Decision Algorithms

for Ostrowski-Automatic Sequences

Aseem Baranwal

Supervised by Jeffrey Shallit

School of Computer Science
University of Waterloo

Acknowledgements

This work would not have been possible without

- Jeffrey Shallit,
- Luke Schaeffer,
- Hamoon Mousavi,
- Narad Rampersad, and
- many others for discussions, feedback and reviews.

What is this about?

- We extend the notion of k-automatic sequences (Schaeffer, Shallit, Mousavi, Du, Allouche, Rowland) to Ostrowski-automatic sequences
- We develop a procedure to computationally decide certain combinatorial and enumeration questions about these sequences that can be expressed as predicates in first-order logic.

Numeration system

A numeration system is a triple $\mathcal{N} = (\Sigma, L, f_{\mathcal{N}})$

- a finite alphabet Σ

Numeration system

A numeration system is a triple $\mathcal{N} = (\Sigma, L, f_{\mathcal{N}})$

- a finite alphabet Σ
- a language $L \subseteq \Sigma^*$

Numeration system

A numeration system is a triple $\mathcal{N} = (\Sigma, L, f_{\mathcal{N}})$

- a finite alphabet Σ
- a language $L \subseteq \Sigma^*$
- an onto function $f_{\mathcal{N}} : \Sigma^* \to \mathbb{N}$

Numeration system

A numeration system is a triple $\mathcal{N} = (\Sigma, L, f_{\mathcal{N}})$

- a finite alphabet Σ
- a language $L \subseteq \Sigma^*$
- an onto function $f_{\mathcal{N}} : \Sigma^* \to \mathbb{N}$

Numeration system

A numeration system is a triple $\mathcal{N} = (\Sigma, L, f_{\mathcal{N}})$

- a finite alphabet Σ
- a language $L \subseteq \Sigma^*$
- an onto function $f_{\mathcal{N}} : \Sigma^* \to \mathbb{N}$

$w \in L$ is a **representation** for $N \geq 0$ if $f_{\mathcal{N}}(w) = N$.

Notation: $[w]_{\mathcal{N}} = f_{\mathcal{N}}(w)$.

Numeration system

A numeration system is a triple $\mathcal{N} = (\Sigma, L, f_{\mathcal{N}})$

- a finite alphabet Σ
- a language $L \subseteq \Sigma^*$
- an onto function $f_{\mathcal{N}} : \Sigma^* \to \mathbb{N}$

$w \in L$ is a **representation** for $N \geq 0$ if $f_{\mathcal{N}}(w) = N$.

Notation: $[w]_{\mathcal{N}} = f_{\mathcal{N}}(w)$.

Example

Standard base-b number system: $(\Sigma_b, \Sigma_b^*, f_b)$.

$$f_b = [a_{n-1}a_{n-2}\cdots a_0]_b = \sum_{0 \leq i < n} a_i b^i.$$

e.g.: $[31]_8 = [25]_{10}$.

Continued fraction

Notation: $\alpha = [d_0; d_1, d_2, \ldots]$, if $\alpha = d_0 + \cfrac{1}{d_1 + \cfrac{1}{d_2 + \cdots}}$.

Continued fraction

Notation: $\alpha = [d_0; d_1, d_2, \ldots]$, if $\alpha = d_0 + \cfrac{1}{d_1 + \cfrac{1}{d_2 + \cdots}}$.

Examples

- The golden ratio $\varphi = [\bar{1}] = [1; 1, 1, \ldots]$.

Continued fraction

Notation: $\alpha = [d_0; d_1, d_2, \ldots]$, if $\alpha = d_0 + \cfrac{1}{d_1 + \cfrac{1}{d_2 + \cdots}}$.

Examples

- The golden ratio $\varphi = [\overline{1}] = [1; 1, 1, \ldots]$.
- $e = [2; 1, 2, 1, 1, 4, 1, 1, 6, \ldots]$.

Continued fraction

Notation: $\alpha = [d_0; d_1, d_2, \ldots]$, if $\alpha = d_0 + \cfrac{1}{d_1 + \cfrac{1}{d_2 + \cdots}}$.

Examples

- The golden ratio $\varphi = [\overline{1}] = [1; 1, 1, \ldots]$.
- $e = [2; 1, 2, 1, 1, 4, 1, 1, 6, \ldots]$.
- The real number $\frac{63-\sqrt{10}}{107} = 0.55923\cdots = [0; 1, 1, 3, \overline{1, 2, 1}]$.
  Preperiod: $0, 1, 1, 3$.
  Period: $1, 2, 1$.

Convergents

For a real $\alpha = [d_0; d_1, d_2, \ldots]$,

Convergents

For a real $\alpha = [d_0; d_1, d_2, \dots]$,

$$\frac{p_n}{q_n} = [d_0; d_1, d_2, \dots, d_n]$$

is a **convergent** of α, where:

Convergents

For a real $\alpha = [d_0; d_1, d_2, \ldots]$,

$$\frac{p_n}{q_n} = [d_0; d_1, d_2, \ldots, d_n]$$

is a **convergent** of α, where:

$$\begin{aligned} p_0 &= d_0, & p_1 &= d_1 d_0 + 1, & p_n &= d_n p_{n-1} + p_{n-2}, \\ q_0 &= 1, & q_1 &= d_1, & q_n &= d_n q_{n-1} + q_{n-2}. \end{aligned}$$

Convergents

For a real $\alpha = [d_0; d_1, d_2, \dots]$,

$$\frac{p_n}{q_n} = [d_0; d_1, d_2, \dots, d_n]$$

is a **convergent** of α, where:

$$p_0 = d_0, \qquad p_1 = d_1 d_0 + 1, \qquad p_n = d_n p_{n-1} + p_{n-2},$$
$$q_0 = 1, \qquad q_1 = d_1, \qquad q_n = d_n q_{n-1} + q_{n-2}.$$

Ostrowski numeration system

Named after **Alexander Markowich Ostrowski** (1922).

Use continued fraction of an irrational α to represent integers.

