The Critical Exponent is Computable for Automatic Sequences

Jeffrey Shallit
School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada
shallit@cs.uwaterloo.ca
http://www.cs.uwaterloo.ca/~shallit

Powers in words

A *square* is a nonempty word of the form xx.

Powers in words

A *square* is a nonempty word of the form xx.

Examples include

- `murmur` and `hotshots` in English

Powers in words

A *square* is a nonempty word of the form xx.

Examples include

- `murmur` and `hotshots` in English
- `jenjen` and `taktak` in Czech

Powers in words

A *square* is a nonempty word of the form xx.

Examples include

- `murmur` and `hotshots` in English
- `jenjen` and `taktak` in Czech

Similarly, a *cube* is a nonempty word of the form xxx.

Fractional powers

We can extend the notion of integer power of a word to *fractional powers*.

Fractional powers

We can extend the notion of integer power of a word to *fractional powers*.

A word w is a fractional power if it can be written in the form $w = x^n x'$, where $n \geq 1$ and x' is a prefix of x.

Fractional powers

We can extend the notion of integer power of a word to *fractional powers*.

A word w is a fractional power if it can be written in the form $w = x^n x'$, where $n \geq 1$ and x' is a prefix of x.

We say w has *period* $|x|$ and *exponent* $|w|/|x|$. The shortest period is *the* period and the largest exponent is *the* exponent.

Fractional powers

We can extend the notion of integer power of a word to *fractional powers*.

A word w is a fractional power if it can be written in the form $w = x^n x'$, where $n \geq 1$ and x' is a prefix of x.

We say w has *period* $|x|$ and *exponent* $|w|/|x|$. The shortest period is *the* period and the largest exponent is *the* exponent.

For example, the exponent of the English word `alfalfa` is $7/3$.

Fractional powers

We can extend the notion of integer power of a word to *fractional powers*.

A word w is a fractional power if it can be written in the form $w = x^n x'$, where $n \geq 1$ and x' is a prefix of x.

We say w has *period* $|x|$ and *exponent* $|w|/|x|$. The shortest period is *the* period and the largest exponent is *the* exponent.

For example, the exponent of the English word `alfalfa` is $7/3$.

The exponent of the Czech words `jajaj` and `jejej` is $5/2$.

Morphisms

Let $h : \Sigma^* \rightarrow \Sigma^*$ be a morphism.

Morphisms

Let $h : \Sigma^* \to \Sigma^*$ be a morphism.

If there exists k such that $|h(a)| = k$ for all $a \in \Sigma$, then we say h is *k-uniform*.

Morphisms

Let $h : \Sigma^* \to \Sigma^*$ be a morphism.

If there exists k such that $|h(a)| = k$ for all $a \in \Sigma$, then we say h is *k-uniform*.

If h is 1-uniform, it is called a *coding*.

Morphisms

Let $h : \Sigma^* \to \Sigma^*$ be a morphism.

If there exists k such that $|h(a)| = k$ for all $a \in \Sigma$, then we say h is *k-uniform*.

If h is 1-uniform, it is called a *coding*.

If there is a letter $a \in \Sigma$ such that

(a) $h(a) = ax$ for some $x \in \Sigma^*$; and

Morphisms

Let $h : \Sigma^* \to \Sigma^*$ be a morphism.

If there exists k such that $|h(a)| = k$ for all $a \in \Sigma$, then we say h is *k-uniform*.

If h is 1-uniform, it is called a *coding*.

If there is a letter $a \in \Sigma$ such that

(a) $h(a) = ax$ for some $x \in \Sigma^*$; and

(b) $h^i(x) \neq \epsilon$ for all $i \geq 0$

Morphisms

Let $h : \Sigma^* \rightarrow \Sigma^*$ be a morphism.

If there exists k such that $|h(a)| = k$ for all $a \in \Sigma$, then we say h is *k-uniform*.

If h is 1-uniform, it is called a *coding*.

If there is a letter $a \in \Sigma$ such that

(a) $h(a) = ax$ for some $x \in \Sigma^*$; and

(b) $h^i(x) \neq \epsilon$ for all $i \geq 0$

then we say h is *prolongable* on a.

