Combinatorial problems on closed and privileged words

Daniel Gabrić

University of Guelph

July 3, 2025

Basic Definitions: Bordered and Unbordered words

Definition A word u is said to be a **border** of a word w if u is a nonempty proper prefix and suffix of w.

Definition A word u is said to be a **border** of a word w if u is a nonempty proper prefix and suffix of w.

Note: Sometimes borders are allowed to be any prefix that is a suffix.

Basic Definitions: Bordered and Unbordered words

Definition A word u is said to be a **border** of a word w if u is a nonempty proper prefix and suffix of w.

Note: Sometimes borders are allowed to be any prefix that is a suffix.

Definition A word w is said to be **bordered** if it has a border. Otherwise, it is said to be **unbordered**.

Basic Definitions: Bordered and Unbordered words

Definition A word u is said to be a **border** of a word w if u is a nonempty proper prefix and suffix of w.

Note: Sometimes borders are allowed to be any prefix that is a suffix.

Definition A word w is said to be **bordered** if it has a border. Otherwise, it is said to be **unbordered**.

The word `alfalfa` is bordered. It has two borders `alfa`, and `a`.

The word `bordered` is in fact unbordered. It has no valid borders.

Basic Definitions: Closed and Privileged words

Definition We say that a word w is **closed by** a word u if u is a border of w and it occurs exactly twice in w.

Basic Definitions: Closed and Privileged words

Definition We say that a word w is **closed by** a word u if u is a border of w and it occurs exactly twice in w.

Definition A word w is said to be **closed** if $|w| \leq 1$ or if w is closed by some word.

Basic Definitions: Closed and Privileged words

Definition We say that a word w is **closed by** a word u if u is a border of w and it occurs exactly twice in w.

Definition A word w is said to be **closed** if $|w| \leq 1$ or if w is closed by some word.

Definition A word w is said to be **privileged** if $|w| \leq 1$ or if w is closed by a privileged word.

Basic Definitions: Closed and Privileged words

Definition We say that a word w is **closed by** a word u if u is a border of w and it occurs exactly twice in w.

Definition A word w is said to be **closed** if $|w| \leq 1$ or if w is closed by some word.

Definition A word w is said to be **privileged** if $|w| \leq 1$ or if w is closed by a privileged word.

The English word `bonobo` is a closed word, closed by `bo`. Since $|\texttt{bo}| > 1$ and `bo` is unbordered and therefore not privileged, we have that `bonobo` is not privileged.

Basic Definitions: Closed and Privileged words

Definition We say that a word w is **closed by** a word u if u is a border of w and it occurs exactly twice in w.

Definition A word w is said to be **closed** if $|w| \leq 1$ or if w is closed by some word.

Definition A word w is said to be **privileged** if $|w| \leq 1$ or if w is closed by a privileged word.

The English word `bonobo` is a closed word, closed by `bo`. Since $|\texttt{bo}| > 1$ and `bo` is unbordered and therefore not privileged, we have that `bonobo` is not privileged.

The French word `entente` is closed by `ente`. Furthermore `ente` is closed by `e`. But $|\texttt{e}| \leq 1$, so `ente` is privileged and therefore so is `entente`.

Background: Closed and Privileged words

Closed words were first defined by Fici (2011) as a way to classify Trapezoidal and Sturmian words.

Background: Closed and Privileged words

Closed words were first defined by Fici (2011) as a way to classify Trapezoidal and Sturmian words.

However, there are a few different names for closed words that were introduced earlier.

Background: Closed and Privileged words

Closed words were first defined by Fici (2011) as a way to classify Trapezoidal and Sturmian words.

However, there are a few different names for closed words that were introduced earlier.

Closed words are also called **periodic-like** words, first defined by Carpi and de Luca (2001).

Background: Closed and Privileged words

Closed words were first defined by Fici (2011) as a way to classify Trapezoidal and Sturmian words.

However, there are a few different names for closed words that were introduced earlier.

Closed words are also called **periodic-like** words, first defined by Carpi and de Luca (2001).

- A **period** of a word $w = w_1 w_2 \cdots w_n$ is an integer $p \leq n$ such that $w_i = w_{i+p}$ for all $1 \leq i \leq n - p$.
- A length-n word is said to be **periodic** if it has a period of length $\leq n/2$.

In the analysis of DNA sequences, and other long words, the minimal period is generally much larger than half the length of the word.

Periodic-like words were considered as a generalization of periodic words that preserve some features of periodic words.

Background: Closed and Privileged words

Closed words also appear in frame synchronization applications dating back to 1960.

Background: Closed and Privileged words

Closed words also appear in frame synchronization applications dating back to 1960.

- There is a 1-1 correspondence between closed words and codewords in *prefix-synchronized codes*.

Background: Closed and Privileged words

Closed words also appear in frame synchronization applications dating back to 1960.

- There is a 1-1 correspondence between closed words and codewords in *prefix-synchronized codes*.

Closed words are also known as **complete first return words**.

Background: Closed and Privileged words

Closed words also appear in frame synchronization applications dating back to 1960.

- There is a 1-1 correspondence between closed words and codewords in *prefix-synchronized codes*.

Closed words are also known as **complete first return words**.

- A **complete first return** to a word u is a word that starts and ends with u, and contains exactly two occurrences of u.
- A word is closed if it is a complete first return to one of its non-empty proper prefixes.

Background: Closed and Privileged words

Closed words also appear in frame synchronization applications dating back to 1960.

- There is a 1-1 correspondence between closed words and codewords in *prefix-synchronized codes*.

Closed words are also known as **complete first return words**.

- A **complete first return** to a word u is a word that starts and ends with u, and contains exactly two occurrences of u.
- A word is closed if it is a complete first return to one of its non-empty proper prefixes.

Privileged words were originally defined by Kellendonk et al. (2011) in the context of dynamical systems and discrete geometry by iterating the definition of a complete first return.

Background: Closed and Privileged words

Closed words also appear in frame synchronization applications dating back to 1960.

