Factorizations of Infinite Words

Jean Berstel

Institut Gaspard–Monge, Université de Marne–la–Vallée and CNRS (UMR 8049)

Workshop on Fibonacci Words, september 2006

Outline

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.

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	$=$	$abaababaabaababaababaabaababaabaab$

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

Interpretation of numerical properties

A numerical relation: $F_n = 2 + F_0 + F_1 + \cdots + F_{n-2}$
e.g. $F_6 = 21 = 2 + 1 + 2 + 3 + 5 + 8$.

Three string interpretations:

1. $f_n = abf_0f_1 \cdots f_{n-2}$
   e.g. $f_6 = abaababaabaababaababa$
2. $f_n = f_0^R f_1^R \cdots f_{n-2}^R (ba|ab)$
   e.g. $f_6 = abaababaabaababaababa$
3. $f_n = aw_0w_1 \cdots w_{n-2}(a|b)$
   e.g. $f_6 = abaababaabaababaababa$

Lyndon factorization

The *singular factorization*

$$f_6 = abaababaabaababaababa$$

is closely related to the *Lyndon factorization*

$$f_6 = abaababaabaababaababa = \ell_1\ell_3\ell_5 a$$

where

$$\ell_1 = ab = aw_0$$

$$\ell_3 = aabab = w_1w_2$$

$$\ell_5 = aabaababaabab = w_3w_4$$

and more generally $\ell_{2n+1} = w_{2n-1}w_{2n}$.

Ziv-Lempel and Crochemore

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

$$z(x) = (y_1, y_2, \ldots, y_m, \ldots)$$

where y_m is the shortest factor that did not appear before.

The *Crochemore* or *c-factorization* is

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

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

Example

$x = a|abaaccbaabaabaa$

$c(x) = (a, \ldots)$

Ziv-Lempel and Crochemore

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

$$z(x) = (y_1, y_2, \ldots, y_m, \ldots)$$

where y_m is the shortest factor that did not appear before.

The *Crochemore* or *c-factorization* is

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

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

Example

$$x = aa|baaccbaabaabaa$$

$$c(x) = (a, a, \ldots)$$

Ziv-Lempel and Crochemore

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

$$z(x) = (y_1, y_2, \ldots, y_m, \ldots)$$

where y_m is the shortest factor that did not appear before.

The *Crochemore* or *c-factorization* is

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

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

Example

$$x = aab|aaccbaabaabaa$$

$$c(x) = (a, a, b, \ldots)$$

Ziv-Lempel and Crochemore

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

$$z(x) = (y_1, y_2, \ldots, y_m, \ldots)$$

where y_m is the shortest factor that did not appear before.

The *Crochemore* or *c-factorization* is

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

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

Example

$$x = aabaa|ccbaabaabaa$$

$$c(x) = (a, a, b, aa, \ldots)$$

Ziv-Lempel and Crochemore

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

$$z(x) = (y_1, y_2, \ldots, y_m, \ldots)$$

where y_m is the shortest factor that did not appear before.

The *Crochemore* or *c-factorization* is

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

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

Example

$$x = aabaac|cbaabaabaa$$

$$c(x) = (a, a, b, aa, c, \ldots)$$

Ziv-Lempel and Crochemore

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

$$z(x) = (y_1, y_2, \ldots, y_m, \ldots)$$

where y_m is the shortest factor that did not appear before.

The *Crochemore* or *c-factorization* is

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

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

Example

$$x = aabaacc|baabaabaa$$

$$c(x) = (a, a, b, aa, c, c, \ldots)$$

Ziv-Lempel and Crochemore

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

$$z(x) = (y_1, y_2, \ldots, y_m, \ldots)$$

where y_m is the shortest factor that did not appear before.

The *Crochemore* or *c-factorization* is

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

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

Example

$$x = aabaaccbaa|baabaa$$

$$c(x) = (a, a, b, aa, c, c, baa, \ldots)$$

Ziv-Lempel and Crochemore

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

$$z(x) = (y_1, y_2, \ldots, y_m, \ldots)$$

where y_m is the shortest factor that did not appear before.

The *Crochemore* or *c-factorization* is

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

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

Example

$$x = aabaaccbaabaabaa$$

$$c(x) = (a, a, b, aa, c, c, baa, baabaa)$$
$$z(x) = (a, ab, aac, cb, aabaab, aa)$$

Z and C for Fibonacci

The *Ziv-Lempel* factorization of the Fibonacci word f is

$$z(f) = abaababaabaababaabab\cdots$$

that is the singular factorization.

The *Crochemore* factorization of the Fibonacci word f is

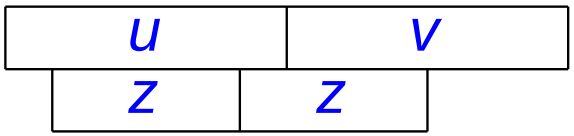
$$c(f) = abaababaabaababaabab\cdots$$

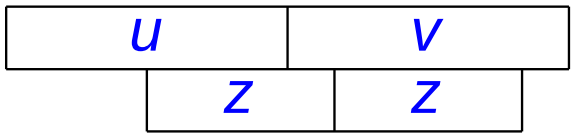
which is basically the factorization into reversals.

