On the k-Abelian Equivalence Relation

Robert Mercaș

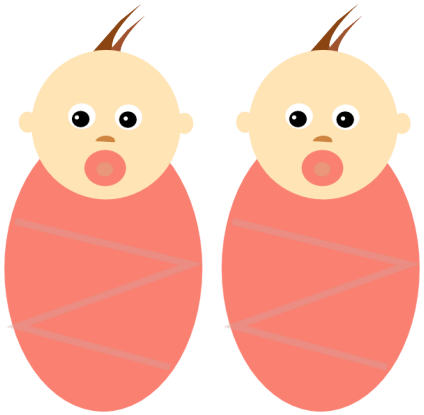
Kiel University

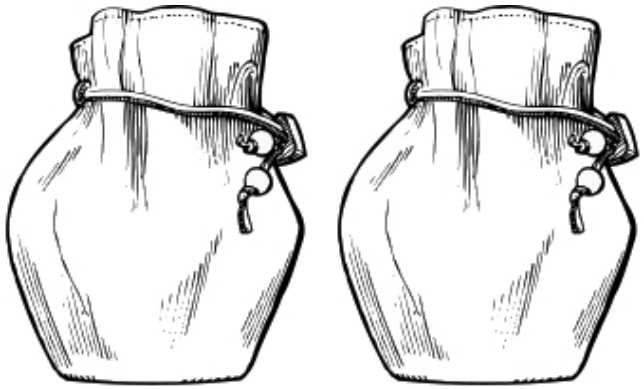
King's College London

Theory Day in Computer Science

Bucharest, September 2014

Motivation





[Source: Google Images]

Definitions

Words u and v over Σ are *equivalent* if $u = v$.

Words u and v over Σ are *abelian equivalent* if $|u|_a = |v|_a$ for every $a \in \Sigma$.

Definition

Two words u and v are *k-abelian equivalent*:

- if $|u|_t = |v|_t$ for every word t of length at most k.
- if $|u|_t = |v|_t$ for every word t of length k, $\mathrm{pref}_{k-1}(u) = \mathrm{pref}_{k-1}(v)$ and $\mathrm{suff}_{k-1}(u) = \mathrm{suff}_{k-1}(v)$.

We denote this by $u \equiv_k v$.

A *k-abelian nth power* is a word $u_1 u_2 \dots u_n$, where $u_1, u_2 \dots, u_n$ are k-abelian equivalent.

Example k-abelian equivalence

$$u \equiv_3 v$$

$$u = abbbaaaba,\ v = abaaabbba, \text{ and } k = 3$$

$$(u,3) = \{(aaa,1),(aab,1),(aba,1),(abb,1),(bba,1),(bbb,1)\} = (w,3)$$
$$(u,2) = \{(aa,2),(ab,2),(ba,2),(bb,2)\} = (w,2)$$
$$(u,1) = \{(a,5),(b,4)\} = (w,1)$$

$$x \not\equiv_2 v$$

$$x = aba,\ y = bab, \text{ and } k = 2$$

$$(x,2) = \{(ab,1),(ba,1)\} = (y,2)$$
$$(x,1) = \{(a,2),(b,1)\} \neq \{(a,1),(b,2)\} = (y,2)$$

Classical avoidability

We say that an n-power is avoidable over some alphabet size if we can construct an infinite word containing no consecutive n repetitions.

hotshots

bananana

Theorem (Prouhet-Thue-Morse)

There exists an infinite binary word containing no consecutive three repetitions of the same factor.

Theorem (Thue)

There exists a ternary infinite word that contains no consecutive repetitions.

Abelian avoidability

We say that an abelian-n-power is avoidable over some alphabet size if we can construct an infinite word containing no consecutive n factors that are permutations of each other.

*intes**tines***

*de**ed**ed*

Theorem (Dekking)

There exists an infinite word defined over a ternary alphabet that contains no 3 consecutive permutations of the same factor.