- There is a 1-1 correspondence between closed words and codewords in *prefix-synchronized codes*.

Closed words are also known as **complete first return words**.

- A **complete first return** to a word u is a word that starts and ends with u, and contains exactly two occurrences of u.
- A word is closed if it is a complete first return to one of its non-empty proper prefixes.

Privileged words were originally defined by Kellendonk et al. (2011) in the context of dynamical systems and discrete geometry by iterating the definition of a complete first return.

A word is privileged if and only if it is the empty word, a single letter, or a complete first return to a shorter privileged word.

Combinatorial Problems

For the rest of this talk, we implicitly assume that all words are over a k-letter alphabet for some $k \geq 2$.

We will explore some results and open questions related to the following three combinatorial problems.

Combinatorial Problems

For the rest of this talk, we implicitly assume that all words are over a k-letter alphabet for some $k \geq 2$.

We will explore some results and open questions related to the following three combinatorial problems.

1. Enumeration

   Can we efficiently count the number of length-n closed words or length-n privileged words? If not, can we find good asymptotic bounds for these numbers?

Combinatorial Problems

For the rest of this talk, we implicitly assume that all words are over a k-letter alphabet for some $k \geq 2$.

We will explore some results and open questions related to the following three combinatorial problems.

1. Enumeration new results! open problems!

   Can we efficiently count the number of length-n closed words or length-n privileged words? If not, can we find good asymptotic bounds for these numbers?

Combinatorial Problems

For the rest of this talk, we implicitly assume that all words are over a k-letter alphabet for some $k \geq 2$.

We will explore some results and open questions related to the following three combinatorial problems.

1. Enumeration new results! open problems!

   Can we efficiently count the number of length-n closed words or length-n privileged words? If not, can we find good asymptotic bounds for these numbers?

2. Ranking/Unranking

   Given a listing of closed or privileged words, can we efficiently compute the position of a specific element? Can we find an element given its position?

Combinatorial Problems

For the rest of this talk, we implicitly assume that all words are over a k-letter alphabet for some $k \geq 2$.

We will explore some results and open questions related to the following three combinatorial problems.

1. Enumeration new results! open problems!

   Can we efficiently count the number of length-n closed words or length-n privileged words? If not, can we find good asymptotic bounds for these numbers?

2. Ranking/Unranking no results. open problems!

   Given a listing of closed or privileged words, can we efficiently compute the position of a specific element? Can we find an element given its position?

Combinatorial Problems

For the rest of this talk, we implicitly assume that all words are over a k-letter alphabet for some $k \geq 2$.

We will explore some results and open questions related to the following three combinatorial problems.

1. Enumeration new results! open problems!

   Can we efficiently count the number of length-n closed words or length-n privileged words? If not, can we find good asymptotic bounds for these numbers?

2. Ranking/Unranking no results. open problems!

   Given a listing of closed or privileged words, can we efficiently compute the position of a specific element? Can we find an element given its position?

3. Generation

   Can we efficiently list all closed or privileged words in any order? In a specific order?

Combinatorial Problems

For the rest of this talk, we implicitly assume that all words are over a k-letter alphabet for some $k \geq 2$.

We will explore some results and open questions related to the following three combinatorial problems.

1. Enumeration new results! open problems!

   Can we efficiently count the number of length-n closed words or length-n privileged words? If not, can we find good asymptotic bounds for these numbers?

2. Ranking/Unranking no results. open problems!

   Given a listing of closed or privileged words, can we efficiently compute the position of a specific element? Can we find an element given its position?

3. Generation partial results. open problems!

   Can we efficiently list all closed or privileged words in any order? In a specific order?

Enumeration

Let $C_k(n)$ (resp. $P_k(n)$) denote the number of length-n closed (resp. privileged) words over a k-letter alphabet.

Enumeration

Let $C_k(n)$ (resp. $P_k(n)$) denote the number of length-n closed (resp. privileged) words over a k-letter alphabet.

It seems hard to count $C_k(n)$ and $P_k(n)$ exactly (and quickly), so most research has been dedicated to finding good upper and lower bounds for them.

Enumeration

Let $C_k(n)$ (resp. $P_k(n)$) denote the number of length-n closed (resp. privileged) words over a k-letter alphabet.

It seems hard to count $C_k(n)$ and $P_k(n)$ exactly (and quickly), so most research has been dedicated to finding good upper and lower bounds for them.

Every privileged word is also closed, so $P_k(n) \leq C_k(n)$.

Enumeration

Let $C_k(n)$ (resp. $P_k(n)$) denote the number of length-n closed (resp. privileged) words over a k-letter alphabet.

It seems hard to count $C_k(n)$ and $P_k(n)$ exactly (and quickly), so most research has been dedicated to finding good upper and lower bounds for them.

Every privileged word is also closed, so $P_k(n) \leq C_k(n)$.

In 2016, Forsyth et al. showed that

$$P_2(n) \in \Omega\left(\frac{2^n}{n^2}\right).$$

In 2018, Nicholson and Rampersad showed that

$$P_k(n) \in \Omega\left(\frac{k^n}{n(\log_k(n))^2}\right).$$

Problem: Enumeration

In 2020, Rukavicka proved the very first nontrivial upper bounds on privileged and closed words. He showed that

$$C_k(n) \in O\left(\ln n \frac{k^n}{\sqrt{n}}\right).$$

Problem: Enumeration

In 2020, Rukavicka proved the very first nontrivial upper bounds on privileged and closed words. He showed that

$$C_k(n) \in O\left(\ln n \frac{k^n}{\sqrt{n}}\right).$$

In 2022, Rukavicka improved his upper bound on privileged words by showing that for every $j \geq 3$,

$$P_k(n) \in O\left(\frac{k^n \sqrt{\ln n}}{\sqrt{n}} \ln^{\circ j}(n) \prod_{i=2}^{j-1} \sqrt{\ln^{\circ i}(n)}\right)$$

where $\ln^{\circ 0}(n) = n$ and $\ln^{\circ j}(n) = \ln(\ln^{\circ j-1}(n))$.

