

Computation with Absolutely No Space Overhead

Lane Hemaspaandra¹ Proshanto Mukherji¹ Till Tantau²

¹Department of Computer Science
University of Rochester

²Fakultät für Elektrotechnik und Informatik
Technical University of Berlin

Developments in Language Theory Conference, 2003

The Model of Overhead-Free Computation

The Standard Model of Linear Space

Our Model of Absolutely No Space Overhead

The Power of Overhead-Free Computation

Palindromes

Linear Languages

Context-Free Languages with a Forbidden Subword

Languages Complete for Polynomial Space

Limitations of Overhead-Free Computation

Linear Space is Strictly More Powerful

The Model of Overhead-Free Computation

The Standard Model of Linear Space

Our Model of Absolutely No Space Overhead

The Power of Overhead-Free Computation

Palindromes

Linear Languages

Context-Free Languages with a Forbidden Subword

Languages Complete for Polynomial Space

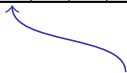
Limitations of Overhead-Free Computation

Linear Space is Strictly More Powerful

The Standard Model of Linear Space

tape

0	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **is larger than** input alphabet

The Standard Model of Linear Space

tape

\$	0	1	0	0	1	0	0
----	---	---	---	---	---	---	---



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **is larger than** input alphabet

The Standard Model of Linear Space

tape

\$	0	1	0	0	1	0	0
----	---	---	---	---	---	---	---



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **is larger than** input alphabet

The Standard Model of Linear Space

tape

\$	0	1	0	0	1	0	\$
----	---	---	---	---	---	---	----



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **is larger than** input alphabet

The Standard Model of Linear Space

tape

\$	0	1	0	0	1	0	\$
----	---	---	---	---	---	---	----



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **is larger than** input alphabet

The Standard Model of Linear Space

tape

\$	\$	1	0	0	1	0	\$
----	----	---	---	---	---	---	----



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **is larger than** input alphabet

The Standard Model of Linear Space

tape

\$	\$	1	0	0	1	0	\$
----	----	---	---	---	---	---	----



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **is larger than** input alphabet

The Standard Model of Linear Space

tape

\$	\$	1	0	0	1	\$	\$
----	----	---	---	---	---	----	----



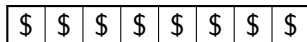
Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **is larger than** input alphabet

The Standard Model of Linear Space

tape



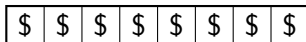
Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **is larger than** input alphabet

The Standard Model of Linear Space

tape

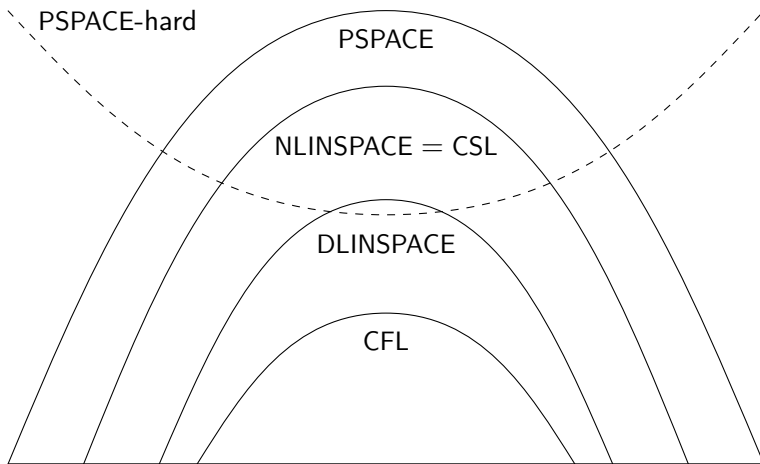


Turing machine

Characteristics

- ▶ Input fills **fixed-size tape**
- ▶ Input may be **modified**
- ▶ Tape alphabet **is larger than** input alphabet

Linear Space is a Powerful Model



Our Model of “Absolutely No Space Overhead”

tape

0	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **equals** input alphabet

Our Model of “Absolutely No Space Overhead”

tape

1	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **equals** input alphabet

Our Model of “Absolutely No Space Overhead”

tape

1	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **equals** input alphabet

Our Model of “Absolutely No Space Overhead”

tape

1	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **equals** input alphabet

Our Model of “Absolutely No Space Overhead”

tape

1	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **equals** input alphabet

Our Model of “Absolutely No Space Overhead”

tape

1	1	1	0	0	1	0	1
---	---	---	---	---	---	---	---



Turing machine

Characteristics

- ▶ Input fills **fixed-size** tape
- ▶ Input may be **modified**
- ▶ Tape alphabet **equals** input alphabet

Our Model of “Absolutely No Space Overhead”



Intuition

- ▶ Tape is used like a RAM module.



