Topological Sorting under Regular Constraints

Antoine Amarilli¹, Charles Paperman²
July 12th, 2018

¹Télécom ParisTech
²Université de Lille
Constrained Topological Sorting

- Fix an alphabet: e.g., $\Sigma = \{a, b\}$
Constrained Topological Sorting

- Fix an **alphabet**: e.g., $\Sigma = \{a, b\}$
- Fix a **language**: e.g., $L = (ab)^*$
Constrained Topological Sorting

• Fix an alphabet: e.g., $\Sigma = \{a, b\}$

• Fix a language: e.g., $L = (ab)^*$

• We study constrained topological sorting:
Constrained Topological Sorting

• Fix an **alphabet**: e.g., $\Sigma = \{a, b\}$

• Fix a **language**: e.g., $L = (ab)^*$

• We study **constrained topological sorting**:
  
  - **Input**: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
Constrained Topological Sorting

- Fix an alphabet: e.g., \( \Sigma = \{a, b\} \)
- Fix a language: e.g., \( L = (ab)^* \)
- We study constrained topological sorting:
  - **Input:** directed acyclic graph (DAG) with vertices labeled with \( \Sigma \)
  - **Output:** is there a topological sort that falls in \( L \)?
Constrained Topological Sorting

- Fix an **alphabet**: e.g., $\Sigma = \{a, b\}$
- Fix a **language**: e.g., $L = (ab)^*$
- We study **constrained topological sorting**:
  - **Input**: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - **Output**: is there a topological sort that falls in $L$?

![Diagram](https://via.placeholder.com/150)
Constrained Topological Sorting

- Fix an alphabet: e.g., $\Sigma = \{a, b\}$
- Fix a language: e.g., $L = (ab)^*$
- We study constrained topological sorting:
  - Input: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - Output: is there a topological sort that falls in $L$?
• Fix an **alphabet**: e.g., $\Sigma = \{a, b\}$
• Fix a **language**: e.g., $L = (ab)^*$
• We study **constrained topological sorting**:
  - **Input**: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - **Output**: is there a **topological sort** that falls in $L$?
Constrained Topological Sorting

- Fix an alphabet: e.g., $\Sigma = \{a, b\}$
- Fix a language: e.g., $L = (ab)^*$
- We study constrained topological sorting:
  - Input: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - Output: is there a topological sort that falls in $L$?

```
a b a b
```

```
\begin{itemize}
  \item a
  \item b
  \item a
  \item b
\end{itemize}
```
Constrained Topological Sorting

- **Fix an alphabet:** e.g., $\Sigma = \{a, b\}$
- **Fix a language:** e.g., $L = (ab)^*$
- **We study** constrained topological sorting:
  - **Input:** directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - **Output:** is there a topological sort that falls in $L$?
• Fix an alphabet: e.g., $\Sigma = \{a, b\}$

• Fix a language: e.g., $L = (ab)^*$

• We study constrained topological sorting:
  
  • Input: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  
  • Output: is there a topological sort that falls in $L$?
Constrained Topological Sorting

- Fix an **alphabet**: e.g., $\Sigma = \{a, b\}$
- Fix a **language**: e.g., $L = (ab)^*$
- We study **constrained topological sorting**:
  - **Input**: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - **Output**: is there a **topological sort** that falls in $L$?

```
a b a b a b a
```
... not in $L$!
Constrained Topological Sorting

• Fix an **alphabet**: e.g., $\Sigma = \{a, b\}$
• Fix a **language**: e.g., $L = (ab)^*$
• We study **constrained topological sorting**:
  • **Input**: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  • **Output**: is there a **topological sort** that falls in $L$?
Constrained Topological Sorting

- Fix an **alphabet**: e.g., $\Sigma = \{a, b\}$
- Fix a **language**: e.g., $L = (ab)^*$
- We study **constrained topological sorting**:
  - **Input**: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - **Output**: is there a topological sort that falls in $L$?
Constrained Topological Sorting

- **Fix an alphabet**: e.g., $\Sigma = \{a, b\}$
- **Fix a language**: e.g., $L = (ab)^*$
- **We study constrained topological sorting**:
  - **Input**: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - **Output**: is there a topological sort that falls in $L$?
• Fix an **alphabet**: e.g., $\Sigma = \{a, b\}$

• Fix a **language**: e.g., $L = (ab)^*$

• We study **constrained topological sorting**:
  
  • **Input**: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  
  • **Output**: is there a **topological sort** that falls in $L$?
Constrained Topological Sorting

- Fix an alphabet: e.g., $\Sigma = \{a, b\}$
- Fix a language: e.g., $L = (ab)^*$
- We study constrained topological sorting:
  - **Input:** directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - **Output:** is there a topological sort that falls in $L$?
Constrained Topological Sorting

- Fix an **alphabet**: e.g., $\Sigma = \{a, b\}$
- Fix a **language**: e.g., $L = (ab)^*$
- We study **constrained topological sorting**:
  - **Input**: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - **Output**: is there a topological sort that falls in $L$?
Constrained Topological Sorting

• Fix an alphabet: e.g., $\Sigma = \{a, b\}$

• Fix a language: e.g., $L = (ab)^*$

• We study constrained topological sorting:
  
  • Input: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  
  • Output: is there a topological sort that falls in $L$?
Constrained Topological Sorting

- Fix an **alphabet**: e.g., $\Sigma = \{a, b\}$
- Fix a **language**: e.g., $L = (ab)^*$
- We study **constrained topological sorting**:
  - **Input**: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - **Output**: is there a topological sort that falls in $L$?

```
a b a b a b ...
```
... in $L$!
Constrained Topological Sorting