Prolongable morphisms and fixed points

If h is prolongable, we can generate an infinite fixed point of h by iteration:

Prolongable morphisms and fixed points

If h is prolongable, we can generate an infinite fixed point of h by iteration:

$$h^{\omega}(a) \quad := \quad \lim_{i \to \infty} h^{i}(a)$$

Prolongable morphisms and fixed points

If h is prolongable, we can generate an infinite fixed point of h by iteration:

$$\begin{aligned} h^\omega(a) &:= \lim_{i\to\infty} h^i(a) \\ &= a\ x\ h(x)\ h^2(x)\ h^3(x)\cdots \end{aligned}$$

Classes of morphic words

If an infinite word $\mathbf{w}$ is generated by iterating a morphism, it is called *pure morphic*.

Classes of morphic words

If an infinite word $\mathbf{w}$ is generated by iterating a morphism, it is called *pure morphic*.

If $\mathbf{w} = \tau(\mathbf{x})$ for a pure morphic word $\mathbf{x}$, and a coding τ, it is called *morphic*.

Classes of morphic words

If an infinite word $\mathbf{w}$ is generated by iterating a morphism, it is called *pure morphic*.

If $\mathbf{w} = \tau(\mathbf{x})$ for a pure morphic word $\mathbf{x}$, and a coding τ, it is called *morphic*.

If an infinite word $\mathbf{w}$ is generated by iterating a uniform morphism, it is called *pure uniform morphic*.

Classes of morphic words

If an infinite word $\mathbf{w}$ is generated by iterating a morphism, it is called *pure morphic*.

If $\mathbf{w} = \tau(\mathbf{x})$ for a pure morphic word $\mathbf{x}$, and a coding τ, it is called *morphic*.

If an infinite word $\mathbf{w}$ is generated by iterating a uniform morphism, it is called *pure uniform morphic*.

If $\mathbf{w} = \tau(\mathbf{x})$ for a k-uniform morphic word $\mathbf{x}$, and a coding τ, it is called *k-automatic*.

Automatic sequences

By Cobham's theorem, we know that automatic sequences can be characterized in two different ways:

Automatic sequences

By Cobham's theorem, we know that automatic sequences can be characterized in two different ways:

- as the image (under a coding) of the fixed point of a k-uniform morphism

Automatic sequences

By Cobham's theorem, we know that automatic sequences can be characterized in two different ways:

- as the image (under a coding) of the fixed point of a k-uniform morphism

- as the infinite word generated by an automaton taking the base-k expansion of n as input, and producing the n'th term of the sequence as output

Exponent of an infinite word

The *critical exponent* of an infinite word is defined to be the sup, over all factors, of the exponent of that factor.

Exponent of an infinite word

The *critical exponent* of an infinite word is defined to be the sup, over all factors, of the exponent of that factor.

It could be infinite: consider $010101010101 \cdots$.

Exponent of an infinite word

The *critical exponent* of an infinite word is defined to be the sup, over all factors, of the exponent of that factor.

It could be infinite: consider $010101010101\cdots$.

It could be irrational: it is known that the critical exponent of the Fibonacci word $01001010\cdots$ generated by iterating $0 \to 01$ and $1 \to 0$, is $(3+\sqrt{5})/2$ (Mignosi & Pirillo, 1992).

Exponent of an infinite word

The *critical exponent* of an infinite word is defined to be the sup, over all factors, of the exponent of that factor.

It could be infinite: consider $010101010101\cdots$.

It could be irrational: it is known that the critical exponent of the Fibonacci word $01001010\cdots$ generated by iterating $0 \to 01$ and $1 \to 0$, is $(3+\sqrt{5})/2$ (Mignosi & Pirillo, 1992).

It can be rational & attained: the critical exponent of the Thue-Morse word $\mathbf{t} = 01101001\cdots$, generated by iterating $0 \to 01$ and $1 \to 01$, is 2, and it is attained.

Exponent of an infinite word

The *critical exponent* of an infinite word is defined to be the sup, over all factors, of the exponent of that factor.

It could be infinite: consider $010101010101 \cdots$.

It could be irrational: it is known that the critical exponent of the Fibonacci word $01001010 \cdots$ generated by iterating $0 \rightarrow 01$ and $1 \rightarrow 0$, is $(3+\sqrt{5})/2$ (Mignosi & Pirillo, 1992).

It can be rational & attained: the critical exponent of the Thue-Morse word $\mathbf{t} = 01101001 \cdots$, generated by iterating $0 \rightarrow 01$ and $1 \rightarrow 01$, is 2, and it is attained.

