Algorithmic problems on compressed words

Markus Lohrey

Universität Leipzig

October 11, 2011

Motivation

Try to develop algorithms that directly work on compressed data.

Motivation

Try to develop algorithms that directly work on compressed data.

Goal: Beat straightforward decompress and analyze strategy.

Motivation

Try to develop algorithms that directly work on compressed data.

Goal: Beat straightforward decompress and analyze strategy.

In this talk: focus on compressed strings

- Algorithms for analyzing compressed strings/trees
- Lower complexity bounds for algorithmic problems on compressed strings/trees.

Motivation

Try to develop algorithms that directly work on compressed data.

Goal: Beat straightforward decompress and analyze strategy.

In this talk: focus on compressed strings

- Algorithms for analyzing compressed strings/trees
- Lower complexity bounds for algorithmic problems on compressed strings/trees.

Applications:

- all domains, where massive string/tree data arise and have to be processed, e.g. bioinformatics, XML

Motivation

Try to develop algorithms that directly work on compressed data.

Goal: Beat straightforward decompress and analyze strategy.

In this talk: focus on compressed strings

- Algorithms for analyzing compressed strings/trees
- Lower complexity bounds for algorithmic problems on compressed strings/trees.

Applications:

- all domains, where massive string/tree data arise and have to be processed, e.g. bioinformatics, XML
- large (and highly compressible) data often occur as intermediate data structures.

Motivation

Try to develop algorithms that directly work on compressed data.

Goal: Beat straightforward decompress and analyze strategy.

In this talk: focus on compressed strings

- Algorithms for analyzing compressed strings/trees
- Lower complexity bounds for algorithmic problems on compressed strings/trees.

Applications:

- all domains, where massive string/tree data arise and have to be processed, e.g. bioinformatics, XML
- large (and highly compressible) data often occur as intermediate data structures.

  Examples in: combinatorial group theory, computational topology, program analysis, verification, ...

Compressed strings and straight-line programs

Dictionary-based compression (LZ77, LZ78) exploits text repetition.

Compressed strings and straight-line programs

Dictionary-based compression (LZ77, LZ78) exploits text repetition.

Straight-line programs are a general representation for compressed strings, which covers most dictionary-based algorithms.

Compressed strings and straight-line programs

Dictionary-based compression (LZ77, LZ78) exploits text repetition.

Straight-line programs are a general representation for compressed strings, which covers most dictionary-based algorithms.

Definition (Straight-line program (SLP))

An SLP over the alphabet Γ is a sequence of definitions

$$\mathbb{A} = \langle A_i := \alpha_i \rangle_{0 \le i \le n},$$

where either $\alpha_i \in \Gamma$ or $\alpha_i = A_j A_k$ for some $j, k > i$.

Compressed strings and straight-line programs

Dictionary-based compression (LZ77, LZ78) exploits text repetition.

Straight-line programs are a general representation for compressed strings, which covers most dictionary-based algorithms.

Definition (Straight-line program (SLP))

An SLP over the alphabet Γ is a sequence of definitions

$$\mathbb{A} = \langle A_i := \alpha_i \rangle_{0 \le i \le n},$$

where either $\alpha_i \in \Gamma$ or $\alpha_i = A_j A_k$ for some $j, k > i$.

Alternatively: a context-free grammar that generates exactly one string.

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2} \text{ for } 0 \leq i \leq 3, \quad A_4 := b, \quad A_5 := a\rangle$.

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2} \text{ for } 0 \leq i \leq 3, \quad A_4 := b, \quad A_5 := a\rangle$.

$$A_0 = A_1 A_2$$

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2} \text{ for } 0 \leq i \leq 3, \quad A_4 := b, \quad A_5 := a \rangle.$

$$\begin{aligned} A_0 &= A_1 A_2 \\ &= A_2 A_3 A_3 A_4 \end{aligned}$$

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2}$ for $0 \le i \le 3, \quad A_4 := b, \quad A_5 := a\rangle$.

$$
\begin{aligned}
A_0 &= A_1A_2 \\
&= A_2A_3A_3A_4 \\
&= A_3A_4A_4A_5A_4A_5b
\end{aligned}
$$

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2} \text{ for } 0 \leq i \leq 3, \quad A_4 := b, \quad A_5 := a\rangle$.

