Pushdown Compression

Elvira Mayordomo, Philippe Moser

January 24-30, 2009
Czech Republic

SOFSEM 2009
Contents

1. Introduction
   - Motivation

2. Introduction
   - Lempel Ziv
   - Pushdown Compression
   - Plogon compression

3. Our results
   - plogon beats Lempel Ziv
   - Lempel Ziv beats plogon
   - invPD beats plogon
   - plogon beats PD

4. Conclusion
Contents

1 Introduction
   • Motivation

2 Introduction
   • Lempel Ziv
   • Pushdown Compression
   • Plogon compression

3 Our results
   • plogon beats Lempel Ziv
   • Lempel Ziv beats plogon
   • invPD beats plogon
   • plogon beats PD

4 Conclusion
Lempel-Ziv compression algorithm (gzip)

- Widely used, fast, lossless online compression algorithm.
- Is finite-state universal (compresses as well than any finite-state compressor).
Motivation

Lempel-Ziv compression algorithm (gzip)

- Widely used, fast, lossless online compression algorithm.
- Is finite-state universal (compresses as well than any finite-state compressor).
Pushdown Compression

Introduction

Motivation

Pushdown compressors

- Used as a compression scheme for XML.
- Is finite-state universal (compresses as well than any finite-state compressor).
Pushdown compressors

- Used as a compression scheme for XML.
- Is finite-state universal (compresses as well than any finite-state compressor).
Plogon compressors

Plogon = Polylog space, online
- Models compression in the data stream setting.
- Small memory, read large stream of data.
Plogon compressors

Plogon = Polylog space, online
- Models compression in the data stream setting.
- Small memory, read large stream of data.
Previous Work (STACS08)

Pushdown compressors and Lempel-Ziv are incomparable.
Our result

How does plogon compression compare to PD compression and LZ compression?

We show that they are all incomparable.
Our result

How does plogon compression compare to PD compression and LZ compression?

We show that they are all incomparable.
Motivation

Compression algorithm yield a dimension notion on complexity classes

- Notion of dimension of a complexity class.
- Property: $\dim(C) = \sup_{A \in C} \dim(A)$.
  $\rightarrow$ study of the dimension of individual languages (= infinite sequences).
- If $S \in \{0, 1\}^\mathbb{N}$ then
  \[
  \dim(S) = \limsup_{n \to \infty} \frac{K(S[1..n])}{n}.
  \]

- Resource bounds $\rightarrow$ effectivity, compression algorithms.

Here: compression by pushdown automata for the study of small classes.
Motivation

Compression algorithm yield a dimension notion on complexity classes

- Notion of dimension of a complexity class.
- Property: \( \dim(C) = \sup_{A \in C} \dim(A) \).
  → study of the dimension of individual languages (= infinite sequences).
- If \( S \in \{0, 1\}^\mathbb{N} \) then
  \[
  \dim(S) = \limsup_{n \to \infty} \frac{K(S[1..n])}{n}.
  \]

- Resource bounds → effectivity, compression algorithms.

Here: compression by pushdown automata for the study of small classes.
Compress algorithm yield a dimension notion on complexity classes

- Notion of dimension of a complexity class.
- Property: $\dim(C) = \sup_{A \in C} \dim(A)$.
  → study of the dimension of individual languages (= infinite sequences).
- If $S \in \{0, 1\}^\mathbb{N}$ then
  $$\dim(S) = \limsup_{n \to \infty} \frac{K(S[1..n])}{n}.$$ 

- Resource bounds → effectivity, compression algorithms.

Here: compression by pushdown automata for the study of small classes.
Lossless compression.

Compressor: injective and computable function
\( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \).

Compression ratio on a finite word \( x \):
\[
\rho_f(x) = \frac{|f(x)|}{|x|}.
\]

Compression ratio on an infinite sequence \( S \):
\[
\rho_f(S) = \lim_{n \to \infty} \rho_f(S[1..n]).
\]
Compression

- **Lossless compression.**

- **Compressor:** injective and computable function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$.

Compression ratio on a finite word $x$:

$$\rho_f(x) = \frac{|f(x)|}{|x|}.$$ 

Compression ratio on an infinite sequence $S$:

$$\rho_f(S) = \lim_{n \rightarrow \infty} \rho_f(S[1..n]).$$
Compression

- Lossless compression.

- Compressor: injective and computable function $f : \{0, 1\}^* \to \{0, 1\}^*$.