Theorem (Keränen)

There exists an infinite word defined over a quartic alphabet that contains no abelian repetition.

[Huova, Karhumäki, Saarela, Saari: Local Squares, Periodicity and Finite Automata, 2011]

The longest ternary word that avoids 2-abelian squares has length 537:

abcbabcacbacabacbabcbacabcbabcabacabcacbacabacbabcbacbcacbabcacbcabc
babcabacbabcbacbcacbacabacbabcbacabcbabcabacabcacbacabacbabcbacbcacb
acabacbcabacabcacbcabcbacbcacbacabacbabcbacbcacbabcacbcabcbabcabacba
bcbacbcacbacabacbabcbacabcbabcabacabcacbacabacbabcbacbcacbacabacbcab
acabcacbcabcbabcabacabcacbacabacbabcbacabcbabcabacabcacbcabcbabcabac
babcbacbcacbabcacbcabcbabcabacabcacbcabcbacbcacbacabacbcabacabcacbca
bcbabcabacabcacbacabacbabcbacabcbabcabacabcacbcabcbabcabacbabcbacbca
cbabcacbcabcbabcabacabcacbacabacbabcbacabcbabcabacabcacbabcba

Problem

Does there exist an infinite binary word that avoids 2-abelian cubes?
Does there exist an infinite binary word that avoids 3-abelian squares?

What we know!

Avoidability of squares				
Size alphabet	type of equivalence			
	$=$	$\equiv_3$	$\equiv_2$	$\equiv_a$
2	−	−	−	−
3	+	?	−	−
4	+	+	+	+

Avoidability of cubes			
Size alphabet	type of equivalence		
	$=$	$\equiv_2$	$\equiv_a$
2	+	?	−
3	+	+	+

Theorem

Over a binary alphabet 8-abelian-cubes are avoidable.

[Huova, Karhumäki, Saarela.: Problems in between words and abelian words: k-abelian avoidability , 2012]

Theorem

The word $\rho(w)$ is 6-abelian cube-free for all abelian square-free words w.

$$\rho(a) \mapsto babaab\ a\ abbaba\ a\ baabba\ b,$$
$$\rho(b) \mapsto babaab\ a\ abbaba\ b\ baabba\ b,$$
$$\rho(c) \mapsto baabba\ a\ babaab\ a\ abbaba\ b,$$
$$\rho(d) \mapsto baabba\ a\ babaab\ b\ abbaba\ b.$$

Theorem

If w is an abelian cube-free word that has no factor of the form $apbqbrc$ for any abelian equivalent words p, q, r, then $\delta(w)$ is 5-abelian cube-free.

$$\delta(a) \mapsto babaab\ a\ abbaba\ b\ baabba\ a,$$
$$\delta(b) \mapsto baabba\ a\ babaab\ b\ abbaba\ a,$$
$$\delta(c) \mapsto baabba\ a\ babaab\ b\ abbaba\ b.$$

[M., Saarela.: 5-abelian cubes are avoidable on binary alphabets, 2012]

Twist: use of partial words result

$$\xi(a) = babaab\diamond abbaba\diamond \qquad \text{and} \qquad \xi(b) = baabba\diamond.$$

This morphism provides a way to construct an infinite cube-free partial word in which each length seven factor contains a hole symbol, denoted $\diamond$.

Theorem

Mapping the Prouhet-Thue-Morse sequence with ξ creates a string that remains cube-free no matter how we replace the $\diamond$ symbols by letters.

further results

Theorem

The morphism defined by

$$a \mapsto babbabaabba, \qquad b \mapsto babbabaabaab, \qquad b \mapsto babbaabbaabaa,$$

maps every abelian cube-free ternary word to a 4-abelian cube-free word.

Theorem

The morphism defined by

$$a \mapsto ababa, \qquad b \mapsto abbaaba, \qquad c \mapsto abbabba,$$

maps the Dekking word to a 3-abelian cube-free word.

