k-Universality of Regular Languages

Duncan Adamson
Leverhulme Centre for Functional Material Design, University of Liverpool, UK

Pamela Fleischmann
Department of Computer Science, Kiel University, Germany

Annika Huch
Department of Computer Science, Kiel University, Germany

Tore Koß
Department of Computer Science, University of Göttingen, Germany

Florin Manea
Department of Computer Science, University of Göttingen, Germany

Dirk Nowotka
Department of Computer Science, Kiel University, Germany

Abstract

A subsequence of a word w is a word u such that $u = w[i_1]w[i_2]\dots w[i_k]$, 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 study the intersection between the set of k-subsequence universal words over some alphabet Σ and regular languages over Σ. We call a regular language L k-$\exists$-*subsequence universal* if there exists a k-subsequence universal word in L, and k-$\forall$-*subsequence universal* if every word of L is k-subsequence universal. We give algorithms solving the problems of deciding if a given regular language, represented by a finite automaton recognising it, is k-$\exists$-*subsequence universal* and, respectively, if it is k-$\forall$-*subsequence universal*, for a given k. The algorithms are FPT w.r.t. the size of the input alphabet, and their run-time does not depend on k; they run in polynomial time in the number n of states of the input automaton when the size of the input alphabet is $O(\log n)$. Moreover, we show that the problem of deciding if a given regular language is k-$\exists$-*subsequence universal* is NP-complete, when the language is over a large alphabet. Further, we provide algorithms for counting the number of k-subsequence universal words (paths) accepted by a given deterministic (respectively, nondeterministic) finite automaton, and ranking an input word (path) within the set of k-subsequence universal words accepted by a given finite automaton.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases String Algorithms, Regular Languages, Finite Automata, Subsequences

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2023.4

Related Version *Full Version*: `http://arxiv.org/abs/2311.10658`

Funding *Duncan Adamson*: Leverhulme Trust via the Leverhulme Research Centre for Functional Material Design
Tore Koß: partly supported by the German Research Foundation (DFG), via the project number 389613931 (research grant)
Florin Manea: partly supported by the German Research Foundation (DFG), via the project number 466789228 (Heisenberg grant)

1 Introduction

Words and subsequences are two fundamental combinatorial objects. Informally, a subsequence of a word w is a word u that can be obtained by deleting some of w's letters while preserving the order of the rest. For instance, `taunt` and `salty` are sub-

sequences of `automatauniversality`, while `trauma` is not since these letters do not occur in the correct order. Subsequences are a heavily studied object within computer science [6, 9, 13, 22, 31, 35, 38, 40, 45, 49, 50, 51] and beyond, with applications in a wide number of fields including bioinformatics [23, 47], database theory [5, 17, 32, 33], and modelling concurrency [46]. A survey of combinatorial pattern matching algorithms for subsequences has been provided by Kosche et al. [36], highlighting a series of recent results for problems on finding subsequences in words as well as their applications and connections to other areas of computer science, to which we refer the reader for further details and references.

This paper considers *k-subsequence universal* words. A word w is k-subsequence universal over an alphabet $\Sigma = \{1, \ldots, \sigma\}$ if w contains every word of length k over Σ as a subsequence. These words were first defined by Karandikar and Schnoebelen [28, 44] as *k-rich words*, however more recent work has used the term *k-subsequence* (or *scattered factor*) *universality* [2, 4, 6, 9, 12, 14, 35, 45], which we use here. The study of these words follows from the seminal work by Simon [48] where a congruence – nowadays known as *Simon's congruence* – is introduced. Two words w, v are k-congruent, denoted $w \sim_k v$, if w and v share the same set of subsequences up to length k. As such, k-subsequence universal words are those which are k-congruent to the word $(1 \cdots \sigma)^k$.