Heuristic Argument

What is the true bound for $C_k(n)$? We give a heuristic lower bound argument.

Heuristic Argument

What is the true bound for $C_k(n)$? We give a heuristic lower bound argument.

Consider a "random" length-n word w.

Let $\ell = \log_k(n) + c$ for some constant c.

Heuristic Argument

What is the true bound for $C_k(n)$? We give a heuristic lower bound argument.

Consider a "random" length-n word w.

Let $\ell = \log_k(n) + c$ for some constant c.

There is a $\frac{1}{k^\ell} = \frac{1}{k^c n}$ chance that w has a length-ℓ bordered.

Heuristic Argument

What is the true bound for $C_k(n)$? We give a heuristic lower bound argument.

Consider a "random" length-n word w.

Let $\ell = \log_k(n) + c$ for some constant c.

There is a $\frac{1}{k^\ell} = \frac{1}{k^c n}$ chance that w has a length-ℓ bordered.

If w has a length-ℓ border u, then drop the first and last character of w to get w'.

Heuristic Argument

What is the true bound for $C_k(n)$? We give a heuristic lower bound argument.

Consider a "random" length-n word w.

Let $\ell = \log_k(n) + c$ for some constant c.

There is a $\frac{1}{k^\ell} = \frac{1}{k^c n}$ chance that w has a length-ℓ bordered.

If w has a length-ℓ border u, then drop the first and last character of w to get w'.

If w' were randomly chosen (which it is not), then by the linearity of expectation, the expected number of occurrences of u in w' is approximately $(n-1-\ell)k^{-\ell} \approx k^{-c}$.

Heuristic Argument

What is the true bound for $C_k(n)$? We give a heuristic lower bound argument.

Consider a "random" length-n word w.

Let $\ell = \log_k(n) + c$ for some constant c.

There is a $\frac{1}{k^\ell} = \frac{1}{k^c n}$ chance that w has a length-ℓ bordered.

If w has a length-ℓ border u, then drop the first and last character of w to get w'.

If w' were randomly chosen (which it is not), then by the linearity of expectation, the expected number of occurrences of u in w' is approximately $(n-1-\ell)k^{-\ell} \approx k^{-c}$.

For c large enough we have that u does not occur in w' with high probability, so w is closed.

There are at least approximately $k^{n-\ell} \in \Theta\left(\frac{k^n}{n}\right)$ length-n closed words.

Enumeration Results

Theorem

For any integer $k \geq 2$, we have $C_k(n) \in \Theta\left(\frac{k^n}{n}\right)$.

Theorem

Let $k \geq 2$ be an integer. Let $\log_k^{\circ 0}(n) = n$ and $\log_k^{\circ j}(n) = \log_k(\log_k^{\circ j-1}(n))$ for $j \geq 1$.

For all $j \geq 0$, we have

$$P_k(n) \in \Omega\left(\frac{k^n}{n \log_k^{\circ j}(n) \prod_{i=1}^{j} \log_k^{\circ i}(n)}\right)$$

and

$$P_k(n) \in O\left(\frac{k^n}{n \prod_{i=1}^{j} \log_k^{\circ i}(n)}\right).$$

Enumeration Results

Theorem

For any integer $k \geq 2$, we have $C_k(n) \in \Theta\left(\frac{k^n}{n}\right)$.

Theorem

Let $k \geq 2$ be an integer. Let $\log_k^{\circ 0}(n) = n$ and $\log_k^{\circ j}(n) = \log_k(\log_k^{\circ j-1}(n))$ for $j \geq 1$.

For all $j \geq 0$, we have

$$P_k(n) \in \Omega\left(\frac{k^n}{n \log_k^{\circ j}(n) \prod_{i=1}^{j} \log_k^{\circ i}(n)}\right)$$

and

$$P_k(n) \in O\left(\frac{k^n}{n \prod_{i=1}^{j} \log_k^{\circ i}(n)}\right).$$

D. Gabrić, Asymptotic bounds for the number of closed and privileged words, E-JC, Vol. 32 (2024) #P2.45

Proof Sketch: Closed words

Let $B_k(n, u)$ denote the number of length-n words closed by the word u.

Proof Sketch: Closed words

Let $B_k(n, u)$ denote the number of length-n words closed by the word u.

Lower Bound

Proof Sketch: Closed words

Let $B_k(n, u)$ denote the number of length-n words closed by the word u.

Lower Bound

Nicholson and Rampersad (2018) showed that there are $\Omega\left(\frac{k^n}{n^2}\right)$ length-n words that are closed by a word of length “around” $\log_k(n)$.

Proof Sketch: Closed words

Let $B_k(n, u)$ denote the number of length-n words closed by the word u.

Lower Bound

Nicholson and Rampersad (2018) showed that there are $\Omega\left(\frac{k^n}{n^2}\right)$ length-n words that are closed by a word of length “around” $\log_k(n)$.

In other words, they showed that $B_k(n, u) \in \Omega\left(\frac{k^n}{n^2}\right)$ for $|u| \approx \log_k(n)$.

Proof Sketch: Closed words

Let $B_k(n,u)$ denote the number of length-n words closed by the word u.

Lower Bound

Nicholson and Rampersad (2018) showed that there are $\Omega\left(\frac{k^n}{n^2}\right)$ length-n words that are closed by a word of length “around” $\log_k(n)$.

In other words, they showed that $B_k(n,u) \in \Omega\left(\frac{k^n}{n^2}\right)$ for $|u| \approx \log_k(n)$.

There are $\Omega(n)$ words of length $\approx \log_k(n)$.

Proof Sketch: Closed words

Let $B_k(n, u)$ denote the number of length-n words closed by the word u.

Lower Bound

Nicholson and Rampersad (2018) showed that there are $\Omega\left(\frac{k^n}{n^2}\right)$ length-n words that are closed by a word of length "around" $\log_k(n)$.

In other words, they showed that $B_k(n, u) \in \Omega\left(\frac{k^n}{n^2}\right)$ for $|u| \approx \log_k(n)$.