[M., Saarela.: 3-abelian cubes are avoidable on binary alphabets, 2013]

Something on squares as well

Theorem

Every ternary infinite pure morphic word contains a 3-abelian square.

Theorem

Every ternary infinite pure morphic word contains a k-abelian square for any $k \geq 1$.

[Huova.: On Unavoidability of k-abelian Squares in Pure Morphic Words, 2012]

Theorem

There exists an infinite word that avoids 64-abelian squares.

[Huova: Existence of an infinite ternary 64-abelian square-free word, 2013]

2-abelian-cube-free

Following an idea of Carpi for abelian n-power free preserving morphisms, Rao creates a series of conditions for k-abelian-n-power-free morphisms.

[Rao: On some generalizations of abelian power avoidability, 2013]

Theorem

$$a \mapsto aabaababaabbaabaababaabaabbaabaabbaababbabab$$
$$b \mapsto aabaabbaabaabbaabbabbaabbabbaabaabbaabbabbabab$$
$$c \mapsto aabbabbababbabaababbababbabaababaabaabbabbabab$$

Theorem

$$a \mapsto aabaababaabbaabaababaabaabbaabaabbaabbabb$$
$$b \mapsto ababbabbaabbabbaabbaabaabbaabbabb$$
$$c \mapsto ababbabaababaabbabbaabbabb$$

3-abelian-square-free

Theorem

$$
\begin{aligned}
a &\mapsto abacabcacbabcabacabcbacbc \\
b &\mapsto abacbabcabacbcabcbabcacbc \\
c &\mapsto abacbabcbacbcacbacabcacbc \\
d &\mapsto abcbacabcacbacabcbabcacbc
\end{aligned}
$$

Theorem

$$
\begin{aligned}
a &\mapsto abcabacabcacbcabcbabcabacb \\
b &\mapsto abcacbcabcbacbcabacb \\
c &\mapsto abcacbacabacb \\
d &\mapsto abcbacabcb
\end{aligned}
$$

Open Problems

1. Is there a pure morphic binary word which avoids 2-abelian-cubes? (Huova - 2003)
2. Which is the smallest k (if any) such that arbitrarily long k-abelian-squares are avoidable over a ternary alphabet? (Rao - 2013)
3. Can we avoid abelian-squares of the form uv, with $|u| \geq 2$, over a binary alphabet? (Mäkelä - 2003)
4. Can we avoid abelian-cubes of the form uvw, with $|u| \geq 2$, over a binary alphabet? (Mäkelä - 2003)

Fresh:

[Rao, Rosenfeld: Avoidability of long k-abelian repetitions, NEXT WEEK]

Claim

There is no infinite word over a binary alphabet avoiding abelian cubes of period at least 2 (longest word has length 290).

Theorem

The morphism h is $(3,3)$-abelian-square-free:

$$\begin{aligned} h(a) &\mapsto aaaaababababb \\ h(b) &\mapsto aaaabbababb \\ h(c) &\mapsto aaabbb \\ h(d) &\mapsto aababaabb \end{aligned}$$

Moreover for every abelian-square-free word w, $h(w)$ contains only 4 different 3-abelian-squares: a^2, b^2, $(aa)^2$, and $(ab)^2$.

even more

Theorem

The morphism h' is $(2,2)$-abelian-square-free:

$$\begin{aligned} h(a) &\mapsto aaacb \\ h(b) &\mapsto aabbb \\ h(c) &\mapsto abbcb \\ h(d) &\mapsto abccb \end{aligned}$$

Therefore, there exist infinite ternary words containing no 2-abelian square of length greater than 3.