- Fix an **alphabet**: e.g., $\Sigma = \{a, b\}$
- Fix a **language**: e.g., $L = (ab)^*$
- We study **constrained topological sorting**:
  
  - **Input**: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - **Output**: is there a topological sort that falls in $L$?
- **Fix an alphabet**: e.g., $\Sigma = \{a, b\}$
- **Fix a language**: e.g., $L = (ab)^*$
- **We study** constrained topological sorting:
  - **Input**: directed acyclic graph (DAG) with vertices labeled with $\Sigma$
  - **Output**: is there a topological sort that falls in $L$?
- **Question**: when is this problem tractable?
Motivation

• How we really ended up studying this problem:
Motivation

• How we **really ended up** studying this problem:

|------|------|------|------|------|------|------|------|
| Probabilistic XML
| XML versioning |
Motivation

- How we **really ended up** studying this problem:

<table>
<thead>
<tr>
<th>Year</th>
<th>Probabilistic XML</th>
<th>Order-uncertain databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>XML versioning</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td></td>
<td>Top-$k$</td>
</tr>
<tr>
<td>2015</td>
<td></td>
<td>Aggregate queries</td>
</tr>
<tr>
<td>2016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Motivation

• How we **really ended up** studying this problem:


- Probabilistic XML
- Order-uncertain databases
- XML versioning
- Top-$k$
- Aggregate queries
- Possible answers

Possible answers

- Scheduling with constraints!
- Verification for concurrent code!
- Computational biology!
- Blockchain!

(joke)

• But why do we actually care?

→ Natural problem and apparently nothing was known about it!
Motivation

• How we **really ended up** studying this problem:

- Probabilistic XML
- XML versioning
- Order-uncertain databases
- Top-$k$
- Aggregate queries
- Possible answers
- Algebraic language theory?!
Motivation

• How we **really ended up** studying this problem:


<table>
<thead>
<tr>
<th>Probabilistic XML</th>
<th>Order-uncertain databases</th>
<th>DAGs</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML versioning</td>
<td>Top-$k$</td>
<td></td>
</tr>
<tr>
<td>Possible answers</td>
<td>Aggregate queries</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Algebraic language theory?!</td>
<td></td>
</tr>
</tbody>
</table>

• Which **a-posteriori motivation** did we invent for the problem?
Motivation

• How we **really ended up** studying this problem:


<table>
<thead>
<tr>
<th>Probabilistic XML</th>
<th>Order-uncertain databases</th>
<th>DAGs</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML versioning</td>
<td>Top-$k$</td>
<td>Aggregate queries</td>
</tr>
<tr>
<td>Possible answers</td>
<td>Algebraic language theory?!</td>
<td></td>
</tr>
</tbody>
</table>

• Which **a-posteriori motivation** did we invent for the problem?

→ **Scheduling** with constraints! → **Verification for concurrent code**!
Motivation

• How we **really ended up** studying this problem:

- Probabilistic XML
- XML versioning
- Order-uncertain databases
- Top-\(k\)
- Aggregate queries
- Possible answers
- Algebraic language theory?!

• Which **a-posteriori motivation** did we invent for the problem?

→ **Scheduling** with constraints! → Verification for **concurrent code**!
→ **Computational biology**! → **Blockchain**!
• How we **really ended up** studying this problem:

<table>
<thead>
<tr>
<th>Year</th>
<th>2011</th>
<th>2012</th>
<th>2013</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
<th>2017</th>
<th>2018</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Probabilistic XML</td>
<td>Order-uncertain databases</td>
<td>DAGs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XML versioning</td>
<td>Top-$k$</td>
<td>Aggregate queries</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Possible answers</td>
<td>Algebraic language theory?!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Which **a-posteriori motivation** did we invent for the problem?

→ **Scheduling** with constraints!  
→ Verification for **concurrent code**!  
→ **Computational biology**!  
→ **Blockchain**! *(joke)*
Motivation

• How we **really ended up** studying this problem:

  - 2011: Probabilistic XML
  - 2012: XML versioning
  - 2013: Order-uncertain databases
  - 2014: Possible answers
  - 2015: Top-k
  - 2016: Aggregate queries
  - 2017: Algebraic language theory?!?
  - 2018: DAGs

• Which **a-posteriori motivation** did we invent for the problem?
  → **Scheduling** with constraints!
  → Verification for **concurrent code**!
  → **Computational biology**!
  → **Blockchain**! (joke)

• But why do we **actually care**?
Motivation

• How we **really ended up** studying this problem:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilistic XML</td>
<td>Order-uncertain databases</td>
<td>DAGs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML versioning</td>
<td>Top-(k)</td>
<td>Aggregate queries</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Possible answers</td>
<td>Algebraic language theory?!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Which **a-posteriori motivation** did we invent for the problem?

→ **Scheduling** with constraints!   → Verification for **concurrent code**!
→ **Computational biology**!   → **Blockchain**! *(joke)*

• But why do we **actually care**?