Examples

$\alpha = 1/\phi^2 = [0; 2, \overline{1}]$ $(q_i)_{i\geq 0} = 1, 2, 3, 5, 8, \ldots$	$\alpha = 2 - \sqrt{3} = [0; 3, \overline{1, 2}]$ $(q_i)_{i\geq 0} = 1, 3, 4, 11, 15, 41, \ldots$
• $[1010]_\alpha = 7$ • $[100]_\alpha = 3$	• $[1000]_\alpha = 11$ • $[110]_\alpha = 7$

Examples

$\alpha = 1/\phi^2 = [0; 2, \overline{1}]$ $(q_i)_{i\geq 0} = 1, 2, 3, 5, 8, \ldots$	$\alpha = 2 - \sqrt{3} = [0; 3, \overline{1, 2}]$ $(q_i)_{i\geq 0} = 1, 3, 4, 11, 15, 41, \ldots$
• $[1010]_\alpha = 7$ • $[100]_\alpha = 3$	• $[1000]_\alpha = 11 = [210]_\alpha$? • $[110]_\alpha = 7 = [103]_\alpha$?

Examples

$\alpha = 1/\phi^2 = [0; 2, \overline{1}]$ $(q_i)_{i\geq 0} = 1, 2, 3, 5, 8, \ldots$	$\alpha = 2 - \sqrt{3} = [0; 3, \overline{1, 2}]$ $(q_i)_{i\geq 0} = 1, 3, 4, 11, 15, 41, \ldots$
• $[1010]_\alpha = 7$ • $[100]_\alpha = 3$	• $[1000]_\alpha = 11 = [210]_\alpha$? • $[110]_\alpha = 7 = [103]_\alpha$?

Unique representation

A representation $[a_{k-1}a_{k-2}\cdots a_0]_\alpha = \sum_i a_i q_i$ is **canonical** if

- $0 \leq a_0 < d_1$,
- $0 \leq a_i \leq d_{i+1}$, for $i \geq 1$, and
- for all $i \geq 1$, if $a_i = d_{i+1}$ then $a_{i-1} = 0$.

Automatic sequence

A sequence $\mathbf{a} = (a_n)_{n\geq 0}$ is automatic if there exists a DFAO M and a number system $\mathcal{N}$, such that $M([n]_{\mathcal{N}}) = \mathbf{a}[n]$.

Automatic sequence

A sequence $\mathbf{a} = (a_n)_{n\geq 0}$ is automatic if there exists a DFAO M and a number system $\mathcal{N}$, such that $M([n]_{\mathcal{N}}) = \mathbf{a}[n]$.

The Thue-Morse sequence $\mathbf{t} = t_0t_1t_2\cdots = \texttt{01101001}\cdots$ is given by the DFAO below, where t_n is the output associated with the state reached on completely reading n in base 2.

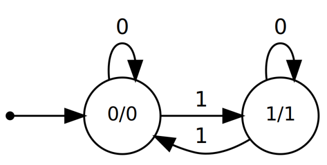


Figure 1: DFAO computing the Thue-Morse sequence $\mathbf{t}$.

- The logical theory $\text{Th}(\mathbb{N}, +)$ is decidable.
- Büchi showed that adding $V_k(n) = k^e$, where $e = \max\{i : k^i | n\}$ maintains decidability.

Theorem 1

There exists an algorithm that, given a proposition $\mathcal{P}$ phrased using only $\forall, \exists, +, -$, comparisons, logical operations, and indexing into one or more automatic sequences, will decide the truth of $\mathcal{P}$.

- The logical theory Th($\mathbb{N}, +$) is decidable.
- Büchi showed that adding $V_k(n) = k^e$, where $e = \max\{i : k^i | n\}$ maintains decidability.

Theorem 1

There exists an algorithm that, given a proposition $\mathcal{P}$ phrased using only $\forall, \exists, +, -$, comparisons, logical operations, and indexing into one or more automatic sequences, will decide the truth of $\mathcal{P}$.

- Schaeffer and Shallit (2012) showed that this is possible for k-automatic sequences.
- Hamoon Mousavi implemented the decision procedures in `Walnut` (2016).

Example for the standard case

Goal

We need an automaton that reads in 3 inputs x, y, z in parallel, and accepts if and only if $x + y = z$.

Goal

We need an automaton that reads in 3 inputs x, y, z in parallel, and accepts if and only if $x + y = z$.

Define:

$$x = x_{b-1} \cdots x_0,$$
$$y = y_{b-1} \cdots y_0,$$
$$z = z_{b-1} \cdots z_0,$$
$$w_i = z_i - (x_i + y_i) \text{ for } 0 \leq i < b.$$

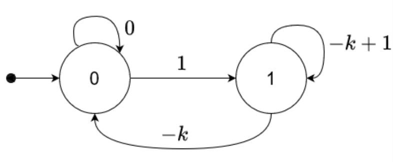


Figure 2: Base-k adder automaton.

Goal

We need an automaton that reads in 3 inputs x, y, z in parallel, and accepts if and only if $x + y = z$.

Define:

$$x = x_{b-1} \cdots x_0,$$
$$y = y_{b-1} \cdots y_0,$$
$$z = z_{b-1} \cdots z_0,$$
$$w_i = z_i - (x_i + y_i) \text{ for } 0 \leq i < b.$$

- Transition $r \to s$ is of the form $-rk + s$.

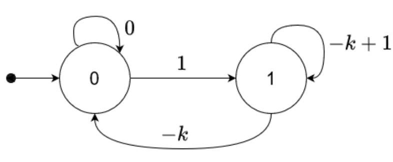


Figure 2: Base-k adder automaton.

Goal

We need an automaton that reads in 3 inputs x, y, z in parallel, and accepts if and only if $x + y = z$.

Define:

$$x = x_{b-1} \cdots x_0,$$
$$y = y_{b-1} \cdots y_0,$$
$$z = z_{b-1} \cdots z_0,$$
$$w_i = z_i - (x_i + y_i) \text{ for } 0 \le i < b.$$

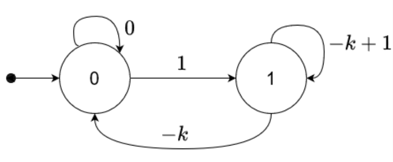


Figure 2: Base-k adder automaton.

- Transition $r \to s$ is of the form $-rk + s$.
- Example: $[0100]_2 + [0110]_2 = [1010]_2$.

Example for the standard case

- Define state r such that reading $w_{i-1} \cdots w_0$ while in state r leads to an accepting state iff $\sum_{0 \leq j < i} w_j k^j = -rk^i$.