It can be rational & not be attained: the word $210201210120210201202101210 \cdots$, which counts the run lengths of 1's in $\mathbf{t}$, and is generated by $2 \rightarrow 210$, $1 \rightarrow 20$, and $0 \rightarrow 1$, has critical exponent 2, but it is not attained.

Critical exponents

More generally: any real number > 1 can be the critical exponent of a word (over a sufficiently large finite alphabet) (Krieger & JOS, 2007).

Critical exponents

More generally: any real number > 1 can be the critical exponent of a word (over a sufficiently large finite alphabet) (Krieger & JOS, 2007).

Any real number ≥ 2 can be the critical exponent of a binary word (Currie and Rampersad, 2008).

Critical exponents

More generally: any real number > 1 can be the critical exponent of a word (over a sufficiently large finite alphabet) (Krieger & JOS, 2007).

Any real number ≥ 2 can be the critical exponent of a binary word (Currie and Rampersad, 2008).

Further, for words that are fixed points of morphisms, the critical exponent lies in the field extension generated by the eigenvalues of the associated incidence matrix (Krieger, 2006).

Computing the critical exponent

- rational and computable for fixed points of uniform binary morphisms (Krieger, 2009)

Computing the critical exponent

- rational and computable for fixed points of uniform binary morphisms (Krieger, 2009)

- computable in many cases for pure morphic words (Krieger)

Computing the critical exponent

- rational and computable for fixed points of uniform binary morphisms (Krieger, 2009)

- computable in many cases for pure morphic words (Krieger)

- it is decidable, for an infinite word generated by iterating an arbitrary morphism, if its critical exponent is $< \infty$ (Ehrenfeucht & Rozenberg, 1983)

Computing the critical exponent

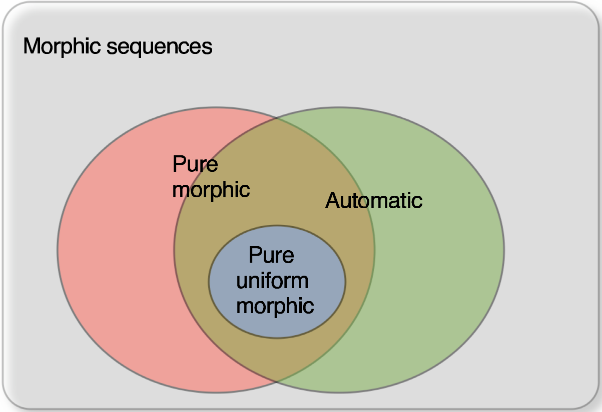
- rational and computable for fixed points of uniform binary morphisms (Krieger, 2009)

- computable in many cases for pure morphic words (Krieger)

- it is decidable, for an infinite word generated by iterating an arbitrary morphism, if its critical exponent is $< \infty$ (Ehrenfeucht & Rozenberg, 1983)

- in this talk: it is rational and computable for all k-automatic sequences

Morphic, pure morphic, and automatic sequences



Encoding natural numbers

Fix an integer $k \geq 2$, and let $\Sigma_k = \{0, 1, 2, \ldots, k-1\}$.

Encoding natural numbers

Fix an integer $k \geq 2$, and let $\Sigma_k = \{0, 1, 2, \ldots, k-1\}$.

We can represent natural numbers $n \geq 0$ in base-k. A representation is *canonical* if it has no leading zeroes.

Encoding natural numbers

Fix an integer $k \geq 2$, and let $\Sigma_k = \{0, 1, 2, \ldots, k-1\}$.

We can represent natural numbers $n \geq 0$ in base-k. A representation is *canonical* if it has no leading zeroes.

If m is a natural number, then $(m)_k$ denotes its canonical representation.

Encoding natural numbers

Fix an integer $k \geq 2$, and let $\Sigma_k = \{0, 1, 2, \ldots, k-1\}$.

We can represent natural numbers $n \geq 0$ in base-k. A representation is *canonical* if it has no leading zeroes.

If m is a natural number, then $(m)_k$ denotes its canonical representation.

If w is a word, then $[w]_k$ is the integer it represents in base k.

Encoding natural numbers

Fix an integer $k \geq 2$, and let $\Sigma_k = \{0, 1, 2, \ldots, k-1\}$.