There are $\Omega(n)$ words of length $\approx \log_k(n)$.

Therefore, there are $\Omega\left(\frac{k^n}{n}\right)$ length-n closed words.

Proof Sketch: Closed words

Upper Bound

Proof Sketch: Closed words

Upper Bound

Notice that every closed word is closed by its longest border.

We can therefore partition all closed words into sets based on the lengths of their longest borders.

Proof Sketch: Closed words

Upper Bound

Notice that every closed word is closed by its longest border.

We can therefore partition all closed words into sets based on the lengths of their longest borders.

Let $C_k(n, t)$ denote the number of length-n closed words that have a longest border of length t.

We have

$$C_k(n) = \sum_{t=1}^{n-1} C_k(n, t).$$

Proof Sketch: Closed words

Upper Bound

Notice that every closed word is closed by its longest border.

We can therefore partition all closed words into sets based on the lengths of their longest borders.

Let $C_k(n,t)$ denote the number of length-n closed words that have a longest border of length t.

We have

$$C_k(n) = \sum_{t=1}^{n-1} C_k(n,t).$$

We consider two cases, one where longest border is of length $t > \lfloor n/2 \rfloor$ and the other where the length is $t \leq \lfloor n/2 \rfloor$.

Proof Sketch: Closed words

Upper Bound

Proof Sketch: Closed words

Upper Bound

If $t \geq \lfloor n/2 \rfloor$, then there are at most $nk^{\lceil n/2 \rceil}$ such words.

Proof Sketch: Closed words

Upper Bound

If $t \geq \lfloor n/2 \rfloor$, then there are at most $nk^{\lceil n/2 \rceil}$ such words.

Each such word is determined by its minimal period, which is of length $n - t \leq \lceil n/2 \rceil$ and there are less than n such periods.

Proof Sketch: Closed words

Upper Bound

If $t \geq \lfloor n/2 \rfloor$, then there are at most $nk^{\lceil n/2 \rceil}$ such words.

Each such word is determined by its minimal period, which is of length $n - t \leq \lceil n/2 \rceil$ and there are less than n such periods.

If $t \leq \lfloor n/2 \rfloor$, then the borders do not "overlap".

Thus, for any length-n closed word w closed by a length-t word u, we can write $w = uvu$.

Proof Sketch: Closed words

Upper Bound

If $t \geq \lfloor n/2 \rfloor$, then there are at most $nk^{\lceil n/2 \rceil}$ such words.

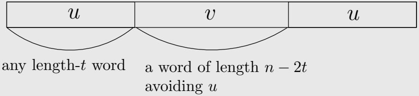
Each such word is determined by its minimal period, which is of length $n - t \leq \lceil n/2 \rceil$ and there are less than n such periods.

If $t \leq \lfloor n/2 \rfloor$, then the borders do not "overlap".

Thus, for any length-n closed word w closed by a length-t word u, we can write $w = uvu$.

The word u can be any length-t word, and the word v is a word of length $n - 2t$ that does not contain u as a factor.

- The word v has some extra constraints which are unimportant in proving an upper bound.



Proof Sketch: Closed words

Upper Bound

Let $A_k(n, u)$ denote the number of length-n words avoiding u as a factor.

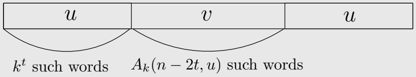
$$\underbrace{\boxed{\quad u \quad}}_{k^t \text{ such words}}\underbrace{\boxed{\quad v \quad}}_{A_k(n-2t,\,u) \text{ such words}}\boxed{\quad u \quad}$$

We can upper bound $C_k(n, t)$ by $\sum_{|u|=t} A_k(n - 2t, u)$.

Proof Sketch: Closed words

Upper Bound

Let $A_k(n, u)$ denote the number of length-n words avoiding u as a factor.



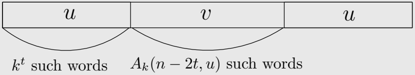
We can upper bound $C_k(n, t)$ by $\sum_{|u|=t} A_k(n - 2t, u)$.

We now need to find an upper bound for $A_k(n - 2t, u)$.

Proof Sketch: Closed words

Upper Bound

Let $A_k(n,u)$ denote the number of length-n words avoiding u as a factor.



We can upper bound $C_k(n,t)$ by $\sum_{|u|=t} A_k(n-2t,u)$.

We now need to find an upper bound for $A_k(n-2t,u)$.

A crucial result by Guibas and Odlyzko (1981) shows that $A_k(n,u) \le A_k(n,0^{|u|})$ for all $n \ge 1$.

Which means

$$C_k(n,t) \le k^t A_k(n-2t,0^t).$$

Proof Sketch: Closed words

Upper Bound

For $t = 1$, we have $A_k(n, 0^t) = (k-1)^n$ for all $n \geq 0$

For $t \geq 2$, the sequence $(A_k(n, 0^t))_{n \geq 0}$ is exactly the t-bonacci sequence, which has the following two well-known recurrences:

$$A_k(n, 0^t) = \begin{cases} k^n, & n < t; \\ (k-1) \sum_{i=1}^{t} A_k(n-i, 0^t), & n \geq t. \end{cases}$$

$$A_k(n, 0^t) = \begin{cases} k^n, & n < t; \\ k^n - 1, & n = t; \\ kA_k(n-1, 0^t) - A_k(n-1-t, 0^t), & n > t. \end{cases}$$

Proof Sketch: Closed words

Upper Bound

For $t = 1$, we have $A_k(n, 0^t) = (k-1)^n$ for all $n \geq 0$

For $t \geq 2$, the sequence $(A_k(n, 0^t))_{n \geq 0}$ is exactly the t-bonacci sequence, which has the following two well-known recurrences:

$$A_k(n, 0^t) = \begin{cases} k^n, & n < t; \\ (k-1) \sum_{i=1}^{t} A_k(n-i, 0^t), & n \geq t. \end{cases}$$

$$A_k(n, 0^t) = \begin{cases} k^n, & n < t; \\ k^n - 1, & n = t; \\ kA_k(n-1, 0^t) - A_k(n-1-t, 0^t), & n > t. \end{cases}$$

Using these recurrences, one can show that $A_k(n, 0^t) \leq (k - (k-1)k^{-t-1})^n$ for all $t \geq 2$, $n \geq 1$.