- Compression ratio on a finite word $x$:

$$\rho_f(x) = \frac{|f(x)|}{|x|}.$$ 

Compression ratio on an infinite sequence $S$:

$$\rho_f(S) = \lim_{n \to \infty} \rho_f(S[1..n]).$$
Contents

1 Introduction
   • Motivation

2 Introduction
   • Lempel Ziv
     • Pushdown Compression
     • Plogon compression

3 Our results
   • plogon beats Lempel Ziv
   • Lempel Ziv beats plogon
   • invPD beats plogon
   • plogon beats PD

4 Conclusion
Example

Sequence to be compressed:

0 1 2 3 4 5 6 7 8 9 10
\epsilon 0 1 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 0 0 0 1 1 1

Compressed sequence:

\epsilon; (0, 0); (0, 1); (1, 0); (1, 1); (4, 1); (2, 0); (6, 0); (7, 0); (3, 1); (2, 1)
Example

Sequence to be compressed:

\[012345678910\]
\[\epsilon/010010110100100000111\]

Compressed sequence:

\[\epsilon; (0, 0); (0, 1); (1, 0); (1, 1); (4, 1); (2, 0); (6, 0); (7, 0); (3, 1); (2, 1)\]
Introduction

Lempel Ziv

Example

Sequence to be compressed:

\[ \epsilon; (0, 0); (0, 1); (1, 0); (1, 1); (4, 1); (2, 0); (6, 0); (7, 0); (3, 1); (2, 1) \]
Pushdown Compression

Introduction

Lempel Ziv

Example

Sequence to be compressed:

\[ \text{\epsilon}; (0, 0); (0, 1); (1, 0); (1, 1); (4, 1); (2, 0); (6, 0); (7, 0); (3, 1); (2, 1) \]
Sequence to be compressed:

\[ \begin{align*}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\

\epsilon & /0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & /1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{align*} \]

Compressed sequence:

\[ \epsilon; (0, 0); (0, 1); (1, 0); (1, 1); (4, 1); (2, 0); (6, 0); (7, 0); (3, 1); (2, 1) \]
Example

Sequence to be compressed:

\[01 2 3 4 5 6 7 8 9 10\]
\[\epsilon/0/1/0 0/0 1/0 1 1/1 0/1 0 0 1 0 0 0/0 0 1 1 1\]

Compressed sequence:

\[\epsilon; (0, 0); (0, 1); (1, 0); (1, 1); (4, 1); (2, 0); (6, 0); (7, 0); (3, 1); (2, 1)\]
Example

Sequence to be compressed:

$\epsilon; 0 1 2 3 4 5 6 7 8 9 10$

$\epsilon; 0/1 0 0 1 0 1 1 1 0/1 0 0 1 0 0 0 0 0 1 1 1$

Compressed sequence:

$\epsilon; (0, 0); (0, 1); (1, 0); (1, 1); (4, 1); (2, 0); (6, 0); (7, 0); (3, 1); (2, 1)$
Example

Sequence to be compressed:

\[01\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\]
\[\epsilon/0/1/0\ 0/0\ 1/0\ 1/1\ 0/1\ 0\ 0/0/0\ 0\ 1/1\ 1\]

Compressed sequence:

\[\epsilon;(0, 0);(0, 1); (1, 0);(1, 1);(4, 1); (2, 0);(6, 0);(7, 0);(3, 1);(2, 1)\]
Description of the $LZ$ algorithm

Formal description of the Lempel Ziv algorithm:

- Input $x \in \Sigma^*$

- $LZ$ parses $x$ into phrases $x = x_1 x_2 \ldots x_n$, $x_i \in \Sigma^*$, $i = 1, \ldots, n$

- $\forall \ y \sqsubseteq x_i, \exists j < i$ such that $y = x_j$

- so, for every $i = 1, \ldots, n$, $x_i = x_{l(i)} b_i$, with $l(i) < i$ and $b_i \in \Sigma$. 
Description of the **LZ** algorithm

Formal description of the Lempel Ziv algorithm:

- **Input** $x \in \Sigma^*$

- **LZ** parses $x$ into phrases $x = x_1 x_2 \ldots x_n$, $x_i \in \Sigma^*, i = 1, \ldots, n$

- $\forall y \sqsubset x_i, \exists j < i$ such that $y = x_j$

- so, for every $i = 1, \ldots, n$, $x_i = x_{l(i)} b_i$, with $l(i) < i$ and $b_i \in \Sigma$. 
Formal description of the Lempel Ziv algorithm:

- **Input** $x \in \Sigma^*$

- **LZ** parses $x$ into phrases $x = x_1x_2 \ldots x_n$, $x_i \in \Sigma^*, i = 1, \ldots, n$

- $\forall y \sqsubseteq x_i, \exists j < i$ such that $y = x_j$

- so, for every $i = 1, \ldots, n$, $x_i = x_{l(i)}b_i$, with $l(i) < i$ and $b_i \in \Sigma$. 