$$\begin{aligned} A_0 &= A_1A_2 \\ &= A_2A_3A_3A_4 \\ &= A_3A_4A_4A_5A_4A_5b \\ &= A_4A_5bbabab \end{aligned}$$

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2} \text{ for } 0 \leq i \leq 3, \quad A_4 := b, \quad A_5 := a\rangle$.

$$\begin{aligned}
A_0 &= A_1A_2 \\
&= A_2A_3A_3A_4 \\
&= A_3A_4A_4A_5A_4A_5b \\
&= A_4A_5bbabab \\
&= babbabab
\end{aligned}$$

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2} \text{ for } 0 \le i \le 3, \quad A_4 := b, \quad A_5 := a \rangle$.

$$
\begin{aligned}
A_0 &= A_1A_2 \\
&= A_2A_3A_3A_4 \\
&= A_3A_4A_4A_5A_4A_5b \\
&= A_4A_5bbabab \\
&= babbabab \;=\; \mathrm{val}(\mathbb{A})
\end{aligned}
$$

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2} \text{ for } 0 \leq i \leq 3, \quad A_4 := b, \quad A_5 := a\rangle$.

$$\begin{aligned} A_0 &= A_1A_2 \\ &= A_2A_3A_3A_4 \\ &= A_3A_4A_4A_5A_4A_5b \\ &= A_4A_5bbabab \\ &= babbabab \ = \ \mathrm{val}(\mathbb{A}) \end{aligned}$$

Grammar-based compression:

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2}$ for $0 \leq i \leq 3, \quad A_4 := b, \quad A_5 := a\rangle$.

$$\begin{aligned} A_0 &= A_1A_2 \\ &= A_2A_3A_3A_4 \\ &= A_3A_4A_4A_5A_4A_5b \\ &= A_4A_5bbabab \\ &= babbabab = \mathrm{val}(\mathbb{A}) \end{aligned}$$

Grammar-based compression:

- The size of an SLP $\mathbb{A} = (A_i := \alpha_i)_{1 \leq i \leq n}$ is $|\mathbb{A}| = n$.

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2} \text{ for } 0 \leq i \leq 3, \quad A_4 := b, \quad A_5 := a\rangle$.

$$\begin{aligned} A_0 &= A_1A_2 \\ &= A_2A_3A_3A_4 \\ &= A_3A_4A_4A_5A_4A_5b \\ &= A_4A_5bbabab \\ &= babbabab = \mathrm{val}(\mathbb{A}) \end{aligned}$$

Grammar-based compression:

- The size of an SLP $\mathbb{A} = (A_i := \alpha_i)_{1 \leq i \leq n}$ is $|\mathbb{A}| = n$.
- One may have $|\mathrm{val}(\mathbb{A})| = 2^{|\mathbb{A}|}$.

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2}$ for $0 \le i \le 3, \quad A_4 := b, \quad A_5 := a\rangle$.

$$\begin{aligned} A_0 &= A_1A_2 \\ &= A_2A_3A_3A_4 \\ &= A_3A_4A_4A_5A_4A_5b \\ &= A_4A_5bbabab \\ &= babbabab \ = \ \mathrm{val}(\mathbb{A}) \end{aligned}$$

Grammar-based compression:

- The size of an SLP $\mathbb{A} = (A_i := \alpha_i)_{1 \le i \le n}$ is $|\mathbb{A}| = n$.
- One may have $|\mathrm{val}(\mathbb{A})| = 2^{|\mathbb{A}|}$.
- An SLP $\mathbb{A}$ can be seen as a compressed representation of $\mathrm{val}(\mathbb{A})$.

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2} \text{ for } 0 \leq i \leq 3, \quad A_4 := b, \quad A_5 := a\rangle$.

$$\begin{aligned} A_0 &= A_1A_2 \\ &= A_2A_3A_3A_4 \\ &= A_3A_4A_4A_5A_4A_5b \\ &= A_4A_5bbabab \\ &= babbabab \;=\; \mathrm{val}(\mathbb{A}) \end{aligned}$$

Grammar-based compression:

- The size of an SLP $\mathbb{A} = (A_i := \alpha_i)_{1 \leq i \leq n}$ is $|\mathbb{A}| = n$.
- One may have $|\mathrm{val}(\mathbb{A})| = 2^{|\mathbb{A}|}$.
- An SLP $\mathbb{A}$ can be seen as a compressed representation of $\mathrm{val}(\mathbb{A})$.