Turing machine

Definition of Overhead-Free Computations

Definition

A Turing machine is **overhead-free** if

- ▶ it has only a single tape,
- ▶ writes only on input cells,
- ▶ writes only symbols drawn from the input alphabet.

Overhead-Free Computation Complexity Classes

Definition

A language $L \subseteq \Sigma^*$ is in

DOF if L is accepted by a deterministic overhead-free machine with input alphabet Σ ,

Overhead-Free Computation Complexity Classes

Definition

A language $L \subseteq \Sigma^*$ is in

DOF if L is accepted by a deterministic overhead-free machine with input alphabet Σ ,

DOF_{poly} if L is accepted by a deterministic overhead-free machine with input alphabet Σ in polynomial time.

Overhead-Free Computation Complexity Classes

Definition

A language $L \subseteq \Sigma^*$ is in

DOF if L is accepted by a deterministic overhead-free machine with input alphabet Σ ,

DOF_{poly} if L is accepted by a deterministic overhead-free machine with input alphabet Σ in polynomial time.

NOF is the nondeterministic version of DOF,

Overhead-Free Computation Complexity Classes

Definition

A language $L \subseteq \Sigma^*$ is in

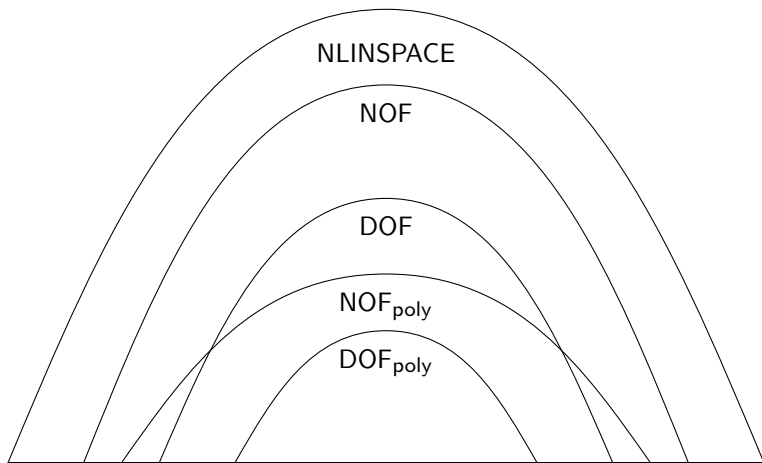
DOF if L is accepted by a deterministic overhead-free machine with input alphabet Σ ,

DOF_{poly} if L is accepted by a deterministic overhead-free machine with input alphabet Σ in polynomial time.

NOF is the nondeterministic version of DOF,

NOF_{poly} is the nondeterministic version of DOF_{poly}.

Simple Relationships among Overhead-Free Computation Classes



The Model of Overhead-Free Computation

The Standard Model of Linear Space

Our Model of Absolutely No Space Overhead

The Power of Overhead-Free Computation

Palindromes

Linear Languages

Context-Free Languages with a Forbidden Subword

Languages Complete for Polynomial Space

Limitations of Overhead-Free Computation

Linear Space is Strictly More Powerful

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

0	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

1	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

1	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

1	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

1	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

0	1	1	0	0	1	0	1
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

0	1	1	0	0	1	0	1
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

0	1	1	0	0	1	1	0
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

0	1	1	0	0	1	1	0
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

0	0	1	0	0	1	1	0
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

0	0	1	0	0	1	1	0
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

0	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

0	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Palindromes Can be Accepted in an Overhead-Free Way

Algorithm

tape

0	0	0	1	1	0	0	0
---	---	---	---	---	---	---	---



overhead-free machine

Phase 1:

Compare first and last bit

Place left end marker

Place right end marker

Phase 2:

Compare bits next to end markers

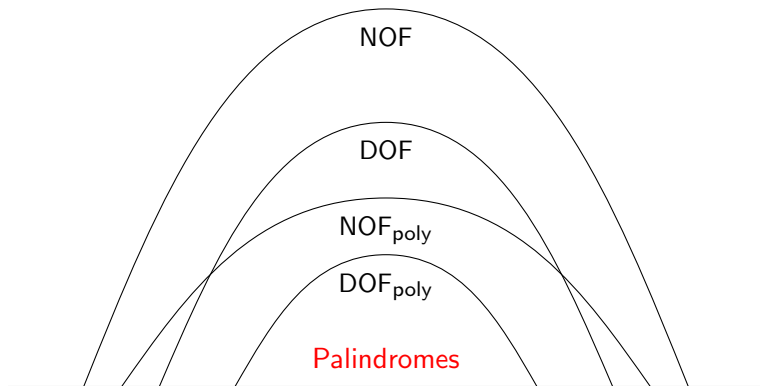
Find left end marker

Advance left end marker

Find right end marker

Advance right end marker

Relationships among Overhead-Free Computation Classes



A Review of Linear Grammars

Definition

A grammar is **linear** if it is context-free and there is only one nonterminal per right-hand side.

