Ranking and Unranking k-subsequence universal words

Duncan Adamson

*Leverhulme Research Centre for Functional Materials Design, The University of Liverpool, UK**

d.a.adamson@liverpool.ac.uk

Abstract

A subsequence of a word w is a word u such that $u = w[i_1]w[i_2], \dots w[i_{|u|}]$, for some set of indices $1 \leq i_1 < i_2 < \cdots < i_k \leq |w|$. A word w is k-subsequence universal over an alphabet Σ if every word in Σ^k appears in w as a subsequence. In this paper, we provide new algorithms for k-subsequence universal words of fixed length n over the alphabet $\Sigma = \{1, 2, \dots, \sigma\}$. Letting $\mathcal{U}(n, k, \sigma)$ denote the set of n-length k-subsequence universal words over Σ, we provide:

- an $O(nk\sigma)$ time algorithm for counting the size of $\mathcal{U}(n, k, \sigma)$;
- an $O(nk\sigma)$ time algorithm for ranking words in the set $\mathcal{U}(n, k, \sigma)$;
- an $O(nk\sigma)$ time algorithm for unranking words from the set $\mathcal{U}(n, k, \sigma)$;
- an algorithm for enumerating the set $\mathcal{U}(n, k, \sigma)$ with $O(n\sigma)$ delay after $O(nk\sigma)$ preprocessing.

1 Introduction

Words and subsequences are two fundamental combinatorial objects. Informally, a subsequence of a word w is a word u that can be found by deleting some subset of the symbols w. Subsequences are a heavily studied object within computer science [Bar+20; Day+21; Fle+22; HSZ17; Kos+21; Lot97; MSY04; Sim03; Tro03; Zet16] and beyond, with applications in a wide number of fields including bioinformatics [HWG20; Shi+19], database theory [Art+17], and modelling concurrency [Sha78]. A recent survey of subsequence algorithms has been provided by Kosche et al. [Kos+22], highlighting major results for problems on funding subsequences in words.

This paper considers *k-subsequence universal* words. A word w is k-subsequence universal over an alphabet Σ if w contains every word of length k over Σ as a subsequence. These words were first defined by Karandikar and Schnoebelen [KS16; SK19] as *k-rich words*, however more recent work has used the term *k-subsequence universality* [Bar+20; Day+21; Kos+21], which we will use here. The study of these words follows from work on *Simon's congruence* [Sim75]. Informally, two words w, v are k-congruent if w and v share the same set of subsequences of length k. This relationship has been heavily studied [FK18; Sim03; Tro03; Zet16], with a recent asymptotically optimal algorithm derived for testing if two words are k-congruent [Gaw+21].

Most relevant to this work are the papers by Barker et al. [Bar+20], and Day et al. [Day+21], directly addressing k-subsequence universal words. In [Bar+20], the authors show that it is possible to determine, in linear time, if a word is k-subsequence universal or not, as well as the shortest k-subsequence universal prefix of a given word. Additionally, they provide results showing that the minimal set of ℓ-factors of a word w, $w_1 w_2 \dots, w_\ell$ such that $w_1 w_2 \dots w_\ell$ is k-subsequence universal, and the index i such that w^i is k-subsequence universal can be determined efficiently.

*This work was completed at, and partially funded by the University of Göttingen. 4

This is built on by [Day+21], in which the authors provide a set of algorithmic results for minimising the number of edit operations to transform a word into a k-subsequence universal word, providing results on *insertions*, *deletions*, and *substitutions*. They show that the minimum number of *insertions* and *substitutions* needed to transform a word w into a k-subsequence universal word w' can be done in $O(nk)$ time, assuming that $k < n$. Additionally, they show that the number of *deletions* needed to reduce the universality index (the maximum k such that the word is k-subsequence universal) of a word to k can be determined in $O(nk)$ time.

This paper is interested in providing algorithms for some of the basic operations on classes of words, *counting*, *ranking*, *unranking*, and *enumerating* for the class of k-subsequence universal words. In providing these algorithms, we aim to expand the understanding of the space of k-subsequence universal words of a fixed length n. We use $\mathcal{U}(n, k, \sigma)$ to denote the set of k-subsequence universal words of length n over an alphabet of size σ (assumed to be the alphabet $\{1, 2, \ldots, \sigma\}$). The counting problem asks for the number of words in a given class. The ranking problem takes as input a word w and determines the number of words within the set which are lexicographically smaller than w. The unranking problem is the inverse of the ranking problem, taking a rank i and asking for the word in the set with the rank i. Finally, the enumeration problem asks for the explicit outputting of every word within the set in some fixed order. Each of these problems has been heavily studied for other classes of words, including cyclic words [Ada22; Ada+21; FM78; GR61; KRR14; SW17] and Gray codes [FM78; KRR14; Sav97].

Our Results. This paper builds upon the existing body of work on k-subsequence universal words to build a stronger understanding of the space of k-subsequence universal words of a fixed length. We provide a suite of algorithmic results for k-subsequence universal words of fixed length n. We denote the set of all k-subsequence universal words over the alphabet $1, 2, \ldots, \sigma$ with length n by $\mathcal{U}(n, k, \sigma)$. In Section 3, we provide an an $O(nk\sigma)$ time algorithm for counting the size of $\mathcal{U}(n, k, \sigma)$. In Section 4, we use the observations from this counting algorithm to provide an $O(nk\sigma)$ time algorithm for ranking words in the set $\mathcal{U}(n, k, \sigma)$. Finally, in Section 5 we provide an $O(n\sigma)$ time algorithm for unranking within the set $\mathcal{U}(n, k, \sigma)$, with $O(nk\sigma)$ time preprocessing. We note this unranking algorithm directly provides an enumeration algorithm for the set $\mathcal{U}(n, k, \sigma)$ with $O(n\sigma)$ delay.