→ **Natural** problem and apparently **nothing** was known about it!
Fix a regular language $L$ on an finite alphabet $\Sigma$
Formal problem statement

• Fix a **regular language** $L$ on an **finite alphabet** $\Sigma$

• **Constrained topological sort** problem $\text{CTS}(L)$:
  • **Input:** a DAG $G$ with vertices labeled by letters of $\Sigma$
  • **Output:** is there a **topological sort** of $G$ such that the sequence of vertex labels is a **word** of $L$
Formal problem statement

• Fix a regular language $L$ on an finite alphabet $\Sigma$

• Constrained topological sort problem $\text{CTS}(L)$:
  • Input: a DAG $G$ with vertices labeled by letters of $\Sigma$
  • Output: is there a topological sort of $G$ such that the sequence of vertex labels is a word of $L$

• Special case: the constrained shuffle problem $\text{CSh}(L)$:
  • Input: a set of words $w_1, \ldots, w_n$ of $\Sigma^*$
  • Output: is there a shuffle of $w_1, \ldots, w_n$ which is in $L$
Formal problem statement

• Fix a regular language $L$ on an finite alphabet $\Sigma$

• Constrained topological sort problem $\text{CTS}(L)$:
  • Input: a DAG $G$ with vertices labeled by letters of $\Sigma$
  • Output: is there a topological sort of $G$ such that the sequence of vertex labels is a word of $L$

• Special case: the constrained shuffle problem $\text{CSh}(L)$:
  • Input: a set of words $w_1, \ldots, w_n$ of $\Sigma^*$
  • Output: is there a shuffle of $w_1, \ldots, w_n$ which is in $L$

• This is like CTS but the input DAG is an union of paths
Formal problem statement

• Fix a regular language $L$ on an finite alphabet $\Sigma$

• Constrained topological sort problem $\text{CTS}(L)$:
  • Input: a DAG $G$ with vertices labeled by letters of $\Sigma$
  • Output: is there a topological sort of $G$ such that the sequence of vertex labels is a word of $L$

• Special case: the constrained shuffle problem $\text{CSh}(L)$:
  • Input: a set of words $w_1, \ldots, w_n$ of $\Sigma^*$
  • Output: is there a shuffle of $w_1, \ldots, w_n$ which is in $L$

• This is like $\text{CTS}$ but the input DAG is an union of paths

→ Question: What is the complexity of $\text{CTS}(L)$ and $\text{CSh}(L)$, depending on the fixed language $L$?
Dichotomy

For every regular language $L$, exactly one of the following holds:

- $L$ has [some nice property] and $\text{CTS}(L)$ is in $\text{NL}$
- $L$ has [some nasty property] and $\text{CTS}(L)$ is $\text{NP-hard}$
**Dichotomy Conjecture**

**Conjecture**

For every *regular language* L, exactly one of the following holds:

- L has [some nice property] and \( \text{CTS}(L) \) is **in NL**
- L has [some nasty property] and \( \text{CTS}(L) \) is **NP-hard**

\[\wedge(\ツ)\wedge\]
Dichotomy Conjecture

<table>
<thead>
<tr>
<th>Conjecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>For every <strong>regular language</strong> $L$, exactly one of the following holds:</td>
</tr>
<tr>
<td>• $L$ has [some nice property] and $\text{CTS}(L)$ is <strong>in NL</strong></td>
</tr>
<tr>
<td>• $L$ has [some nasty property] and $\text{CTS}(L)$ is <strong>NP-hard</strong></td>
</tr>
</tbody>
</table>

Here’s what we actually know:

• $\text{CTS}$ and $\text{CSh}$ are **NP-hard** for some languages, including $(ab)^*$
Conjecture

For every regular language $L$, exactly one of the following holds:

- $L$ has [some nice property] and $\text{CTS}(L)$ is in NL
- $L$ has [some nasty property] and $\text{CTS}(L)$ is NP-hard

Here’s what we actually know:

- CTS and CSh are NP-hard for some languages, including $(ab)^*$
- They are in NL for some language families (monomials, groups)
**Conjecture**

For every *regular language* $L$, exactly one of the following holds:

- $L$ has [some nice property] and $\text{CTS}(L)$ is *in NL* 🙄
- $L$ has [some nasty property] and $\text{CTS}(L)$ is *NP-hard* 🙄

Here’s what we actually know:

- $\text{CTS}$ and $\text{CSh}$ are *NP-hard* for some languages, including $(ab)^*$
- They are *in NL* for *some language families* (monomials, groups)
- Some languages are tractable for *seemingly unrelated* reasons
Conjecture

For every regular language $L$, exactly one of the following holds:

- $L$ has [some nice property] and $\text{CTS}(L)$ is in $\text{NL}$
- $L$ has [some nasty property] and $\text{CTS}(L)$ is NP-hard

Here’s what we actually know:

- $\text{CTS}$ and $\text{CSh}$ are NP-hard for some languages, including $(ab)^*$
- They are in $\text{NL}$ for some language families (monomials, groups)
- Some languages are tractable for seemingly unrelated reasons

→ Very mysterious landscape! (to us)
Hardness Results
On the Complexity of Iterated Shuffle

MANFRED K. WARMUTH† AND DAVID HAUSSLER‡

Department of Computer Science,
University of Colorado, Boulder, Colorado 80309

It is demonstrated that the following problems are NP complete:

(1) Given words $w$ and $w_1, w_2, ..., w_n$, is $w$ in the shuffle of $w_1, w_2, ..., w_n$?
... but the target is a word which is provided as input!
Existing Hardness Result

... but the target is a word which is provided as input!

→ Does not directly apply for us, because we fix the target language
Hardness for CTS

- We can reduce their problem to CSh for the language \((aA + bB)^*\).
- To determine if the shuffle of \(aab\) and \(bb\) contains \(ababb\) ...