Relationship to dictionary-based compression (Rytter 2003):

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2} \text{ for } 0 \leq i \leq 3, \quad A_4 := b, \quad A_5 := a\rangle$.

$$\begin{aligned} A_0 &= A_1A_2 \\ &= A_2A_3A_3A_4 \\ &= A_3A_4A_4A_5A_4A_5b \\ &= A_4A_5bbabab \\ &= babbabab = \mathrm{val}(\mathbb{A}) \end{aligned}$$

Grammar-based compression:

- The size of an SLP $\mathbb{A} = (A_i := \alpha_i)_{1 \leq i \leq n}$ is $|\mathbb{A}| = n$.
- One may have $|\mathrm{val}(\mathbb{A})| = 2^{|\mathbb{A}|}$.
- An SLP $\mathbb{A}$ can be seen as a compressed representation of $\mathrm{val}(\mathbb{A})$.

Relationship to dictionary-based compression (Rytter 2003):

- From an SLP $\mathbb{A}$ one can compute in polynomial time $\mathrm{LZ77}(\mathrm{val}(\mathbb{A}))$.

Example: $\mathbb{A} = \langle A_i := A_{i+1}A_{i+2} \text{ for } 0 \le i \le 3, \quad A_4 := b, \quad A_5 := a\rangle$.

$$\begin{aligned} A_0 &= A_1 A_2 \\ &= A_2 A_3 A_3 A_4 \\ &= A_3 A_4 A_4 A_5 A_4 A_5 b \\ &= A_4 A_5 bbabab \\ &= babbabab = \mathrm{val}(\mathbb{A}) \end{aligned}$$

Grammar-based compression:

- The size of an SLP $\mathbb{A} = (A_i := \alpha_i)_{1 \le i \le n}$ is $|\mathbb{A}| = n$.
- One may have $|\mathrm{val}(\mathbb{A})| = 2^{|\mathbb{A}|}$.
- An SLP $\mathbb{A}$ can be seen as a compressed representation of $\mathrm{val}(\mathbb{A})$.

Relationship to dictionary-based compression (Rytter 2003):

- From an SLP $\mathbb{A}$ one can compute in polynomial time $\mathrm{LZ77}(\mathrm{val}(\mathbb{A}))$.
- From $\mathrm{LZ77}(w)$ one can compute in polynomial time an SLP $\mathbb{A}$ with $\mathrm{val}(\mathbb{A}) = w$.

Algorithms on SLP-compressed strings

The mother of all algorithms on SLP -compressed strings:

Algorithms on SLP-compressed strings

The mother of all algorithms on SLP -compressed strings:

Plandowski 1994

The following problem can be solved in polynomial time:

INPUT: SLPs $\mathbb{A}, \mathbb{B}$
QUESTION: $\mathsf{val}(\mathbb{A}) = \mathsf{val}(\mathbb{B})$?

Algorithms on SLP-compressed strings

The mother of all algorithms on SLP -compressed strings:

Plandowski 1994

The following problem can be solved in polynomial time:

INPUT: SLPs $\mathbb{A}, \mathbb{B}$
QUESTION: $\mathsf{val}(\mathbb{A}) = \mathsf{val}(\mathbb{B})$?

Note: The decompress-and-compare strategy does not work here.
We cannot compute $\mathsf{val}(\mathbb{A})$ and $\mathsf{val}(\mathbb{B})$ in polynomial time.

Algorithms on SLP-compressed strings

The mother of all algorithms on SLP -compressed strings:

Plandowski 1994

The following problem can be solved in polynomial time:

INPUT: SLPs $\mathbb{A}, \mathbb{B}$
QUESTION: $\mathsf{val}(\mathbb{A}) = \mathsf{val}(\mathbb{B})$?

Note: The decompress-and-compare strategy does not work here.
We cannot compute $\mathsf{val}(\mathbb{A})$ and $\mathsf{val}(\mathbb{B})$ in polynomial time.

Plandowski's algorithm uses combinatorics on words, in particular the Periodicity Lemma of Fine and Wilf.