Computational Model. In this paper, we assume the unit cost RAM computational model, in this case, equivalent to the unit cost word-RAM with word size $O(\log(N)\log(\sigma))$, where N is the larger of the input or the output. We note that this remains logarithmic relative to the number of k-subsequence universal words of length n, and thus the bits required to output the integer representation of the number of such words. All our complexities can be readjusted into the unit cost RAM computational model with word size $O(\log(n)\log(\sigma))$ where n is the size of the input, by applying a multiplicative factor of $O(n/\log(n))$ to the stated bounds. We avoid a factor of $O(n^2/\log^2(n))$ by noting that these algorithms only perform multiplications where at least one integer has size at most σ or addition between integers of size at most σ^n.

2 Preliminaries

We use the following notation. Given a pair of natural numbers $m, n \in \mathbb{N}$, the notation $[m, n]$ denotes the ordered set $\{m, m+1, \ldots, n\}$, or the empty set if $m > n$. A word w is an ordered sequence of symbols over some alphabet Σ. The set of words of length n over the alphabet Σ is denoted Σ^n, and the set of all words over the alphabet Σ by Σ^*. The length of a word w is denoted $|w|$. The notation $w[i]$ is used to denote the i^{th} symbol in the word w, and $w[i, j]$ is used to denote the contiguous sequence within w corresponding to the word $w[i]w[i+1]\ldots w[j]$ (or the empty word ε if $i > j$). A word v is a *factor* of a word w if there exists some pair of indices i, j such that $v = w[i, j]$.

$$w = 11234, 4321, 22314, 33214, 4$$
$$v = 12234, 323134, 11234, 4412$$

Figure 1: An example of the arch-factorisation of two words $w, v \in \Sigma^{20}$ where $\Sigma = \{1, 2, 3, 4\}$. Each arch (or the ending suffix) is separated by a comma, and highlighted in a seperate colour. Note that w is 4-subsequence universal while v is only 3-subsequence universal, despite sharing the same Parikh vector $(5, 5, 5, 5)$.

We assume the alphabet $\Sigma = [1, \sigma]$ for some natural number $\sigma \geq 1$. Given two words $w, v \in \Sigma^n$, w is *lexicographically smaller* than v if there exists some index $i \in [1, n]$ such that $w[1, i-1] = v[1, i-1]$ and $w[i] < v[i]$. Given two words $w, v \in \Sigma^*$, v is a subsequence of w if and only if there exists some series of indices $1 \leq i_1 < i_2 < \cdots < i_{|v|} \leq |w|$ such that $v = w[i_1]w[i_2] \dots w[i_{|v|}]$.

Definition 1 (k-subsequence universality)**.** *A word $w \in \Sigma^n$ is k-subsequence universal if and only if every word $v \in \Sigma^k$ is a subsequence of w. The set of words of length n that are k-subsequence universal over an alphabet of size σ is denoted $\mathcal{U}(n, k, \sigma)$.*

The *subsequence universality index* of a word w is the largest value k such that w is k-subsequence universal. In order to determine if a word is k-subsequence universal, we use *arch-factorisations*, first introduced by Hebard [Heb91]. Informally, an *arch* of a word is a minimal length factor containing each symbol in the alphabet Σ at least once. For the remainder of this paper, we use the following formal definition:

Definition 2 (Arches)**.** *An* Arch *over the alphabet $\Sigma = \{1, 2, \dots, \sigma\}$ is a word containing every symbol in Σ at least once. The* universal subsequence *of an arch v is the set of indices $(i_1, i_2, \dots, i_\sigma)$ satisfying the following:*

- *$v[i_1], v[i_2], \dots, v[i_\sigma]$ contains every symbol in Σ exactly once.*
- *The index i_j is the first position in v where the symbol $v[i_j]$ appears.*
- *The index i_σ is the last position in v.*

Any symbol not in the universal subsequence is called a free symbol*.*

We note that this definition of an arch corresponds to a 1-subsequence universal word where the last symbol is unique, i.e. it does not appear anywhere else in the word.

Definition 3 ([Heb91], Arch Factorisations)**.** *The arch-factorisation of a word $w \in \Sigma^n$ with a universality index of k, is a set of factors $\{w_1, w_2, \dots, w_k, v\}$, denoted $Arch(w)$, such that w_i is a factor of w that is an arch for every $i \in [1, k]$, v is a suffix of w that does not contain any arch as a factor, and $w_1 w_2 \dots w_k v = w$.*

An example of this factorisation is given in Figure 1. Day et al. [Day+21] expanded upon Definition 3 to show that a word is k-subsequence universal if and only if there exists an arch-factorisation of w containing at least k-arches, and further, that such a factorisation can be computed in time linear to the length of the word.

Theorem 1 ([Day+21])**.** *A word $w \in \Sigma^n$ is k-subsequence universal over Σ if and only if $Arch(w)$ contains at least k arches. Further, $Arch(w)$ can be computed in $O(n)$ time.*

In this paper, we use the following technical Lemma from [Day+21].