Example

$$G_1: S \rightarrow 00S0 \mid 1.$$

$$G_2: S \rightarrow 0S10 \mid 0.$$

A Review of Linear Grammars

Definition

A grammar is **linear** if it is context-free and there is only one nonterminal per right-hand side.

Example

$G_1: S \rightarrow 00S0 \mid 1.$

$G_2: S \rightarrow 0S10 \mid 0.$

Definition

A grammar is **deterministic** if
“there is always only one rule that can be applied.”

Example

G_1 is deterministic.

G_2 is not deterministic.

Deterministic Linear Languages Can Be Accepted in an Overhead-Free Way

Theorem

Every deterministic linear language is in DOF_{poly} .

Continued Review of Linear Grammars

Definition

A language is **metilinear** if it is the concatenation of linear languages.

Continued Review of Linear Grammars

Definition

A language is **metilinear** if it is the concatenation of linear languages.

Example

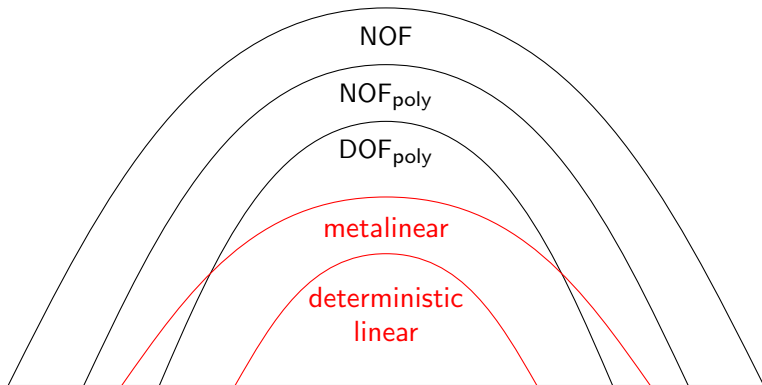
$\text{TRIPLE-PALINDROME} = \{uvw \mid u, v, \text{ and } w \text{ are palindromes}\}.$

Metalinear Languages Can Be Accepted in an Overhead-Free Way

Theorem

Every metalinear language is in NOF_{poly} .

Relationships among Overhead-Free Computation Classes



Definition of Almost-Overhead-Free Computations

Definition

A Turing machine is **almost-overhead-free** if

- ▶ it has only a single tape,

Definition of Almost-Overhead-Free Computations

Definition

A Turing machine is **almost-overhead-free** if

- ▶ it has only a single tape,
- ▶ writes only on input cells,

Definition of Almost-Overhead-Free Computations

Definition

A Turing machine is **almost-overhead-free** if

- ▶ it has only a single tape,
- ▶ writes only on input cells,
- ▶ writes only symbols drawn from the input alphabet
plus one special symbol.

Context-Free Languages with a Forbidden Subword Can Be Accepted in an Overhead-Free Way

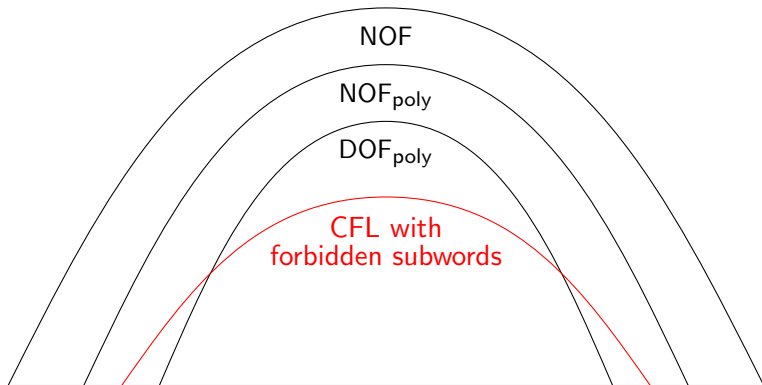
Theorem

Let L be a context-free language with a forbidden word.

Then $L \in \text{NOF}_{\text{poly}}$.

The proof is based on the fact that every context-free language can be accepted by a nondeterministic almost-overhead-free machine in polynomial time.