Description of the *LZ* algorithm

Formal description of the Lempel Ziv algorithm:

- Input $x \in \Sigma^*$
- *LZ* parses $x$ into phrases $x = x_1x_2\ldots x_n$, $x_i \in \Sigma^*, i = 1, \ldots, n$

\[ \forall y \sqsubset x_i, \ \exists j < i \text{ such that } y = x_j \]

so, for every $i = 1, \ldots, n$, $x_i = x_{l(i)}b_i$, with $l(i) < i$ and $b_i \in \Sigma$. 
Description of the \textit{LZ} algorithm

Formal description of the Lempel Ziv algorithm:

- Input $x \in \Sigma^*$
- \textit{LZ} parses $x$ into phrases $x = x_1 x_2 \ldots x_n$, $x_i \in \Sigma^*$, $i = 1, \ldots, n$
- $\forall y \sqsubseteq x_i, \ \exists j < i$ such that $y = x_j$
- so, for every $i = 1, \ldots, n$, $x_i = x_{l(i)} b_i$, with $l(i) < i$ and $b_i \in \Sigma$. 

Description of the $LZ$ algorithm

Then (remember: $x_i = x_{l(i)} b_i$):

$x_i$ is encoded by a prefix free encoding of $l(i)$ and the symbol $b_i$, that is

$LZ(x) = c_{l(1)} b_1 c_{l(2)} b_2 \ldots c_{l(n)} b_n$
Then (remember: $x_i = x_{l(i)} b_i$):

$x_i$ is encoded by a prefix free encoding of $l(i)$ and the symbol $b_i$, that is

$$LZ(x) = c_{l(1)} b_1 c_{l(2)} b_2 \ldots c_{l(n)} b_n$$
LZ is universal for Finite-State

Theorem (Lempel,Ziv)

On every infinite sequence $S \in \{0, 1\}^\mathbb{N}$, Lempel-Ziv does better than any finite-state compressor, that is,

$$\rho_{LZ}(S) \leq \rho_{FS}(S).$$

This universality is no longer true for the natural generalization from FS to PD compressors.
LZ is universal for Finite-State

Theorem (Lempel,Ziv)

On every infinite sequence $S \in \{0, 1\}^\mathbb{N}$, Lempel-Ziv does better than any finite-state compressor, that is,

$$\rho_{LZ}(S) \leq \rho_{FS}(S).$$

This universality is no longer true for the natural generalization from FS to PD compressors.
Contents

1 Introduction
   - Motivation

2 Introduction
   - Lempel Ziv
   - Pushdown Compression
   - Plogon compression

3 Our results
   - plogon beats Lempel Ziv
   - Lempel Ziv beats plogon
   - invPD beats plogon
   - plogon beats PD

4 Conclusion
Finite-state transducer: finite-state automaton that outputs symbols at each transition.

Output function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$. 

![Finite-state transducer diagram]
Finite-state transducer: finite-state automaton that outputs symbols at each transition.

Output function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$. 
Finite-state transducer: finite-state automaton that outputs symbols at each transition.

Output function \( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \).
Finite-state transducer: finite-state automaton that outputs symbols at each transition.

Output function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$.

For example:

0000011100 $\rightarrow$ ?
Finite-state transducer: finite-state automaton that outputs symbols at each transition.

Output function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$. 

```
0 \epsilon
1 01
0 0
1 011
```
**Finite-state transducer**: finite-state automaton that outputs symbols at each transition.

Output function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$. 

```
0 / \epsilon
0 / 0
1 / 011
0 / 0
1 / 01
```

*0000011100* $\rightarrow$ *0*
**Finite-state transducer:** finite-state automaton that outputs symbols at each transition.

Output function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$. 

![Finite-state transducer diagram]

$0000011100 \rightarrow 0$
Finite-state transducer: finite-state automaton that outputs symbols at each transition.