Improvements of Plandowski's result

Gasieniec, Karpinski, Miyazaki, Plandowski, Rytter, Shinohara, Takeda (mid 90's)

The following problem can be solved in polynomial time (fully compressed pattern matching):

INPUT: SLPs $\mathbb{P}, \mathbb{T}$

QUESTION: Is $\mathsf{val}(\mathbb{P})$ a factor of $\mathsf{val}(\mathbb{T})$, i.e., $\exists u, v : \mathsf{val}(\mathbb{T}) = u \, \mathsf{val}(\mathbb{P}) \, v$?

Improvements of Plandowski's result

Gasieniec, Karpinski, Miyazaki, Plandowski, Rytter, Shinohara, Takeda (mid 90's)

The following problem can be solved in polynomial time (fully compressed pattern matching):

INPUT: SLPs $\mathbb{P}, \mathbb{T}$

QUESTION: Is $\mathrm{val}(\mathbb{P})$ a factor of $\mathrm{val}(\mathbb{T})$, i.e., $\exists u, v : \mathrm{val}(\mathbb{T}) = u \,\mathrm{val}(\mathbb{P})\, v$?

The best known algorithm has a running time of $\mathcal{O}(|\mathbb{P}| \cdot |\mathbb{T}|^2)$ (Lifshits 2006).

Parsing compressed strings: finite automata

Plandowski, Rytter 1999

The following problem can be solved in polynomial time:

INPUT: A nondeterministic automaton A and an SLP $\mathbb{B}$.

QUESTION: Does A accept $\mathrm{val}(\mathbb{B})$?

Parsing compressed strings: finite automata

Plandowski, Rytter 1999

The following problem can be solved in polynomial time:

INPUT: A nondeterministic automaton A and an SLP $\mathbb{B}$.

QUESTION: Does A accept $\mathrm{val}(\mathbb{B})$?

The precise time bound is $O(v \cdot s^3)$, where:

- $v = |\mathbb{B}|$, and
- s is the number of states of A.

Parsing compressed strings: finite automata

Plandowski, Rytter 1999

The following problem can be solved in polynomial time:

INPUT: A nondeterministic automaton A and an SLP $\mathbb{B}$.

QUESTION: Does A accept $\text{val}(\mathbb{B})$?

The precise time bound is $O(v \cdot s^3)$, where:

- $v = |\mathbb{B}|$, and
- s is the number of states of A.

Proof:

Parsing compressed strings: finite automata

Plandowski, Rytter 1999

The following problem can be solved in polynomial time:

INPUT: A nondeterministic automaton A and an SLP $\mathbb{B}$.

QUESTION: Does A accept $\mathrm{val}(\mathbb{B})$?

The precise time bound is $O(v \cdot s^3)$, where:

- $v = |\mathbb{B}|$, and
- s is the number of states of A.

Proof:

The automaton A can be represented by Boolean matrices $A_a \in \{0,1\}^{s \times s}$ for each input letter a.

Parsing compressed strings: finite automata

Plandowski, Rytter 1999

The following problem can be solved in polynomial time:

INPUT: A nondeterministic automaton A and an SLP $\mathbb{B}$.

QUESTION: Does A accept val($\mathbb{B}$)?

The precise time bound is $O(v \cdot s^3)$, where:

- $v = |\mathbb{B}|$, and
- s is the number of states of A.

Proof:

The automaton A can be represented by Boolean matrices $A_a \in \{0,1\}^{s \times s}$ for each input letter a.

Evaluate SLP $\mathbb{B}$ over $\{0,1\}^{s \times s}$.

Parsing compressed strings: finite automata

Plandowski, Rytter 1999

The following problem can be solved in polynomial time:

INPUT: A nondeterministic automaton A and an SLP $\mathbb{B}$.

QUESTION: Does A accept $\mathrm{val}(\mathbb{B})$?

The precise time bound is $O(v \cdot s^3)$, where:

- $v = |\mathbb{B}|$, and
- s is the number of states of A.

Proof:

The automaton A can be represented by Boolean matrices $A_a \in \{0,1\}^{s \times s}$ for each input letter a.

Evaluate SLP $\mathbb{B}$ over $\{0,1\}^{s \times s}$.

$\rightsquigarrow$ v multiplications in $\{0,1\}^{s \times s}$.

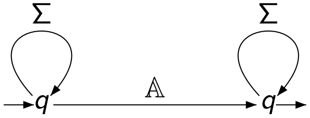
Parsing compressed strings: compressed finite automata

A compressed automata is an ordinary finite automaton, where every transition is labelled with an SLP.