Relationships among Overhead-Free Computation Classes



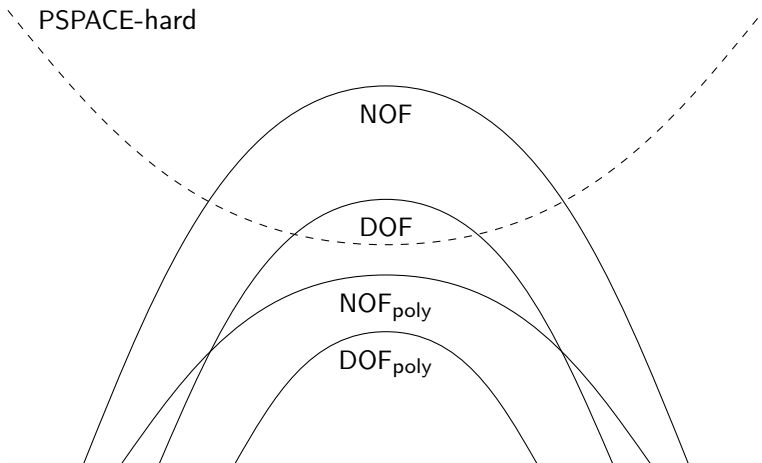
Some PSPACE-complete Languages Can Be Accepted in an Overhead-Free Way

Theorem

DOF contains languages that are complete for PSPACE.

The proof is based on the fact that for every $L \in \text{DLINSPACE}$ there exists an isometric homomorphism h such that $h(L) \in \text{DOF}$.

Relationships among Overhead-Free Computation Classes



The Model of Overhead-Free Computation

The Standard Model of Linear Space

Our Model of Absolutely No Space Overhead

The Power of Overhead-Free Computation

Palindromes

Linear Languages

Context-Free Languages with a Forbidden Subword

Languages Complete for Polynomial Space

Limitations of Overhead-Free Computation

Linear Space is Strictly More Powerful

Some Context-Sensitive Languages Cannot be Accepted in an Overhead-Free Way

Theorem

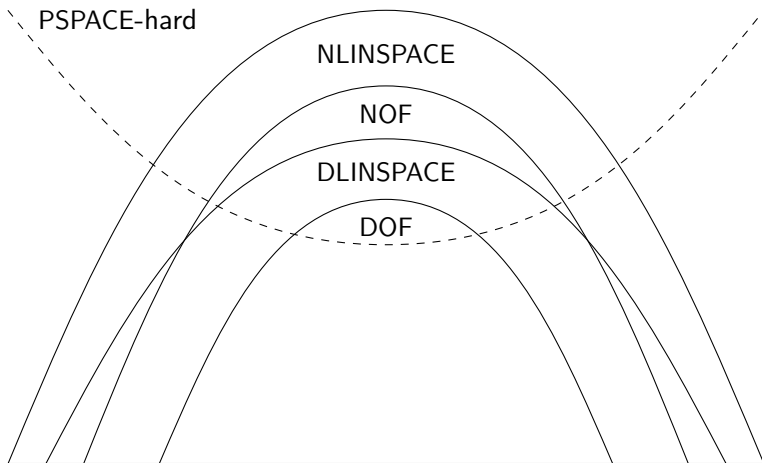
$\text{DOF} \subsetneq \text{DLINSPACE}$.

Theorem

$\text{NOF} \subsetneq \text{NLINSPACE}$.

The proofs are based on old diagonalisations due to Feldman, Owings, and Seiferas.

Relationships among Overhead-Free Computation Classes



Candidates for Languages that Cannot be Accepted in an Overhead-Free Way

Conjecture

DOUBLE-PALINDROMES \notin DOF.

Conjecture

$\{ww \mid w \in \{0, 1\}^*\} \notin$ NOF.

Proving the first conjecture would show $\text{DOF} \subsetneq \text{NOF}$.

Summary

- ▶ Overhead-free computation is a more faithful **model of fixed-size memory**.
- ▶ Overhead-free computation is **less powerful** than linear space.
- ▶ **Many** context-free languages can be accepted by overhead-free machines.
- ▶ We conjecture that **all** context-free languages are in NOF_{poly} .
- ▶ Our results can be seen as new results on the power of **linear bounded automata with fixed alphabet** size.

For Further Reading



A. Salomaa.

Formal Languages.

Academic Press, 1973.



E. Dijkstra.

Smoothsort, an alternative for sorting in situ.

Science of Computer Programming, 1(3):223–233, 1982.



E. Feldman and J. Owings, Jr.

A class of universal linear bounded automata.

Information Sciences, 6:187–190, 1973.



P. Jančar, F. Mráz, M. Plátek, and J. Vogel.

Restarting automata.

FCT Conference 1995, LNCS 985, pages 282–292. 1995.