Proof Sketch: Closed words

Upper Bound

We end up with

$$C_k(n) = \sum_{t=1}^{n-1} C_k(n,t) \leq (k-1)^n + nk^{\lceil n/2 \rfloor} + \sum_{t=2}^{\lfloor n/2 \rfloor} k^t (k - (k-1)k^{-t-1})^{n-2t}$$

Proof Sketch: Closed words

Upper Bound

We end up with

$$C_k(n) = \sum_{t=1}^{n-1} C_k(n,t) \leq (k-1)^n + nk^{\lceil n/2 \rfloor} + \sum_{t=2}^{\lfloor n/2 \rfloor} k^t (k - (k-1)k^{-t-1})^{n-2t}$$

Using standard techniques, which are uninteresting for the purpose of this talk, one can show that there is some constant $N > 0$ such that

$$C_k(n) \leq c\frac{k^n}{n}$$

for all $n > N$ where $c > 0$ is some constant which depends on k.

Proof Sketch: Privileged words

Since every privileged word is also a closed word, we have that a privileged word is also closed by its largest border (which must be privileged).

We can write, similar to the closed case, that

$$P_k(n) = \sum_{t=1}^{n-1} P_k(n,t)$$

where $P_k(n,t)$ is the number of length-n privileged words with longest border of length t.

Proof Sketch: Privileged words

Since every privileged word is also a closed word, we have that a privileged word is also closed by its largest border (which must be privileged).

We can write, similar to the closed case, that

$$P_k(n) = \sum_{t=1}^{n-1} P_k(n,t)$$

where $P_k(n,t)$ is the number of length-n privileged words with longest border of length t.

Lower Bound

Consider a privileged word w of length n with longest border u of length $t \approx \log_k(n)$.

For n large enough, we can write $w = uvu$ for some word v.

Proof Sketch: Privileged words

Since every privileged word is also a closed word, we have that a privileged word is also closed by its largest border (which must be privileged).

We can write, similar to the closed case, that

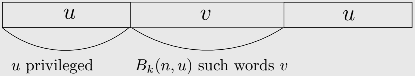
$$P_k(n) = \sum_{t=1}^{n-1} P_k(n,t)$$

where $P_k(n,t)$ is the number of length-n privileged words with longest border of length t.

Lower Bound

Consider a privileged word w of length n with longest border u of length $t \approx \log_k(n)$.

For n large enough, we can write $w = uvu$ for some word v.



Proof Sketch: Privileged words

Lower Bound

We have

$$P_k(n,t) = \sum_{|u|=t \text{ and } u \text{ privileged}} B_k(n,u).$$

By Nicholson and Rampersad (2018), we have $B_k(n,u) \in \Omega\left(\frac{k^n}{n^2}\right)$ for $|u| \approx \log_k(n)$, and so

$$P_k(n) \geq P_k(n,t) \geq \sum_{|u|=t \text{ and } u \text{ privileged}} c\frac{k^n}{n^2} = cP_k(t)\frac{k^n}{n^2}$$

for all n large enough, where $c > 0$ is some constant, and $t \approx \log_k(n)$.

Proof Sketch: Privileged words

Lower Bound

We have

$$P_k(n,t) = \sum_{|u|=t \text{ and } u \text{ privileged}} B_k(n,u).$$

By Nicholson and Rampersad (2018), we have $B_k(n,u) \in \Omega\left(\frac{k^n}{n^2}\right)$ for $|u| \approx \log_k(n)$, and so

$$P_k(n) \geq P_k(n,t) \geq \sum_{|u|=t \text{ and } u \text{ privileged}} c\frac{k^n}{n^2} = cP_k(t)\frac{k^n}{n^2}$$

for all n large enough, where $c > 0$ is some constant, and $t \approx \log_k(n)$.

We get the lower bound by iteratively applying the above recursive inequality.

We can prove by induction that:

$$P_k(n) \geq c_0 P_k(\log_k(n))\frac{k^n}{n^2} \geq c_1 P_k(\log_k(\log_k(n)))\frac{k^n}{n(\log_k(n))^2} \geq \dots$$
$$\geq c_j \frac{k^n}{n \log_k^{\circ j}(n) \prod_{i=1}^{j} \log_k^{\circ i}(n)}.$$

Proof Sketch: Privileged words

Upper Bound

Similar to the upper bound proof for closed words, we split the sum

$$P_k(n) = \sum_{t=1}^{n-1} P_k(n,t)$$

into two separate sums, one where $t > \lfloor n/2 \rfloor$ and one where $t \leq \lfloor n/2 \rfloor$.

Proof Sketch: Privileged words

Upper Bound

Similar to the upper bound proof for closed words, we split the sum

$$P_k(n) = \sum_{t=1}^{n-1} P_k(n,t)$$

into two separate sums, one where $t > \lfloor n/2 \rfloor$ and one where $t \leq \lfloor n/2 \rfloor$.

The same reasoning as in the closed words case shows that there are at most $nk^{\lceil n/2 \rceil}$ length-n privileged words with a longest border of length $t > \lfloor n/2 \rfloor$.

Proof Sketch: Privileged words

Upper Bound

Similar to the upper bound proof for closed words, we split the sum

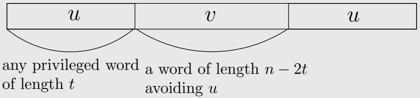
$$P_k(n) = \sum_{t=1}^{n-1} P_k(n,t)$$

into two separate sums, one where $t > \lfloor n/2 \rfloor$ and one where $t \leq \lfloor n/2 \rfloor$.

The same reasoning as in the closed words case shows that there are at most $nk^{\lceil n/2 \rceil}$ length-n privileged words with a longest border of length $t > \lfloor n/2 \rfloor$.