Simon's congruence relation is well studied [11, 13, 14, 49, 50, 51], with recent asymptotically optimal algorithms for testing if two words are k-congruent [6] and for computing the largest k for which two words are k-congruent [18], as well as for pattern matching under Simon's congruence [31]. Indeed, besides the usage of k-subsequence universal words in [28, 44] in a context related to the study of the height of piecewise testable languages and the logic of subsequence, the idea of universality itself is quite important in formal languages, automata theory, but also in combinatorics. In this context, the universality problem [25] is whether a given language L (over an alphabet Σ, given as an automaton) is equal to Σ^*. This problem for various classes of languages and language accepting/generating formalisms is studied in, e.g., [41, 37, 19] and the references therein. The universality problem was considered for contiguous factors (substrings) of words [39, 10] and partial words [7, 21], as well. In that context, one analyses, the (partial) words w over an alphabet Σ which have exactly one occurrence of each string of length ℓ over Σ as a substring. De Bruijn sequences [10] fulfil this property and have many applications in computer science or combinatorics, see [7, 21] and the references therein. While it is a perfectly valid problem to investigate (partial) words where each substring occurs exactly once, in the case of subsequence universality this is a trivial restriction, as in each long-enough word there will be subsequences occurring more than once [6].

Coming closer to the topic of this work, we recall the works by Barker et al. [6], Day et al. [9], and Schnoebelen and Veron [45], which directly address k-subsequence universal words. In [6], the authors show that it is possible to determine in linear time (1) whether a word is k-subsequence universal and (2) the shortest k-subsequence universal prefix of a given word. Additionally, they show that the minimal set of factors $w_1, w_2, \ldots, w_\ell$ of a word w, such that $w_1 w_2 \cdots w_\ell$ is k-subsequence universal and the exponent i such that w^i is k-subsequence universal can be determined efficiently. Further, [9] proposes a set of algorithmic results for computing the minimum number of edit operations (insertion, deletions, substitutions) to apply to a word w in order to make it k-subsequence universal; in general, their algorithms run in $O(|w|k)$ time, for $k \leq |w|$ (the problems are trivial, otherwise). Interestingly, the problems approached in that paper can be seen as determining the minimum number Δ such that the (finite) language containing the words found at edit distance at most Δ from w contains a k-universal word. Finally, [45] presents an algorithm computing the largest k for which a word, given in a compressed form, is k-universal.

Another paper which is also strongly related to our work is [29] (as well as its follow-up [30]). In this paper, the authors investigate the language of words which are k-congruent to u, for a given word u, called the k-closure of u. They show that this language is regular, and effectively construct a finite automaton recognising it (of size exponential in σ, the size of the input alphabet). Now, testing whether another given finite (or regular) language contains a word which is k-congruent to u can be reduced to deciding the emptiness of the intersection between the k-closure of u and the given language.

Our work builds on [9, 29], as well as on the works whose main object is the downward closure of languages (see [51] and the references therein), and investigates the problem of efficiently detecting k-subsequence universal words within regular languages. In other words, we are interested in identifying the words of a given regular language L which are k-subsequence universal, i.e., in the k-closure of the target word $(1 \cdots \sigma)^k$. There is, however, a fundamental difference between the problems considered here and those of [29, 30]. The input of the problems studied here is a regular language L and a number k (given in binary representation, similarly to [9]), and we want to detect the words of L which are k-universal, without having to explicitly write those words or the target word $(1 \cdots \sigma)^k$ (whose length is at least $k\sigma$, so exponential in the size of the binary representation of k, our input). On the other hand, in the setting of [29, 30] we are explicitly given a target word w for which we construct the finite automaton recognising its k-closure; applying this approach to our setting would lead to dealing explicitly with the target word $w = (1 \cdots \sigma)^k$, so we would directly have an exponential blow-up both at this step and when the automaton is constructed. Hence, the problem discussed in this paper extends significantly the one of [9] by looking at the intersection between the language of k-subsequence universal words and arbitrary regular languages, rather than a very particular class of finite languages. Moreover, it addresses a highly-relevant particular case of the theory presented in [29, 30], by considering the class of k-subsequence universal words, with the interesting property that this case can be succinctly specified, and with the hope that more direct and efficient algorithms can be obtained in this setting, without having to go through the general framework.

Going more into the details of our approach, we start our investigation by defining two notions for k-subsequence universality of regular languages. A regular language L is *existence k-subsequence universal* (k-$\exists$-subsequence universal) if there exists at least one k-subsequence universal word in L, and it is *universal k-subsequence universal* (k-$\forall$-subsequence universal) if every word of L is k-subsequence universal. We assume that regular languages are given by finite automata which recognises them, so, canonically, we will call an automaton k-$\exists$-subsequence universal (respectively, k-$\forall$-subsequence universal) if the language accepted by it is k-$\exists$-subsequence universal (respectively, k-$\forall$-subsequence universal).