Parsing compressed strings: compressed finite automata

A compressed automata is an ordinary finite automaton, where every transition is labelled with an SLP.

Example: The compressed automaton

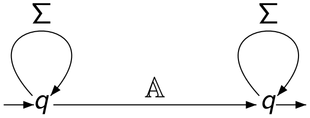


accepts all words that have $\mathrm{val}(\mathbb{A})$ as a factor.

Parsing compressed strings: compressed finite automata

A compressed automata is an ordinary finite automaton, where every transition is labelled with an SLP.

Example: The compressed automaton



accepts all words that have $\mathrm{val}(\mathbb{A})$ as a factor.

The size $|\mathcal{A}|$ of the compressed automaton $\mathcal{A}$ is

$$|\mathcal{A}| = \sum_{p \xrightarrow{\mathbb{A}} q} |\mathbb{A}|.$$

Parsing compressed strings: compressed finite automata

Compressed membership for compressed automata:

INPUT: A compressed automaton $\mathcal{A}$ and an SLP $\mathbb{B}$.

QUESTION: $\mathsf{val}(\mathbb{B}) \in L(\mathcal{A})$?

Parsing compressed strings: compressed finite automata

Compressed membership for compressed automata:

INPUT: A compressed automaton $\mathcal{A}$ and an SLP $\mathbb{B}$.

QUESTION: $\mathsf{val}(\mathbb{B}) \in L(\mathcal{A})$?

Plandowski, Rytter 1999

- Compressed membership for compressed automata is in PSPACE.
- Compressed membership for compressed automata over a unary alphabet is NP-complete.

Parsing compressed strings: compressed finite automata

Conjecture (Plandowski, Rytter 1999)

- Compressed membership for compressed automata is NP-complete (for every alphabet size).
- Compressed membership for compressed deterministic automata belongs to P.

  A compressed automaton $\mathcal{A}$ is deterministic, if for all transitions $p \xrightarrow{\mathbb{A}} q$, $p \xrightarrow{\mathbb{B}} r$ that start in the same state p, neither $\mathrm{val}(\mathbb{A})$ is a prefix of $\mathrm{val}(\mathbb{B})$ nor $\mathrm{val}(\mathbb{B})$ is a prefix of $\mathrm{val}(\mathbb{A})$.

Parsing compressed strings: pushdown automata

Theorem (L 2010)

The following problem is PSPACE-complete:

INPUT: A pushdown automaton A and an SLP $\mathbb{B}$.

QUESTION: Does A accept $\mathrm{val}(\mathbb{B})$?

Parsing compressed strings: pushdown automata

Theorem (L 2010)

The following problem is PSPACE-complete:

INPUT: A pushdown automaton A and an SLP $\mathbb{B}$.

QUESTION: Does A accept $\mathrm{val}(\mathbb{B})$?

PSPACE-hardness holds already for the special case that A is a fixed deterministic pushdown automaton.

Parsing compressed strings: pushdown automata

Theorem (L 2010)

The following problem is PSPACE-complete:

INPUT: A pushdown automaton A and an SLP $\mathbb{B}$.

QUESTION: Does A accept val($\mathbb{B}$)?

PSPACE-hardness holds already for the special case that A is a fixed deterministic pushdown automaton.

The proof uses a characterization of PSPACE based on leaf languages (Hertrampf, Lautemann, Schwentick, Vollmer, Wagner; 1993).

Other hard problems for compressed strings

A string $a_1 a_2 \cdots a_m$ is a subsequence of a string $b_1 b_2 \cdots b_n$ if there exist $i_1 < i_2 < \cdots < i_m$ with $a_1 = b_{i_1}$, $a_2 = b_{i_2}, \ldots, a_m = b_{i_m}$.

Other hard problems for compressed strings

A string $a_1 a_2 \cdots a_m$ is a subsequence of a string $b_1 b_2 \cdots b_n$ if there exist $i_1 < i_2 < \cdots < i_m$ with $a_1 = b_{i_1}$, $a_2 = b_{i_2}, \ldots, a_m = b_{i_m}$.

The following problems are hard for the complexity class PP (probabilistic polynomial time) (and belong to PSPACE).