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)

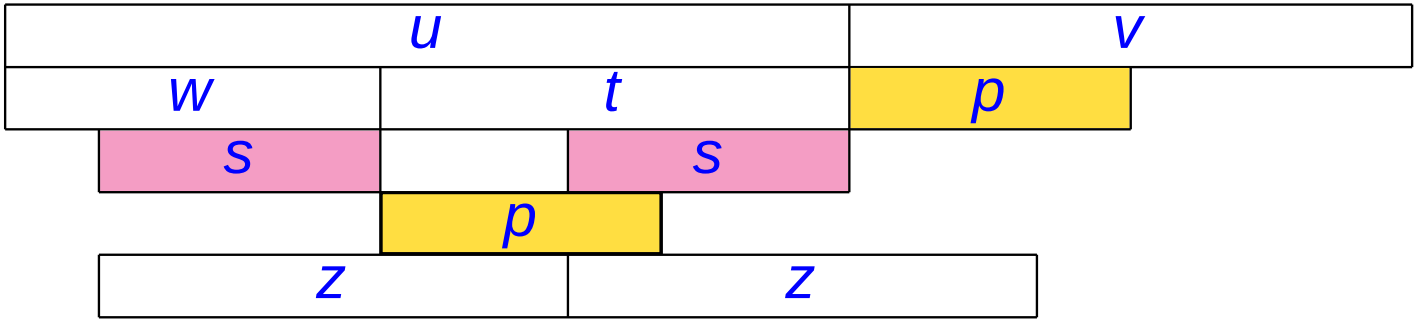
- A square zz is left-centered (right-centered) in (u, v) if zz is a square in uv and the right (left) z overlaps (u, v):





- 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 square-freeness is based on the *Crochemore*-factorization:

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

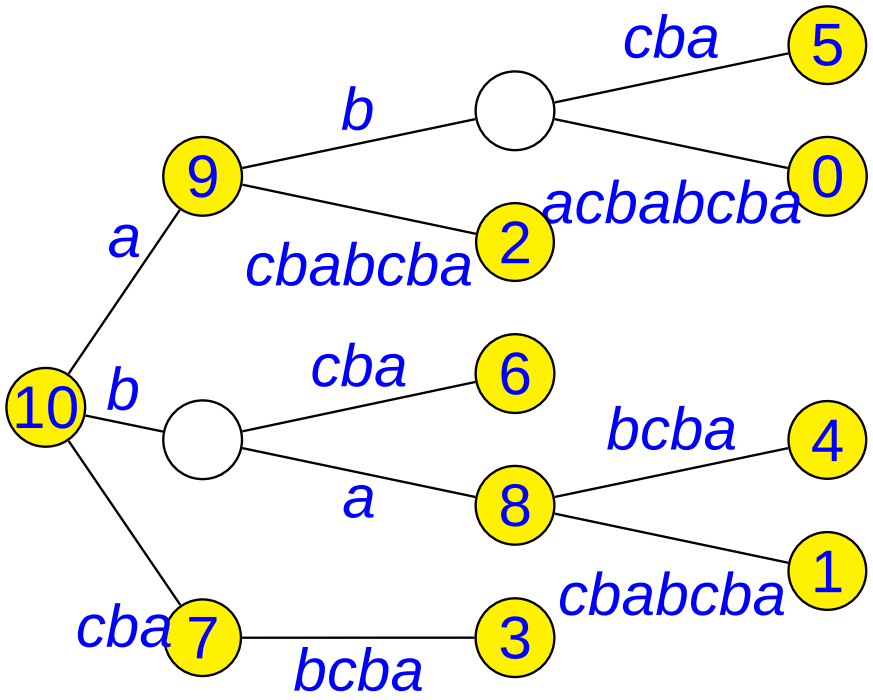
where each x_k is either a fresh letter, or is the longest factor that appears already before.