Alongside the categorisation problems (given an automaton, decide whether the language it recognises is k-$\exists$-subsequence universal or k-$\forall$-subsequence universal), we consider the problems of *counting* and *ranking* the number of ℓ-length k-subsequence universal words accepted by a given finite automaton $\mathcal{A}$. The counting problem asks for the total number of ℓ-length k-subsequence universal words accepted by $\mathcal{A}$. The ranking problem takes a word w as input and asks for the number of k-subsequence universal words accepted by $\mathcal{A}$ that are lexicographically smaller than w. Both of these problems have been heavily studied for other classes of words, including cyclic words [1, 2, 3, 16, 20, 34, 43] and Gray codes [16, 34, 42].

Our Contributions. In Section 2 we introduce the novel notions of k-$\exists$-subsequence universality or k-$\forall$-subsequence universality for regular languages and finite automata.

In Section 3, we give algorithms solving the problems of deciding if the language accepted

by a given finite automaton with n states is k-$\exists$-*subsequence universal* and, respectively, if it is k-$\forall$-*subsequence universal*, for a given k. These algorithms are fixed parameter tractable with respect to σ, the size of the input alphabet. If we have additionally $\sigma \in O(\log n)$, they run in polynomial time, and their run-time does not depend at all on k. Note that one could easily devise solutions for both these problems using the framework of [29, 30], but their complexity would have been exponential both in $\log k$ (the size of the representation of k in our input) and in σ. Moreover, we show that, if no bound is placed on the size of the input alphabet, the problem of deciding if a given regular language is k-$\exists$-*subsequence universal* is NP-complete. The NP-hardness of this problem follows from [29]; it is worth noting that, on the one hand, the problem is hard even if $k = 1$, but also, on the other hand, that the hardness proof is indeed based on the fact that the input alphabet is large (equal to the number of states in the input automaton). Showing that this problem is in NP, as well as our algorithms, requires a series of combinatorial insights on the structure of k-universal words accepted by finite automata.

Further, in Section 4, building on the aforementioned understanding of the combinatorial properties of k-universal words accepted by finite automata, we provide algorithms for counting the number of k-subsequence universal words (respectively, paths) accepted by a given deterministic (respectively, non-deterministic) finite automaton, and ranking an input word within the set of k-subsequence universal words (paths) accepted by a given deterministic (respectively, non-deterministic) finite automaton. Again, this approach extends non-trivially the approach from [2], where problems related to counting and ranking subsequence universal words (unrestricted by any regular membership constraint) were approached for the first time.

2 Preliminaries

Let $\mathbb{N} = \{1, 2, \ldots\}$ denote the natural numbers and set $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ as well as $[n] = \{1, \ldots, n\}$ and $[i, n] = \{i, i+1, \ldots, n\}$ for all $i, n \in \mathbb{N}_0$ with $i \leq n$.

An *alphabet* $\Sigma = \{1, 2, \ldots, \sigma\}$ is a finite set of symbols, called *letters* (w.l.o.g., we can assume that the letters are integers). A *word* w is a finite sequence of letters from a given alphabet and its length $|w|$ is the number of w's letters. For $i \in [|w|]$ let $w[i]$ denote w's i^{th} letter. The set of all finite words (aka strings) over the alphabet Σ, denoted by Σ^*, is the free monoid generated by Σ with concatenation as operation and the neutral element is the empty word ε, i.e., the word of length 0. Let Σ^n denote all words in Σ^* exactly of length $n \in \mathbb{N}_0$ and $\Sigma^{\leq n}$ the set of all words of Σ^* up to length $n \in \mathbb{N}_0$. Set $\text{alph}(w) = \{\mathtt{a} \in \Sigma \mid \exists i \in [|w|] : w[i] = \mathtt{a}\}$ as w's alphabet. For $u, w \in \Sigma^*$, u is called a *factor* of w, if $w = xuy$ for some words $x, y \in \Sigma^*$. If $x = \varepsilon$ (resp., $y = \varepsilon$) then u is called a *prefix* (resp., *suffix*) of w. For $1 \leq i \leq j \leq |w|$ define the factor from w's i^{th} letter to the j^{th} letter by $w[i, j] = w[i] \cdots w[j]$. Futher, given a pair of indices $i < j$, $w[j, i] = \varepsilon$. Let $<$ be an order relation on Σ (e.g., the natural order on integers). We extend this order relation to the lexicrographical order on Σ^* in the following way: the word u is lexicographically smaller than the word w ($u < w$) iff either u is a prefix of w or there exists $x, y_1, y_2 \in \Sigma^*$ and $\mathtt{a}, \mathtt{b} \in \Sigma$ with $u = x\mathtt{a}y_1$, $w = x\mathtt{b}y_2$ and $\mathtt{a} < \mathtt{b}$.