[Huova, Karhumäki, Saarela, Saari: Local Squares, Periodicity and Finite Automata, 2011]

Equivalence classes of 2-abelian binary words (# is $n^2 - n + 2$):

$$aa^k b^l (ab)^m a^n \qquad \text{or} \qquad bb^k a^l (ba)^m b^n$$

Equivalence classes of 3-abelian binary words (# is $\Omega(n^4)$):

$$\left.\begin{array}{l} aaa^k b^l (aabb)^m \\ bbb^k a^l (aabb)^m \\ abb^k a^l (aabb)^m \\ baa^k b^l (aabb)^m \end{array}\right\} * \text{ connected with } * \left\{\begin{array}{l} (aab)^g (ab)^h b^i a^j \\ (abb)^g (ab)^h b^i a^j \end{array}\right.$$

Theorem

The number of k-abelian equivalence classes over Σ^n is $\mathcal{O}(n^{|\Sigma|^k - 1})$.

Equivalence classes

For a word $\omega \in \Sigma^{\mathbb{N}}$, let $\mathcal{F}_\omega(n)$ denote its set of distinct factors of length n. The *factor complexity function* of ω is defined as $\rho_\omega(n) = Card(\mathcal{F}_\omega(n))$. Therefore, the *k-abelian factor complexity function* of ω is defined as

$$\mathcal{P}_\omega(n) = Card(\mathcal{F}_\omega(n)/\sim_k).$$

Theorem

Let $k \geq 1$ and $m \geq 2$ be fixed numbers and let Σ be an m-letter alphabet. The number of k-abelian equivalence classes of Σ^n is $\Theta(n^{m^k - m^{k-1}})$.

[Karhumäki, Saarela, Zamboni: On a generalization of abelian equivalence and complexity of infinite words, 2013]

Theorem

A bi-infinite word w over a finite alphabet is periodic if and only if:

- *(Morse, Hedlund) $\mathcal{P}_w^\infty(n) < n+1$ for some $n \geq 1$.*
- *(Coven, Hedlund) $\mathcal{P}_w^1(n) < 2$ for some $n \geq 1$.*
- *(Karhumki, Saarela, Zamboni) $\mathcal{P}_w^k(n) < \min\{n+1, 2k\}$ for some $k \in \mathbb{N} \cup \{\infty\}$ and $n \geq 1$.*

Theorem

An aperiodic one-sided infinite word w is Sturmian if and only if:

- *(Morse, Hedlund) $\mathcal{P}_w^\infty(n) = n+1$ for all $n \geq 1$.*
- *(Coven, Hedlund) $\mathcal{P}_w^1(n) = 2$ for all $n \geq 1$. (k)*
- *(Karhumki, Saarela, Zamboni) $\mathcal{P}_w^k(n) = \min\{n+1, 2k\}$ for all $k \in \mathbb{N} \cup \{\infty\}$ and $n \geq 1$.*

[Karhumäki, Saarela, Zamboni: Variations of the Morse-Hedlund theorem for k-abelian equivalence, 2014]

Ultimately periodic words

Consider the following function:

$$q^{(k)}(n) = \begin{cases} n+1 & \text{for } n \leq 2k-1, \\ 2k & \text{for } n \geq 2k, \end{cases}$$

Theorem

Let $\omega \in \Sigma^{\mathbb{N}}$ and $k \in \mathbb{Z}^+ \cup \{+\infty\}$. If $\mathcal{P}_\omega^{(k)}(n_0) < q^{(k)}(n_0)$ for some $n_0 \geq 1$, then ω is ultimately periodic.

Corollary

Let $\omega \in \Sigma^{\mathbb{Z}}$ and $k \in \mathbb{Z}^+ \cup \{+\infty\}$. If $\mathcal{P}_\omega^{(k)}(n_0) < q^{(k)}(n_0)$ for some $n_0 \geq 1$, then ω is ultimately periodic.

Fine and Wilf

A word w is k-abelian p-periodic if there exist k-abelian equivalent words $u_0, u_1, \ldots, u_n, u_{n+1}$, each with length p, and some integer $r \geq 0$ such that

$$w = \mathrm{suff}_r(u_0)u_1 \cdots u_n \mathrm{pref}_{|w|-r-np}(u_{n+1})$$

We say that the period p is initial if $r = 0$.

Let $L_k(p, q)$ be the length of the longest word that has initial k-abelian periods p and q but does not have initial k-abelian period $\gcd(p, q)$.

