACE \subseteq \#_d \text{EXPTIME} \subsetneq \#_d \text{EXPSPACE}\]

\[\#P \subseteq \#PSPACE \subseteq \# \text{EXPTIME} \subseteq \# \text{NEXPTIME} \subseteq \# \text{EXPSPACE} \subseteq \# \text{2EXPTIME}\]
Counting Complexity

Lemma

\[ \#_d \text{Pspace} \subseteq \#_d \text{Exptime} \subset \#_d \text{Expspace} \subseteq \#_d 2\text{Exptime} \]

\[ \# \text{P} \subseteq \# \text{Pspace} \subseteq \# \text{Exptime} \subseteq \# \text{NExptime} \subseteq \# \text{Expspace} \subseteq \# 2\text{Exptime} \]
Lemma

\[
\text{\#PSPACE} \subseteq \text{\#d Exptime} \subseteq \text{\#d Expspace} \subseteq \text{\#d 2Exptime}
\]

\[
\text{\#P} \subseteq \text{\#Pspace} \subseteq \text{\#Exptime} \subseteq \text{\#NExptime} \subseteq \text{\#Expspace} \subseteq \text{\#2Exptime}
\]
Lemma

\[
\#_dPspace \subseteq \#_dExptime \subsetneq \#dExpspace \subseteq \#d2Exptime
\]

\[
\#P \subseteq \#Pspace \subseteq \#Exptime \subseteq \#NExptime \subseteq \#Expspace \subseteq \#2Exptime
\]

Reductions:

- \( f \) is \#P-hard, if there is a polynomial time computable function \( r \) s. t. \( f(r(M, w)) \) is equal to the number of accepting runs of \( M \) on \( w \).
Lemma

$$\text{#}_d \text{Pspace} \subseteq \text{#}_d \text{Exptime} \subset \text{#}_d \text{Expspace} \subseteq \text{#}_d \text{2Exptime}$$

$$\cup \downarrow \quad \cup \downarrow \quad \cup \downarrow \quad \cup \downarrow$$

$$\text{#P} \subseteq \# \text{Pspace} \subseteq \# \text{Exptime} \subseteq \# \text{NExptime} \subseteq \# \text{Expspace} \subseteq \# \text{2Exptime}$$

Reductions:

- $f$ is #P-hard, if there is a polynomial time computable function $r$ s. t. $f(r(M, w))$ is equal to the number of accepting runs of $M$ on $w$.
- Hardness for other classes analogously.
- Completeness as usual.
**Theorem**

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

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

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

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

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

Lower bound: PSPACE-hardness of LTL satisfiability [SC85] made one-to-one
Counting Word-Models

Theorem

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

**Lower bound:** \( PSPACE \text{-hardness of LTL satisfiability [SC85]} \) made one-to-one

**Upper bound:** Guess word of length \( k \) and model-check it
Theorem
The following problem is $\#_d \mathsf{EXPTIME}$-complete: Given an LTL formula $\varphi$ and a bound $k$ (in unary), how many $k$-tree-models does $\varphi$ have?
Theorem

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

- **Lower bound:**

   \[
   \begin{align*}
   c_1 & \quad c_2 \\
   2^p(n) & \quad 2^p(n) \\
   \text{left} & \quad \text{right}
   \end{align*}
   \]

   \[
   \begin{align*}
   c_{2p(n) - 1} & \quad c_{2p(n)} \\
   p(n) & \quad p(n)
   \end{align*}
   \]
Theorem
The following problem is \( \#_d \text{EXPTIME}\)-complete: Given an LTL formula \( \varphi \) and a bound \( k \) (in unary), how many \( k \)-tree-models does \( \varphi \) have?

- **Lower bound:**

- **Upper bound:** Guess tree of height \( k \) and model-check it.
Theorem
The following problem is $\#d\text{Expspace}$-hard and in $\#d2\text{Exptime}$: Given an LTL formula $\varphi$ and a bound $k$ (in binary), how many $k$-tree-models does $\varphi$ have?
Theorem
The following problem is \#dEXPSPACE-hard and in \#d2EXPTIME: Given an LTL formula \( \varphi \) and a bound \( k \) (in binary), how many \( k \)-tree-models does \( \varphi \) 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 $\#_d \text{EXPSPACE}-hard$ and in $\#_d 2 \text{EXPTIME}$: Given an LTL formula $\varphi$ and a bound $k$ (in binary), how many $k$-tree-models does $\varphi$ 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
## Conclusion

### Overview of results:

<table>
<thead>
<tr>
<th></th>
<th>unary</th>
<th>binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>(#P)-compl.</td>
<td>(#dPSPACE)-compl.</td>
</tr>
<tr>
<td>trees</td>
<td>(#dEXPTIME)-compl.</td>
<td>(#dEXPSPACE)-hard/(#d2EXPTIME)</td>
</tr>
</tbody>
</table>
Conclusion

Overview of results:

<table>
<thead>
<tr>
<th></th>
<th>unary</th>
<th>binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>#P-compl.</td>
<td>(#_d\text{PSPACE-compl.})</td>
</tr>
<tr>
<td>trees</td>
<td>(#_d\text{EXPTIME-compl.})</td>
<td>(#_d\text{EXPSPACE-hard/#}_d\text{2EXPTIME})</td>
</tr>
</tbody>
</table>

Lower bounds: safety LTL, upper bounds: full LTL
## Conclusion

### Overview of results:

<table>
<thead>
<tr>
<th></th>
<th>unary</th>
<th>binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>$#P$-compl.</td>
<td>$#_dPSPACE$-compl.</td>
</tr>
<tr>
<td>trees</td>
<td>$#_dEXPTIME$-compl.</td>
<td>$#_dEXPTIME$-hard/$#_d2EXPTIME$</td>
</tr>
</tbody>
</table>

Lower bounds: safety LTL, upper bounds: full LTL

### Open problems:
- Close the gap!
Conclusion

Overview of results:

<table>
<thead>
<tr>
<th></th>
<th>unary</th>
<th>binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>$#P$-compl.</td>
<td>$#_dPSPACE$-compl.</td>
</tr>
<tr>
<td>trees</td>
<td>$#_dEXPTIME$-compl.</td>
<td>$#_dEXPSPACE$-hard/$#_d2EXPTIME$</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th></th>
<th>unary</th>
<th>binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>$#P$-compl.</td>
<td>$#dPSPACE$-compl.</td>
</tr>
<tr>
<td>trees</td>
<td>$#dEXPTIME$-compl.</td>
<td>$#dEXPSPACE$-hard/$#d2EXPTIME$</td>
</tr>
</tbody>
</table>

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?