Lemma 1 ([Day+21]). *Let $\Delta(w, i, j)$ denote the number unique symbols in $w[i, j]$ for some $w \in \Sigma^n$. We can compute in $O(n)$ the values of $\Delta(w, 1, j)$ for every $j \in [1, n]$.*

We combine Theorem 1 and Lemma 1 to make the following observation.

Observation 1. *Let $w \in \Sigma^n$ be a k-subsequence universal word with the arch-decomposition $w_1, w_2, \ldots, w_k, v$, and further let $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ denote the set of indices where $A_\ell = 1 + \sum_{i \in [1,\ell-1]} |w_i|$, i.e. the set of indices in w corresponding to the first position of an arch in $Arch(w)$. Then, the values of $\Delta(w, A_\ell, i_\ell)$ can be computed in $O(n)$ time for every $\ell \in [k]$ and $i_\ell \in [A_\ell, A_{\ell+1} - 1]$, where $\Delta(w, A_\ell, i_\ell)$ denotes the number unique symbols in $w[A_\ell, i_\ell]$.*

Using this notation, we provide a formal definition of the ranking and unranking problems as considered in this paper. The *rank* of a word w within an ordered set of words $\mathcal{S}$ is the number of words in $\mathcal{S}$ that are smaller than w under the ordering of the set. In this paper, we assume that the set of k-subsequence universal words is ordered lexicographically, and therefore the rank of a word w in the set $\mathcal{U}(n, k, \sigma)$ is the number of words in $\mathcal{U}(n, k, \sigma)$ that are lexicographically smaller than w. The *ranking* problem takes as input a word w integer triple $n, k, \sigma \in \mathbb{N}$ such that $n \geq k\sigma$, and returns the number of words in $\mathcal{U}(n, k, \sigma)$ lexicographically smaller than w. The *unranking* problem is conceptually the inverse of the ranking problem. Given an integer $i \in [1, |\mathcal{S}|]$, the unranking problem asks for the word in $\mathcal{S}$ with a rank of i. In this paper, the unranking problem takes as input a rank i and integer triple $n, k, \sigma \in \mathbb{N}$ such that $n \geq k\sigma$, and returns the word in $\mathcal{U}(n, k, \sigma)$ with a rank of i.

3 Counting Arches and k-subsequence universal words

Idea: start with $(\sigma!)^k$, then insert symbols form $\sigma - 1$ such that, when x is inserted between y_1 and y_2, x can be any value other than y_2, that way we avoid overcounting.

First, we present a tool for counting the size of $\mathcal{U}(n, k, \sigma)$, i.e. number of k-subsequence universal words of length n over the alphabet $\Sigma = \{1, 2, \ldots, \sigma\}$. As well as being an interesting result in and of itself, this provides the foundation for our tools for both ranking and unranking.

This section is split into two sections. First, we provide formulae for counting the number of arches, 0-subsequence universal words, and 1-universal words of length n over an alphabet of size σ. Second, we provide a recursive technique to count the number of k-subsequence universal words of length n over an alphabet of size σ.

3.1 Arches, 0-subsequence universal and 1-subsequence universal words

We first consider how to count the number of arches, 0-subsequence universal and 1-subsequence universal word of length n. We note that these three special cases are closely interlinked. First, note that any word that is not 0-subsequence universal must be at least 1-subsequence universal. Therefore, the number of 1-subsequence universal words is equal to the number of words minus the number of 0-subsequence universal words. Similarly, the number of n-length arches is equal to the number of $(n-1)$-length 1-subsequence universal words over an alphabet of size $\sigma - 1$, multiplied by σ. We start with 0-subsequence universal words.

Lemma 2. *The number of n-length 0-subsequence universal words over an alphabet $\Sigma = \{1, 2, \ldots, \sigma\}$ is given by:*

$$\sum_{i \in [1,\sigma]} (-1)^{i+1} \binom{\sigma}{i} (\sigma - i)^n.$$

Proof. Let $\varsigma \subseteq \Sigma$ be a i-length alphabet. Note first that the number of n-length words over ς is given by i^n, and further, as there are $\binom{\sigma}{i}$ such alphabets, the total number of words over *any* i-length alphabet is given by $\binom{\sigma}{i} i^n$. Observe that any string in ς^n is also in $(\varsigma \cup \{x\})^n$,

for some $x \in \Sigma \setminus \varsigma$, and further, there are $\sigma - i$ such $i+1$-length alphabets containing every symbol in ς. More generally, there are $\binom{\sigma-i}{j}$ alphabets of size $i+j$ containing every symbol in the i-length alphabet ς. Therefore, taking the sum of n-length words in all i-length alphabets, given by $\binom{\sigma}{i}(i)^n$, will also count every word in a $j < i$-length language $\binom{\sigma-j}{i-j}$ times. Combining this with the well-known binomial coefficient identities gives the equation for the total number of unique words in any alphabet in the set $\{\Sigma \setminus \{x\} \mid x \in \Sigma\}$ as:

$$\binom{\sigma}{1}(\sigma-1)^{n-1} - \binom{\sigma}{2}(\sigma-2)^{n-1} + \binom{\sigma}{3}(\sigma-3)^{n-1} \dots (-1)^{\sigma+1}$$
$$= \sum_{i \in [1,\sigma]} (-1)^{i+1}\binom{\sigma}{i}(\sigma-i)^n$$

□

Using Lemma 2, the counting of n-length arches and 1-subsequence universal words follows directly.