We can represent natural numbers $n \geq 0$ in base-k. A representation is *canonical* if it has no leading zeroes.

If m is a natural number, then $(m)_k$ denotes its canonical representation.

If w is a word, then $[w]_k$ is the integer it represents in base k.

This gives a 1–1 correspondence between $\mathbb{N}$ and elements of $\{\epsilon\} \cup (\Sigma_k - \{0\})\Sigma_k^*$.

Encoding pairs of natural numbers

A pair of natural numbers (m, n) can be encoded as a word over the larger alphabet Σ_k^2.

Encoding pairs of natural numbers

A pair of natural numbers (m, n) can be encoded as a word over the larger alphabet Σ_k^2.

To do so, we take the base-k representations of m and n, and pad the shorter with leading zeroes, if necessary, so it is the same length as the longer.

Encoding pairs of natural numbers

A pair of natural numbers (m, n) can be encoded as a word over the larger alphabet Σ_k^2.

To do so, we take the base-k representations of m and n, and pad the shorter with leading zeroes, if necessary, so it is the same length as the longer.

Then we pair up the digits and consider them as elements of Σ_k^2.

Encoding pairs of natural numbers

A pair of natural numbers (m, n) can be encoded as a word over the larger alphabet Σ_k^2.

To do so, we take the base-k representations of m and n, and pad the shorter with leading zeroes, if necessary, so it is the same length as the longer.

Then we pair up the digits and consider them as elements of Σ_k^2.

For example, if $m = 23$ and $n = 10$, then $(m)_2 = 10111$ and $(n)_2 = 1010$. We pad the representation of n to get 01010 and then pair up with the digits of m to get

$$(23, 10)_2 = [1, 0][0, 1][1, 0][1, 1][1, 0].$$

Encoding sets of integers and pairs

A language $L \subseteq \Sigma_k^*$ encodes a set S of integers as follows: $n \in S$ if and only if the canonical base-k representation of n is contained in L.

Encoding sets of integers and pairs

A language $L \subseteq \Sigma_k^*$ encodes a set S of integers as follows: $n \in S$ if and only if the canonical base-k representation of n is contained in L.

If L is regular, we get the class of k-automatic sets.

Encoding sets of integers and pairs

A language $L \subseteq \Sigma_k^*$ encodes a set S of integers as follows: $n \in S$ if and only if the canonical base-k representation of n is contained in L.

If L is regular, we get the class of k-automatic sets.

In a similar way, we can encode a set of pairs of integers as a language over $(\Sigma_k^2)^*$.

Representing rational numbers

We can represent a rational number p/q as the base-k encoding of the pair (p, q).

Representing rational numbers

We can represent a rational number p/q as the base-k encoding of the pair (p, q).

Note that we do not insist that p/q be in lowest terms.

Representing rational numbers

We can represent a rational number p/q as the base-k encoding of the pair (p, q).

Note that we do not insist that p/q be in lowest terms.

We define $f((p, q)_k) = p/q$.

Representing rational numbers

We can represent a rational number p/q as the base-k encoding of the pair (p, q).

Note that we do not insist that p/q be in lowest terms.

We define $f((p, q)_k) = p/q$.

Similarly, we can encode a set of rationals as a language L over $(\Sigma_k^2)^*$.

Representing rational numbers

We can represent a rational number p/q as the base-k encoding of the pair (p, q).

Note that we do not insist that p/q be in lowest terms.

We define $f((p, q)_k) = p/q$.

Similarly, we can encode a set of rationals as a language L over $(\Sigma_k^2)^*$.

A given rational may have one or more representations in L.

The main theorem

Theorem. Suppose $L \subseteq (\Sigma_k^2)^*$ is a regular language. Then

$$\alpha := \sup_{x \in L} f(x)$$

is either infinite or rational.

The main theorem

Theorem. Suppose $L \subseteq (\Sigma_k^2)^*$ is a regular language. Then

$$\alpha := \sup_{x \in L} f(x)$$

is either infinite or rational.

Further, given an automaton for L, we can compute α.

Outline of the proof

1. We assume that $\alpha = \sup_{x \in L} f(x)$ is finite and irrational and L is accepted by a DFA of n states.

Outline of the proof

1. We assume that $\alpha = \sup_{x \in L} f(x)$ is finite and irrational and L is accepted by a DFA of n states.
2. Therefore we can find $x \in L$ with $f(x)$ arbitrarily close to α.