Other hard problems for compressed strings

A string $a_1 a_2 \cdots a_m$ is a subsequence of a string $b_1 b_2 \cdots b_n$ if there exist $i_1 < i_2 < \cdots < i_m$ with $a_1 = b_{i_1}$, $a_2 = b_{i_2}, \ldots, a_m = b_{i_m}$.

The following problems are hard for the complexity class PP (probabilistic polynomial time) (and belong to PSPACE).

- Given SLPs $\mathbb{A}$, $\mathbb{B}$, is $\mathrm{val}(\mathbb{A})$ a subsequence of $\mathrm{val}(\mathbb{B})$?

Other hard problems for compressed strings

A string $a_1 a_2 \cdots a_m$ is a subsequence of a string $b_1 b_2 \cdots b_n$ if there exist $i_1 < i_2 < \cdots < i_m$ with $a_1 = b_{i_1}$, $a_2 = b_{i_2}, \ldots, a_m = b_{i_m}$.

The following problems are hard for the complexity class PP (probabilistic polynomial time) (and belong to PSPACE).

- Given SLPs $\mathbb{A}$, $\mathbb{B}$, is $\mathrm{val}(\mathbb{A})$ a subsequence of $\mathrm{val}(\mathbb{B})$?
- Given SLPs $\mathbb{A}$, $\mathbb{B}$ and $n \in \mathbb{N}$, do $\mathrm{val}(\mathbb{A})$ and $\mathrm{val}(\mathbb{B})$ have a common subsequence of length at least n?

Other hard problems for compressed strings

A string $a_1 a_2 \cdots a_m$ is a subsequence of a string $b_1 b_2 \cdots b_n$ if there exist $i_1 < i_2 < \cdots < i_m$ with $a_1 = b_{i_1}$, $a_2 = b_{i_2}, \ldots, a_m = b_{i_m}$.

The following problems are hard for the complexity class PP (probabilistic polynomial time) (and belong to PSPACE).

- Given SLPs $\mathbb{A}$, $\mathbb{B}$, is $\mathrm{val}(\mathbb{A})$ a subsequence of $\mathrm{val}(\mathbb{B})$?
- Given SLPs $\mathbb{A}$, $\mathbb{B}$ and $n \in \mathbb{N}$, do $\mathrm{val}(\mathbb{A})$ and $\mathrm{val}(\mathbb{B})$ have a common subsequence of length at least n?
- Given SLPs $\mathbb{A}$, $\mathbb{B}$ and $n \in \mathbb{N}$, are $\mathrm{val}(\mathbb{A})$ and $\mathrm{val}(\mathbb{B})$ subsequences of a string of length at most n?

Other hard problems for compressed strings

A string $a_1 a_2 \cdots a_m$ is a subsequence of a string $b_1 b_2 \cdots b_n$ if there exist $i_1 < i_2 < \cdots < i_m$ with $a_1 = b_{i_1}, a_2 = b_{i_2}, \ldots, a_m = b_{i_m}$.

The following problems are hard for the complexity class PP (probabilistic polynomial time) (and belong to PSPACE).

- Given SLPs $\mathbb{A}$, $\mathbb{B}$, is val($\mathbb{A}$) a subsequence of val($\mathbb{B}$)?
- Given SLPs $\mathbb{A}$, $\mathbb{B}$ and $n \in \mathbb{N}$, do val($\mathbb{A}$) and val($\mathbb{B}$) have a common subsequence of length at least n?
- Given SLPs $\mathbb{A}$, $\mathbb{B}$ and $n \in \mathbb{N}$, are val($\mathbb{A}$) and val($\mathbb{B}$) subsequences of a string of length at most n?

PP is the class of all problems A for which there exists a probabilitstic polynomial time machine M such that

$$\forall x : x \in A \iff \text{Prob}[M \text{ accepts } x] > 1/2$$

Toda 1991: P^{PP} contains the polynomial time hierarchy.

Application in computational group theory

Application in computational group theory

Let G be a group, finitely generated by A.

Application in computational group theory

Let G be a group, finitely generated by A.

The compressed word problem for G is the following problem:

INPUT: SLP $\mathbb{A}$ over $A \cup A^{-1}$.

QUESTION: $\mathrm{val}(\mathbb{A}) = 1$ in G?

Application in computational group theory

Let G be a group, finitely generated by A.

The compressed word problem for G is the following problem:

INPUT: SLP $\mathbb{A}$ over $A \cup A^{-1}$.