[Karhumäki, Puzynina, Saarela: Fine and Wilf's theorem for k-abelian periods, 2013]

Existing bounds

Theorem

- *If* $p, q > gcd(p, q) = k$, *then* $L_k(p, q) = \frac{pq}{k} - 1$.
- *If* $p, q > \gcd(p, q) = d$, *then*

$$L_2(p, q) = \begin{cases} \max\{m_1 p, n_{-1} q\} & \text{if } d = 1, \\ pq/2 - 1 & \text{if } d = 2, \\ \infty & \text{if } d \geq 3 \end{cases}$$

- *If* $p = dp'$, $q = dq'$, *and* $k - 1 = dk'$ *such that* $gcd(p', q') = 1$ *and* $1 \leq k' \leq p'/4 \leq q'/4$, *then* w *has period* d *whenever*

$$|w| \geq \frac{pq}{d} - \frac{2(k-1)q}{d} + q + k - 1.$$

Periodicity and Complexity

1. Still a lot of work to do regarding the complexity description of this kind of words.
2. There are more dependencies between these words than considered in general. Use them in connection with the periodicity relation.
3. Work more on the Fine and Wilf variant for k-abelian periods.

[Cassaigne, Karhumäki, Saarela: On growth and fluctuation of k-abelian complexity, 2014]

Classical Results

Theorem

For a pattern $P \in \Sigma^*$ and a text $T \in \Sigma^*$, we can identify all factors of T that are abelian equivalent to P in $\mathcal{O}(|T| + |P|)$ time.

Theorem

For a word $w \in \Sigma^*$, we can identify all factors of w that are abelian repetitions in $\mathcal{O}(|w|^2)$ time.

[Ehlers, Manea, M., Nowotka: k-abelian pattern matching, 2014]

Tools

The suffix array of u, Suf_u is defined such that $Suf_u[i] = j$ if and only if $u[j..n]$ is the i^{th} suffix of u, in the lexicographical order.

Observation

Let $w \in \Sigma^n$. If there exist integers i and j with $1 \leq i < j \leq n$, and a word $u \in \Sigma^*$ such that $u \leq_p w[Suf_w[i]..n]$ and $u \leq_p w[Suf_w[j]..n]$, then for any $i \leq \ell \leq j$ we have $u \leq_p w[Suf_w[\ell]..n]$.

The array LCP_u is defined by $LCP_u[1] = 1$ and $LCP_u[r]$ is the length of the longest common prefix of the suffixes found on positions r and $r-1$ in the suffix array, i.e., $u[Suf_u[r-1]..n]$ and $u[Suf_u[r]..n]$. Moreover, after a linear time preprocessing one can answer in constant time to longest common prefix queries "$LCPref(i, j)$: What is the length of the longest common prefix of $u[i..n]$ and $u[j..n]$?".

Tool: the $\#(w, k)$ encoding

$w = aabbccaabbcc$ and $k = 2$

7	1	8	2	9	3	10	4	12	6	11	5
a	a	a	a	b	b	b	b	c	c	c	c
a	a	b	b	b	b	c	c		a	c	c
b	b	b	b	c	c	c	c		a		a
b	b	c	c	c	c		a		b		a
c	c	c	c		a		a		b		b
c	c		a		a		b		c		b
	a		a		b		b		c		c
	a		b		b		c				c
	b		b		c		c				
	b		c		c						
	c		c								
	c										

7	1	8	2	9	3	10	4	12	6	11	5
1	1	2	2	3	3	4	4	-	5	6	6

$\#(w, 2) = 23456512346$

Properties of encodings

Lemma

Let $w \in \Sigma^*$ be a word of length n. We can compute $\#(w, k)$ in $\mathcal{O}(n)$ time.