Corollary 1. *The number of n-length 1-universal words over an alphabet $\Sigma = \{1, 2, \dots, \sigma\}$ is given by:*

$$\sigma^n - \sum_{i \in [1,\sigma]} (-1)^{i+1}\binom{\sigma}{i}(\sigma-i)^n$$

Corollary 2. *The number of n-length Arches over an alphabet $\Sigma = \{1, 2, \dots, \sigma\}$ is given by:*

$$\sigma(\sigma-1)^{n-1} - \sum_{i \in [2,\sigma]} (-1)^i i\binom{\sigma}{i}(\sigma-i)^{n-1}.$$

3.2 Counting k-subsequence universal words

To count k-subsequence universal words with an arbitrary value of k, we employ a recursive approach. The high-level idea is to count the number of suffixes of k-subsequence universal words sharing a given prefix v. Let $\mathcal{S}(v)$ be the set of words of length $n - \mid v \mid$ such that for every word $u \in \mathcal{S}(v)$, the word vu is a k-subsequence universal word. Let $Arch(v) = v_1, v_2, \dots, v_\ell, v'$ be the arch factorisation of v, or the set of the first k arches of v'. In order to count the size of $\mathcal{S}(v)$, we observe that every word $u \in \mathcal{S}(v)$ must contain a prefix u' such that $v'u'$ is an arch and the suffix $u[|u'|+1, |u|]$ must contain $k - \ell - 1$ arches. Our recursive approach is based on the observation that the size of $\mathcal{S}(v)$ is equal to the size of $\bigcup_{x \in \Sigma} \mathcal{S}(vx)$. This leaves two major problems: determining whether or not the set $\mathcal{S}(vx)$ is empty, and ensuring that the total size of $\mathcal{S}(v)$ can be computed without having to explicitly check $\mathcal{S}(vw)$ for every suffix $w \in \Sigma^{n-|v|}$.

We solve these problems by introducing a new function, $\mathrm{CS}(q, m, c)$ (**C**ount **S**uffixes) such that $|\mathcal{S}(v)| = \mathrm{CS}(q, m, c)$ where:

- q is the number of unique symbols in v'.
- m is the number of free symbols in every word $u \in \mathcal{S}(v)$, i.e. the number of symbols in u that do not belong to any universal subsequence of the first $k-\ell-1$ arches of $u[|u'|+1, |u|]$ or in the universal subsequence of u' in the word $v'u'$.
- c is the minimum number of arches in $v'u$, equal to $k - \ell$.

The value of $\mathrm{CS}(q, m, c)$ is determined in a recursive manner. We first provide the base cases. If $m = 0$, then every remaining symbol must be in the universal subsequence for one of the remaining c arches, giving $\mathrm{CS}(q, 0, c) = (\sigma - q)!(\sigma!)^{c-1}$. On the other hand, if $c = 0$, then the remaining symbols can be chosen arbitrarily from Σ, giving $\mathrm{CS}(q, m, 0) = \sigma^m$. Assuming both

c and m are greater than 0, then the value of $\mathrm{CS}(q, m, k)$ is determined recursively. If $q = \sigma$, then the next symbol must be the first symbol of the $(k-c)^{th}$ arch of the word. As there are σ such possible symbols, followed by one of $\mathrm{CS}(1, m, c-1)$ suffixes, the value of $\mathrm{CS}(\sigma, m, c)$ is $\sigma\,\mathrm{CS}(1, m, c-1)$. Otherwise, the next symbol can either be one of the q symbols already in the universal subsequence of the current arch, or one of the $\sigma - q$ symbols not in the universal subsequence, giving $\mathrm{CS}(q, m, c) = (\sigma - q)\,\mathrm{CS}(q+1, m, c) + q\,\mathrm{CS}(q, m-1, c)$. Putting this together, the function $\mathrm{CS}(q, m, c)$ can be defined as:

$$\mathrm{CS}(q, m, c) = \begin{cases} (\sigma - q)!(\sigma!)^{c-1} & m = 0 \\ \sigma^m & c = 0 \\ \sigma\,\mathrm{CS}(1, m, c-1) & q = \sigma, m > 0, c > 0 \\ (\sigma - q)\,\mathrm{CS}(q+1, m, c) + q\,\mathrm{CS}(q, m-1, c) & q > 0, m > 0, c > 0 \end{cases}$$

Lemma 3. *Let $\mathcal{S}(v)$ denote the set of suffixes such that for every $u \in \mathcal{S}(v)$, the word vu is an n-length k-subsequence word. Further, let ℓ be the number of complete arches in v, and let v' be the suffix of v such that $Arch(v) = v_1 v_2 \dots v_{\ell-1} v_\ell v' = v$ where v_i is the i^{th} arch in the arch factorisation for v. Then, the size of $\mathcal{S}(v)$ is equal to $\mathrm{CS}(q, m, k-\ell)$ where q is the number of unique symbols in v', and $m = n - (|v| + \sigma(k - \ell - 1))$.*

Proof. We will assume, for notational simplicity, that if $v' = \varepsilon$, then $q = \sigma$. We start with the base cases. If $m = 0$, every symbol in the suffixes of $\mathcal{S}(v)$ must be in the universal subsequence of one of the remaining arches. As there are q symbols in v', there are $(\sigma - q)!$ possible ways of extending v' to become an arch, and $\sigma!$ arches of length σ, the total number of suffixes in $\mathcal{S}(v)$ is $(\sigma - q)!(\sigma!)^{k-\ell-1} = (\sigma - q)(\sigma!)^{c-1}$. Alternatively, if $\ell \geq k$ (meaning that v is already a k-subsequence universal word), then every remaining symbol in the suffix is a free symbol, and as such there are no constraints on the contents of the suffix. Therefore in this case, there are σ^m suffixes in $\mathcal{S}(v)$. Note that in the case $m = 0$ and $\ell \geq k$, both of these formulae return 0, corresponding to the empty word.