QUESTION: $\mathrm{val}(\mathbb{A}) = 1$ in G?

Why is the compressed word problem interesting in group theory?

Application in computational group theory

Let G be a group, finitely generated by A.

The compressed word problem for G is the following problem:

INPUT: SLP $\mathbb{A}$ over $A \cup A^{-1}$.

QUESTION: $\mathrm{val}(\mathbb{A}) = 1$ in G?

Why is the compressed word problem interesting in group theory?

Observation

Assume that the compressed word problem for G can be solved in polynomial time.

Application in computational group theory

Let G be a group, finitely generated by A.

The compressed word problem for G is the following problem:

INPUT: SLP $\mathbb{A}$ over $A \cup A^{-1}$.

QUESTION: $\mathrm{val}(\mathbb{A}) = 1$ in G?

Why is the compressed word problem interesting in group theory?

Observation

Assume that the compressed word problem for G can be solved in polynomial time.

Then, for every finitely generated subgroup of $\mathrm{Aut}(G)$ the (standard) word problem can be solved in polynomial time.

Application in computational group theory

Classes of groups, where CWP can be solved in polynomial time:

- graph groups (in particular free groups)

Application in computational group theory

Classes of groups, where CWP can be solved in polynomial time:

- graph groups (in particular free groups)
- nilpotent groups

Application in computational group theory

Classes of groups, where CWP can be solved in polynomial time:

- graph groups (in particular free groups)
- nilpotent groups
- fully residually-free groups (Macdonald 2010)

Application in computational group theory

Classes of groups, where CWP can be solved in polynomial time:

- graph groups (in particular free groups)
- nilpotent groups
- fully residually-free groups (Macdonald 2010)
- Coxeter groups

Application in computational group theory

Classes of groups, where CWP can be solved in polynomial time:

- graph groups (in particular free groups)
- nilpotent groups
- fully residually-free groups (Macdonald 2010)
- Coxeter groups

Closure properties of the class of groups with polynomial time CWP:

Application in computational group theory

Classes of groups, where CWP can be solved in polynomial time:

- graph groups (in particular free groups)
- nilpotent groups
- fully residually-free groups (Macdonald 2010)
- Coxeter groups

Closure properties of the class of groups with polynomial time CWP:

- going to f.g. subgroups

Application in computational group theory

Classes of groups, where CWP can be solved in polynomial time:

- graph groups (in particular free groups)
- nilpotent groups
- fully residually-free groups (Macdonald 2010)
- Coxeter groups

Closure properties of the class of groups with polynomial time CWP:

- going to f.g. subgroups
- finite extensions

Application in computational group theory

Classes of groups, where CWP can be solved in polynomial time:

- graph groups (in particular free groups)
- nilpotent groups
- fully residually-free groups (Macdonald 2010)
- Coxeter groups

Closure properties of the class of groups with polynomial time CWP:

- going to f.g. subgroups
- finite extensions
- graph products (in particular free and direct products)

Application in computational group theory

Classes of groups, where CWP can be solved in polynomial time:

- graph groups (in particular free groups)
- nilpotent groups
- fully residually-free groups (Macdonald 2010)
- Coxeter groups

Closure properties of the class of groups with polynomial time CWP:

- going to f.g. subgroups
- finite extensions
- graph products (in particular free and direct products)
- HNN-extensions and amalgamated free products over finite groups

Application in computational group theory

Theorem

Assume that $K \lhd G$, $Q = G/K$ (with K, Q, G finitely generated).

Application in computational group theory

Theorem

Assume that $K \lhd G$, $Q = G/K$ (with K, Q, G finitely generated).
Moreover, assume that:

- CWP for K can be solved in polynomial time.

Application in computational group theory

Theorem

Assume that $K \lhd G$, $Q = G/K$ (with K, Q, G finitely generated).
Moreover, assume that:

- CWP for K can be solved in polynomial time.
- Q has polynomial Dehn function.

Application in computational group theory

Theorem

Assume that $K \lhd G$, $Q = G/K$ (with K, Q, G finitely generated).
Moreover, assume that:

- CWP for K can be solved in polynomial time.
- Q has polynomial Dehn function.
- The word search problem for Q can be solved in polynomial time.