Conjecture:

If \(F\) is finite, then CTS \((F^*)\) is NP-hard unless it contains a power of each of its letters.
Hardness for CTS

- We can reduce their problem to CSh for the language \((aA + bB)^*\)
- To determine if the shuffle of \(aab\) and \(bb\) contains \(ababb\) ... solve the CSh-problem for \(aab\) and \(bb\) and \(ABABB\)
Hardness for CTS

• We can reduce their problem to **CSh** for the language \((aA + bB)^*\)

• To determine if the shuffle of \(aab\) and \(bb\) contains \(ababb\) ... solve the **CSh**-problem for \(aab\) and \(bb\) and \(ABABB\)

→ **CSh**\(((aA + bB)^*)\) is **NP-hard** and the same holds for CTS
Hardness for CTS

• We can reduce their problem to CSh for the language \((aA + bB)^*\)
• To determine if the shuffle of \(aab\) and \(bb\) contains \(ababb\) ...
solve the CSh-problem for \(aab\) and \(bb\) and \(ABABB\)
→ CSh\(((aA + bB)^*)\) is NP-hard and the same holds for CTS

• Similar technique: CSh\(((ab)^*)\) is NP-hard
Hardness for CTS

- We can reduce their problem to $CSh$ for the language $(aA + bB)^*$
- To determine if the shuffle of $aab$ and $bb$ contains $ababb$ ... solve the $CSh$-problem for $aab$ and $bb$ and $ABABB$
  $\rightarrow CSh((aA + bB)^*)$ is NP-hard and the same holds for CTS

- Similar technique: $CSh((ab)^*)$ is NP-hard

- Custom reduction technique to show NP-hardness for:
  - $(ab + b)^*$
  - $(aa + bb)^*$
  - $u^*$ if $u$ contains two different letters
Hardness for CTS

- We can reduce their problem to $\text{CSh}$ for the language $(aA + bB)^*$.
- To determine if the shuffle of $aab$ and $bb$ contains $ababb$ ... solve the $\text{CSh}$-problem for $aab$ and $bb$ and $ABABB$.
- $\text{CSh}((aA + bB)^*)$ is NP-hard and the same holds for CTS.

- Similar technique: $\text{CSh}((ab)^*)$ is NP-hard.

- Custom reduction technique to show NP-hardness for:
  - $(ab + b)^*$
  - $(aa + bb)^*$
  - $u^*$ if $u$ contains two different letters.

- Conjecture: if $F$ is finite then $\text{CTS}(F^*)$ is NP-hard unless it contains a power of each of its letters.
Hardness for CTS

- We can reduce their problem to \textbf{CSh} for the language \((aA + bB)^*\)
- To determine if the shuffle of \(aab\) and \(bb\) contains \(ababb\) ... solve the \textbf{CSh}-problem for \(aab\) and \(bb\) and \(ABABB\)
  \(\rightarrow \) \textbf{CSh}((\(aA + bB)^*)) \text{ is NP-hard} and the same holds for \text{CTS}

- Similar technique: \textbf{CSh}((\(ab)^*)) \text{ is NP-hard}

- Custom reduction technique to show NP-hardness for:
  - \((ab + b)^*\)
  - \((aa + bb)^*\)
  - \(u^*\) if \(u\) contains two different letters

- Conjecture: if \(F\) is finite then \textbf{CTS}(\(F^*)\) is \text{NP-hard} unless it contains a power of each of its letters
Tractability Results
Tractability for Monomials

- **Monomial:** language of the form $A_1^* a_1 A_2^* a_2 \cdots A_n^* a_n A_{n+1}^*$
  where $a_1, \ldots, a_n \in \Sigma$ and $A_1, \ldots, A_{n+1} \subseteq \Sigma$

- **Union of monomials:** union of finitely many such languages
Tractability for Monomials

- **Monomial**: language of the form $A_1^* a_1 A_2^* a_2 \cdots A_n^* a_n A_{n+1}^*$ where $a_1, \ldots, a_n \in \Sigma$ and $A_1, \ldots, A_{n+1} \subseteq \Sigma$

- **Union of monomials**: union of finitely many such languages
  - **Example**: pattern matching $\Sigma^* \text{word1} \Sigma^* + \Sigma^* \text{word2} \Sigma^*$
• **Monomial**: language of the form $A_1^* a_1 A_2^* a_2 \cdots A_n^* a_n A_{n+1}^*$ where $a_1, \ldots, a_n \in \Sigma$ and $A_1, \ldots, A_{n+1} \subseteq \Sigma$

• **Union of monomials**: union of finitely many such languages
  • Example: pattern matching $\Sigma^* \text{word1} \Sigma^* + \Sigma^* \text{word2} \Sigma^*$
  • Logical interpretation: languages definable in $\Sigma_2[\prec]$
Tractability for Monomials

• **Monomial:** language of the form $A_1^* a_1 A_2^* a_2 \cdots A_n^* a_n A_{n+1}^*$
  where $a_1, \ldots, a_n \in \Sigma$ and $A_1, \ldots, A_{n+1} \subseteq \Sigma$

• **Union of monomials:** union of finitely many such languages
  
  • Example: pattern matching $\Sigma^* \text{word1} \Sigma^* + \Sigma^* \text{word2} \Sigma^*$
  
  • Logical interpretation: languages definable in $\Sigma_2[\langle\rangle]$  

---

**Theorem**

*For any union of monomials $L$, the problem $\text{CTS}(L)$ is in $\text{NL}$*
Tractability for Monomials

- **Monomial:** language of the form $A_1^* a_1 A_2^* a_2 \cdots A_n^* a_n A_{n+1}^*$
  where $a_1, \ldots, a_n \in \Sigma$ and $A_1, \ldots, A_{n+1} \subseteq \Sigma$

- **Union of monomials:** union of finitely many such languages
  
  - **Example:** pattern matching $\Sigma^* \text{word1} \Sigma^* + \Sigma^* \text{word2} \Sigma^*$
  - **Logical interpretation:** languages definable in $\Sigma_2[\lt]$

**Theorem**

*For any union of monomials $L$, the problem $\text{CTS}(L)$ is in $\text{NL}$*

**Proof idea:**

- Tractable languages are clearly closed under union
Tractability for Monomials

- **Monomial**: language of the form $A_1^* a_1 A_2^* a_2 \cdots A_n^* a_n A_{n+1}^*$ where $a_1, \ldots, a_n \in \Sigma$ and $A_1, \ldots, A_{n+1} \subseteq \Sigma$