When $t \leq \lfloor n/2 \rfloor$, we have that the longest borders do not overlap.

For any length-n privileged word closed by a length-t privileged word u, we can write $w = uvu$.



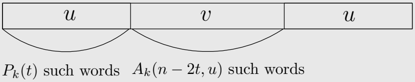
Proof Sketch: Privileged words

Upper Bound

$$\underbrace{u}_{P_k(t)\text{ such words}}\ \underbrace{v}_{A_k(n-2t,\,u)\text{ such words}}\ u$$

Proof Sketch: Privileged words

Upper Bound

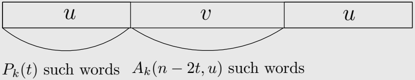


We already have an upper bound on $A_k(n-2t,u)$ from before, so we get the following recursive upper bound:

$$P_k(n,t) \le \sum_{|u|=t \text{ and } u \text{ privileged}} A_k(n-2t,u) \le \sum_{|u|=t \text{ and } u \text{ privileged}} A_k(n-2t,0^t)$$
$$\le P_k(t)(k-(k-1)k^{-t-1})^{n-2t}.$$

Proof Sketch: Privileged words

Upper Bound



We already have an upper bound on $A_k(n-2t,u)$ from before, so we get the following recursive upper bound:

$$P_k(n,t) \leq \sum_{|u|=t \text{ and } u \text{ privileged}} A_k(n-2t,u) \leq \sum_{|u|=t \text{ and } u \text{ privileged}} A_k(n-2t,0^t)$$
$$\leq P_k(t)(k-(k-1)k^{-t-1})^{n-2t}.$$

We can show by induction on j, that

$$P_k(n) \leq c_j \frac{k^n}{n\prod_{i=1}^{j}\log_k^{\circ i}(n)}$$

for all n large enough where $c_j > 0$ is some constant, and the base case when $j=0$ is $P_k(n) \leq C_k(n) \in O\left(\frac{k^n}{n}\right)$.

Open Problems

Let ℓ_n denote the smallest positive integer such that $\log_k^{\circ \ell_n + 1}(n) \leq 1$.

Define

$$\log_k^*(n) := \prod_{j=1}^{\ell_n} \log_k^{\circ j}(n).$$

Conjecture

For all $k \geq 2$, we have

$$P_k(n) \in \Theta\left(\frac{k^n}{n \log_k^*(n)}\right).$$

Open Problems

Let ℓ_n denote the smallest positive integer such that $\log_k^{\circ \ell_n + 1}(n) \leq 1$.

Define

$$\log_k^*(n) \coloneqq \prod_{j=1}^{\ell_n} \log_k^{\circ j}(n).$$

Conjecture

For all $k \geq 2$, we have

$$P_k(n) \in \Theta\left(\frac{k^n}{n \log_k^*(n)}\right).$$

This should be provable by a careful analysis of the multiplicative constants introduced in the proofs of the upper and lower bounds of $P_k(n)$.

Open Problems

Open Question

Does the limit

$$\lim_{n\to\infty} \frac{C_k(n)n}{k^n}$$

exist?

Open Problems

Open Question

Does the limit

$$\lim_{n\to\infty} \frac{C_k(n)n}{k^n}$$

exist?

There is reason to think that this limit does not exist.

Open Problems

Open Question

Does the limit

$$\lim_{n\to\infty} \frac{C_k(n)n}{k^n}$$

exist?

There is reason to think that this limit does not exist.

A **prefix-synchronized code** of length n is a set of codewords of length n that all begin with a fixed prefix u of length t, and every codeword is a prefix of a closed word of length $n+t$ that is closed by u.

A prefix-synchronized code is said to be **maximal** if it contains every possible codeword that begins with the fixed prefix.

Open Problems

Open Question

Does the limit

$$\lim_{n\to\infty} \frac{C_k(n)n}{k^n}$$

exist?

There is reason to think that this limit does not exist.

A **prefix-synchronized code** of length n is a set of codewords of length n that all begin with a fixed prefix u of length t, and every codeword is a prefix of a closed word of length $n+t$ that is closed by u.

A prefix-synchronized code is said to be **maximal** if it contains every possible codeword that begins with the fixed prefix.

Guibas and Odlyzko (1978) showed that for $k = 2, 3, 4$ a prefix-synchronized code is maximal if the fixed length-t prefix u is unbordered and $t \approx \log_k(n)$.

Open Problems

They showed that the ratio between the size of a maximal prefix-synchronized code of length n and $\frac{k^n}{n}$ oscillates between $0.3679\ldots$, and $0.3465\ldots$ (for $k=2$), $0.3171\ldots$ (for $k=3$), and 0.2911 (for $k=4$).

Open Problems

They showed that the ratio between the size of a maximal prefix-synchronized code of length n and $\frac{k^n}{n}$ oscillates between $0.3679\ldots$, and $0.3465\ldots$ (for $k=2$), $0.3171\ldots$ (for $k=3$), and 0.2911 (for $k=4$).

However, just looking experimentally, it seems that the ratio $\frac{nC_k(n)}{k^n}$ tends to a constant.

n	$\frac{nC_2(n)}{2^n}$	n	$\frac{nC_3(n)}{3^n}$	n	$\frac{nC_4(n)}{4^n}$
42	1.633719	19	1.044144	14	0.808427
43	1.630123	20	1.039228	15	0.802991
44	1.626657	21	1.034529	16	0.799168
45	1.623312	22	1.030038	17	0.796606
46	1.620082	23	1.025759	18	0.794974
47	1.616962	24	1.021695	19	0.793994
48	1.613948	25	1.017851	20	0.793439
49	1.611034	26	1.014232	21	0.793127
50	1.608218	27	1.010838	22	0.792917
51	1.605495	28	1.007667	23	0.792707

Open Problems

Question

Is it possible to compute the number of length-n closed words (resp. privileged words) in $o(C_k(n))$ (resp. $o(P_k(n))$) time?

Open Problems

Question

Is it possible to compute the number of length-n closed words (resp. privileged words) in $o(C_k(n))$ (resp. $o(P_k(n))$) time?