Lemma

Let $u, v \in \Sigma^*$ be two words of length n. If $u \equiv_k v$ for some k, then

$$\begin{array}{rcl} u[1..k-1] & = & v[1..k-1], \text{ and} \\ \#(u,k) & \equiv_1 & \#(v,k). \end{array}$$

Testing k-equivalence

Lemma

Let $u, v \in \Sigma^*$ be two words of length n and k be a positive integer with $1 \leq k \leq n$. We can decide whether $u \equiv_k v$ in $\mathcal{O}(n)$ time.

Construct $w = u0v$, and compute Suf_w and $\#(w, k)$.
Set $u' = \#(w, k)[1..n - k + 1]$ and $v' = \#(w, k)[n + 2..2n - k + 2]$.
The fact that u' and v' contain exactly the same letters is equivalent to them having exactly the same multi-set of factors of length k.

Then $u \equiv_k u$ if and only if

$$\begin{array}{rcl} u[1..k-1] & = & v[1..k-1], \text{ and} \\ u' & \equiv_1 & v'. \end{array}$$

Pattern Matching Results

Remark

The k-abelian pattern matching problem can be reduced to the classical abelian pattern matching problem.

Theorem

For a pattern $P \in \Sigma^*$, a text $T \in \Sigma^*$, and an integer k, we can identify all factors of T that are k-abelian equivalent to P in $\mathcal{O}(|T| + |P|)$ time.

Theorem

For a word $w \in \Sigma^*$ and an integer k, we can identify all k-abelian repetitions in w in $\mathcal{O}(|w|^2)$ time.

Another problem

Theorem

Given two words of same length n, we can find the largest positive integer k such that the words are k-abelian equivalent linear time $\mathcal{O}(n)$.

Construct the word $w = u0v$, its Suf_w, and $LCPref_w$ data structures. Set $u' = \#(w, k)[1..n-k+1]$ and $v' = \#(w, k)[n+2..2n-k+2]$.

Set $\ell = \min(\max_{pref}(u', v'), \max_{suff}(u', v')) + 1$. Then $k \leq \ell$.

Going through Suf_w we do $LCPref_w$ queries for the i^{th} suffix of u (in lexicographic order) and the i^{th} suffix of v. Let ℓ' be the minimum value such that two such suffixes share a common prefix of length exactly ℓ', while both have length at least $\ell' + 1$.

Then $k = \min\{\ell', \ell\}$.

Online problem

Problem

Preprocess a word $P \in \Sigma^*$ and a positive integer k such that when given a text $T \in \Sigma^*$, in letter by letter manner, to be able to answer, at each moment, queries asking whether the part of T read so far ends with a factor which is k-abelian equivalent to P or not.

Construct the $\#(P, k-1)$ and $\#(P, k)$ encodings over $\{1, \ldots, \ell_2, \ldots, \ell_1\}$.

Compute $L = \{(i, a, j) \mid 1 \leq i \leq \ell_1, 1 \leq j \leq \ell_2, a \in \Sigma$, and $f_1[i]a = f_2[j]\}$, where $f_1[i]$ ($f_2[i]$) is the factor of length $k-1$ (k) of P, whose position in the lexicographically ordered set of all factors of length $k-1$ (k) of P is i (j).

Online Solution

Different implementations for L

- An $m \times \sigma$ table M, with $M[i][a] = j$ if and only if $(i, a, j) \in L$.
- A hash table, using perfect hashing.
- A van Emde Boas tree associating to each i a pair $(a, \cdot)$, if this exists.

Compute for each $1 \leq j \leq \ell_2$ the values $suf[j], pref[j] \leq \ell_1$, such that $suf[j] = i$ if $f_2[j] = af_1[i]$ for $a \in \Sigma$, while $pref[j] = i$ if $f_2[j] = f_1[i]a$.

If last read part of T is $\#(P', k) = j_1 \dots j_{m-k} j_{m-k+1}$ and we read letter a, then the new last part is $\#(P'', k) = j_2 \dots j_{m-k} j_{m-k+1} suf[j_{m-k+1}]a$.