- **Union of monomials**: union of finitely many such languages
  - **Example**: pattern matching $\Sigma^* \text{word1} \Sigma^* + \Sigma^* \text{word2} \Sigma^*$
  - **Logical interpretation**: languages definable in $\Sigma_2[\prec]$

**Theorem**

*For any union of monomials $L$, the problem $CTS(L)$ is in NL*

**Proof idea:**

- Tractable languages are clearly **closed under union**
- We can **guess** the positions of the individual $a_i$
Tractability for Monomials

- **Monomial**: language of the form \( A_1^* a_1 A_2^* a_2 \cdots A_n^* a_n A_{n+1}^* \)
  where \( a_1, \ldots, a_n \in \Sigma \) and \( A_1, \ldots, A_{n+1} \subseteq \Sigma \)

- **Union of monomials**: union of finitely many such languages
  - **Example**: pattern matching \( \Sigma^* \text{word1} \Sigma^* + \Sigma^* \text{word2} \Sigma^* \)
  - **Logical interpretation**: languages definable in \( \Sigma_2[\prec] \)

---

**Theorem**

For any union of monomials \( L \), the problem \( \text{CTS}(L) \) is in NL

**Proof idea:**

- Tractable languages are clearly **closed under union**
- We can **guess** the positions of the individual \( a_i \)
- Check that the other vertices **can fit** in the \( A_i^* \) (uses NL = co-NL)
The Algebraic Approach

Can we just study algebraically the tractable languages?

• Not closed under intersection
• Not closed under complement
• Not closed under inverse morphism
• Not closed under concatenation (not in paper, only known for CTS)
• For CSh: not closed under quotient

Remark: For the language $L = b\Sigma^* + a\Sigma^* + (ab)^*$

• CTS($L$) is NP-hard because $(ab) - one.osf L = (ab)^*$
• CSh($L$) is in NL: trivial if there is more than one word
Can we just study algebraically the tractable languages? Not really...

• Not closed under intersection
• Not closed under complement
• Not closed under inverse morphism
• Not closed under concatenation (not in paper, only known for CTS)
• For CSh: not closed under quotient

Remark: For the language $L = b\Sigma^* + aa\Sigma^* + (ab)^*$

• $\text{CTS}(L)$ is NP-hard because $(ab)^*/\text{one.osf} L = (ab)^*$
• $\text{CSh}(L)$ is in NL: trivial if there is more than one word
Can we just study algebraically the tractable languages? Not really...

- Not closed under intersection
- Not closed under complement
- Not closed under inverse morphism
- Not closed under concatenation
  (not in paper, only known for CTS)
- For CSh: not closed under quotient
The Algebraic Approach Fails

Can we just study algebraically the tractable languages? Not really...

- Not closed under intersection
- Not closed under complement
- Not closed under inverse morphism
- Not closed under concatenation
  (not in paper, only known for CTS)
- For CSh: not closed under quotient

Remark: For the language \( L = b\Sigma^* + aa\Sigma^* + (ab)^* \)

- \( \text{CTS}(L) \) is \( \text{NP-hard} \) because \((ab)^{-1}L = (ab)^*\)
- \( \text{CSh}(L) \) is in \( \text{NL} \): trivial if there is more than one word
• \( \text{CSh}(L) \) is in NL for any regular language \( L \) if we assume that there are at most \( k \) input words \( w_1, \ldots, w_k \) for a constant \( k \in \mathbb{N} \).
• **CSh**(*L*) is in **NL** for any regular language *L* if we assume that there are at most *k* input words *w*₁, . . . , *w*ₖ for a constant *k* ∈ **N**

→ Need *k* counters to remember the current position in each word, plus automaton state
Tractability Based on Width

• $\text{CSh}(L)$ is in NL for any regular language $L$ if we assume that there are at most $k$ input words $w_1, \ldots, w_k$ for a constant $k \in \mathbb{N}$
  
  → Need $k$ counters to remember the current position in each word, plus automaton state

• $\text{CTS}(L)$ is in NL for any regular language $L$ if the input DAG $G$ has \textbf{width} $\leq k$ for constant $k \in \mathbb{N}$
• **CSh(L)** is in NL for any regular language $L$ if we assume that there are at most $k$ input words $w_1, \ldots, w_k$ for a constant $k \in \mathbb{N}$
  
  $\rightarrow$ Need $k$ counters to remember the current position in each word, plus automaton state

• **CTS(L)** is in NL for any regular language $L$ if the input DAG $G$ has width $\leq k$ for constant $k \in \mathbb{N}$
Tractability Based on Width

- **CSh**(\(L\)) is in NL for any regular language \(L\) if we assume that there are at most \(k\) input words \(w_1, \ldots, w_k\) for a constant \(k \in \mathbb{N}\)
  - Need \(k\) counters to remember the current position in each word, plus automaton state

- **CTS**(\(L\)) is in NL for any regular language \(L\) if the input DAG \(G\) has width \(\leq k\) for constant \(k \in \mathbb{N}\)
  - **Width**: size of the largest antichain (subset of pairwise incomparable vertices)
Tractability Based on Width

• \( \text{CSh}(L) \) is in \( \text{NL} \) for any regular language \( L \) if we assume that there are at most \( k \) input words \( w_1, \ldots, w_k \) for a constant \( k \in \mathbb{N} \)
  
  → Need \( k \) counters to remember the current position in each word, plus automaton state

• \( \text{CTS}(L) \) is in \( \text{NL} \) for any regular language \( L \) if the input DAG \( G \) has width \( \leq k \) for constant \( k \in \mathbb{N} \)
  
  • width: size of the largest antichain
    
    (subset of pairwise incomparable vertices)
Tractability Based on Width

- CSh($L$) is in NL for any regular language $L$ if we assume that there are at most $k$ input words $w_1, \ldots, w_k$ for a constant $k \in \mathbb{N}$.
  