Outline of the proof

1. We assume that $\alpha = \sup_{x \in L} f(x)$ is finite and irrational and L is accepted by a DFA of n states.

2. Therefore we can find $x \in L$ with $f(x)$ arbitrarily close to α.

3. Using the pumping lemma, we construct $x' \in L$ with $f(x') > \alpha$, a contradiction.

A useful lemma

Let u, v, w be words over $(\Sigma_k^2)^*$.

A useful lemma

Let u, v, w be words over $(\Sigma_k^2)^*$.

Then exactly one of the following cases holds:

A useful lemma

Let u, v, w be words over $(\Sigma_k^2)^*$.

Then exactly one of the following cases holds:
(a) $f(uv^i w) = f(uv^{i+1} w)$ for all $i \geq 0$;

A useful lemma

Let u, v, w be words over $(\Sigma_k^2)^*$.

Then exactly one of the following cases holds:
(a) $f(uv^i w) = f(uv^{i+1} w)$ for all $i \geq 0$;
(b) $f(uv^i w) < f(uv^{i+1} w)$ for all $i \geq 0$;

A useful lemma

Let u, v, w be words over $(\Sigma_k^2)^*$.

Then exactly one of the following cases holds:
(a) $f(uv^i w) = f(uv^{i+1}w)$ for all $i \geq 0$;
(b) $f(uv^i w) < f(uv^{i+1}w)$ for all $i \geq 0$;
(c) $f(uv^i w) > f(uv^{i+1}w)$ for all $i \geq 0$.

A useful lemma

Let u, v, w be words over $(\Sigma_k^2)^*$.

Then exactly one of the following cases holds:
(a) $f(uv^iw) = f(uv^{i+1}w)$ for all $i \geq 0$;
(b) $f(uv^iw) < f(uv^{i+1}w)$ for all $i \geq 0$;
(c) $f(uv^iw) > f(uv^{i+1}w)$ for all $i \geq 0$.

This allows us to construct an x' with $f(x') > \alpha$, a contradiction.

Computability

The same argument shows that $\alpha = \sup_{x \in L} f(x)$ lies in an easily describable set, so it is computable.

Computability

The same argument shows that $\alpha = \sup_{x \in L} f(x)$ lies in an easily describable set, so it is computable.

More precisely,

$$\alpha \in S_1 \bigcup S_2,$$

Computability

The same argument shows that $\alpha = \sup_{x \in L} f(x)$ lies in an easily describable set, so it is computable.

More precisely,

$$\alpha \in S_1 \bigcup S_2,$$

where

$$S_1 = \{p/q \; : \; 0 \leq p < k^n, \; 1 \leq q < k^n\}$$

Computability

The same argument shows that $\alpha = \sup_{x \in L} f(x)$ lies in an easily describable set, so it is computable.

More precisely,

$$\alpha \in S_1 \bigcup S_2,$$

where

$$\begin{aligned}
S_1 &= \{p/q \ : \ 0 \leq p < k^n, \ 1 \leq q < k^n\} \\
S_2 &= \{\frac{[u_1]_k + \frac{[v_1]_k}{k^a - 1}}{[u_2]_k + \frac{[v_2]_k}{k^a - 1}} \ : \ |u_1 v_1| \leq n, |u_2 v_2| \leq n, |u_1| = |u_2|, \\
&\quad |v_1| = |v_2| = a \geq 1\}.
\end{aligned}$$

The same argument shows that $\alpha = \sup_{x \in L} f(x)$ lies in an easily describable set, so it is computable.

More precisely,

$$\alpha \in S_1 \bigcup S_2,$$

where

$$\begin{aligned} S_1 &= \{p/q \ : \ 0 \le p < k^n, \ 1 \le q < k^n\} \\ S_2 &= \{\frac{[u_1]_k + \frac{[v_1]_k}{k^a - 1}}{[u_2]_k + \frac{[v_2]_k}{k^a-1}} \ : \ |u_1 v_1| \le n, |u_2 v_2| \le n, |u_1| = |u_2|, \\ &\quad |v_1| = |v_2| = a \ge 1\}. \end{aligned}$$

where n is the number of states in the minimal DFA accepting L.