In the general case, assume that the size of $\mathcal{S}(vx)$ is equal to $\mathrm{CS}(q', m', c')$, where q' is the number of unique symbols in $v'x$, m' is equal to $n - (|v| + 1) - (q + \sigma(k - \ell))$, and c' is $k - \ell$. Observe that the total number of suffixes in $\mathcal{S}(v)$ is equal to $\sum_{x \in \Sigma} |\mathcal{S}(vx)|$. If $q = \sigma$, then the next symbol must belong to the universal subsequence of the next arch, equal to the value of $\mathrm{CS}(1, m, k - \ell - 1)$. If x is one of the q symbols that have already appeared in v', then the size of $\mathcal{S}(vx)$ is $\mathrm{CS}(q, m-1, c)$. Otherwise, the size of $\mathcal{S}(vx)$ is $\mathrm{CS}(q+1, m, c)$. As there are q unique symbols in v', the number of suffixes is given by the sum $q\,\mathrm{CS}(q, m-1, c) + (\sigma - q)\,\mathrm{CS}(q+1, m, c)$. Hence the size of $\mathcal{S}(v)$ is given by $\mathrm{CS}(q, m, k - \ell)$. □

Lemma 4. *The values of $\mathrm{CS}(q, m, c)$ can be computed for every $q \in [1, \sigma], m \in [0, n], c \in [0, k]$ in $O(nk\sigma)$ time.*

Proof. The correctness follows for the arguments above. We assume that the values of $(\sigma - q)!(\sigma!)^c$ have been precomputed for every $q \in [1, \sigma], c \in [0, k]$, requiring $O(k\sigma)$ time, and the values of σ^m have been precomputed for every $m \in [0, n]$ requiring $O(n)$ time. To determine the time complexity, note that the value of $\mathrm{CS}(q, m, c)$ can be computed in constant time assuming that the values of $\mathrm{CS}(q+1, m, c)$, $\mathrm{CS}(1, m, c-1)$, and $\mathrm{CS}(q, m-1, c)$ have already been computed. As the base cases of $c = 0$, and $m = 0$ can be computed in constant time, and for every other case the values of m and $(\sigma - q) + \sigma c$ are monotonically decreasing, the values of $\mathrm{CS}(q, m, c)$ can be computed for every $q \in [1, \sigma], m \in [0, n], c \in [0, k]$ in a dynamic manner, starting with the base cases, and proceeding in increasing value of m, $\sigma - q$ and c. We note that the order in which m, q and c are incremented is irrelevant provided $\mathrm{CS}(q+1, m, c)$, $\mathrm{CS}(1, m, c-1)$, and $\mathrm{CS}(q, m-1, c)$ are computed before $\mathrm{CS}(q, m, c)$. Therefore, the total time complexity of computing the values of $\mathrm{CS}(q, m, c)$ for every $q \in [1, \sigma], m \in [0, n], c \in [0, k]$, is $O(nk\sigma)$. □

Note that the number of k-subsequence universal words is equal to the number of words in the set $\mathcal{S}(\varepsilon)$, i.e. the number of n-length words with an empty prefix. Therefore, from Lemma 4, it follows that the size of $\mathcal{U}(n,k,\sigma)$ can be computed by counting the size of $\mathcal{S}(\varepsilon)$, equivalent to evaluating $\sigma\,\mathrm{CS}(1,n-(k\sigma),k)$. Theorem 2 follows from this observation.

Theorem 2. *The size of $\mathcal{U}(n,k,\sigma)$ can be computed in $O(nk\sigma)$ time.*

4 Ranking

Using the counting techniques outlined in Section 3, we can now rank a given word $w \in \Sigma^n$ amongst the set of n-length k-subsequence universal words. This is done in an iterative manner. For each $i \in [1,n]$, we count the number of words of n-length k-subsequence universal words with the prefix $w[1,i-1]x$, where x is some symbol lexicographically smaller than $w[i]$. Taking the sum of such words for every $i \in [1,n]$ gives the total number of n-length k-subsequence universal words that are lexicographically smaller than w. By taking the sum of such words for each prefix, the total number of n-length k-subsequence universal words that are lexicographically smaller than w can be computed.

Let $Arch(w) = w_1, w_2, \ldots, w_m v$. The first key observation is that given any word $u = w_1 w_2 \ldots w_i u'$, for some $i \leq k$, u is k-subsequence universal if and only if u' is $(k-i)$-subsequence universal. Secondly, given a word $s = w_1 w_2 \ldots w_i[1:j]s'$, s is k-subsequence universal if and only if $w_i[1:j]s'$ is $(k-i+1)$-subsequence universal.