Output function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$. 

\[
\begin{array}{c}
0 \\
1 / 011 \\
0 / 0 \\
1 / 01 \\
0 / \epsilon
\end{array}
\]

0000011100 $\rightarrow$ 00
**Finite-state transducer**: finite-state automaton that outputs symbols at each transition.

Output function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$. 

$$
0 / 0
1 / 011
0 / 0
0 / \epsilon
1 / 01
$$

$$
0000011100 \rightarrow 00
$$
**Finite-state transducer**: finite-state automaton that outputs symbols at each transition.

Output function \( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \).

\[
\begin{align*}
0 & / \epsilon \\
1 & / 011 \\
0 & / 0 \\
1 & / 01 \\
0000011100 & \rightarrow & 0001
\end{align*}
\]
Finite-state transducer: finite-state automaton that outputs symbols at each transition.

Output function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$. 

$$0 \not\in \epsilon$$

$$1 \in 01$$

$$0 \in 0$$

$$1 \in 01$$

$$0000011100 \rightarrow 0001011$$
Finite-state transducer: finite-state automaton that outputs symbols at each transition.

Output function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$.

```
0000011100    →    00010111011
```
**Finite-state transducer**: finite-state automaton that outputs symbols at each transition.

Output function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$.

\[
\begin{align*}
0 &\rightarrow \epsilon \\
1 &\rightarrow 011 \\
0 &\rightarrow 0 \\
1 &\rightarrow 01
\end{align*}
\]

\[
\begin{align*}
0000011100 &\rightarrow 0001011011
\end{align*}
\]
Finite-state transducer: finite-state automaton that outputs symbols at each transition.

Output function \( f : \{0, 1\}^* \rightarrow \{0, 1\}^*. \)

0000011100 → 00010110110110
**Finite-state transducer**: finite-state automaton that outputs symbols at each transition.

Output function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$.

![Finite-state automaton diagram]

**Finite-state compressor**: injective finite-state transducer (given the final state)
**Finite-state transducer:** finite-state automaton that outputs symbols at each transition.

Output function \( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \).

Finite-state compressor: \( x \leftrightarrow (f(x), q) \)
Pushdown compressors

- **Pushdown compressor**: finite-state compressor equipped with a stack.
  - The transition is done according both to the symbol read and to the topmost symbol of the stack.
  - Each transition either pushes or pops symbols from the stack.
  - $\lambda$-rules only can pop one symbol of the top of the stack.
  - For feasibility: the PD compressor is required to be invertible by a PD transducer.
Pushdown compressors

- **Pushdown compressor**: finite-state compressor equipped with a stack.
- The transition is done according both to the symbol read and to the topmost symbol of the stack.
- Each transition either pushes or pops symbols from the stack.
- $\lambda$-rules only can pop one symbol of the top of the stack.
- For feasibility: the PD compressor is required to be invertible by a PD transducer.
Pushdown compressors

- **Pushdown compressor**: finite-state compressor equipped with a stack.
- The transition is done according both to the symbol read and to the topmost symbol of the stack.
- Each transition either pushes or pops symbols from the stack.
  - $\lambda$-rules only can pop one symbol of the top of the stack.
  - For feasibility: the PD compressor is required to be invertible by a PD transducer.
Pushdown compressors

- **Pushdown compressor**: finite-state compressor equipped with a stack.
- The transition is done according both to the symbol read and to the topmost symbol of the stack.
- Each transition either pushes or pops symbols from the stack.
- $\lambda$-rules only can pop one symbol of the top of the stack.
- For feasibility: the PD compressor is required to be invertible by a PD transducer.
Pushdown compressors

- **Pushdown compressor**: finite-state compressor equipped with a stack.
- The transition is done according both to the symbol read and to the topmost symbol of the stack.
- Each transition either pushes or pops symbols from the stack.
- $\lambda$-rules only can pop one symbol of the top of the stack.
- For feasibility: the PD compressor is required to be invertible by a PD transducer.
Invertible Pushdown compressors

**Definition**

\((C, D)\) is an invertible PD compressor if \(C\) is an ILPDC and \(D\) is a PD transducer s.t. \(D\) in input both \(C(w)\) and the final state, outputs \(w\).
Contents