  → Need $k$ counters to remember the current position in each word, plus automaton state.

- CTS($L$) is in NL for any regular language $L$ if the input DAG $G$ has width $\leq k$ for constant $k \in \mathbb{N}$.
  
  - **Width**: size of the largest antichain (subset of pairwise incomparable vertices).
  
  → Partition $G$ in $k$ chains (Dilworth’s theorem), and conclude by NL algorithm.

\[
\begin{align*}
\text{Graph:} & \\
& a \rightarrow b \rightarrow a \rightarrow b \rightarrow a
\end{align*}
\]
Tractability Based on Width

• $\text{CSh}(L)$ is in NL for any regular language $L$ if we assume that there are at most $k$ input words $w_1, \ldots, w_k$ for a constant $k \in \mathbb{N}$
  → Need $k$ counters to remember the current position in each word, plus automaton state

• $\text{CTS}(L)$ is in NL for any regular language $L$ if the input DAG $G$ has width $\leq k$ for constant $k \in \mathbb{N}$
  • Width: size of the largest antichain (subset of pairwise incomparable vertices)
  → Partition $G$ in $k$ chains (Dilworth’s theorem), and conclude by NL algorithm
Tractability Based on Width

- \( \text{CSh}(L) \) is in NL for any regular language \( L \) if we assume that there are at most \( k \) input words \( w_1, \ldots, w_k \) for a constant \( k \in \mathbb{N} \)
  - Need \( k \) counters to remember the current position in each word, plus automaton state

- \( \text{CTS}(L) \) is in NL for any regular language \( L \) if the input DAG \( G \) has width \( \leq k \) for constant \( k \in \mathbb{N} \)
  - Width: size of the largest antichain (subset of pairwise incomparable vertices)
  - Partition \( G \) in \( k \) chains (Dilworth’s theorem), and conclude by NL algorithm

→ These results are making an additional assumption, but...
• Fix $\Sigma = \{a, b\}$, take any regular language $L$ and constant $k \in \mathbb{N}$, we know that $\text{CTS}$ is in $\text{NL}$ for $L + \Sigma^*(a^k + b^k)\Sigma^*$
Tractability Based on Width (2)

- Fix $\Sigma = \{a, b\}$, take any regular language $L$ and constant $k \in \mathbb{N}$, we know that CTS is in NL for $L + \Sigma^*(a^k + b^k)\Sigma^*$
  - If the input DAG has width < $2k$, use the result for bounded width.
Tractability Based on Width (2)

- Fix $\Sigma = \{a, b\}$, take any regular language $L$ and constant $k \in \mathbb{N}$, we know that $\text{CTS}$ is in $\text{NL}$ for $L + \Sigma^*(a^k + b^k)\Sigma^*$
  - If the input DAG has width $< 2k$, use the result for bounded width
  - Otherwise we can achieve $a^k$ or $b^k$ with a large antichain
Fix $\Sigma = \{a, b\}$, take any regular language $L$ and constant $k \in \mathbb{N}$, we know that CTS is in NL for $L + \Sigma^* (a^k + b^k) \Sigma^*$

- If the input DAG has width $< 2k$, use the result for bounded width
- Otherwise we can achieve $a^k$ or $b^k$ with a large antichain

A similar technique shows that $(ab)^* + \Sigma^* aa \Sigma^*$ is tractable

→ Does it suffice to bound the width of all letters but one?
• Fix $\Sigma = \{a, b\}$, take any regular language $L$ and constant $k \in \mathbb{N}$, we know that CTS is in NL for $L + \Sigma^*(a^k + b^k)\Sigma^*$
  • If the input DAG has width $< 2k$, use the result for bounded width
  • Otherwise we can achieve $a^k$ or $b^k$ with a large antichain

• A similar technique shows that $(ab)^* + \Sigma^*aa\Sigma^*$ is tractable

→ Does it suffice to bound the width of all letters but one?
  → Unknown for $L + \Sigma^*a^k\Sigma^*$ with arbitrary $L$ and $k > 2$!  ゚_(_;)_/\
• **Group language:** the underlying monoid is a **finite group**
  → Automata where each letter acts **bijectively**
• **Group language:** the underlying monoid is a **finite group**
  → Automata where each letter **acts bijectively**

• **District group monomial:** language $G_1 \ a_1 \ \cdots \ G_n \ a_n \ G_{n+1}$
  where $a_1, \ldots, a_n \in \Sigma$ and $G_1, \ldots, G_n$ are group languages on **subsets** of the alphabet $\Sigma$
Tractability Based on the Structure of Groups

- **Group language:** the underlying monoid is a **finite group**
  → Automata where each letter **acts bijectively**

- **District group monomial:** language $G_1 \ a_1 \ · · · \ G_n \ a_n \ G_{n+1}$
  where $a_1, \ldots, a_n \in \Sigma$ and $G_1, \ldots, G_n$ are **group languages**
  on **subsets** of the alphabet $\Sigma$

### Theorem

For any union $L$ of district group monomials, $\text{CSh}(L)$ is **in NL**
Tractability Based on the Structure of Groups

- **Group language**: the underlying monoid is a **finite group**
  - Automata where each letter acts bijectively

- **District group monomial**: language $G_1 a_1 \cdots G_n a_n G_{n+1}$
  where $a_1, \ldots, a_n \in \Sigma$ and $G_1, \ldots, G_n$ are group languages on subsets of the alphabet $\Sigma$

**Theorem**

*For any union $L$ of district group monomials, $\text{CSh}(L)$ is in $\text{NL}$*

→ Only for $\text{CSh}$; complexity for $\text{CTS}$ is unknown!  