Preprocessing. In order to make our ranking algorithm more efficient, we first provide an overview of the preprocessing that is performed before the main ranking algorithm. Let $Arch(w) = w_1, w_2, \ldots, w_k, v$. Using the notation from Observation 1, let $A_1, A_2, \ldots, A_k$ be the indices such that A_ℓ corresponds to the first position in w at which the arch w_ℓ appears in w, i.e. $w_\ell = w[A_\ell, A_{\ell+1} - 1]$. Further, let $\Delta(w, A_\ell, i)$ be the number of unique symbols in the i-length prefix of w_ℓ. We assume that the values of $\Delta(w, A_\ell, i_\ell)$ have been computed for every $i \in [A_\ell, A_{\ell+1} - 1]$, $\ell \in [0,k]$.

In order to count the number of free symbols within each suffix of w, let m be an n-length array such that $m[i]$ contains the number of free symbols in the i-length suffix of w. The values of $m[i]$ are computed by starting with $i = n$, and working in decreasing value of i. Note that the value of $m[n]$ is equal to 0. In the general case, the value of $m[i]$ is either $m[i+1]$, if $w[i]$ belongs to the universal subsequence of some arch in the arch decomposition, of $m[i+1]+1$ otherwise. Letting ℓ be the index such that $A_\ell \leq i < A_{\ell+1}$, note that if $w[i]$ is in the universal subsequence of w_ℓ, then $\Delta(w, A_\ell, i) = \Delta(w, A_\ell, i-1) + 1$, otherwise $\Delta(w, A_\ell, i) = \Delta(w, A_\ell, i-1)$. Hence using the previous computation, the values of $m[i]$ can be determined in $O(n)$ time for every $i \in [1,n]$.

In order to determine the number of symbols smaller than $w[i]$, an additional n-length array l such that $l[i]$ contains the set of symbols that appear between A_ℓ and i in w, where ℓ is the index such that $A_\ell \leq i < A_{\ell+1}$. This complements m by ensuring providing a quick method of checking if a given symbol x has already been used by $w_\ell[1, i+1-A_\ell]$. The array l is computed in $O(n\sigma)$ time as follows. For each $i \in [1,n]$, note that the value of $l[i]$ is either $l[i-1] \cup \{w[i]\}$, if $i \neq A_\ell$ for every $\ell \in [0,k]$, or $\{w[i]\}$ otherwise. By storing each array as a σ-length binary vector, requiring at most $O(n\sigma)$ time to initialise, the values of $l[i]$ can be computed for every $i \in [1,n]$ in $O(n)$ time. Finally, we assume that the value of $\mathrm{CS}(q,m,c)$ has been precomputed for every $q \in [1,\sigma], m \in [n]$ and $c \in [k]$.

Ranking. We now have the tools we need to rank the input word $w \in \Sigma^n$. We note that w does not have to be a k-subsequence universal word, allowing this tool to be used in a more general setting. At a high level, our approach is to take each prefix of w, $w[1,i]$, and count the number of words in $\mathcal{U}(k,n,\sigma)$ that are lexicographically smaller than w with the prefix $w[1,i]x$,

where $x < w[i+1]$. By taking the sum of such words for each prefix of w, the total number of words smaller than w can be determined.

Let $\mathcal{R}(i) = \{u \in \mathcal{U}(k,n,\sigma) \mid u < w, u[1,i] = w[1,i], u[i+1] < w[i+1]\}$, and let ℓ be the index such that $A_\ell \leq i < A_{\ell+1}$. Note that the number of possible values for the symbol $u[i+1]$ is equal to $w[i+1]-1$. Further, the number of words in $\mathcal{R}(i)$ with the prefix $w[1,i]x$ for some fixed $x < w[i+1]$ is equal to either $\mathrm{CS}(\Delta(w,A_\ell,i)+1, m[i], k+1-\ell)$, if x is in the universal subsequence of w_ℓ or $\mathrm{CS}(\Delta(w,A_\ell,i), m[i]-1, k+1-\ell)$ otherwise. Recall that the array l contains at position i the set of unique symbols in the factor of w $w[A_\ell, i]$. Therefore, the size of $\mathcal{R}(i)$ can be computed with this following sum:

$$|\mathcal{R}(i)| = \sum_{x \in [1, w[i+1]-1]} \begin{cases} \mathrm{CS}(\Delta(w,A_\ell,i)+1, m[i], k+1-\ell) & x \in l[i] \\ \mathrm{CS}(\Delta(w,A_\ell,i), m[i]-1, k+1-\ell) & x \notin l[i] \end{cases}$$

Using $\mathcal{R}(i)$, rank of w in the set $\mathcal{U}(n,k,\sigma)$, denoted $rank(w)$, is given by:

$$rank(w) = \sum_{i \in [0,n-1]} |\mathcal{R}(i)|$$

Theorem 3. *The rank of a given word $w \in \Sigma^n$ can be determined in $O(nk\sigma)$ time.*

Proof. Observe that for any word of the form $w[i]xv$ to be k-subsequence universal, the suffix v must belong to $\mathcal{S}(w[i]x)$. Let ℓ be the index such that $A_\ell \leq i < A_{\ell+1}$, q be the number of unique symbols in $w[A_\ell, i]$ and m the number of free symbols following $w[1,i]x$. Note that the number of possible values of v is either $\mathrm{CS}(q+1, m, k-\ell)$, if x is not in the universal subsequence of $w[A_\ell, i]$, or $\mathrm{CS}(q, m-1, k-\ell)$ if x has already appeared in $w[A_\ell, i]$. Using the list l, it can be determined in constant time if the symbol x appears in $w[A_\ell, i]$. By extension, the total number of n-length k-subsequence universal words with the prefix $w[i]x$ can be computed in constant time, assuming that the values of $\mathrm{CS}(q, m, k-\ell)$ has been precomputed, and hence the value of $\mathcal{R}(i)$ can be computed in $O(\sigma)$ time. As there are n possible prefixes of w, the total rank of w within $\mathcal{U}(n,k,\sigma)$ can be computed in $O(n\sigma)$ time after $O(n\sigma k)$ preprocessing. □

5 Unranking

We complement our counting and ranking techniques by showing how to unrank n-length k-subsequence universal words. Note that an efficient unranking technique may be used as an effective tool to enumerate the set of all k-subsequence universal words. We assume that the values of $\mathrm{CS}(q,m,k)$ have been precomputed for every $q \in [1,\sigma], m \in [0,n]$, and $c \in [0,k]$.