As we are interested in investigating the k-subsequence universality of regular languages, we firstly introduce the basic concepts related to subsequences. We then present the definitions for the transformation of these notions to the domain of regular languages and finite automata.

▶ **Definition 1.** *Let $w \in \Sigma^*$ and $n \in \mathbb{N}_0$. A word $u \in \Sigma^*$ is called* subsequence *of w ($u \in \text{SubSeq}(w)$) if there exist $v_1, \ldots, v_{n+1} \in \Sigma^*$ such that $w = v_1 u[1] v_2 u[2] \cdots v_n u[n] v_{n+1}$.*

Set $\mathrm{SubSeq}_k(w) = \{u \in \mathrm{SubSeq}(w) \mid |u| = k\}$.

▶ **Example 2.** Subsequences of `automatauniversality` are `auto`, `tomata`, `salty`, `mate`, and `atom` while `star` and `alien` are not because their letters do not occur in the correct order.

In [6], the authors investigated words which have, for a given $k \in \mathbb{N}_0$, all words from Σ^k as subsequence, namely k-subsequence universal words. Note that this notion is similar to the one of richness introduced and investigated in [27, 28]. We stick here to the notion of k-subsequence universality since our focus are regular languages and thus the well-known notion of the universality of automata and formal languages, i.e., $L(\mathcal{A}) = \Sigma^*$ for a given finite automaton $\mathcal{A}$, is close to the one of subsequence universality of words.

▶ **Definition 3.** *A word $w \in \Sigma^*$ is called* k-subsequence universal *(w.r.t. Σ), for $k \in \mathbb{N}_0$, if $\mathrm{SubSeq}_k(w) = \Sigma^k$. If the context is clear we briefly call w k-universal. The universality-index $\iota(w)$ is the largest k such that w is k-universal.*

We denote the set of k-universal words in a given set $\mathcal{M} \subseteq \Sigma^*$ by $\mathrm{Univ}_{\mathcal{M},k}$. Thus, the set of all k-universal words over a given alphabet Σ is denoted $\mathrm{Univ}_{\Sigma^*,k}$.

▶ **Example 4.** Consider the word $w = \mathtt{baaababb} \in \{\mathtt{a},\mathtt{b}\}^*$. We have $|\mathrm{SubSeq}_3(\mathtt{baaababb})| = |\{\mathtt{aaa},\mathtt{aab},\mathtt{aba},\mathtt{abb},\mathtt{baa},\mathtt{bab},\mathtt{bba},\mathtt{bbb}\}| = 8 = 2^3$. Thus, $\mathtt{baaababb} \in \mathrm{Univ}_{\{\mathtt{a},\mathtt{b}\}^*,3}$. Since $\mathtt{abba} \notin \mathrm{SubSeq}_4(\mathtt{baaababb})$, it follows that $\mathtt{baaababb} \notin \mathrm{Univ}_{\{\mathtt{a},\mathtt{b}\}^*,4}$ and $\iota(\mathtt{baaababb}) = 3$.

Further, we recall the *arch factorisation* by Hébrard [24] which factorises words uniquely.

▶ **Definition 5.** *The* arch factorisation *of $w \in \Sigma^*$ is given by $w = \mathrm{ar}_1(w)\cdots\mathrm{ar}_k(w)\,\mathrm{r}(w)$ for $k \in \mathbb{N}_0$ with $\iota(\mathrm{ar}_i(w)) = 1$ and $\mathrm{ar}_i(w)[|\mathrm{ar}_i(w)|] \notin \mathrm{alph}(\mathrm{ar}_i(w)[1,|\mathrm{ar}_i(w)|-1])$ for all $i \in [k]$, as well as $\mathrm{alph}(\mathrm{r}(w)) \subsetneq \Sigma$. The words $\mathrm{ar}_i(w)$ are the* arches *and $\mathrm{r}(w)$ is the* rest *of w.*

▶ **Example 6.** Continuing Example 4, we have the arch factorisation $w = (\mathtt{ba})\cdot(\mathtt{aab})\cdot(\mathtt{ab})\cdot\mathtt{b}$ where the parantheses denote the three arches and the rest `b`.