I suspect yes for closed words, using only what is currently known.

Open Problems

Question

Is it possible to compute the number of length-n closed words (resp. privileged words) in $o(C_k(n))$ (resp. $o(P_k(n))$) time?

I suspect yes for closed words, using only what is currently known.

Definition The **autocorrelation** u of a length-n binary word w is a binary word of length n that has a 1 at position i if and only if w has a border of length $n - i + 1$.

In this definition, borders are allowed to be non-proper.

For example, the word `redelivered` has autocorrelation 10000000100.

Open Problems

There is a recurrence for $B_k(n, u)$ that depends on n and the autocorrelation of u.

This implies that $B_k(n, u) = B_k(n, v)$ if u and v share the same autocorrelation.

Open Problems

There is a recurrence for $B_k(n, u)$ that depends on n and the autocorrelation of u.

This implies that $B_k(n, u) = B_k(n, v)$ if u and v share the same autocorrelation.

Guibas and Odlyzko (1981) presented a recurrence to count the number of words having a specific autocorrelation.

Guibas and Odlyzko also showed that there are a subexponential/superpolynomial number of length-n autocorrelations.

Open Problems

There is a recurrence for $B_k(n, u)$ that depends on n and the autocorrelation of u.

This implies that $B_k(n, u) = B_k(n, v)$ if u and v share the same autocorrelation.

Guibas and Odlyzko (1981) presented a recurrence to count the number of words having a specific autocorrelation.

Guibas and Odlyzko also showed that there are a subexponential/superpolynomial number of length-n autocorrelations.

To count closed words in $o(C_k(n))$ time, all we would need to do is

1. Count the number of periodic words of length n, handling the case where the longest border is of length $> \lfloor n/2 \rfloor$.
2. For the case where the longest border is of length $\leq \lfloor n/2 \rfloor$. Generate all autocorrelations of length $\leq \lfloor n/2 \rfloor$, compute $B_k(n, u)$ for every individual autocorrelation, and compute the number of words having every one of these autocorrelations.

Open Problems

However, this procedure takes superpolynomial time.

Open Problems

However, this procedure takes superpolynomial time.

Question

Is it possible to compute the number of length-n closed words in polynomial time?

Open Problems

However, this procedure takes superpolynomial time.

Question

Is it possible to compute the number of length-n closed words in polynomial time?

I have no idea how to count privileged words in $o(P_k(n))$ time.

Ranking/Unranking

Assume all closed/privileged words are listed in some order, perhaps lexicographic.

Ranking/Unranking

Assume all closed/privileged words are listed in some order, perhaps lexicographic.

Ranking: Given a length-n closed (resp. privileged) word, compute its position in the listing.

Unranking: Given a number between 1 and $C_k(n)$ (resp. $P_k(n)$), compute the length-n closed (resp. privileged) word at that position in the listing.

Ranking/Unranking

Assume all closed/privileged words are listed in some order, perhaps lexicographic.

Ranking: Given a length-n closed (resp. privileged) word, compute its position in the listing.

Unranking: Given a number between 1 and $C_k(n)$ (resp. $P_k(n)$), compute the length-n closed (resp. privileged) word at that position in the listing.

Example

Consider the length-5 closed words in lexicographic order.

Closed word	Position
00000	1
00100	2
01001	3
01010	4
01101	5
01110	6

Closed word	Position
10001	7
10010	8
10101	9
10110	10
11011	11
11111	12

Ranking: The position of the closed word 01101 is 5.
Unranking: The closed word at position 8 is 10010.

Ranking/Unranking

Of course, we could always rank/unrank by generating the entire list. But this is uninteresting.

Ranking/Unranking

Of course, we could always rank/unrank by generating the entire list. But this is uninteresting.

Can we rank/unrank more efficiently than generating the whole list?

Ranking/Unranking

Of course, we could always rank/unrank by generating the entire list. But this is uninteresting.

Can we rank/unrank more efficiently than generating the whole list?

Conjecture

Closed words can be ranked/unranked in lexicographic order in $o(C_k(n))$ time.

Ranking/Unranking

Of course, we could always rank/unrank by generating the entire list. But this is uninteresting.

Can we rank/unrank more efficiently than generating the whole list?

Conjecture

Closed words can be ranked/unranked in lexicographic order in $o(C_k(n))$ time.

It should be possible to derive a recurrence that counts the number of length-n closed words with a fixed prefix u.

- This recurrence would depend on n, u, and all autocorrelations of length $\leq \lfloor n/2 \rfloor$ that are compatible with u.

Ranking/Unranking

Of course, we could always rank/unrank by generating the entire list. But this is uninteresting.

Can we rank/unrank more efficiently than generating the whole list?

Conjecture

Closed words can be ranked/unranked in lexicographic order in $o(C_k(n))$ time.

It should be possible to derive a recurrence that counts the number of length-n closed words with a fixed prefix u.

- This recurrence would depend on n, u, and all autocorrelations of length $\leq \lfloor n/2 \rfloor$ that are compatible with u.

Then ranking a length-n closed word w involves summing this recurrence over at most kn words of the form va where v is a prefix of w and $a < w[|v|+1]$.

- This represents all closed words that are smaller than w in lexicographic order.

Ranking/Unranking

Of course, we could always rank/unrank by generating the entire list. But this is uninteresting.

Can we rank/unrank more efficiently than generating the whole list?

Conjecture

Closed words can be ranked/unranked in lexicographic order in $o(C_k(n))$ time.

It should be possible to derive a recurrence that counts the number of length-n closed words with a fixed prefix u.

- This recurrence would depend on n, u, and all autocorrelations of length $\leq \lfloor n/2 \rfloor$ that are compatible with u.

Then ranking a length-n closed word w involves summing this recurrence over at most kn words of the form va where v is a prefix of w and $a < w[|v| + 1]$.

- This represents all closed words that are smaller than w in lexicographic order.

Once ranking is solved, unranking can also be solved by repeated ranking.