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

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

The suffix tree of a word

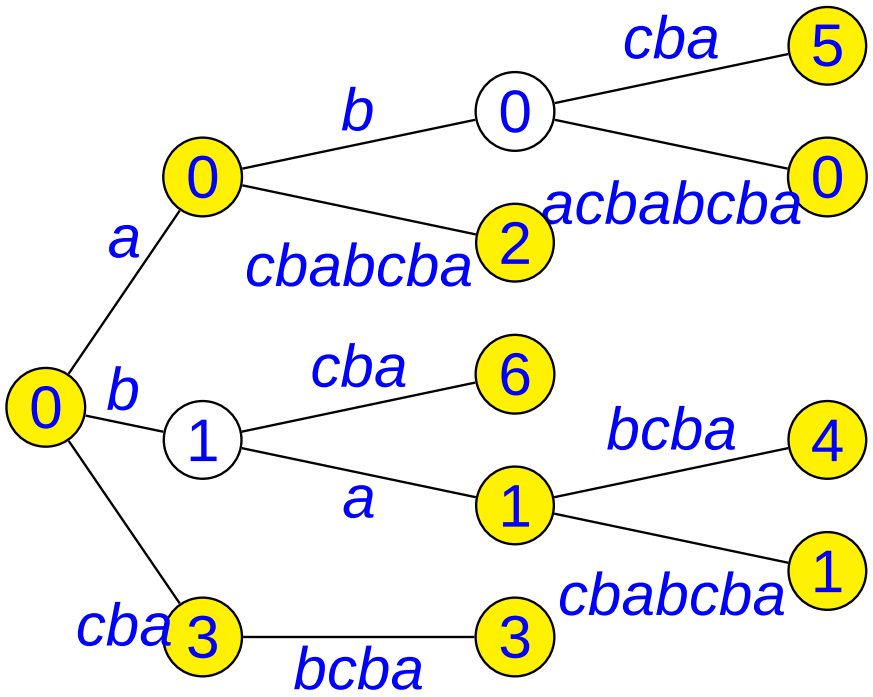
This is the suffix tree of *abacbabcba*. (In yellow the starting index of the position.)



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 (Crochemore)

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.

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

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	$\cdots$		

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.

Prouhet-Thue-Morse

For $n = 3$, take the morphism

$$0 \mapsto 012 \quad 1 \mapsto 120 \quad 2 \mapsto 201$$

One gets

$$s = 012\,120\,201\,120\,201\,012\cdots$$

Also $s(n)$ is the sum of the digits of n in base 3 modulo 3.

E. M. Wright in: Prouhet's 1851 solution of the Tarry-Escott problem of 1910:

Prouhet's note is no more than an "Extrait par l'auteur" of a "Mémoire présenté" to the Academy. The secretaire-archiviste of the Academy was kind enough to inform me that the memoir was returned to the author in 1852. But there appears to be no trace in the literature of its publication.

Construction

"In order to group integers, write clockwise the indices $0, 1, 2, \ldots, n-1$ on a circle. Then read the indices clockwise, and omit one index at each tour, two indices after n tours,three after n^2 tours, et ainsi de suite. The indices show to which group beongs each term."

For $n = 3$

0	·	5	7	·	11	13	15	·	19	21	·	26
1	3	·	8	9	·	14	16	·	20	22	24	·
2	4	6	·	10	12	·	17	18	·	23	25	·

or equivalently

$$012\not{0}120\not{1}201\not{2}012\not{0}120\not{1}201\not{2}012\not{0}120\not{1}201\not{2}012\not{0}120\not{1}201\not{2}012\cdots$$

Thue-Morse

For $n = 2$:

$$t = 0110100110010110\cdots$$

with partition

0				3			5	6			9	10		12			15
	1	2			4					7	8				11			13	14	

A solution to the Tarry-Escott problem:

$$0 + 3 = 1 + 2$$

$$0^2 + 3^2 + 5^2 + 6^2 = 1^2 + 2^2 + 4^2 + 7^2 \quad (= 70)$$

$$0^3 + 3^3 + 5^3 + 6^3 + 9^3 + 10^3 + 12^3 + 15^3$$

$$= 1^3 + 2^3 + 4^3 + 7^3 + 8^3 + 11^3 + 13^3 + 14^3$$

$$(= 7200)$$

Crochemore factorization of the Thue-Morse word

The *Thue-Morse infinite word* is obtained by iterating the morphism τ defined by $\tau(a) = ab$, $\tau(b) = ba$. One gets

$$c(t) = abbabaabbaababbabaab|abbaabbabaabbaababba\cdots$$

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

Synchronizes quite late !

Crochemore factorization of generalized Thue-Morse words

Let $t^{(m)}$ be the word on $\{0, \ldots, m-1\}$ obtained by the morphism τ_m defined by $\tau_m(a) = a(a+1)\cdots(m-1)01(a-1)$ for $a = 0, \ldots, m-1$.

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 $n > 3$.

$$c(t^{(3)}) = 012120201$$
$$120201012201012120$$
$$120\,201\,012\,201\,012\,120012120201\cdots$$

Period-doubling word

Define $\delta(a) = ab$, $\delta(b) = aa$, and set $q_0 = a$ and $q_{n+1} = \delta(q_n)$.
Thus $q_0 = a$ $q_1 = ab$ $q_2 = abaa \cdots$.
The limit is the *period doubling sequence*

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

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.

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

Crochemore versus Ziv-Lempel

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

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

Thus

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

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	$\cdots$
c :	a	b	a	a	a	b	a	b	a	b	a	a	a	b	a	a	$\cdots$

Open problems

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