Crochemore factorization of Sturmian and other infinite words

Jean Berstel and Alessandra Savelli
Institut Gaspard–Monge
Université de Marne–la–Vallée and CNRS (UMR 8049)

Workshop on Words and Automata
St Petersburg, 7 june 2006

Outline

I. Motivation

- A long motivation
  - Square-free words: A construction by A. Thue
  - Words with many squares
- An even longer motivation: testing square-freeness
  - A $O(n \log n)$ algorithm
  - Centered squares
  - Looking for centered squares in linear time
- The true motivation: testing square-freeness in linear time
  - Crochemore factorization
  - Suffix trees for computing the Crochemore factorization
  - The linear time algorithm

Outline continued)

II. Crochemore factorizations

- The Crochemore factorization of the Fibonacci word
- The Crochemore factorization of standard Sturmian words
- The Crochemore factorization of the Thue-Morse word
- Crochemore factorization and Ziv–Lempel factorization

Repetitions

- A square is a sequence that is repeated. For instance `ti` is a square in `repetition`.
- A square is called a tandem repeat in computational biology.
- A word is square-free if it contains no square.

Questions

- Finding squares is difficult ?
- Avoiding squares is possible ?
- How many square may a word contain ?
- How many square-free words exist ?

A square-free word given by Axel Thue

- Axel Thue gives in 1906 an infinite ternary square-free word, constructed as follows.

- Three step construction, starting with a square-free word, e. g. $abac$

  1. Replace c by $\bar{b}\,\bar{a}$ if c is preceded by a, by $\bar{a}\,\bar{b}$ otherwise:

  $$abac \longrightarrow aba\bar{b}\,\bar{a}$$

  2. Insert a c after each letter:

  $$aba\bar{b}\,\bar{a} \longrightarrow acbcac\bar{b}c\bar{a}c$$

  3. Replace each a by aba and each b by bab, and then erase bars:

  $$acbcac\bar{b}c\bar{a}c \longrightarrow abacbabcabacbcac$$

- Repeat the construction.

Other constructions of this word

The word is

$$abac\ babc\ abac\ bcac\ babc\ abac\ babc\ acbc\ abac\ babc\ abac\ bcac\ babc\ acbc\ abac\ \cdots$$

1. By iterating a (modified) substitution:

$$\begin{array}{lclclcll} a & \longmapsto & abac & \qquad & c & \longmapsto & bcac & \text{if } c \text{ is preceded by } a \\ b & \longmapsto & babc & & c & \longmapsto & acbc & \text{otherwise} \end{array}$$

2. By iterating a substitution on four letters and then identifying two of them:

$$\begin{array}{lclclcl} a & \longmapsto & abac' & \qquad & c' & \longmapsto & bc''ac' \\ b & \longmapsto & babc'' & & c'' & \longmapsto & ac'bc'' \end{array}$$

and then erase the primes and seconds.

3. By a finite automaton yields explicitly the value of the word at each position:

Words with many squares

Theorem *At most $2n$ distinct squares may occur in a word of length n.*

This has been improved to $2n - \Theta(\log n)$.

Example The word $ababaababaabab$ of length 14 contains 9 squares (this is maximal for a 14-letter word):

a
ab, ba
aba
$ababa, babaa, abaab, baaba, aabab$

Open It is not known whether there exists a word of length n having more than n occurrences of distinct squares.

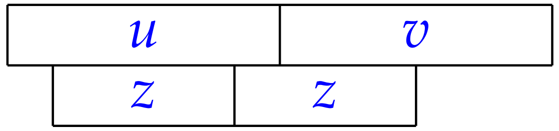
Consider the word $u_n = w_1 w_2 \cdots w_n$, where $w_i = 0^{i+1}10^i10^{i+1}1$.
It has length $3n^2/2 + 13n/2$ and more than $3n^2/2 + 7n/2 - 2$ distinct squa-res.

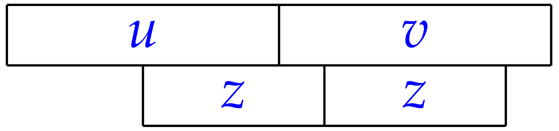
Example The word $u_2 = 00101001\,00010010001$ has length 19 and 11 squa-res.