¯\_(ツ)_/¯
Tractability Based on All Sorts of Strange Reasons

- \((aa + b)^*\) is in NL for CSh:
Tractability Based on All Sorts of Strange Reasons

• \((aa + b)^*\) is in NL for CSh:
  • Ad-hoc greedy algorithm: consume chain with most odd a blocks
• $(aa + b)^*$ is in NL for CSh:
  • Ad-hoc greedy algorithm: consume chain with most odd $a$ blocks
  • Complexity open for CTS! 

Tractability argument: another ad hoc greedy algorithm

Hardness argument: from $k$-clique encoded to a bipartite graph
Tractability Based on All Sorts of Strange Reasons

- \((aa + b)^*\) is in \textbf{NL} for CSh:
  - \textbf{Ad-hoc greedy algorithm}: consume chain with most odd \(a\) blocks
  - Complexity \textbf{open} for CTS! \(\(_(ツ)_\)/\)
  - Complexity \textbf{open} for \((a^k + b)^*\) for \(k > 2\)! \(\(_(ツ)_\)/\)
Tractability Based on All Sorts of Strange Reasons

• \((aa + b)^*\) is in NL for CSh:
  • Ad-hoc greedy algorithm: consume chain with most odd \(a\) blocks
  • Complexity open for CTS!
  • Complexity open for \((a^k + b)^*\) for \(k > 2\)
  • What about similar languages like \((aa + bb + ab)^*\)?
Tractability Based on All Sorts of Strange Reasons

• \((aa + b)^*\) is in NL for CSh:
  • Ad-hoc greedy algorithm: consume chain with most odd a blocks
  • Complexity open for CTS!
  • Complexity open for \((a^k + b)^*\) for \(k > 2\)
  • What about similar languages like \((aa + bb + ab)^*\)?

• \((caa)^*d(cbb)^*d\Sigma^* + \Sigma^*cc\Sigma^*\) is in NL for CSh but NP-hard for CTS
  • Tractability argument: another ad hoc greedy algorithm
  • Hardness argument: from k-clique encoded to a bipartite graph
Conclusion
### Summary and Future Work

<table>
<thead>
<tr>
<th>Language</th>
<th>( (ab)^* ), ( u^* ) with different letters</th>
<th>CSh (shuffle)</th>
<th>CTS (top. sort)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( NP-hard )</td>
<td>( NP-hard )</td>
<td></td>
</tr>
</tbody>
</table>
## Summary and Future Work

<table>
<thead>
<tr>
<th>Language</th>
<th>CSh (shuffle)</th>
<th>CTS (top. sort)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((ab)^<em>, u^</em>) with different letters</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td>Monomials (A_1^*a_1 \cdots A_n^<em>a_nA_{n+1}^</em>) Groups, district group monomials</td>
<td>in NL</td>
<td>in NL</td>
</tr>
</tbody>
</table>

Essentially all other languages... Thanks for your attention!
## Summary and Future Work

<table>
<thead>
<tr>
<th>Language</th>
<th>CSh (shuffle)</th>
<th>CTS (top. sort)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(ab)^<em>, u^</em>$ with different letters</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td>Monomials $A_1^*a_1 \cdots A_n^<em>a_nA_{n+1}^</em>$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>Groups, district group monomials</td>
<td>in NL</td>
<td>_(ツ)_/_</td>
</tr>
<tr>
<td>$b\Sigma^* + aa\Sigma^* + (ab)^*$</td>
<td>in NL</td>
<td>NP-hard</td>
</tr>
</tbody>
</table>
## Summary and Future Work

<table>
<thead>
<tr>
<th>Language</th>
<th>CSh (shuffle)</th>
<th>CTS (top. sort)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(ab)^<em>, u^</em>$ with different letters</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td>Monomials $A_1^*a_1 \cdots A_n^*a_nA_{n+1}$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>Groups, district group monomials</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>$b\Sigma^* + aa\Sigma^* + (ab)^*$</td>
<td>in NL</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$L + \Sigma^<em>(a^k + b^k)\Sigma^</em>$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>$(ab)^* + \Sigma^<em>a^2\Sigma^</em>$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>$L + \Sigma^<em>a^k\Sigma^</em>$</td>
<td>(___)</td>
<td>(___)</td>
</tr>
</tbody>
</table>

Thanks for your attention!
## Summary and Future Work

<table>
<thead>
<tr>
<th>Language</th>
<th>CSh (shuffle)</th>
<th>CTS (top. sort)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(ab)^<em>, u^</em>$ with different letters</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td>Monomials $A_1^*a_1 \cdots A_n^<em>a_nA_{n+1}^</em>$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>Groups, district group monomials</td>
<td>in NL</td>
<td>(__(__)__)</td>
</tr>
<tr>
<td>$b\Sigma^* + aa\Sigma^* + (ab)^*$</td>
<td>in NL</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$L + \Sigma^<em>(a^k + b^k)\Sigma^</em>$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>$(ab)^* + \Sigma^<em>a^2\Sigma^</em>$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>$L + \Sigma^<em>a^k\Sigma^</em>$</td>
<td>(__(__)__)</td>
<td>(__(__)__)</td>
</tr>
<tr>
<td>$(aa + bb)^<em>, (ab + a)^</em>$</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$(aa + b)^*$</td>
<td>in NL</td>
<td>(__(__)__)</td>
</tr>
<tr>
<td>$(a^k + b)^*$</td>
<td>(__(__)__)</td>
<td>(__(__)__)</td>
</tr>
</tbody>
</table>

Essentially all other languages... Thanks for your attention!
Summary and Future Work