Ranking/Unranking

This method of ranking/ranking, if it works, runs in superpolynomial time.

Ranking/Unranking

This method of ranking/ranking, if it works, runs in superpolynomial time.

Is it possible to do better than this?

Ranking/Unranking

This method of ranking/ranking, if it works, runs in superpolynomial time.

Is it possible to do better than this?

Lexicographic order would be preferred, but is it possible in any order?

Question

Can closed words be ranked/unranked in any order in polynomial time?

Ranking/Unranking

This method of ranking/ranking, if it works, runs in superpolynomial time.

Is it possible to do better than this?

Lexicographic order would be preferred, but is it possible in any order?

Question

Can closed words be ranked/unranked in any order in polynomial time?

I have no idea how to, even in principle, rank/unrank privileged words in $o(P_k(n))$ time.

Ranking/Unranking

This method of ranking/ranking, if it works, runs in superpolynomial time.

Is it possible to do better than this?

Lexicographic order would be preferred, but is it possible in any order?

Question

Can closed words be ranked/unranked in any order in polynomial time?

I have no idea how to, even in principle, rank/unrank privileged words in $o(P_k(n))$ time.

The first algorithms to rank/unrank bordered and unbordered words were recently presented:

D. Gabrić. Ranking and unranking bordered and unbordered words. Information Processing Letters. Vol. 184 (2024) 106452

J. Radoszewski, W. Rytter, T. Waleń. Faster Algorithms for Ranking/Unranking Bordered and Unbordered Words. String Processing and Information Retrieval. SPIRE 2024. Lecture Notes in Computer Science, vol 14899. (2025)

Generation

Generation: Compute every length-n closed (resp. privileged) word in lexicographic order (or any order) in time proportional to $C_k(n)$ (resp. $P_k(n)$), i.e., in constant-amortized time per word.

Generation

Generation: Compute every length-n closed (resp. privileged) word in lexicographic order (or any order) in time proportional to $C_k(n)$ (resp. $P_k(n)$), i.e., in constant-amortized time per word.

Naïvely, we can generate every length-n word in lexicographic order, and then perform a $O(n^2)$ test to determine whether it is closed.

Generation

Generation: Compute every length-n closed (resp. privileged) word in lexicographic order (or any order) in time proportional to $C_k(n)$ (resp. $P_k(n)$), i.e., in constant-amortized time per word.

Naïvely, we can generate every length-n word in lexicographic order, and then perform a $O(n^2)$ test to determine whether it is closed.

We can improve the test to $O(n)$ by realizing that a word is closed if and only if its longest border is strictly longer than the longest borders of all of its proper prefixes.

The failure function from the KMP string matching algorithm computes the length of the longest border of every prefix of a word, and can be computed in $O(n)$ time.

Generation

Generation: Compute every length-n closed (resp. privileged) word in lexicographic order (or any order) in time proportional to $C_k(n)$ (resp. $P_k(n)$), i.e., in constant-amortized time per word.

Naïvely, we can generate every length-n word in lexicographic order, and then perform a $O(n^2)$ test to determine whether it is closed.

We can improve the test to $O(n)$ by realizing that a word is closed if and only if its longest border is strictly longer than the longest borders of all of its proper prefixes.

The failure function from the KMP string matching algorithm computes the length of the longest border of every prefix of a word, and can be computed in $O(n)$ time.

Can we do better?

Conjecture

All length-n closed words can be generated in constant-amortized time per word in lexicographic order.

Generation

Conjecture solved! unpublished

All length-n closed words can be generated in constant-amortized time per word in lexicographic order.

Generation

Conjecture solved! unpublished

All length-n closed words can be generated in constant-amortized time per word in lexicographic order.

Idea

Generation

Conjecture solved! unpublished

All length-n closed words can be generated in constant-amortized time per word in lexicographic order.

Idea

The failure function can be computed symbol by symbol from left-to-right.

Generation

Conjecture solved! unpublished

All length-n closed words can be generated in constant-amortized time per word in lexicographic order.

Idea

The failure function can be computed symbol by symbol from left-to-right.

Start recursively generating all words in lexicographic order and compute the failure function in parallel.

Generation

Conjecture solved! unpublished

All length-n closed words can be generated in constant-amortized time per word in lexicographic order.

Idea

The failure function can be computed symbol by symbol from left-to-right.

Start recursively generating all words in lexicographic order and compute the failure function in parallel.

Based on the failure function, infer whether a prefix can be uniquely extended to a closed word.

Generation

Using the algorithm for closed words, one can compute all privileged words in amortized time between $\Omega(\log_k(n))$ and $O(\log_k^2(n))$ per word.

Generation

Using the algorithm for closed words, one can compute all privileged words in amortized time between $\Omega(\log_k(n))$ and $O(\log_k^2(n))$ per word.

Forsyth et al. (2016) gave an algorithm to decide whether a length-n word is privileged in $O(n)$ time.

This algorithm is based on the failure function and can be computed left-to-right symbol-by-symbol.

Generation

Using the algorithm for closed words, one can compute all privileged words in amortized time between $\Omega(\log_k(n))$ and $O(\log_k^2(n))$ per word.

Forsyth et al. (2016) gave an algorithm to decide whether a length-n word is privileged in $O(n)$ time.

This algorithm is based on the failure function and can be computed left-to-right symbol-by-symbol.

All we need to do is compute this algorithm in parallel to the recursive algorithm to generate all closed words.

Generation

Using the algorithm for closed words, one can compute all privileged words in amortized time between $\Omega(\log_k(n))$ and $O(\log_k^2(n))$ per word.

Forsyth et al. (2016) gave an algorithm to decide whether a length-n word is privileged in $O(n)$ time.

This algorithm is based on the failure function and can be computed left-to-right symbol-by-symbol.

All we need to do is compute this algorithm in parallel to the recursive algorithm to generate all closed words.

I have not figured out how to improve this algorithm to run in constant-amortized time per word.

Question

Is it possible to generate all length-n privileged words in constant-amortized time per word? In lexicographic order? In any order?

Thank you!