Detecting squares in a word: A $O(n \log n)$ algorithm

There exists a linear time algorithm for testing whether a word is square-free.

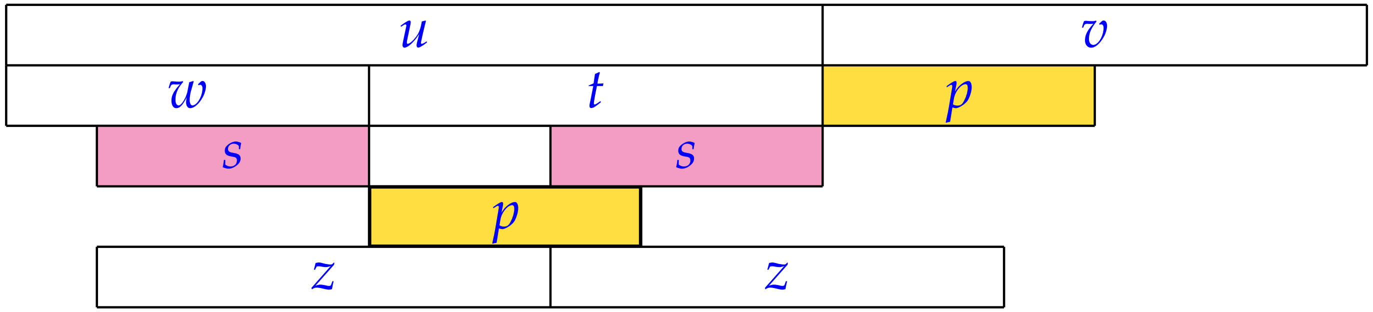
- A square zz is left-centered (right-centered) in (u, v) if zz is a square in uv and the right (left) z overlaps (u, v):





- A word $x = uv$ is square-free if u and v are square-free and if (u, v) has no centered square.
- If one can test centered squarefreeness in linear time, then this gives an $O(n \log n)$ algorithm ($n = |x|$).

Detecting centered squares in a word



- $p = t \wedge v$ is the longest common prefix of t and v
- $s = w \vee u$ is the longest common suffix of w and u
- (u, v) has a left-centered square if and only if there is a factorization $u = wt$, with nonempty t, such that

$$|p| + |s| \geq |t| \,.$$

- First miracle: the computation of all $t \wedge v$, for all suffixes t of u, can be performed in time $O(|u|)$.
- So, testing whether (u, v) has no left (right) centered square can be done in time $(O(|u|)$ (resp. $(O(|v|)$).

A linear time algorithm

A linear time algorithm for testing whether a word is square-free is based on the socalled c-factorization (for *Crochemore*-factorization):

$$c(x) = (x_1, x_2, \ldots, x_m)$$

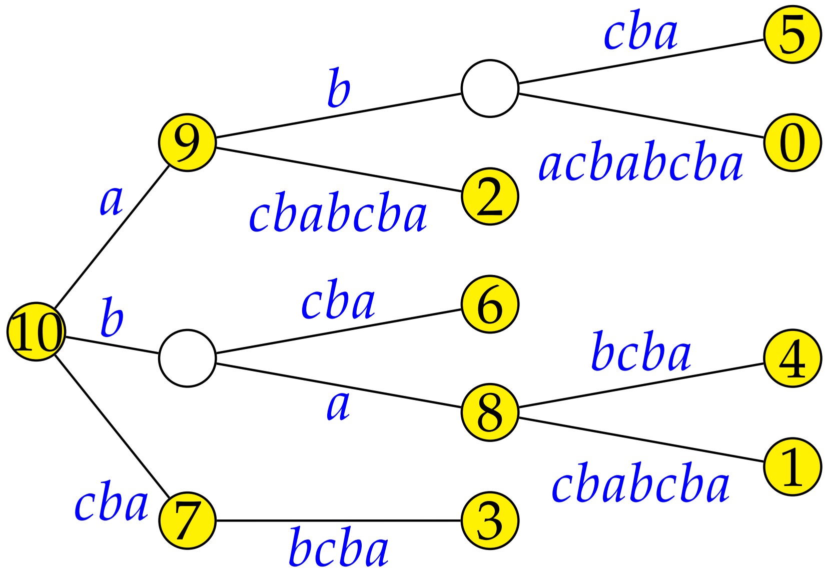
where each x_k is either a fresh letter, or is the longest factor that appears already before.

$$\begin{aligned} c(ababaab) &= a|b|aba|ab \\ c(abacbabcba) &= a|b|a|c|ba|b|cba \end{aligned}$$

The efficient computation of the c-factorization of x uses the suffix tree of the word x.