- Define state r such that reading $w_{i-1} \cdots w_0$ while in state r leads to an accepting state iff $\sum_{0 \leq j < i} w_j k^j = -rk^i$.
- Separate out the first transition:

$$\begin{aligned} w_{i-1}k^{i-1} + \sum_{0 \leq j < i-1} w_j k^j &= -rk^i \\ \implies \sum_{0 \leq j < i-1} w_j k^j &= (rk - s)k^{i-1} - rk^i \\ &= -sk^{i-1}. \end{aligned}$$

Core Framework

Given:	Goal: recognize (x, y, z) s.t. $x + y = z$.
$\alpha = [0; d_1, d_2, \ldots]$,	• $w = w_{k-1} \cdots w_0$, $w_i = z_i - (x_i + y_i)$ for $0 \leq i < k$.
$x = x_{k-1} \cdots x_0$,	
$y = y_{k-1} \cdots y_0$, and	• Automaton reads the pair (d, w).
$z = z_{k-1} \cdots z_0$.	• States are labelled (r, s).

Given:	Goal: recognize (x, y, z) s.t. $x + y = z$.
$\alpha = [0; d_1, d_2, \ldots],$ $x = x_{k-1} \cdots x_0,$ $y = y_{k-1} \cdots y_0,$ and $z = z_{k-1} \cdots z_0.$	• $w = w_{k-1} \cdots w_0,$ $w_i = z_i - (x_i + y_i)$ for $0 \le i < k$. • Automaton reads the pair (d, w). • States are labelled (r, s).

The adder automaton accepts the input $w_{i-1} w_{i-2} \cdots w_0$, starting from state (r, s), if and only if

$$\sum_{0 \le j < i} q_j w_j = r q_{i-1} + s q_i.$$

Given:	Goal: recognize (x, y, z) s.t. $x + y = z$.
$\alpha = [0; d_1, d_2, \ldots]$, $x = x_{k-1} \cdots x_0$, $y = y_{k-1} \cdots y_0$, and $z = z_{k-1} \cdots z_0$.	• $w = w_{k-1} \cdots w_0$, $w_i = z_i - (x_i + y_i)$ for $0 \leq i < k$. • Automaton reads the pair (d, w). • States are labelled (r, s).

The adder automaton accepts the input $w_{i-1}w_{i-2} \cdots w_0$, starting from state (r, s), if and only if

$$\sum_{0 \leq j < i} q_j w_j = rq_{i-1} + sq_i.$$

Naturally, the initial state must be $(0, 0)$ to accept a valid addition.

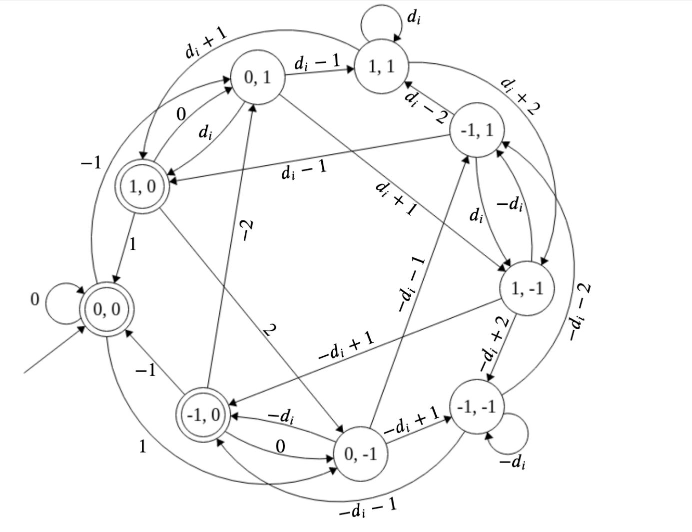


Figure 3: Ostrowski adder automaton.

Main Theorem

If the automaton is to process an input of length i: $w_{i-1} \cdots w_0$, starting in the state (r, s), then meaningful transitions from (r, s) are of the form $w_{i-1} = r + sd_i - t$, and the destination state is (s, t). Here, $r, s, t \in \{-1, 0, 1\}$.

Main Theorem

If the automaton is to process an input of length i: $w_{i-1} \cdots w_0$, starting in the state (r, s), then meaningful transitions from (r, s) are of the form $w_{i-1} = r + sd_i - t$, and the destination state is (s, t). Here, $r, s, t \in \{-1, 0, 1\}$.

Proof sketch:

- Similar construction to the standard base-k number system.
- Look at how transitions change the required sum from the remaining input.
- Create corresponding destination states.

After reading w_{i-1}, the sum of the remaining input is bounded:

$$-2q_{i-1}+2 \leq \sum_{0\leq j<i-1} q_j w_j \leq q_{i-1}-1.$$

After reading w_{i-1}, the sum of the remaining input is bounded:

$$-2q_{i-1} + 2 \leq \sum_{0 \leq j < i-1} q_j w_j \leq q_{i-1} - 1.$$

- Upper bound:
  - $x_j, y_j = 0$ for $j < i - 1$,
  - $[z_{i-2} \cdots z_0]_\alpha = q_{i-1} - 1$.
- Lower bound:
  - $z_j = 0$ for $j < i - 1$,
  - $[x_{i-2} \cdots x_0]_\alpha = [y_{i-2} \cdots y_0]_\alpha = q_{i-1} - 1$.

Separate out the immediate next transition:

$$-2q_{i-1} + 2 \leq (rq_{i-1} + sq_i) - w_{i-1}q_{i-1} \leq q_{i-1} - 1.$$

Separate out the immediate next transition:

$$-2q_{i-1}+2 \le (rq_{i-1}+sq_i) - w_{i-1}q_{i-1} \le q_{i-1}-1.$$

For the upper bound on w_{i-1} we have

$$\begin{aligned} w_{i-1}q_{i-1} &\le rq_{i-1}+sq_i+2q_{i-1}-2 \\ &\le rq_{i-1}+s(d_iq_{i-1}+q_{i-2})+2q_{i-1}-2 \\ &\le (r+sd_i+2)q_{i-1}+sq_{i-2}-2. \end{aligned}$$

Separate out the immediate next transition:

$$-2q_{i-1} + 2 \leq (rq_{i-1} + sq_i) - w_{i-1}q_{i-1} \leq q_{i-1} - 1.$$

For the upper bound on w_{i-1} we have

$$\begin{aligned} w_{i-1}q_{i-1} &\leq rq_{i-1} + sq_i + 2q_{i-1} - 2 \\ &\leq rq_{i-1} + s(d_iq_{i-1} + q_{i-2}) + 2q_{i-1} - 2 \\ &\leq (r + sd_i + 2)q_{i-1} + sq_{i-2} - 2. \end{aligned}$$

We analyze this in two cases:

- $s = 1$, and
- $s \in \{-1, 0\}$.