1 Introduction
   • Motivation

2 Introduction
   • Lempel Ziv
   • Pushdown Compression
   • Plogon compression

3 Our results
   • plogon beats Lempel Ziv
   • Lempel Ziv beats plogon
   • invPD beats plogon
   • plogon beats PD

4 Conclusion
A Turing machine $M$ is a plogon transducer if it has the following properties, for each input string $w$:

- The computation of $M(w)$ reads its input from left to right (no turning back),
- $M(w)$ is given $|w|$ written in binary (on a special tape),
- $M(w)$ writes the output from left to right on a write-only output tape,
- $M(w)$ uses memory bounded by $\log(|w|)^c$, for a constant $c$. 
Plogon compressors

Definition

A Turing machine \( M \) is a plogon transducer if it has the following properties, for each input string \( w \)

- the computation of \( M(w) \) reads its input from left to right (no turning back),
- \( M(w) \) is given \(|w|\) written in binary (on a special tape),
- \( M(w) \) writes the output from left to right on a write-only output tape,
- \( M(w) \) uses memory bounded by \( \log(|w|)^c \), for a constant \( c \).
Definition

A Turing machine $M$ is a plogon transducer if it has the following properties, for each input string $w$

- the computation of $M(w)$ reads its input from left to right (no turning back),
- $M(w)$ is given $|w|$ written in binary (on a special tape),
- $M(w)$ writes the output from left to right on a write-only output tape,
- $M(w)$ uses memory bounded by $\log(|w|)^c$, for a constant $c$. 
Plogon compressors

Definition

A Turing machine $M$ is a plogon transducer if it has the following properties, for each input string $w$

- the computation of $M(w)$ reads its input from left to right (no turning back),
- $M(w)$ is given $|w|$ written in binary (on a special tape),
- $M(w)$ writes the output from left to right on a write-only output tape,
- $M(w)$ uses memory bounded by $\log(|w|)^c$, for a constant $c$. 
Definition

A plogon transducer $C : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is an information lossless compressor (ILpC) if it is 1-1.
Compression ratio

For a sequence $S$ and $C \in \{\text{PD, LZ, Plogon}\}$ the upper and lower compression ratios are given by

$$
\rho_T(S) = \lim_{n \to \infty} \inf \frac{|T(S[1 \ldots n])|}{n}, \quad \text{and}
$$

$$
R_T(S) = \lim_{n \to \infty} \sup \frac{|T(S[1 \ldots n])|}{n}.
$$

Given a sequence $S$ and a class of functions $\mathcal{T}$, the upper and lower compression ratios are given by

$$
\rho_{\mathcal{T}}(S) = \inf_{T \in \mathcal{T}} \rho_T(S), \quad \text{and}
$$

$$
R_{\mathcal{T}}(S) = \inf_{T \in \mathcal{T}} R_T(S).
$$
Compression ratio

For a sequence $S$ and $C \in \{\text{PD, LZ, Plogon}\}$ the upper and lower compression ratios are given by

$$\rho_T(S) = \liminf_{n \to \infty} \frac{|T(S[1\ldots n])|}{n}, \quad \text{and}$$

$$R_T(S) = \limsup_{n \to \infty} \frac{|T(S[1\ldots n])|}{n}.$$

Given a sequence $S$ and a class of functions $T$, the upper and lower compression ratios are given by

$$\rho_T(S) = \inf_{T \in \mathcal{T}} \rho_T(S), \quad \text{and}$$

$$R_T(S) = \inf_{T \in \mathcal{T}} R_T(S).$$
• $\rho_T(S)$ corresponds to the best-case performance of $T$-compressors on $S$
• $R_T(S)$ corresponds to the worst-case performance of $T$-compressors on $S$
- $\rho_T(S)$ corresponds to the best-case performance of $T$-compressors on $S$
- $R_T(S)$ corresponds to the worst-case performance of $T$-compressors on $S$
Example

Let $S = 0^\infty$.

- The compression ratio on $S$ of a finite-state compressor with $k$ states is $\geq 1/k$.
- The compression ratio on $S$ of a pushdown compressor with $k$ states is $\geq 1/k$.
- On $S = 0^\infty$, FS, PD, LZ all have upper and lower compression ratio 0.
Let $S = 0^\infty$.

- The compression ratio on $S$ of a finite-state compressor with $k$ states is $\geq 1/k$.
- The compression ratio on $S$ of a pushdown compressor with $k$ states is $\geq 1/k$.
- On $S = 0^\infty$, FS, PD, LZ all have upper and lower compression ratio 0.
Example

Let $S = 0^\infty$.