Our unranking processes operates in an iterative manner. Let w be the word of rank i that is being unranked. Starting with $j=1$, the value of $w[j]$ is computed by counting the number of n-length k-subsequence universal words with the prefix $w[1,j-1]x$, for $x \in \Sigma$ starting with $x=1$. The value of x is increased until the number of words with a prefix smaller than or equal to $w[1,j-1]x$ is greater than i. Once this value of x has been computed, $w[j]$ is set to $x-1$, and the algorithm proceeds to compute the value of $w[j+1]$.

Theorem 4. *The k-subsequence universal word w of length n with a rank of i can be determined in $O(n\sigma + nk\sigma)$ time.*

Proof. Starting with $w[1]$, note that the number of words with the prefix x, for any $x \in \Sigma$, is given by $\mathrm{CS}(1, n-(k\sigma), k)$. Further, any word with the first symbol x has a rank in the range $(x-1)\,\mathrm{CS}(1, n-(k\sigma), k)+1$ to $x\,\mathrm{CS}(1, n-(k\sigma), k)$. Therefore the value of $w[1]$ is the value of x such that $(x-1)\,\mathrm{CS}(1, n-(k\sigma), k) < i \leq x\,\mathrm{CS}(1, n-(k\sigma), k)$.

More generally, let $t(j)$ be the smallest rank of words with the prefix $w[1,j]$, determined by the sum:

$$t(j) = \sum_{\ell \in [1,j]} \sum_{x \in [1, w[\ell]-1]} |\mathcal{S}(w[1,\ell-1]x)| = t(j-1) + \sum_{x \in [1, w[j]-1]} |\mathcal{S}(w[1,j-1]x)|.$$

Note that the value of $t(j)$ can therefore be computed in $O(\sigma)$ time using $t(j-1)$ and the values of $\mathrm{CS}(q,m,c)$. The value of $w[j+1]$ is, therefore, the symbol x such that $t(j) + \sum_{y\in[1,x-1]} |\mathcal{S}(w[1,j]y)| < i \leq t(j) + \sum_{y\in[1,x]} |\mathcal{S}(w[1,j]y)|$, and further can be computed in $O(\sigma)$ time, giving the total time complexity of the unranking of w as $O(n\sigma)$ after $O(n\sigma k)$ preprocessing. □

Corollary 3. *The set of k-subsequence universal words of length n can be output explicitly with $O(n\sigma)$ delay after $O(nk\sigma)$ preprocessing.*

Proof. Following Theorem 4, each index $i \in [1, |\mathcal{U}(n,k,\sigma)|]$ can be unranked in $O(n\sigma)$ time after at most $O(n\sigma k)$ preprocessing. Hence the set $\mathcal{U}(n,\sigma,k)$ can be enumerated with $O(n\cdot\sigma)$ delay after $O(nk\sigma)$ preprocessing. □

6 Conclusion

In this paper, we provided new tools for understanding the space of k-subsequence universal words. Notably, we have shown how to count, rank, unrank, and enumerate these words with efficient algorithms for words of fixed length. We note that all of these algorithms can be extended to the setting of words of length at most n. We see two key open questions asked in this paper. First, if there is a general formula for counting the number of n-length k-subsequence universal words. Indeed, such a formula may allow for a speed up for the preprocessing of the ranking, unranking, and enumeration algorithms, if it can be extended to count the size of $\mathcal{S}(v)$ efficiently. Secondly, if there is an enumeration algorithm outputting every word in $\mathcal{U}(n,k,\sigma)$ with at most $\mathrm{O}(n)$ delay after polynomial-time preprocessing.

The author thanks the Leverhulme Trust for funding this research via the Leverhulme Research Centre for Functional Materials Design. Further, the author would like to thank the reviewers for their helpful comments that have improved the readability of this paper.

References

[Ada+21] Duncan Adamson, Argyrios Deligkas, Vladimir V. Gusev, and Igor Potapov. "Ranking Bracelets in Polynomial Time". In: *32nd Annual Symposium on Combinatorial Pattern Matching* (2021), pp. 4–17.

[Ada22] Duncan Adamson. "Ranking binary unlabelled necklaces in polynomial time". In: *Descriptional Complexity of Formal Systems: 24th IFIP WG 1.02 International Conference, DCFS 2022, Debrecen, Hungary, August 29–31, 2022, Proceedings*. Springer. 2022, pp. 15–29.

[Art+17] Alexander Artikis et al. "Complex event recognition languages: Tutorial". In: *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*. 2017, pp. 7–10.

[Bar+20] Laura Barker et al. "Scattered factor-universality of words". In: *Developments in Language Theory: 24th International Conference, DLT 2020, Tampa, FL, USA, May 11–15, 2020, Proceedings*. Springer. 2020, pp. 14–28.