Proof: main result

- For $s = 1$, we have

$$w_{i-1}q_{i-1} \leq (r + d_i + 2)q_{i-1} + q_{i-2} - 2.$$

- For $s = 1$, we have

$$w_{i-1}q_{i-1} \leq (r + d_i + 2)q_{i-1} + q_{i-2} - 2.$$

Since $w_{i-1} = z_{i-1} - (x_{i-1} + y_{i-1})$, we have that $w_{i-1} \leq d_i$.

- For $s = 1$, we have

$$w_{i-1}q_{i-1} \leq (r + d_i + 2)q_{i-1} + q_{i-2} - 2.$$

Since $w_{i-1} = z_{i-1} - (x_{i-1} + y_{i-1})$, we have that $w_{i-1} \leq d_i$.
Hence, for $r \in \{-1, 0, 1\}$, we have

$$w_{i-1} \leq r + d_i + 1.$$

- For $s = 1$, we have

$$w_{i-1}q_{i-1} \leq (r + d_i + 2)q_{i-1} + q_{i-2} - 2.$$

Since $w_{i-1} = z_{i-1} - (x_{i-1} + y_{i-1})$, we have that $w_{i-1} \leq d_i$.
Hence, for $r \in \{-1, 0, 1\}$, we have

$$w_{i-1} \leq r + d_i + 1.$$

- For $s \in \{-1, 0\}$, we have

$$\begin{aligned} w_{i-1}q_{i-1} &\leq (r + sd_i + 2)q_{i-1} + sq_{i-2} - 2 \\ &< (r + sd_i + 2)q_{i-1} \\ \implies w_{i-1} &\leq r + sd_i + 1. \end{aligned}$$

For the lower bound:

$$\begin{aligned} w_{i-1}q_{i-1} &\geq rq_{i-1} + sq_i - q_{i-1} + 1 \\ &\geq rq_{i-1} + sd_iq_{i-1} + sq_{i-2} - q_{i-1} + 1 \\ &\geq (r + sd_i - 1)q_{i-1} + sq_{i-2} + 1 \\ \implies w_{i-1} &\geq r + sd_i - 1. \end{aligned}$$

For the lower bound:

$$\begin{aligned} w_{i-1}q_{i-1} &\geq rq_{i-1} + sq_i - q_{i-1} + 1 \\ &\geq rq_{i-1} + sd_iq_{i-1} + sq_{i-2} - q_{i-1} + 1 \\ &\geq (r + sd_i - 1)q_{i-1} + sq_{i-2} + 1 \\ \implies w_{i-1} &\geq r + sd_i - 1. \end{aligned}$$

Therefore, we have the following bounds on w_{i-1}.

$$r + sd_i - 1 \leq w_{i-1} \leq r + sd_i + 1.$$

Proof: main result

For the lower bound:

$$\begin{aligned} w_{i-1}q_{i-1} &\geq rq_{i-1} + sq_i - q_{i-1} + 1 \\ &\geq rq_{i-1} + sd_iq_{i-1} + sq_{i-2} - q_{i-1} + 1 \\ &\geq (r + sd_i - 1)q_{i-1} + sq_{i-2} + 1 \\ \implies w_{i-1} &\geq r + sd_i - 1. \end{aligned}$$

Therefore, we have the following bounds on w_{i-1}.

$$r + sd_i - 1 \leq w_{i-1} \leq r + sd_i + 1.$$

Hence, all **transitions** are of the form $r + sd_i - t$ for $t \in \{-1, 0, 1\}$.

Proof: main result

Consider transition $r + sd_i - t$ from state (r, s). Separate out the immediate next transition, we have that

$$\sum_{0 \le j < i-1} q_j w_j + w_{i-1} q_{i-1} = r q_{i-1} + s q_i$$

$$\sum_{0 \le j < i-1} q_j w_j = r q_{i-1} + s q_i - (r + s d_i - t) q_{i-1}$$

$$= s(d_i q_{i-1} + q_{i-2}) - (s d_i - t) q_{i-1}$$

$$= s q_{i-2} + t q_{i-1}.$$

Consider transition $r + sd_i - t$ from state (r, s). Separate out the immediate next transition, we have that

$$\sum_{0 \le j < i-1} q_j w_j + w_{i-1} q_{i-1} = r q_{i-1} + s q_i$$

$$\begin{aligned} \sum_{0 \le j < i-1} q_j w_j &= r q_{i-1} + s q_i - (r + s d_i - t) q_{i-1} \\ &= s(d_i q_{i-1} + q_{i-2}) - (s d_i - t) q_{i-1} \\ &= s q_{i-2} + t q_{i-1}. \end{aligned}$$

This gives us the transition function,

$$\delta((r, s), r + s d_i - t) = (s, t), \text{ for all } r, s, t \in \{-1, 0, 1\}.$$

We implement the adder for the case where α is a quadratic irrational.

We implement the adder for the case where α is a quadratic irrational.

Why not for all α?

We implement the adder for the case where α is a quadratic irrational.

Why not for all α?

- An adder in Walnut reads 3 inputs.

We implement the adder for the case where α is a quadratic irrational.

Why not for all α?

- An adder in Walnut reads 3 inputs.
- For quadratic irrational α, the c.f. is ultimately periodic.

Walnut implementation

We implement the adder for the case where α is a quadratic irrational.

Why not for all α?

- An adder in Walnut reads 3 inputs.
- For quadratic irrational α, the c.f. is ultimately periodic.

Example

The Fibonacci numeration system has $\alpha = 1/\phi^2 = [0; 2, \overline{1}]$, the preperiod $0, 2$, and the period 1. The following command generates the required automata.

```
ost fib [0 2] [1];
```

Now we can use the new system in predicates like usual:

```
eval test "?msd_fib <predicate>";
```

Applications

We apply the developed procedures to several problems in combinatorics on words.

1. Repetition threshold for balanced words.
2. Critical exponent of rich words.
3. Avoiding antisquares in binary words.
4. Deciding properties of Lucas words.

Balanced word

A word w over Σ is **balanced** if, for all equal-length pair of factors u, v of w, we have $||u|_a - |v|_a| \leq 1$ for all letters a.

Balanced word

A word w over Σ is **balanced** if, for all equal-length pair of factors u, v of w, we have $||u|_a - |v|_a| \leq 1$ for all letters a.

Period

The word $w = \texttt{alfalfa}$ has three periods: $3, 6$, and 7. The corresponding exponents are $7/3, 7/6$, and 1.