The suffix tree of a word

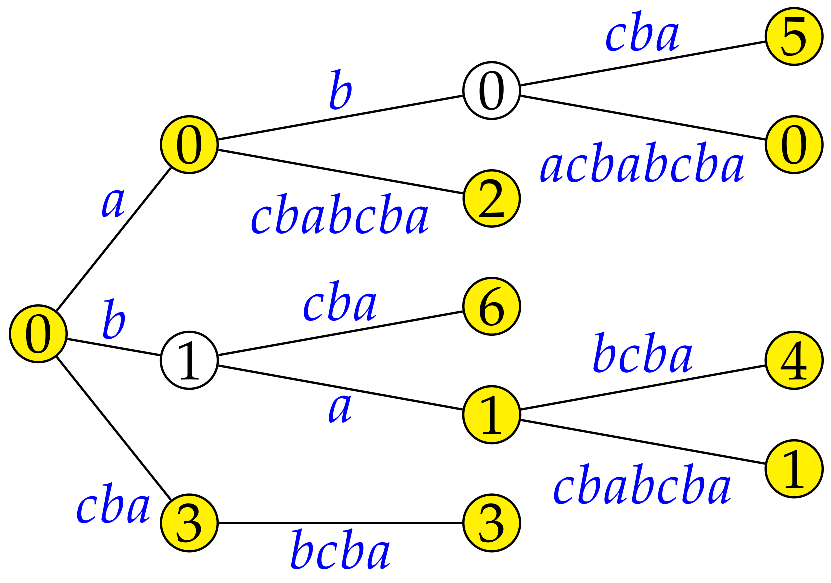
This is the suffix tree of *abacbabcba*.



Second miracle : the suffix tree of a word can be computed in linear time.

Augmented suffix tree

At each node, the first occurrence of the factor is reported. For *abacbabcba*:



This gives in linear time the c-factorization:

$$c(abacbabcba) = a|b|a|c|ba|b|cba$$

Squares in c-factorizations

Theorem *Let $c(x) = (x_1, \ldots, x_k)$ be the c-factorization of x. Then x is square-free iff the following hold for all j*

1. *The occurrence of x_j in $c(x)$ and the first occurrence of x_j do not overlap.*
2. *(x_{j-1}, x_j) has no centered square,*
3. *$(x_1 \cdots x_{j-2}, x_{j-1}x_j)$ has no right centered square.*

Cost for each j:

1. $O(1)$.
2. $|x_{j-1}| + |x_j|$
3. $|x_{j-1}| + |x_j|$

So total cost is linear in the length of x.

Fibonacci word

Defined by $f_0 = a$, $f_1 = ab$, $f_{n+2} = f_{n+1}f_n$. Length of f_n is F_n.

$$\begin{aligned}
F_0 &= 1 & f_0 &= a \\
F_1 &= 2 & f_1 &= ab \\
F_2 &= 3 & f_2 &= aba \\
F_3 &= 5 & f_3 &= abaab \\
F_4 &= 8 & f_4 &= abaababa \\
F_5 &= 13 & f_5 &= abaababaabaab \\
F_6 &= 21 & f_6 &= abaababaabaababaababa \\
F_7 &= 34 & f_7 &= abaababaabaababaababaabaababaabaab
\end{aligned}$$

The infinite Fibonacci word has all finite Fibonacci words as prefixes.

Interpretation of numerical properties

Numerical relation

$$F_n = 2 + F_0 + F_1 + \cdots + F_{n-2}$$

e.g. $F_6 = 21 = 2 + 1 + 2 + 3 + 5 + 8$.