A proper subset of the k-universal words is given by all the words with an empty rest introduced in [12] as the set of *perfect k-universal* words.

▶ **Definition 7.** *We call a word $w \in \Sigma^*$* perfect k-universal *if $\iota(w) = k$ and $\mathrm{r}(w) = \varepsilon$. The set of all these words with $\mathrm{alph}(w) = \Sigma$ is denoted by $\mathrm{PUniv}_{\Sigma^*,k}$.*

▶ **Example 8.** The word `baaababb` from Examples 4 and 6 is not perfect 3-universal since it has $\iota(\mathtt{baaababb}) = 3$ but $\mathrm{r}(\mathtt{baaababb}) = \mathtt{b} \neq \varepsilon$. To give a positive example, consider $\mathtt{abcbbaccbbaacb} \in \mathrm{PUniv}_{\{\mathtt{a},\mathtt{b},\mathtt{c}\}^*,4}$ since its arch factorisation is $(\mathtt{abc})\cdot(\mathtt{bbac})\cdot(\mathtt{cbba})\cdot(\mathtt{acb})$.

▶ **Theorem 9** ([6]). *Let $w \in \Sigma^{\geq k}$ with $\mathrm{alph}(w) = \Sigma$. Then we have $\iota(w) = k$ iff w has exactly k arches.*

In the remainder of this section, we introduce the basic definitions we need to define the k-universality of regular languages. For basic notions on finite automata, we refer to [26]. A *non-deterministic finite automaton (NFA)* $\mathcal{A}$ is a tuple $(Q, \Sigma, q_0, \delta, F)$ with the finite set of states Q (of cardinality $n \in \mathbb{N}$), an initial state $q_0 \in Q$, the set of final states $F \subseteq Q$, an input alphabet Σ, and a transition function $\delta : Q \times \Sigma \to 2^Q$, where 2^Q is the powerset of Q. If we have $|\delta(q,\mathtt{a})| = 1$ for all $q \in Q, \mathtt{a} \in \Sigma$, then $\mathcal{A}$ is called *deterministic* (DFA). We call a sequence $\pi = (q_0, \mathtt{a}_1, q_1, \mathtt{a}_2, \ldots, \mathtt{a}_\ell, q_\ell)$ an ℓ-length path in $\mathcal{A}$ iff $q_i \in Q$ for all $i \in [0,\ell]$ and $q_{i+1} \in \delta(q_i, \mathtt{a}_{i+1})$, for all $i \in [0, \ell-1]$. The word $w_\pi = \mathtt{a}_1\cdots\mathtt{a}_\ell$ is the *word (label) associated to* π. A path is simple if it does not contain the same state

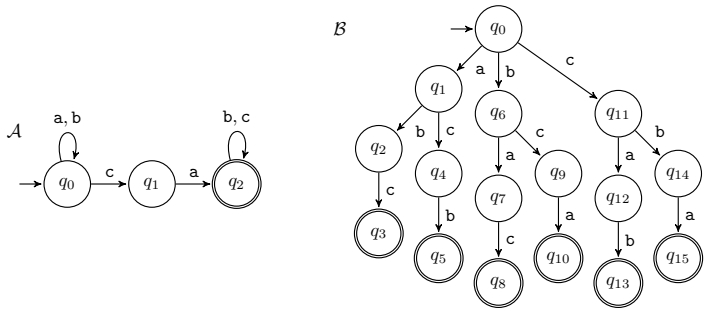

Figure 1 A 2-∃-universal NFA $\mathcal{A}$ and a 1-∀-universal NFA $\mathcal{B}$.