Balanced word

A word w over Σ is **balanced** if, for all equal-length pair of factors u, v of w, we have $||u|_a - |v|_a| \leq 1$ for all letters a.

Period

The word $w = \texttt{alfalfa}$ has three periods: $3, 6$, and 7. The corresponding exponents are $7/3, 7/6$, and 1.

Critical exponent

The **critical exponent** of an infinite word w, denoted $E(w)$, is the supremum of the set of all e such that there exists a nonempty factor of w with exponent e.

Sturmian words

A Sturmian word, denoted by $\mathbf{c}_\alpha$ is produced as the limit of the sequence of *standard words* s_n defined as follows:

$$s_0 = 0, \quad s_1 = 0^{d_1 - 1}1, \quad s_n = s_{n-1}^{d_n} s_{n-2} \text{ for } n \geq 2,$$

where $[d_0, d_1, d_2, \ldots]$ is the continued fraction expansion of α.

Sturmian words

A Sturmian word, denoted by $\mathbf{c}_\alpha$ is produced as the limit of the sequence of *standard words* s_n defined as follows:

$$s_0 = 0, \quad s_1 = 0^{d_1-1}1, \quad s_n = s_{n-1}^{d_n}s_{n-2} \text{ for } n \geq 2,$$

where $[d_0, d_1, d_2, \ldots]$ is the continued fraction expansion of α.

Previous work

Rampersad et al. (2018).

- Constructed infinite balanced words $\mathbf{x}_k$ over Σ_k for $3 \leq k \leq 10$ using $\mathbf{c}_\alpha$ and a characterization by Hubert.
- Computed $E(\mathbf{x}_3)$ and $E(\mathbf{x}_4)$.
- Proved that $\mathbf{x}_3, \mathbf{x}_4$ achieve the minimum possible repetition.

k	α	c.f.
3	$\sqrt{2}-1$	$[0;\overline{2}]$
4	$1/\varphi^2$	$[0;2,\overline{1}]$
5	$\sqrt{2}-1$	$[0;\overline{2}]$
6	$(78-2\sqrt{6})/101$	$[0;1,2,1,1,\overline{1,1,1,2}]$
7	$(63-\sqrt{10})/107$	$[0;1,1,3,\overline{1,2,1}]$
8	$(23+\sqrt{2})/31$	$[0;1,3,1,\overline{2}]$
9	$(23-\sqrt{2})/31$	$[0;1,2,3,\overline{2}]$
10	$(109+\sqrt{13})/138$	$[0;1,4,2,\overline{3}]$

k	α	c.f.
3	$\sqrt{2}-1$	$[0;\overline{2}]$
4	$1/\varphi^2$	$[0;2,\overline{1}]$
5	$\sqrt{2}-1$	$[0;\overline{2}]$
6	$(78-2\sqrt{6})/101$	$[0;1,2,1,1,\overline{1,1,1,2}]$
7	$(63-\sqrt{10})/107$	$[0;1,1,3,\overline{1,2,1}]$
8	$(23+\sqrt{2})/31$	$[0;1,3,1,\overline{2}]$
9	$(23-\sqrt{2})/31$	$[0;1,2,3,\overline{2}]$
10	$(109+\sqrt{13})/138$	$[0;1,4,2,\overline{3}]$

Conjecture (Rampersad et al.)

$$E(\mathbf{x}_k) = \frac{k-2}{k-3}, \text{ for } k \geq 5.$$

We resolve the conjecture for $k \leq 8$.

The words x_k are Ostrowski-automatic. To determine the critical exponent:

- Construct a DFAO producing x_k.
- Assert with first-order predicates that the maximum possible exponent of a factor in x_k is $(k-2)/(k-3)$.

The words x_k are Ostrowski-automatic. To determine the critical exponent:

- Construct a DFAO producing x_k.
- Assert with first-order predicates that the maximum possible exponent of a factor in x_k is $(k-2)/(k-3)$.

Observation

The DFAO are small in size if we do everything in LSD-first notation.

We create the numeration system and assert two first-order statements. Example for x_6:

We create the numeration system and assert two first-order statements. Example for x_6:

1. `ost ns6 [0 1 2 1 1] [1 1 1 2];`

We create the numeration system and assert two first-order statements. Example for $\mathbf{x}_6$:

1. `ost ns6 [0 1 2 1 1] [1 1 1 2];`
2. There exists a factor that has exponent 4/3.

```
eval CritExp "?lsd_ns6 Ei,p
  (i>=1) & (p>=1) & (Aj (3*j<p) =>
  X6[i+j] = X6[i+j+p])";
```

Commands

We create the numeration system and assert two first-order statements. Example for $\mathbf{x}_6$:

1. `ost ns6 [0 1 2 1 1] [1 1 1 2];`
2. There exists a factor that has exponent 4/3.
```
eval CritExp "?lsd_ns6 Ei,p
  (i>=1) & (p>=1) & (Aj (3*j<p) =>
  X6[i+j] = X6[i+j+p])";
```
3. There does not exist a factor that has exponent greater than 4/3.
```
eval CritExp "?lsd_ns6 Ẽ i,p
  (i>=1) & (p>=1) &
  (Aj (3*j<=p) => X6[i+j] = X6[i+j+p])";
```

Both predicates produce the `true` automaton for all $5 \leq k \leq 8$, proving the result.

k	States	Memory	Time
5	24	2 GB	30 seconds
6	210	40 GB	5 minutes
7	591	150 GB	45 minutes
8	781	360 GB	2 hours
9	780	—	—
10	1458	—	—

Table 1: Computational statistics.

Factors achieving the critical exponent.

- The factor of $x_6 = 1203410530214\cdots$, $x_6[7..10] = 0530$, has exponent 4/3.
- The factor of $x_7 = 2031405216041\cdots$, $x_7[2..6] = 03140$, has exponent 5/4.
- The factor of $x_8 = 2340526713254\cdots$, $x_8[1..6] = 234052$, has exponent 6/5.

Rich words

- A finite word w is palindrome-rich, or simply *rich* if it contains $|w|$ nonempty distinct palindromic factors.
- An infinite word is *rich* if all of its factors are rich.

Rich words

- A finite word w is palindrome-rich, or simply *rich* if it contains $|w|$ nonempty distinct palindromic factors.
- An infinite word is *rich* if all of its factors are rich.

Examples

- The word **00010110** is rich – contains 8 distinct nonempty palindromes: 0, 00, 000, 1, 010, 101, 11, 0110.
- The word **00101100** is not rich – only 7 distinct palindromes.