Application in computational group theory

Theorem

Assume that $K \lhd G$, $Q = G/K$ (with K, Q, G finitely generated).
Moreover, assume that:

- CWP for K can be solved in polynomial time.
- Q has polynomial Dehn function.
- The word search problem for Q can be solved in polynomial time.

Then, the (standard) word problem for G can be solved in polynomial time.

Application in computational group theory

Theorem

Assume that $K \lhd G$, $Q = G/K$ (with K, Q, G finitely generated). Moreover, assume that:

- CWP for K can be solved in polynomial time.
- Q has polynomial Dehn function.
- The word search problem for Q can be solved in polynomial time.

Then, the (standard) word problem for G can be solved in polynomial time.

Classes of groups with (i) polynomial Dehn function and (ii) polynomial time word search problem:

Application in computational group theory

Theorem

Assume that $K \lhd G$, $Q = G/K$ (with K, Q, G finitely generated).
Moreover, assume that:

- CWP for K can be solved in polynomial time.
- Q has polynomial Dehn function.
- The word search problem for Q can be solved in polynomial time.

Then, the (standard) word problem for G can be solved in polynomial time.

Classes of groups with (i) polynomial Dehn function and (ii) polynomial time word search problem:

- automatic groups

Application in computational group theory

Theorem

Assume that $K \triangleleft G$, $Q = G/K$ (with K, Q, G finitely generated). Moreover, assume that:

- CWP for K can be solved in polynomial time.
- Q has polynomial Dehn function.
- The word search problem for Q can be solved in polynomial time.

Then, the (standard) word problem for G can be solved in polynomial time.

Classes of groups with (i) polynomial Dehn function and (ii) polynomial time word search problem:

- automatic groups
- nilpotent groups

Open problems

- Is the following problem PSPACE-complete:
  Given SLPs $\mathbb{A}$, $\mathbb{B}$, is $\mathrm{val}(\mathbb{A})$ a subsequence of $\mathrm{val}(\mathbb{B})$?

Open problems

- Is the following problem PSPACE-complete:
  Given SLPs $\mathbb{A}$, $\mathbb{B}$, is $\mathsf{val}(\mathbb{A})$ a subsequence of $\mathsf{val}(\mathbb{B})$?
- Compressed word problem for finitely generated linear groups

Open problems

- Is the following problem PSPACE-complete:
  Given SLPs $\mathbb{A}$, $\mathbb{B}$, is $\mathsf{val}(\mathbb{A})$ a subsequence of $\mathsf{val}(\mathbb{B})$?

- Compressed word problem for finitely generated linear groups

  The standard word problem for a f.g. linear groups can be solved in deterministic logspace (and hence polynomial time).

Open problems

- Is the following problem PSPACE-complete:
  Given SLPs $\mathbb{A}$, $\mathbb{B}$, is $\mathsf{val}(\mathbb{A})$ a subsequence of $\mathsf{val}(\mathbb{B})$?

- Compressed word problem for finitely generated linear groups

  The standard word problem for a f.g. linear groups can be solved in deterministic logspace (and hence polynomial time).

  The CWP for a f.g. linear group belongs to coRP, i.e., the complementary problem can be solved in randomized polynomial time.

Open problems

- Is the following problem PSPACE-complete:
  Given SLPs $\mathbb{A}$, $\mathbb{B}$, is val($\mathbb{A}$) a subsequence of val($\mathbb{B}$)?

- Compressed word problem for finitely generated linear groups

  The standard word problem for a f.g. linear groups can be solved in deterministic logspace (and hence polynomial time).

  The CWP for a f.g. linear group belongs to coRP, i.e., the complementary problem can be solved in randomized polynomial time.

  There is some evidence from complexity theory that RP$=$ coRP $=$ P.

Open problems

- Is the following problem PSPACE-complete:
  Given SLPs $\mathbb{A}$, $\mathbb{B}$, is $\mathsf{val}(\mathbb{A})$ a subsequence of $\mathsf{val}(\mathbb{B})$?
- Compressed word problem for finitely generated linear groups

  The standard word problem for a f.g. linear groups can be solved in deterministic logspace (and hence polynomial time).

  The CWP for a f.g. linear group belongs to coRP, i.e., the complementary problem can be solved in randomized polynomial time.

  There is some evidence from complexity theory that RP= coRP = P.
- Compressed word problem for braid groups, polycyclic groups, and finitely generated metabelian groups