String interpretation

$$f_n = ab f_0 f_1 \cdots f_{n-2}$$

e.g. $f_6 = abaababaabaababaababa$.
Noncommutativity of words gives richer interpretations:

$$f_n = f_0^R f_1^R \cdots f_{n-2}^R (ba|ab)$$

e.g. $f_6 = abaababaabaababaababa$.
One gets even another interpretation:

$$f_n = a w_0 w_1 \cdots w_{n-2} (a|b)$$

e.g. $f_6 = abaababaabaababaababa$.
The second factorization is (almost) the c-factorization.

Crochemore factorization of the Fibonacci word

Comparison of three factorizations:

- h: as a product of finite Fibonacci words
- w: as a product of singular words
- c: as a product of reversals of Fibonacci words

$$
\begin{array}{r|c|c|c|c|c|c|c}
h: & a & b & a & a\ b & a\ b\ a & a\ b\ a\ a\ b & a\ b\ a\ a\ b\ a\ b\ a & \cdots \\
w: & a & b & a\ a & b\ a\ b & a\ a\ b\ a\ a & b\ a\ b\ a\ a\ b\ a\ b & \cdots \\
c: & a & b & a & a\ b\ a & b\ a\ a\ b\ a & a\ b\ a\ b\ a\ a\ b\ a & \cdots
\end{array}
$$

Theorem *The c-factorization of the Fibonacci word f is*

$$c(f) = (a, b, a, aba, baaba, \ldots) = (a, b, a, f_2^R, f_3^R, \ldots)$$

Crochemore factorization of standard Sturmian words

A standard Sturmian word is defined by a directive sequence $(d_1, d_2, \ldots)$. It is the limit of the words s_n with

$$s_{-1} = b,\ s_0 = a, \text{ and } s_n = s_{n-1}^{d_n} s_{n-2},$$

Theorem *Let s be the standard Sturmian word defined by the directive sequence $(d_1, d_2, \ldots)$. Then*

$$c(s) = (a, a^{d_1-1}, b, a^{d_1}\widetilde{s}_1^{d_2-1}, \widetilde{s}_2^{\,d_3}, \widetilde{s}_3^{\,d_4}, \ldots, \widetilde{s}_n^{\,d_{n+1}}, \ldots)$$

Here $\widetilde{w}$ is the reversal of w.

Crochemore factorization of the Thue-Morse word

The *Thue-Morse infinite word* is

$$t = abbabaabbaababba \cdots$$

obtained by iterating the morphism τ defined by $\tau(a) = ab$, $\tau(b) = ba$. One gets

$$c(t) = a|b|b|ab|a|abba|aba|bbabaab|abbaab|babaabbaababba| \cdots$$

Each long enough factor is obtained from a previous one by applying the morphism τ.

Theorem *The c-factorization $c(t) = (c_1, c_2, \ldots)$ of the Thue-Morse sequence is*

$$(a, b, b, ab, a, abba, aba, bbabaab, c_9, c_{10}, \ldots)$$

where $c_{n+2} = \tau(c_n)$ for every $n \geq 8$.

So, $c_9 = abbaab = \tau(aba)$, $c_{10} = babaabbaababba = \tau(bbabaab)$.

Synchronization is late !

Crochemore factorization of generalized Thue-Morse words

Better behaviour !

Let $t^{(m)}$ be the word on $\{a_1, a_2, \ldots, a_m\}$ obtained as the limit of the morphism τ_m defined by

$$\tau_m(a_i) = a_i a_{i+1} \cdots a_m a_1 \cdots a_{i-1} \quad (i = 1, \ldots, m).$$

Theorem *For $m \geq 3$, the c-factorization $c(t^{(m)}) = (c_1^{(m)}, c_2^{(m)}, \ldots)$ satisfies the relation $c_{n+2(m-1)}^{(m)} = \tau_m(c_n)$ for $n > m$.*

Example $m = 3$. Morphism $0 \mapsto 012, 1 \mapsto 120, 2 \mapsto 201$. $c_{n+4}^{(3)} = \tau_3(c_n)$ for $n > 3$.