Rich words

- A finite word w is palindrome-rich, or simply *rich* if it contains $|w|$ nonempty distinct palindromic factors.
- An infinite word is *rich* if all of its factors are rich.

Examples

- The word **00010110** is rich – contains 8 distinct nonempty palindromes: 0, 00, 000, 1, 010, 101, 11, 0110.
- The word **00101100** is not rich – only 7 distinct palindromes.

Problem

What is the repetition threshold for infinite rich words over a k-letter alphabet?

Known results

Known results

- Pelantová and Starosta (2013) showed that every infinite rich word contains a square, implying that the repetition threshold is greater than 2.

Known results

- Pelantová and Starosta (2013) showed that every infinite rich word contains a square, implying that the repetition threshold is greater than 2.
- Vesti gave upper and lower bounds on the length of the longest square-free rich words (2017). He also proposed the open problem of determining the repetition threshold.

Known results

- Pelantová and Starosta (2013) showed that every infinite rich word contains a square, implying that the repetition threshold is greater than 2.
- Vesti gave upper and lower bounds on the length of the longest square-free rich words (2017). He also proposed the open problem of determining the repetition threshold.

Our contributions

- We make the first progress on Vesti's problem by constructing a binary word that achieves the repetition threshold.
- Our approach is computation-based and utilizes the decision procedures we implemented in `Walnut`.

Morphisms

$$\phi\colon \begin{aligned} 0 &\mapsto 01 \\ 1 &\mapsto 02 \\ 2 &\mapsto 022 \end{aligned} \qquad \tau\colon \begin{aligned} 0 &\mapsto 0 \\ 1 &\mapsto 01 \\ 2 &\mapsto 011 \end{aligned}$$

Morphisms

ϕ: $0 \mapsto 01$, $1 \mapsto 02$, $2 \mapsto 022$

τ: $0 \mapsto 0$, $1 \mapsto 01$, $2 \mapsto 011$

Theorem

The infinite word $\mathbf{r} = \tau(\phi^\omega(0)) = 00100110010\cdots$ is rich, and has the critical exponent $2 + \frac{\sqrt{2}}{2}$.

Morphisms

ϕ:	$0 \mapsto 01$	τ:	$0 \mapsto 0$
	$1 \mapsto 02$		$1 \mapsto 01$
	$2 \mapsto 022$		$2 \mapsto 011$

Theorem

The infinite word $\mathbf{r} = \tau(\phi^\omega(0)) = 00100110010\cdots$ is rich, and has the critical exponent $2 + \frac{\sqrt{2}}{2}$.

Remark: Currie et al. have proved that our word achieves the repetition threshold.

Constructing The Automaton

Observation

The lengths $L_i = |\tau(\phi^i(0))|$ follow the recurrence relation:
$L_0 = 1$, $L_1 = 3$, and $L_i = 2L_{i-1} + L_{i-2}$ for all $i \geq 2$.

Observation

The lengths $L_i = |\tau(\phi^i(0))|$ follow the recurrence relation:
$L_0 = 1$, $L_1 = 3$, and $L_i = 2L_{i-1} + L_{i-2}$ for all $i \geq 2$.

The word **r** might be Pell-automatic.

Observation

The lengths $L_i = |\tau(\phi^i(0))|$ follow the recurrence relation:
$L_0 = 1$, $L_1 = 3$, and $L_i = 2L_{i-1} + L_{i-2}$ for all $i \geq 2$.

The word **r** might be Pell-automatic.

- We require an automaton producing the word **r** for `Walnut` to understand predicates involving **r**.
- We create the adder automaton using the following command:
  `ost pell [0] [2];`

- We construct the DFAO producing the word $\mathbf{r}$ using the L* algorithm by Dana Angluin (1987).

Constructing the DFAO

- We construct the DFAO producing the word **r** using the L* algorithm by Dana Angluin (1987).
- An induction based proof verifies that this DFAO produces the same word as $\tau(\phi^{\omega}(0))$.

- We construct the DFAO producing the word **r** using the L* algorithm by Dana Angluin (1987).
- An induction based proof verifies that this DFAO produces the same word as $\tau(\phi^\omega(0))$.

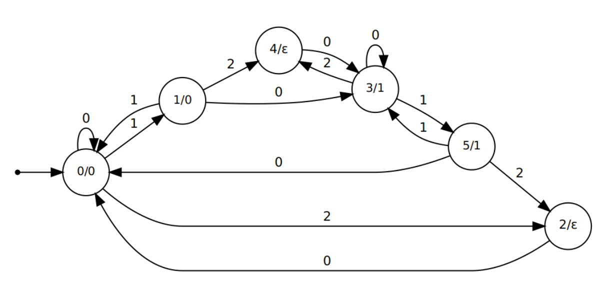


Figure 4: Automaton for the infinite word **r**.

Length-n factors of R starting at indices i and j are equal.

FactorEq(i, j, n)

$\forall k\ (k < n) \implies \mathsf{R}[i+k] = \mathsf{R}[j+k]$

Length-n factor of R starting at index i is a palindrome.

Palindrome(i, n)

$\forall j\ \forall k\ (k < n) \implies \mathsf{R}[i+k] = \mathsf{R}[n-1-k]$

The word $\mathsf{R}[i..i+m-1]$ occurs in the word $\mathsf{R}[j..j+n-1]$.

Occurs(i, j, m, n)

$(m \leq n) \wedge (\exists k\ (k+m \leq n) \wedge \texttt{FactorEq}(i, j+k, m))$

Length-n factors of R starting at indices i and j are equal.

`FactorEq`(i, j, n)	67 states
$\forall k\ (k < n) \implies \mathrm{R}[i + k] = \mathrm{R}[j + k]$	

Length-n factor of R starting at index i is a palindrome.

`Palindrome`(i, n)	27 states
$\forall j\ \forall k\ (k < n) \implies \mathrm{R}[i + k] = \mathrm{R}[n - 1 - k]$	

The word $\mathrm{R}[i..i + m - 1]$ occurs in the word $\mathrm{R}[j..j + n - 1]$.

`Occurs`(i, j, m, n)	1715 states
$(m \leq n) \wedge (\exists k\ (k + m \leq n) \wedge \texttt{FactorEq}(i, j + k, m))$	

Fact

An infinite word is rich if and only if all its factors are rich.
We could also look at only the prefixes instead of all factors.

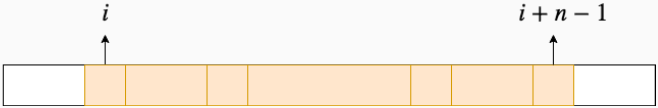
Fact

An infinite word is rich if and only if all its factors are rich.
We could also look at only the prefixes instead of all factors.