- The compression ratio on $S$ of a finite-state compressor with $k$ states is $\geq 1/k$.
- The compression ratio on $S$ of a pushdown compressor with $k$ states is $\geq 1/k$.
- On $S = 0^\infty$, FS, PD, LZ all have upper and lower compression ratio 0.
Our results

- **plogon compressors and Lempel-Ziv are incomparable:**
  - We construct a sequence that Lempel-Ziv compresses optimally but that no plogon transducer compresses at all.
  - Vice-versa: a sequence that plogon compresses but LZ fails to compress.
  - Optimal result: optimal compression is in liminf (almost all prefixes of the sequence are optimally compressible), fail to compress even in limsup (only finitely many prefixes of the sequence are compressible).
Our results

- plogon compressors and Lempel-Ziv are incomparable:
- We construct a sequence that Lempel-Ziv compresses optimally but that no plogon transducer compresses at all.
- Vice-versa: a sequence that plogon compresses but LZ fails to compress.
- Optimal result: optimal compression is in liminf (almost all prefixes of the sequence are optimally compressible), fail to compress even in limsup (only fintely many prefixes of the sequence are compressible).
Our results

- plogon compressors and Lempel-Ziv are incomparable:
- We construct a sequence that Lempel-Ziv compresses optimally but that no plogon transducer compresses at all.
- Vice-versa: a sequence that plogon compresses but LZ fails to compress.
- Optimal result: optimal compression is in liminf (almost all prefixes of the sequence are optimally compressible), fail to compress even in limsup (only finitely many prefixes of the sequence are compressible).
Our results

- plogon compressors and Lempel-Ziv are incomparable:
  - We construct a sequence that Lempel-Ziv compresses optimally but that no plogon transducer compresses at all.
  - Vice-versa: a sequence that plogon compresses but LZ fails to compress.
- Optimal result: optimal compression is in liminf (almost all prefixes of the sequence are optimally compressible), fail to compress even in limsup (only finitely many prefixes of the sequence are compressible).
Our results

- Plogon compressors and Lempel-Ziv are incomparable:
  - We construct a sequence that Lempel-Ziv compresses optimally but that no plogon transducer compresses at all.
  - Vice-versa: a sequence that plogon compresses but LZ fails to compress.

- Optimal result: optimal compression is in liminf (almost all prefixes of the sequence are optimally compressible), fail to compress even in limsup (only finitely many prefixes of the sequence are compressible).
Our results

- plogon compressors and Pushdown compressors are incomparable:
  - Optimal result: PD incompressibility holds even for the more general pushdown model (where the pushdown compressor need not be invertible by a pushdown transducer).
  - Optimality: PD compressibility holds even for the more restrictive pushdown model (where the pushdown compressor is required be invertible by a pushdown transducer).
  - The PD compressibility also holds for visibly PD compressors.
plogon compressors and Pushdown compressors are incomparable:

Optimal result: PD incompressibility holds even for the more general pushdown model (where the pushdown compressor need not be invertible by a pushdown transducer).

Optimality: PD compressibility holds even for the more restrictive pushdown model (where the pushdown compressor is required be invertible by a pushdown transducer).

The PD compressibility also holds for visibly PD compressors.
Our results

- Plogon compressors and Pushdown compressors are incomparable:

- Optimal result: PD incompressibility holds even for the more general pushdown model (where the pushdown compressor need not be invertible by a pushdown transducer).

- Optimality: PD compressibility holds even for the more restrictive pushdown model (where the pushdown compressor is required be invertible by a pushdown transducer).

- The PD compressibility also holds for visibly PD compressors.
Our results

- Plogon compressors and Pushdown compressors are incomparable:
- Optimal result: PD incompressibility holds even for the more general pushdown model (where the pushdown compressor need not be invertible by a pushdown transducer).
- Optimality: PD compressibility holds even for the more restrictive pushdown model (where the pushdown compressor is required be invertible by a pushdown transducer).
- The PD compressibility also holds for visibly PD compressors.
Contents