twice. Moreover, a state $q \in Q$ is called accessible (respectively, co-accessible) in $\mathcal{A}$ if there exists a path connecting q_0 to q (respectively, q to a final state). A path is called *accepting* if $q_\ell \in F$ holds. Define the language of $\mathcal{A}$, i.e., the set of words *accepted* by $\mathcal{A}$, by $L(\mathcal{A}) = \{w \in \Sigma^* \mid \exists \text{ accepting path } \pi \text{ in } \mathcal{A} : w = w_\pi\}$. For abbreviation, we set $\mathcal{A}_n = L(\mathcal{A}) \cap \Sigma^n$ and $\mathcal{A}_{\leq n} = L(\mathcal{A}) \cap \Sigma^{\leq n}$. Note that the class of languages accepted by NFAs is equal to the class of languages accepted by DFAs and it is equal to the class of regular languages. Moreover, for every word $w \in L(\mathcal{A})$ there exists exactly one (in the deterministic case) or a set of (in the non-deterministic case) associated path(s). For a word w accepted by a finite automaton, let π_w denote one accepting path labelled with w; in the case when there are multiple such paths, we simply choose one of them. Since we usually are interested in only one path for a word $w \in L(\mathcal{A})$, we refer to it as π_w.

▶ **Definition 10.** *For a given NFA $\mathcal{A}$, the* reverse transition function $\Delta(q, \mathtt{x})$ *returns the set of states in Q with a transition labelled by* $\mathtt{x}$ *to q, i.e.,* $\Delta(q, \mathtt{x}) = \{q' \in Q \mid \delta(q', \mathtt{x}) = q\}$.

Now, we can define the k-subsequence universality of a regular language L. Here, we distinguish whether at least one or all words of L are k-universal w.r.t. Definition 3. Note that we always take the minimal alphabet Σ such that $L \subseteq \Sigma^*$ as a reference when considering the k-universality of words from L.

▶ **Definition 11.** *Let L be a regular language. L is called* existence k-subsequence universal *(k-∃-universal) for some $k \in \mathbb{N}$, if there exists a k-universal word $w \in L$. L is called* universal k-subsequence universal *(k-∀-universal) for some $k \in \mathbb{N}$, if all $w \in L$ are k-universal.*

If a regular language L, accepted by some finite automaton $\mathcal{A}$ (NFA or DFA), is k-∃-universal (respectively, k-∀-universal) then we also say that the automaton $\mathcal{A}$ is k-∃-universal (respectively, k-∀-universal). Further, we also say that a *path π in $\mathcal{A}$ is k-universal* if the label of π, namely w_π, is k-universal. As an example, a 2-∃-universal NFA $\mathcal{A}$, which is not 3-∃-universal, and a 1-∀-universal NFA $\mathcal{B}$ which is not 2-∀-universal are shown in Figure 1 (note that $\mathcal{B}$ recognises exactly every permutation of $\mathtt{abc}$). Based on this definition we define two associated decision problems.

▶ **Problem 12.** The *existence subsequence universality problem for regular languages (k-ESU)* is to decide for a regular language, given by a finite automaton $\mathcal{A}$ recognising it, and $k \in \mathbb{N}$, given in binary representation, whether L is k-∃-universal.

▶ **Problem 13.** The *universal subsequence universality problem for finite automata (k-ASU)* is to decide for a regular language, given by a finite automaton $\mathcal{A}$ recognising it, and $k \in \mathbb{N}$, given in binary representation, whether L is k-∀-universal.

We finish this section by introducing the *rank* of a word in order to enumerate the k-universal words accepted by a given finite automaton.

▶ **Definition 14.** *The* rank *of a word w within the set* $\mathrm{Univ}_{\mathcal{M},k}$ *is the number of words in* $\mathrm{Univ}_{\mathcal{M},k}$ *lexicographically smaller than w, i.e. $rank(w) = |\{v \in \mathrm{Univ}_{\mathcal{M},k} \mid v < w\}|$.*

▶ Remark 15. By Definition 3, the set of all k-universal words accepted by $\mathcal{A}$ is given by $\mathrm{Univ}_{L(\mathcal{A}),k}$, the set of all n-length k-universal words accepted by $\mathcal{A}$ is given by $\mathrm{Univ}_{\mathcal{A}_n,k}$, and the set of all k-universal words of length at most n accepted by $\mathcal{A}$ is given by $\mathrm{Univ}_{\mathcal{A}_{\leq n},k}$.