<table>
<thead>
<tr>
<th>Language</th>
<th>CSh (shuffle)</th>
<th>CTS (top. sort)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(ab)^<em>, u^</em>$ with different letters</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td>Monomials $A_1a_1 \cdots A_na_nA_{n+1}$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>Groups, district group monomials</td>
<td>in NL</td>
<td></td>
</tr>
<tr>
<td>$b\Sigma^* + aa\Sigma^* + (ab)^*$</td>
<td>in NL</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$L + \Sigma^<em>(a^k + b^k)\Sigma^</em>$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>$(ab)^* + \Sigma^<em>a^2\Sigma^</em>$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>$L + \Sigma^<em>a^k\Sigma^</em>$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(aa + bb)^<em>, (ab + a)^</em>$</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$(aa + b)^*$</td>
<td>in NL</td>
<td></td>
</tr>
<tr>
<td>$(a^k + b)^*$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Essentially all other languages...
Summary and Future Work

<table>
<thead>
<tr>
<th>Language</th>
<th>Informations</th>
<th>MNEMS</th>
<th>Problem definition</th>
</tr>
</thead>
</table>
| (ab)*, u* with different letters | NP-hard | in NL | Fix an alphabet A. An A-DAG is a directed acyclic graph G where each vertex is labeled by a letter of A. A topological sort of G is a linear ordering of the vertices that respects the edges of the DAG, i.e., for every edge (u, v) of G, the vertex u is enumerated before v. The topological sort achieves the word of A* formed by concatenating the labels of the vertices in the order where they are enumerated.

Fix a language L ⊆ A*. The constrained topological sort problem for L, written CTS[L], asks, given an A-DAG G, whether there is a topological sort of G that achieves a word of L.

One problem variant is the multi-letter setting where the input DAG is an A*+DAG, where the vertices are labeled by a word of A*, i.e., a topological sort achieves the word obtained by concatenating the labels of the vertices, but the words labeling each vertex cannot be interleaved with anything else. However in this page we mostly focus on the single-letter setting, i.e., A-DAGs.

Our current main results on the CTS-problem are presented in our paper. We show that CTS[L] is in NL for some regular languages L, and is NP-hard for some other regular languages.

Main dichotomy conjecture: For every regular language L, either CTS[L] is in NL or CTS[L] is NP-hard.

Restrictions on the input DAG

When the input DAG G is an union of paths, the problem is called constrained shuffle problem (CSh), because a topological sort of G corresponds to an interleaving of the strings represented by the paths.

We can consider the problem where the input DAG has bounded height, where the height of a DAG is defined as the length of the longest directed path.

We can consider the problem where the input DAG has bounded width, where the width of a DAG is the size of its largest antichain, i.e., subset of pairwise incomparable vertices. In the case of the CSh problem, the width is the number of paths.

Essentially all other languages...
# Summary and Future Work

<table>
<thead>
<tr>
<th>Language</th>
<th>CTS (top. sort)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(ab)^<em>, u^</em>$ with different letters</td>
<td>NP-hard</td>
</tr>
<tr>
<td>Monomials $A_1^*$</td>
<td>in NL</td>
</tr>
<tr>
<td>Groups, district</td>
<td>in NL</td>
</tr>
<tr>
<td>$b\Sigma^* + aa\Sigma^* +$</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$L + \Sigma^*(a^k + b^k)$</td>
<td>in NL</td>
</tr>
<tr>
<td>$(ab)^* + \Sigma^<em>a^2\Sigma^</em>$</td>
<td>in NL</td>
</tr>
<tr>
<td>$L + \Sigma^<em>a^k\Sigma^</em>$</td>
<td></td>
</tr>
<tr>
<td>$(aa + bb)^<em>, (aa + b)^</em>$</td>
<td></td>
</tr>
<tr>
<td>$(a^k + b)^*$</td>
<td></td>
</tr>
<tr>
<td>Essentially all other languages...</td>
<td></td>
</tr>
</tbody>
</table>

*Thanks for your attention!
## Summary and Future Work

<table>
<thead>
<tr>
<th>Language</th>
<th>CSh (shuffle)</th>
<th>CTS (top. sort)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(ab)^<em>, u^</em>$ with different letters</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td>Monomials $A_1^*a_1 \cdots A_n^<em>a_nA_{n+1}^</em>$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>Groups, district group monomials</td>
<td>in NL</td>
<td>(ツ)<em>(</em>)<em>(</em>)_</td>
</tr>
<tr>
<td>$b\Sigma^* + aa\Sigma^* + (ab)^*$</td>
<td>in NL</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$L + \Sigma^<em>(a^k + b^k)\Sigma^</em>$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>$(ab)^* + \Sigma^<em>a^2\Sigma^</em>$</td>
<td>in NL</td>
<td>in NL</td>
</tr>
<tr>
<td>$L + \Sigma^<em>a^k\Sigma^</em>$</td>
<td>(ツ)<em>(</em>)<em>(</em>)_</td>
<td>(ツ)<em>(</em>)<em>(</em>)_</td>
</tr>
<tr>
<td>$(aa + bb)^<em>, (ab + a)^</em>$</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$(aa + b)^*$</td>
<td>in NL</td>
<td>(ツ)<em>(</em>)<em>(</em>)_</td>
</tr>
<tr>
<td>$(a^k + b)^*$</td>
<td>(ツ)<em>(</em>)<em>(</em>)_</td>
<td>(ツ)<em>(</em>)<em>(</em>)_</td>
</tr>
<tr>
<td>Essentially all other languages...</td>
<td>(ツ)<em>(</em>)<em>(</em>)_</td>
<td>(ツ)<em>(</em>)<em>(</em>)_</td>
</tr>
</tbody>
</table>

Thanks for your attention!

Image Credits