1 Introduction
   • Motivation

2 Introduction
   • Lempel Ziv
   • Pushdown Compression
   • Plogon compression

3 Our results
   • plogon beats Lempel Ziv
   • Lempel Ziv beats plogon
   • invPD beats plogon
   • plogon beats PD

4 Conclusion
Theorem

There exists a sequence $S$ such that

$$R_{plogon}(S) = 0 \quad \text{and} \quad \rho_{LZ}(S) = 1.$$  

Proof idea: use a Copeland-Erdős sequence on which Lempel-Ziv has maximal compression ratio, whereas with logspace each prefix of the sequence can be completely reconstructed from its length.
Our results

plogon beats Lempel Ziv

Theorem

There exists a sequence $S$ such that

$$R_{\text{plogon}}(S) = 0 \quad \text{and} \quad \rho_{\text{LZ}}(S) = 1.$$ 

Proof idea: use a Copeland-Erdös sequence on which Lempel-Ziv has maximal compression ratio, whereas with logspace each prefix of the sequence can be completely reconstructed from its length.
Pushdown Compression

Our results

Lempel Ziv beats plogon

Contents

1 Introduction
   • Motivation

2 Introduction
   • Lempel Ziv
   • Pushdown Compression
   • Plogon compression

3 Our results
   • plogon beats Lempel Ziv
   • Lempel Ziv beats plogon
   • invPD beats plogon
   • plogon beats PD

4 Conclusion
Our results

Lempel Ziv beats plogon

Lempel Ziv beats plogon

Theorem

There exists a sequence $S$ such that

$$R_{LZ}(S) = 0 \quad \text{and} \quad \rho_{\text{plogon}}(S) = 1.$$
Theorem

There exists a sequence $S$ such that

$$R_{LZ}(S) = 0 \quad \text{and} \quad \rho_{plogon}(S) = 1.$$  

Proof idea: based on repetition of Kolmogorov random strings.
Contents

1. Introduction
   - Motivation

2. Introduction
   - Lempel Ziv
   - Pushdown Compression
   - Plogon compression

3. Our results
   - plogon beats Lempel Ziv
   - Lempel Ziv beats plogon
   - invPD beats plogon
   - plogon beats PD

4. Conclusion
Pushdown Compression

Our results

invPD beats plogon

invPD beats plogon

Theorem

For each $\epsilon > 0$ there exists a sequence $S$ such that

$$R_{invPD}(S) \leq \frac{1}{2} \quad \text{and} \quad \rho_{plogon}(S) \geq 1 - \epsilon.$$

Proof idea: Pushdown compresses palindromes with ratio $\approx \frac{1}{2}$... but plogon not always.
invPD beats plogon

**Theorem**

*For each* $\epsilon > 0$ *there exists a sequence* $S$ *such that*

$$R_{\text{invPD}}(S) \leq 1/2 \quad \text{and} \quad \rho_{\text{plogon}}(S) \geq 1 - \epsilon.$$  

Proof idea: Pushdown compresses palindromes with ratio \(\sim 1/2\)… but plogon not always.
Contents

1 Introduction
   • Motivation

2 Introduction
   • Lempel Ziv
   • Pushdown Compression
   • Plogon compression

3 Our results
   • plogon beats Lempel Ziv
   • Lempel Ziv beats plogon
   • invPD beats plogon
   • plogon beats PD

4 Conclusion
Theorem

There exists a sequence $S$ such that

$$R_{\text{plogon}}(S) = 0 \quad \text{and} \quad \rho_{\text{PD}}(S) = 1.$$ 

Proof idea: Use a pumping idea; repeat (pump) a PD incompressible string many times.
Summary

- Study of 3 compression schemes:
  - Lempel-Ziv
  - Pushdown compressors (finite-state with stack)
  - plogon (polylog space online)
- Results: all 3 schemes are incomparable.
Summary

- Study of 3 compression schemes:
  - Lempel-Ziv
    - Pushdown compressors (finite-state with stack)
    - plogon (polylog space online)
  - Results: all 3 schemes are incomparable.
Summary

- Study of 3 compression schemes:
  - Lempel-Ziv
  - Pushdown compressors (finite-state with stack)
    - plogon (polylog space online)
  - Results: all 3 schemes are incomparable.
Summary

- Study of 3 compression schemes:
  - Lempel-Ziv
  - Pushdown compressors (finite-state with stack)
  - plogon (polylog space online)
- Results: all 3 schemes are incomparable.
Study of 3 compression schemes:
- Lempel-Ziv
- Pushdown compressors (finite-state with stack)
- plogon (polylog space online)
Results: all 3 schemes are incomparable.
Future work

- Better separation for “PD beats plogon” (from 1/2 to 0)?
- Is there a separation example with normal sequences?
Future work

- Better separation for “PD beats plogon” (from 1/2 to 0)?
- Is there a separation example with normal sequences?