$$\begin{aligned} c(t^{(3)}) \;=\; & 0|1|2| \\ & 12|0|20|1| \\ & 120201|012|201012|120| \\ & 120\,201\,012\,201\,012\,120|012120201| \cdots \end{aligned}$$

Crochemore factorization of the period-doubling word

Define $\delta(a) = ab$, $\delta(b) = aa$, and set $q_0 = a$ and $q_{n+1} = \delta(q_n)$. Thus

$$\begin{aligned} q_0 &= a & q_3 &= abaaabab \\ q_1 &= ab & q_4 &= abaaababababaaabaa \\ q_2 &= abaa \end{aligned}$$

The limit q is the *period doubling sequence*

$$q = a\,ba\,aaba\,babaaaba\,aabaaabababaaaba \cdots (= q_0^R q_1^R q_2^R q_3^R q_4^R \cdots)$$

Theorem *The c-factorization of q is*

$$c(q) = (a, q_0^S, q_0^R, q_1^S, q_1^R, q_2^S, q_2^R, \ldots).$$

Here w^R is the reversal, and w^S is obtained from w^R by replacing the first letter by its opposite.

$$c(q) = a|b|a|aa|ba|baba|aaba|aabaaaba|babaaaba| \cdots$$

Ziv-Lempel factorization

The *Ziv-Lempel* or *z-factorization* $z(x)$ of a word x is

$$z(x) = (y_1, y_2, \ldots, y_m, y_{m+1}, \ldots)$$

where y_m is the shortest prefix of $y_m y_{m+1} \cdots$ which occurs only once in $y_1 y_2 \cdots y_m$.

Example For $x = aabaaccbaabaabaa$.

$$\begin{aligned} c(x) &= (a, a, b, aa, c, c, baa, baabaa) \\ z(x) &= (a, ab, aac, cb, aabaab, aa). \end{aligned}$$

Crochemore factorization versus Ziv-Lempel factorization

The factorizations are closely related:

Proposition *Let* $(c_1, c_2, \ldots)$ *and* $(z_1, z_2, \ldots)$ *be the Crochemore and the Ziv-Lempel factorizations of a word* w*, then the following hold for each* i, j*.*

- *If* $|c_1 \cdots c_{i-1}| \geq |z_1 \cdots z_{j-1}|$ *and* $|c_1 \cdots c_i| < |z_1 \cdots z_j|$*,* *then* $|z_1 \cdots z_j| = |c_1 \cdots c_i| + 1$*.*
- *If* $|z_1 \cdots z_{j-1}| < |c_1 \cdots c_i| \leq |z_1 \cdots z_j|$*, then* $|c_1 \cdots c_{i+1}| \leq |z_1 \cdots z_{j+1}|$*.*

An example

Consider the word

$$v = abaaababababaaabaa\cdots$$

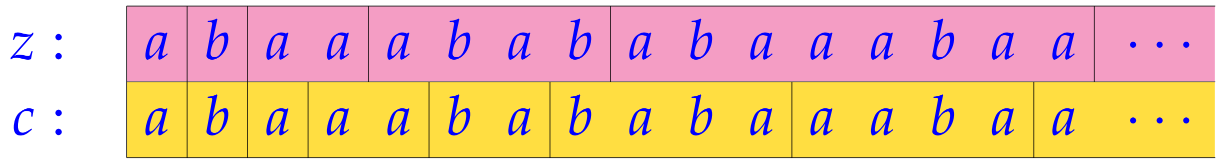
defined as the limit of the sequence

$$v_0 = a, \quad v_{2n+1} = v_{2n} b v_{2n}, \quad v_{2n} = v_{2n-1} a v_{2n-1}$$

Thus

$$\begin{aligned} v_0 &= a & v_2 &= abaaaba \\ v_1 &= aba & v_3 &= abaaababababaaaba \end{aligned}$$

Each Ziv-Lempel factor of v properly includes a Crochemore factor ending just a letter before it, as illustrated in this figure:


$z:$ a | b | a a | a b a b | a b a a a b a a | ⋯
$c:$ a | b | a | a a | b a | b a b a | a a b a | a ⋯

Open problems

- characterize c-factorizations of automatic words.
- are c-factorizations and z-factorizations really different?