- If $(suf[j_{m-k+1}], a, \cdot) \in L$, then use real-time abelian pattern matching between $\#(P'', k)$ and $\#(P, k)$.
- If $(suf[j_{m-k+1}], a, \cdot) \notin L$, then use real-time pattern matching between the suffix of length $k-1$ of the read text and $P[1..k-1]$.

thus...

Theorem

Given a pattern $P \in \Sigma^m$ for $|\Sigma| = \sigma$, and a positive integer k, the online k-abelian pattern matching problem can be solved in:

- $\mathcal{O}(m\sigma)$ *preprocessing time,* $\mathcal{O}(m\sigma)$ *space, and* $\mathcal{O}(1)$ *query time.*
- $\mathcal{O}(m \log\log m)$ *preprocessing time,* $\mathcal{O}(m)$ *space, and* $\mathcal{O}(1)$ *query time.*
- $\mathcal{O}(m)$ *expect. preprocessing time,* $\mathcal{O}(m)$ *space, and* $\mathcal{O}(1)$ *query time.*
- $\mathcal{O}(m)$ *preprocessing time,* $\mathcal{O}(m)$ *space, and* $\mathcal{O}(\log\log\sigma)$ *query time.*

Extended-k-abelian equivalence

Problem

Preprocess a word $P \in \Sigma^*$ and a positive integer k such that when given a text $T \in \Sigma^*$, in letter by letter manner, to be able to answer, at each moment, queries asking whether the part of T read so far ends with a factor which is extended-k-abelian equivalent to P or not.

$$x \not\equiv_2 v$$

$$x = aba,\ y = bab, \text{ and } k = 2$$

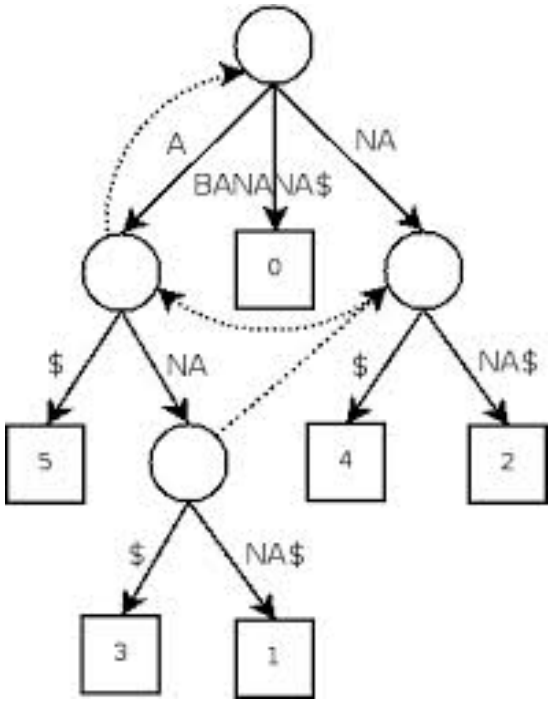
$$(x, 2) = \{(ab, 1), (ba, 1)\} = (y, 2)$$
$$(x, 1) = \{(a, 2), (b, 1)\} \neq \{(a, 1), (b, 2)\} = (y, 2)$$

$$x \equiv_2^e v$$

Online Solution for extended case

Extra tools:

- Suffix tree for P
- Associate to a node of the suffix tree of P the factor $P[i..j]$ if and only if the path from the root to that node is labelled with $P[i..j]$.
- Suffix links (from the node with label aX go to the node with label X)



[Source: Wikipedia]

Strategy

1. Let N_1 be the lowest explicit ancestor of N the node for $P[i..i+\ell-1]$.
2. If exists and edge labeled $a = T[j+1]$ from N to M, then $T[1..j+1]$ has the suffix $P[i..i+\ell-1]a$ and we update the current node to M.
3. Otherwise, let N_2 be the target-node of the suffix link of N_1.Advance along the edge leaving N_2, and update each node we find to be the current N_2 until no longer possible. The current N_2 is the lowest explicit ancestor of the node corresponding to $P[i+1..i+\ell-1]$.
4. Update $i = i+1$ and $N = N_2$, and restart from step 2.

thus...