Applications of the lemma

1. The critical exponent of k-automatic words is either infinite or rational. Furthermore, it is computable.

Applications of the lemma

1. The critical exponent of k-automatic words is either infinite or rational. Furthermore, it is computable.

2. The optimal constant for linear recurrence of k-automatic words is either infinite or rational. Furthermore, it is computable.

Applications of the lemma

1. The critical exponent of k-automatic words is either infinite or rational. Furthermore, it is computable.

2. The optimal constant for linear recurrence of k-automatic words is either infinite or rational. Furthermore, it is computable.

3. The quantities $\limsup_{n \to \infty} F(n)/n$ and $\liminf_{n \to \infty} F(n)/n$ are computable for automatic sequences, where F denotes subword complexity.

Application to the critical exponent

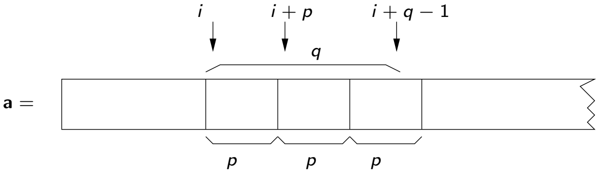
Given an automaton generating the k-automatic sequence $\mathbf{a} = a_0 a_1 \cdots$, we transform it into an automaton M accepting

$$L = \{(p, q)_k \ : \ \exists \text{ a factor of } \mathbf{a} \text{ of length } q \text{ with period } p\ \}.$$

Application to the critical exponent

Given an automaton generating the k-automatic sequence $\mathbf{a} = a_0 a_1 \cdots$, we transform it into an automaton M accepting

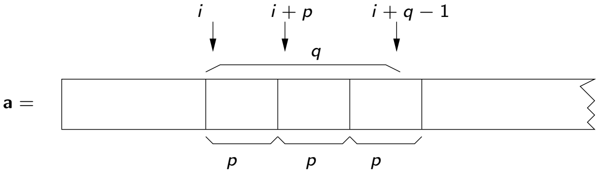
$$L = \{(p,q)_k \ : \ \exists \text{ a factor of } \mathbf{a} \text{ of length } q \text{ with period } p \ \}.$$



Application to the critical exponent

Given an automaton generating the k-automatic sequence $\mathbf{a} = a_0 a_1 \cdots$, we transform it into an automaton M accepting

$$L = \{(p, q)_k \ : \ \exists \text{ a factor of } \mathbf{a} \text{ of length } q \text{ with period } p \ \}.$$

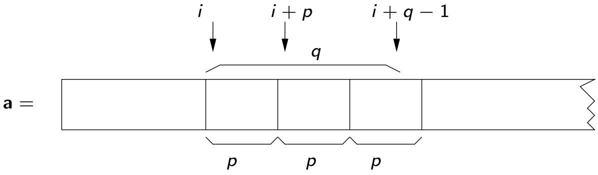


The idea is to nondeterministically choose an index i at which a factor of length q begins in $\mathbf{a}$, and then verify that $\mathbf{a}[j] = \mathbf{a}[j + p]$ for $i \leq j \leq i + q - p - 1$.

Application to the critical exponent

Given an automaton generating the k-automatic sequence $\mathbf{a} = a_0 a_1 \cdots$, we transform it into an automaton M accepting

$$L = \{(p, q)_k \ : \ \exists \text{ a factor of } \mathbf{a} \text{ of length } q \text{ with period } p \}.$$



The idea is to nondeterministically choose an index i at which a factor of length q begins in $\mathbf{a}$, and then verify that $\mathbf{a}[j] = \mathbf{a}[j + p]$ for $i \leq j \leq i + q - p - 1$.

The critical exponent of $\mathbf{a}$ is then $\sup_{x \in L} f(x)$, which is either rational or infinite.

Application to linear recurrence

A sequence **a** is *recurrent* if every factor that occurs, occurs infinitely often.

Application to linear recurrence

A sequence $\mathbf{a}$ is *recurrent* if every factor that occurs, occurs infinitely often.

It is *linearly recurrent* if there exists a constant C such that for all $\ell \geq 0$ and all factors x of length ℓ occurring in $\mathbf{a}$, any two consecutive occurrences of x are separated by at most $C\ell$ positions.

Application to linear recurrence

A sequence $\mathbf{a}$ is *recurrent* if every factor that occurs, occurs infinitely often.