[Day+21] Joel D. Day et al. "The Edit Distance to k-Subsequence Universality". In: *38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021)*. Ed. by Markus Bläser and Benjamin Monmege. Vol. 187. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 25:1–25:19.

[FK18] Lukas Fleischer and Manfred Kufleitner. "Testing Simon's congruence". In: *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2018.

[Fle+22] Pamela Fleischmann et al. “Nearly k-universal words-investigating a part of simon’s congruence”. In: *Descriptional Complexity of Formal Systems: 24th IFIP WG 1.02 International Conference, DCFS 2022, Debrecen, Hungary, August 29–31, 2022, Proceedings*. Springer. 2022, pp. 57–71.

[FM78] Harold Fredricksen and James Maiorana. “Necklaces of beads in k colors and k-ary de Bruijn sequences”. In: *Discrete Mathematics* 23.3 (1978), pp. 207–210.

[Gaw+21] Paweł Gawrychowski et al. “Efficiently Testing Simon’s Congruence”. In: *38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021)*. Ed. by Markus Bläser and Benjamin Monmege. Vol. 187. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 34:1–34:18.

[GR61] Edgar N. Gilbert and John Riordan. “Symmetry types of periodic sequences”. In: *Illinois Journal of Mathematics* 5.4 (1961), pp. 657–665.

[Heb91] Jean-Jacques Hebrard. “An algorithm for distinguishing efficiently bit-strings by their subsequences”. In: *Theoretical computer science* 82.1 (1991), pp. 35–49.

[HSZ17] Simon Halfon, Philippe Schnoebelen, and Georg Zetzsche. “Decidability, complexity, and expressiveness of first-order logic over the subword ordering”. In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. 2017, pp. 1–12.

[HWG20] Renmin Han, Sheng Wang, and Xin Gao. “Novel algorithms for efficient subsequence searching and mapping in nanopore raw signals towards targeted sequencing”. In: *Bioinformatics* 36.5 (2020), pp. 1333–1343.

[Kos+21] Maria Kosche, Tore Koß, Florin Manea, and Stefan Siemer. “Absent Subsequences in Words”. In: *Reachability Problems*. Ed. by Paul C. Bell, Patrick Totzke, and Igor Potapov. Cham: Springer International Publishing, 2021, pp. 115–131.

[Kos+22] Maria Kosche, Tore Koß, Florin Manea, and Stefan Siemer. “Combinatorial Algorithms for Subsequence Matching: A Survey”. In: Proceedings 12th International Workshop on *Non-Classical Models of Automata and Applications,* Debrecen, Hungary, August 26-27, 2022. Ed. by Henning Bordihn, Géza Horváth, and György Vaszil. Vol. 367. Electronic Proceedings in Theoretical Computer Science. Open Publishing Association, 2022, pp. 11–27.

[KRR14] Tomasz Kociumaka, Jakub Radoszewski, and Wojciech Rytter. “Computing k-th Lyndon word and decoding lexicographically minimal de Bruijn sequence”. In: *Combinatorial Pattern Matching. CPM 2014. Lecture Notes in Computer Science, vol 8486*. Springer, 2014, pp. 202–211.

[KS16] Prateek Karandikar and Philippe Schnoebelen. “The height of piecewise-testable languages with applications in logical complexity”. In: *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2016.

[Lot97] Monsieur Lothaire. *Combinatorics on words*. Vol. 17. Cambridge university press, 1997.

[MSY04] Alexandru Mateescu, Arto Salomaa, and Sheng Yu. “Subword histories and Parikh matrices”. In: *Journal of Computer and System Sciences* 68.1 (2004), pp. 1–21.

[Sav97] Carla Savage. “A survey of combinatorial Gray codes”. In: *SIAM review* 39.4 (1997), pp. 605–629.

[Sha78] Alan C. Shaw. “Software descriptions with flow expressions”. In: *IEEE Transactions on Software Engineering* 3 (1978), pp. 242–254.

[Shi+19] Rayhan Shikder, Parimala Thulasiraman, Pourang Irani, and Pingzhao Hu. “An OpenMP-based tool for finding longest common subsequence in bioinformatics”. In: *BMC research notes* 12 (2019), pp. 1–6.

[Sim03] Imre Simon. “Words distinguished by their subwords”. In: *Proc. WORDS 2003* 27 (2003), pp. 6–13.

[Sim75] Imre Simon. “Piecewise testable events”. In: *Automata Theory and Formal Languages: 2nd GI Conference Kaiserslautern, May 20–23, 1975*. Springer. 1975, pp. 214–222.

[SK19] Philippe Schnoebelen and Prateek Karandikar. “The height of piecewise-testable languages and the complexity of the logic of subwords”. In: *Logical Methods in Computer Science* 15 (2019).

[SW17] Joe Sawada and Aaron Williams. “Practical algorithms to rank necklaces, Lyndon words, and de Bruijn sequences”. In: *Journal of Discrete Algorithms* 43 (2017), pp. 95–110.

[Tro03] Zdenêk Troniĉek. “Common subsequence automaton”. In: *Implementation and Application of Automata: 7th International Conference, CIAA 2002 Tours, France, July 3–5, 2002 Revised Papers*. Springer. 2003, pp. 270–275.

[Zet16] Georg Zetzsche. “The complexity of downward closure comparisons”. In: *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Ed. by Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi. Vol. 55. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, 123:1–123:14.