Theorem

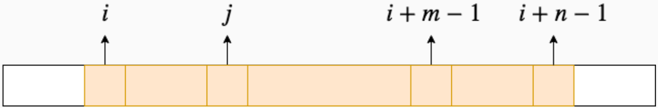
(Glen et al.) A finite word w is rich if and only if every prefix of w has a unioccurrent palindromic suffix.

Figure 5: Constructing the predicate RichFactor(i, n).



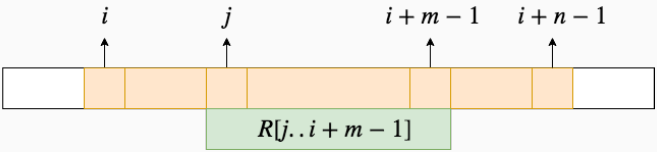
RichFactor(i, n):

Figure 5: Constructing the predicate `RichFactor`(i, n).



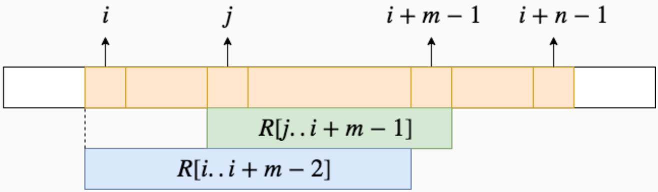
`RichFactor`(i, n): $\forall m, (1 \leq m < n) \implies (\exists j, (i \leq j < i + m) \wedge$

Figure 5: Constructing the predicate `RichFactor`(i, n).



`RichFactor`(i, n): $\forall m, (1 \leq m < n) \implies (\exists j, (i \leq j < i + m) \wedge$
`Palindrome`$(j, i + m - j)$

Figure 5: Constructing the predicate RichFactor(i, n).



RichFactor(i, n): $\forall m, (1 \leq m < n) \implies (\exists j, (i \leq j < i + m) \wedge$ Palindrome($j, i + m - j$) $\wedge \neg$Occurs($j, i, i + m - j, m - 1$))

To determine if **r** is rich, we check if all its prefixes are rich.

R_is_Rich

$\forall n$ `RichFactor`$(0, n)$.

- In `Walnut`, this predicate evaluates to true.
- Conclusion – The infinite word **r** is rich.

Problem

It is difficult to write a first-order predicate to determine the critical exponent because it is an irrational number.

Problem

It is difficult to write a first-order predicate to determine the critical exponent because it is an irrational number.

Solution

Compute the critical exponent as the limit of a monotonic expression for exponents.

Critical exponent

Problem

It is difficult to write a first-order predicate to determine the critical exponent because it is an irrational number.

Solution

Compute the critical exponent as the limit of a monotonic expression for exponents.

Overview

- Compute periods of high powers ($\geq 5/2$).
- Compute the maximal lengths associated with the high-power periods above.

Computing the critical exponent

First, compute periods p such that the word **r** has factors with period p and exponent $> 5/2$.

HighPowerPeriods(p)
$(p \geq 1) \wedge (\exists i \forall j\ (2j \leq 3p) \implies \mathrm{R}[i+j] = \mathrm{R}[i+j+p])$.

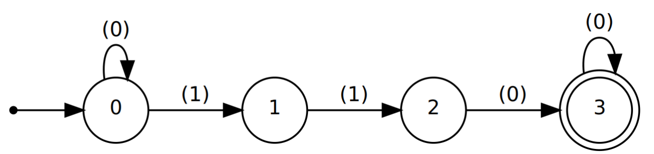


Figure 6: Automaton for the predicate HighPowerPeriods.

Compute pairs (n, p) such that **r** has a factor of length $n + p$ with period p, which cannot be extended with the same period.

MaximalReps(n, p)	17 states
$\exists i (\forall j \; (j < n) \implies \mathsf{R}[i+j] = \mathsf{R}[i+j+p]) \wedge (\mathsf{R}[i+n] \neq \mathsf{R}[i+n+p])$.	

Compute pairs (n, p) such that **r** has a factor of length $n + p$ with period p, which cannot be extended with the same period.

`MaximalReps`(n, p) 17 states

$\exists i(\forall j\ (j < n) \implies \mathrm{R}[i+j] = \mathrm{R}[i+j+p]) \land (\mathrm{R}[i+n] \neq \mathrm{R}[i+n+p])$.

Compute pairs (n, p) where p satisfies `HighPowerPeriods` and $n + p$ is the longest length of any factor with period p.

`HighestPowers`(n, p)

`HighPowerPeriods`$(p)\land$
`MaximalReps`$(n, p)\land$
$(\forall m$ `MaximalReps`$(m, p) \implies m \leq n)$.

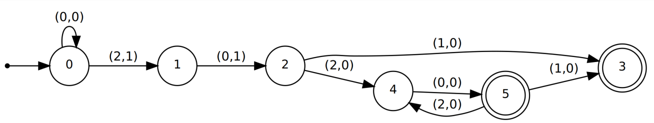


Figure 7: Automaton for the predicate `HighestPowers`.

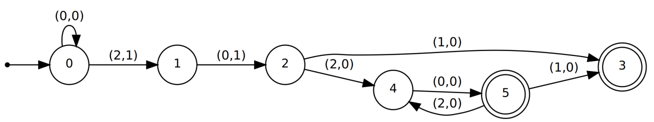


Figure 7: Automaton for the predicate `HighestPowers`.

This automaton accepts pairs (n, p) of the following formats:

1. $\binom{0}{0}^* \binom{2}{1} \binom{0}{1} \binom{2}{0} \binom{0}{0} \left\{ \binom{2}{0} \binom{0}{0} \right\}^*$,
2. $\binom{0}{0}^* \binom{2}{1} \binom{0}{1} \binom{2}{0} \binom{0}{0} \left\{ \binom{2}{0} \binom{0}{0} \right\}^* \binom{1}{0}$.