It is *linearly recurrent* if there exists a constant C such that for all $\ell \geq 0$ and all factors x of length ℓ occurring in $\mathbf{a}$, any two consecutive occurrences of x are separated by at most $C\ell$ positions.

Theorem.
If an automatic sequence $\mathbf{a}$ is linearly recurrent, then the optimal constant C is rational and computable.

Sketch of the proof

Define

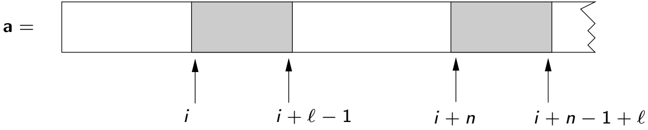
$L = \{(n, l)_k \;:\;$ (a) there exists $i \geq 0$ s. t. for all $j, 0 \leq j < \ell$ we have $\mathbf{a}[i+j] = \mathbf{a}[i+n+j]$ and
(b) there is no $t, 0 < t < n$ s. t. for all $j, 0 \leq j < \ell$ we have $\mathbf{a}[i+j] = \mathbf{a}[i+t+j]\}$

Sketch of the proof

Define

$L = \{(n, l)_k \ : \ $ (a) there exists $i \geq 0$ s. t. for all $j, 0 \leq j < \ell$ we have $\mathbf{a}[i+j] = \mathbf{a}[i+n+j]$ and

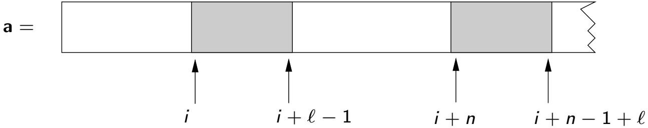
(b) there is no $t, 0 < t < n$ s. t. for all $j, 0 \leq j < \ell$ we have $\mathbf{a}[i+j] = \mathbf{a}[i+t+j]\}$



Sketch of the proof

Define

$L = \{(n, l)_k \ : \ $ (a) there exists $i \geq 0$ s. t. for all $j, 0 \leq j < \ell$ we have $\mathbf{a}[i + j] = \mathbf{a}[i + n + j]$ and
(b) there is no $t, 0 < t < n$ s. t. for all $j, 0 \leq j < \ell$ we have $\mathbf{a}[i + j] = \mathbf{a}[i + t + j]\}$



Another way to say this is that L consists of the base-k representation of those pairs of integers (n, ℓ) such that (a) there is some factor of length ℓ for which there is another occurrence at distance n and (b) this occurrence is actually the very next occurrence.

Sketch of the proof

Now from our Theorem we know $\sup\{n/\ell \ : \ (n,\ell)_k \in L\}$ is either infinite or rational. In the latter case this sup is computable, and this gives the optimal constant C for the linear recurrence of $\mathbf{a}$.

Applications to subword complexity

Let $F(n)$ denote the number of distinct factors of length n in a given k-autmatic sequence (aka subword complexity).

Applications to subword complexity

Let $F(n)$ denote the number of distinct factors of length n in a given k-autmatic sequence (aka subword complexity).

The quantities

$$\limsup_{n\to\infty} F(n)/n$$

and

$$\liminf_{n\to\infty} F(n)/n$$

are computable.

Applications to subword complexity

Let $F(n)$ denote the number of distinct factors of length n in a given k-autmatic sequence (aka subword complexity).

The quantities

$$\limsup_{n \to \infty} F(n)/n$$

and

$$\liminf_{n \to \infty} F(n)/n$$

are computable.

The idea for $\limsup$ is essentially the same as the idea for $\sup$.

Applications to subword complexity

Let $F(n)$ denote the number of distinct factors of length n in a given k-autmatic sequence (aka subword complexity).

The quantities

$$\limsup_{n\to\infty} F(n)/n$$

and

$$\liminf_{n\to\infty} F(n)/n$$

are computable.

The idea for $\limsup$ is essentially the same as the idea for sup.

Hence their equality is computable (cf. Goldstein [2011]).

Open problems

– Extend these ideas to all morphic sequences, not just automatic sequences.

– Extend these ideas to all morphic sequences, not just automatic sequences.

– Is the theorem about $\sup_{x \in L} f(x)$ also true for context-free languages?

For further reading

1. E. Charlier, N. Rampersad, and J. Shallit, Enumeration and decidable properties of automatic sequences. In DLT 2011, pp. 165–179.