The computational model we use is the standard unit-cost RAM with logarithmic word size: for an input of size n, each memory word can hold $\log(n)$ bits. Arithmetic and bitwise operations with numbers in $[1, n]$ are, thus, assumed to take $O(1)$ time. Numbers larger than n, with ℓ bits, are represented in $O(\ell/\log n)$ memory words, and working with them takes time proportional to the number of memory words on which they are represented. In all the problems, we assume that we are given a number k, binary encoded, and one finite automaton $\mathcal{A}$, specified as the set of states, set of input letters, set of transitions, and initial and final states. The size of the input is, as such, the size of the binary encoding of k, which is $\lceil \log_2 k \rceil$, plus the size S of the encoding of $\mathcal{A}$ (which is lower bounded by the number of edges in the graph associated to the automaton). So, one memory word can hold $\log(S + \log_2(k))$ bits.

For a more detailed general discussion on this model see, e. g., [8].

Some of our algorithms run in exponential time and use exponential space w.r.t. the size n of the input. Following the literature dealing with such exponential algorithms (see, e.g., [15]), we will use the O^*-notation. By definition, for functions f and g we write $f(n) = O^*(g(n))$ if $f(n) = O(g(n)n^{O(1)})$. In other words, the O^*-notation hides polynomial factors, just as the O-notation hides constants. Using this notation, we can assume that our single-exponential time and single-exponential space algorithms run on a RAM model where arithmetic and bitwise operations with single exponential numbers (w.r.t. the size n of the input) as well as accessing the memory-words (given their address, like in an usual RAM) are assumed to take $O^*(1)$ time. Working with such a computational model allows us to analyse the actual algorithms rather than the various intricacies of the computational model.

In particular, when expressing the complexity of our algorithms, we get functions which are exponential in σ (the size of the alphabet) but polynomial in the number n of states of the input NFA or, for the enumeration algorithms, in the length m of the enumerated strings or in the value of the input number k (the parameter of the considered problems). For clarity of the exposure, although we use the O^*-notation, we will explicitly write the dependency on n, m, and k, and only hide the polynomial dependency on σ.

3 Decision Problems

In this section, we consider the decision problems k-ESU and k-ASU. We begin with a series of combinatorial observations.

▶ Remark 16. Every path acccepted by an NFA $\mathcal{A}$ is k-universal iff every simple accepting path in it is k-universal. In one direction, if $\mathcal{A}$ accepts any path that is not k-universal, then not every path accepted by $\mathcal{A}$ is k-universal. Otherwise, the set of paths induced by every word accepted by $\mathcal{A}$ must contain, as a subsequence, some non-cyclic path. Therefore, if each non-cyclic path is k-universal, then every path is.

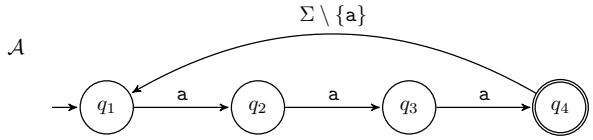

Figure 2 Note that the shortest k-universal word accepted by $\mathcal{A}$ has length $(\sigma-1)kn$.

▶ **Lemma 17.** *Let $q, q' \in Q$ be a pair of states in the NFA $\mathcal{A}$ such that there exists a path π from q to q' where the final transition in π is labelled $\mathtt{x} \in \Sigma$. Then, there exists some path π' of length at most $n = |Q|$ from q to q' where the final transition in the path is labelled $\mathtt{x}$.*

Proof. Let $\hat{q} \in Q$ be the state in π before q', i.e., the state such that $q' \in \delta(\hat{q}, \mathtt{x})$. As there are at most n states in $\mathcal{A}$, given any pair of states $q_1, q_2 \in Q$, if q_2 can be reached from q_1, then there must exist a path from q_1 to q_2 of length at most $n-1$. In particular, there exists a path of length $n-1$ from q to $\hat{q}$. Extending this path by the transition from $\hat{q}$ reading $\mathtt{x}$ to q', we obtain a path π' from q to q' of length at most n. ◀

▶ **Lemma 18.** *For an NFA $\mathcal{A}$ with n states, if there is a k-universal word accepted by $\mathcal{A}$, then there is a k-universal word accepted by $\mathcal{A}$ of length at most $kn\sigma - (n-1)(k-1)$.*