Theorem

Given a pattern $P \in \Sigma^*$ of length n for $|\Sigma| = \sigma$, and a positive integer k, the online extended-k-abelian pattern matching problem can be solved in:

- $\mathcal{O}(m\sigma)$ *preprocessing time,* $\mathcal{O}(m\sigma)$ *space, and* $\mathcal{O}(n)$ *time.*
- $\mathcal{O}(m \log\log m)$ *preprocessing time,* $\mathcal{O}(m)$ *space, and* $\mathcal{O}(n)$ *time.*
- $\mathcal{O}(m)$ *expected preprocessing time,* $\mathcal{O}(m)$ *space, and* $\mathcal{O}(n)$ *time.*
- $\mathcal{O}(m)$ *preprocessing time,* $\mathcal{O}(m)$ *space, and* $\mathcal{O}(n \log\log \sigma)$ *time.*

Real-time solution for extended case

Idea: report only the factors of T that are extended-k-abelian equivalent to P (whenever a length k factor of P is found).

Lemma (GawNicLew14)

We can preprocess a word $P \in \Sigma^*$ in $\mathcal{O}(|P| \log k)$ time and linear space such that, for each i and j with $j - i \leq k$, we can return in $\mathcal{O}(1)$ time the (explicit or implicit) node of the suffix tree of P corresponding to $P[i..j]$.

Use a queue to add the current letter and 2 more operations:

1. dequeue and check whether the current factor of P can be extended;
2. update the P-suffix and its representation accordingly, dequeue, and check whether the new current factor of P can be extended.

Enough, since not being able to extended for some length ℓ, the number of suffixes of factors of P we have to check until reaching again length ℓ equals the number of letters read between these two moments

thus...

Theorem

Given a pattern $P \in \Sigma^*$ of length n for $|\Sigma| = \sigma$, and an integer k, the real-time extended-k-abelian pattern matching problem can be solved in:

- $\mathcal{O}(m(\sigma + \log k))$ *preproc. time,* $\mathcal{O}(m\sigma)$ *space, and* $\mathcal{O}(1)$ *query time.*
- $\mathcal{O}(m(\log(k \log m))$ *preproc. time,* $\mathcal{O}(m)$ *space, and* $\mathcal{O}(1)$ *query time.*
- $\mathcal{O}(m \log k)$ *expec. preproc. time,* $\mathcal{O}(m)$ *space, and* $\mathcal{O}(1)$ *query time.*
- $\mathcal{O}(m \log k)$ *preproc. time,* $\mathcal{O}(m)$ *space, and* $\mathcal{O}(\log \log k)$ *query time.*

Downside: index structures for k-abelian pattern matching

Idea: construct $\#(T, k-1)$ and $\#(T, k)$ and $L = \{(i, a, j)\}$, as before.

Find a location of $P[1..k-1]$ in T in $\mathcal{O}(k)$ time.
Reading $P[k..m]$ letter by letter (checking against L) we produce $\#(P, k)$.
The problem is reduced to producing an index of $\#(T, k)$, useful to check efficiently whether a factor is abelian equivalent to $\#(P, k)$.

The preprocessing time for building an index for the k-abelian pattern matching problem is $\mathcal{O}(n)$ expected or $\mathcal{O}(n \log\log n)$ deterministic time.
The query time is $\mathcal{O}(n + m - k + 1)$.

Open Problems

1. Can we do the online version of the k-abelian pattern matching better (linear time and space)?
2. Can we do the online version of the extended k-abelian pattern matching better (linear time and space)?
3. Most likely the indexing for the k-abelian pattern matching can be improved. But how, and how much?

WORDS 2015: September 14-18

http://words2015.uni-kiel.de

Submission **April 17** Conference **September 14 – 18**

Mathematical theory of WORDS from all points of view:

- combinatorial
- algebraic
- algorithmic
- applications