- **Case 1**: $\binom{0}{0}^* \binom{2}{1} \binom{0}{1} \binom{2}{0} \binom{0}{0} \left\{ \binom{2}{0} \binom{0}{0} \right\}^*$, corresponds to:

$$n = 2 \sum_{1 \le i \le k} P_{2i} = P_{2k+1} - 1, \quad p = P_{2k} + P_{2k-1}$$

- **Case 2**: $\binom{0}{0}^* \binom{2}{1} \binom{0}{1} \binom{2}{0} \binom{0}{0} \left\{ \binom{2}{0} \binom{0}{0} \right\}^* \binom{1}{0}$, corresponds to:

$$n = 1 + 2 \sum_{1 \le i \le k} P_{2i+1} = P_{2k+2} - 1, \quad p = P_{2k+1} + P_{2k}$$

Substituting $m = 2k - 1$ in case 1, and $m = 2k$ in case 2, we notice that the expressions for the exponent coincide:

Substituting $m = 2k - 1$ in case 1, and $m = 2k$ in case 2, we notice that the expressions for the exponent coincide:

$$e = \frac{P_{m+2} + P_{m+1} + P_m - 1}{P_{m+1} + P_m}$$
$$= 2 + \frac{P_{m+1} - 1}{P_{m+1} + P_m}.$$

Substituting $m = 2k - 1$ in case 1, and $m = 2k$ in case 2, we notice that the expressions for the exponent coincide:

$$\begin{aligned} e &= \frac{P_{m+2} + P_{m+1} + P_m - 1}{P_{m+1} + P_m} \\ &= 2 + \frac{P_{m+1} - 1}{P_{m+1} + P_m}. \end{aligned}$$

This expression is increasing with m, and tends to $2 + \sqrt{2}/2$ as $m \to \infty$. Thus, the critical exponent,

$$E(\mathrm{r}) = 2 + \frac{\sqrt{2}}{2}.$$

Definition

Antisquare: a finite word xx' where x' is the binary complement of x.

Definition

Antisquare: a finite word xx' where x' is the binary complement of x.

- Example: 00101101.

Definition

Antisquare: a finite word xx' where x' is the binary complement of x.

- Example: 00101101.
- Meaningful only for the binary alphabet.

Contribution

We construct an infinite binary word that avoids as many antisquares as possible and has a small critical exponent.

Morphisms

$$\varphi\colon \quad \begin{aligned} 0 &\mapsto 001 \\ 1 &\mapsto 01, \end{aligned} \qquad \tau\colon \quad \begin{aligned} 0 &\mapsto 0001 \\ 1 &\mapsto 01. \end{aligned}$$

Morphisms

$$\varphi\colon \quad \begin{aligned} 0 &\mapsto 001 \\ 1 &\mapsto 01, \end{aligned} \qquad \tau\colon \quad \begin{aligned} 0 &\mapsto 0001 \\ 1 &\mapsto 01. \end{aligned}$$

Claim

The infinite word $\mathsf{w} = \tau(\varphi^\omega(0))$ does not have antisquares other than 01 and 10, and has a small critical exponent.

Fibonacci automaton producing the word.

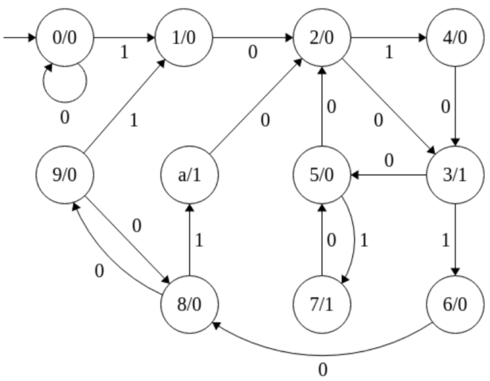


Figure 8: DFAO computing the infinite word w avoiding antisquares.

The word **w** does not contain antisquares other than 01 and 10.

AntisqLengths(p)

$(p >= 1) \wedge (\forall (j < p) \; W[i + j] \neq W[i + j + p])$.

Absence of antisquares

The word `w` does not contain antisquares other than 01 and 10.

`AntisqLengths`(p)

$(p >= 1) \ \wedge \ (\forall \, (j < p) \; W[i+j] \neq W[i+j+p])$.

- `AntisqLengths` accepts only 1 as the input

The word `w` does not contain antisquares other than 01 and 10.

`AntisqLengths`(p)
$(p >= 1) \ \wedge \ (\forall\, (j < p)\ W[i+j] \neq W[i+j+p])$.

- `AntisqLengths` accepts only 1 as the input
  $\Rightarrow$ only antisquares: 01 and 10.

The word **w** does not contain antisquares other than 01 and 10.

`AntisqLengths`(p)
$(p >= 1) \ \wedge \ (\forall \, (j < p) \, W[i + j] \neq W[i + j + p])$.

- `AntisqLengths` accepts only 1 as the input
  $\Rightarrow$ only antisquares: 01 and 10.
- $E(\mathbf{w}) = 2 + \phi$. Computed using Walnut.
  We claim that this is the repetition threshold for infinite binary words avoiding antisquares of length > 2.

Further directions

1. Implementing the 4-input adder that also reads the partial quotients in parallel.

Further directions

1. Implementing the 4-input adder that also reads the partial quotients in parallel.
2. Generalizing the procedures to higher-order numeration systems.

$$q_i = q_{i-1} + q_{i-2} \qquad \text{(Fibonacci)}$$

$$q_i = aq_{i-1} + q_{i-2} \qquad \text{(Ostrowski)}$$

$$q_i = \sum_{i-n \leq j < i} a_j q_j \qquad \text{(Higher-order generalized)}$$

Further directions

1. Implementing the 4-input adder that also reads the partial quotients in parallel.
2. Generalizing the procedures to higher-order numeration systems.

$$q_i = q_{i-1} + q_{i-2} \qquad \text{(Fibonacci)}$$
$$q_i = aq_{i-1} + q_{i-2} \qquad \text{(Ostrowski)}$$
$$q_i = \sum_{i-n \leq j < i} a_j q_j \qquad \text{(Higher-order generalized)}$$

3. An easier way to resolve the repetition threshold conjecture for balanced words. New ideas by Pelantová et al. regarding complementary symmetric Rote words.

Further directions

1. Implementing the 4-input adder that also reads the partial quotients in parallel.
2. Generalizing the procedures to higher-order numeration systems.

$$q_i = q_{i-1} + q_{i-2} \quad \text{(Fibonacci)}$$

$$q_i = aq_{i-1} + q_{i-2} \quad \text{(Ostrowski)}$$

$$q_i = \sum_{i-n \leq j < i} a_j q_j \quad \text{(Higher-order generalized)}$$

3. An easier way to resolve the repetition threshold conjecture for balanced words. New ideas by Pelantová et al. regarding complementary symmetric Rote words.
4. Construct infinite rich words over larger alphabets that achieve the repetition threshold.