Proof. Let w be a k-universal word accepted by $\mathcal{A}$ and for all $i \in [k]$ let $a_{i,1}, \dots a_{i,\sigma} \in \Sigma$ be the letters of Σ in the order they occur in $\mathrm{ar}_i(w)$, that is $a_{i,1} = \mathrm{ar}_i(w)[1]$ and $a_{i,j} = \mathrm{ar}_i(w)[\ell]$ such that $j - 1 = |\operatorname{alph}(\mathrm{ar}_i(w)[1, \ell-1])| < |\operatorname{alph}(\mathrm{ar}_i(w)[1,\ell])| = j$ for $1 < j \leq \sigma$. Then $\mathrm{ar}_i(w) = a_{i,1}u_{i,1}a_{i,2}\cdots u_{i,\sigma-1}a_{i,\sigma}$ where $u_{i,j} \in \Sigma^*$ is a word such that there is a path $\pi_{u_{i,j}}$ in $\mathcal{A}$ labeled with $u_{i,j}$. For the sake of readability we denote the suffix of w starting after $\mathrm{ar}_k(w)$ by $u_{k+1,1}$ in this proof. By Lemma 17, $u_{i,j}$ is accepted by an automaton with at most n states obtainable from $\mathcal{A}$ by changing the initial (respectively, final) state of $\mathcal{A}$ to the starting (respectively, end) state of $\pi_{u_{i,j}}$. Hence we can find a word $v_{i,j}$ of length at most $n-1$ which is the label of a path starting and ending in the same state as $\pi_{u_{i,j}}$. Let w' be the word obtained from w by replacing every $u_{i,j}$ by $v_{i,j}$. Then w' is a k-universal word accepted by $\mathcal{A}$ of length $|w'| = \sum_{i=1}^{k}\left(\sum_{j=1}^{\sigma}|a_{i,j}| + \sum_{j=1}^{\sigma-1}|v_{i,j}|\right) + |v_{k+1,1}| \leq k\left(\sigma + (\sigma-1)(n-1)\right) + n - 1 = kn\sigma - (k-1)(n-1)$ (cf. Figure 2). ◀

We now move to the main results of this section and give fixed parameter tractable (FPT) algorithms w.r.t. σ. The time complexity of these algorithms is polynomial for $\sigma \in O(\log n)$ and does not depend at all on k.

▶ **Lemma 19.** *For a given NFA $\mathcal{A}$ with n states and $|\Sigma| = \sigma$, we can decide in $O^*(n^3 2^\sigma)$ time whether $\mathcal{A}$ accepts words whose universality index is arbitrarily large. If the answer is negative, then we can compute the largest universality index of a word accepted by $\mathcal{A}$.*

Proof. Assume that $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$, as defined in Section 2. Let us assume w.l.o.g. that $\mathcal{A}$ contains only accessible and co-accessible states.

Our approach is based on a series of observations. We first make these observations, and then provide an algorithm proving the statement of the lemma.

Note first that if there exists a state $q \in Q$ (which is both accessible and co-accessible, by assumption) such that there is a path in $\mathcal{A}$ from q to q labelled with a word which contains all letters of Σ, then we can immediately decide that $\mathcal{A}$ accepts words whose universality index is arbitrarily large. Indeed, to obtain an accepted word (accepting path), whose universality index is at least ℓ, we follow the path from q_0 to q, then follow ℓ times the path which

contains all letters of Σ going from q to q, and finally follow a path connecting q to a final state.

Secondly, assume that there exists no state $q \in Q$ such that there is a path in $\mathcal{A}$ from q to q labelled with a word which contains all letters of Σ. In this setting we note that, for each state $q \in Q$, there exists a unique maximal (w.r.t. inclusion) subset $V_q \subsetneq \Sigma$ such that there exists a path from q to q labelled with a word β_q with $\mathrm{alph}(\beta_q) = V_q$. Indeed, if two such different maximal sets V_q' and V_q'' would exist, witnessed by the words w' and w'', then we can follow the path labelled with $w'w''$, which goes from q to q, and $\mathrm{alph}(w'w'') = V_q' \cup V_q''$ (and this includes both V_q' and V_q'').

Finally, consider an accepting path $\pi = (q_0, \mathtt{a}_1, q_1, \mathtt{a}_2, \ldots, \mathtt{a}_\ell, q_\ell)$ of $\mathcal{A}$. We can rewrite the path π as follows:

- Find the rightmost occurrence of q_0 in π; let us assume that this is the h^{th} state on this path, namely q_h. Replace the subpath connecting the initial occurrence of q_0 to q_h with the loop $(q_0, \mathtt{a}_1 \cdots \mathtt{a}_h)^\circ$ (where we use $\circ$ to emphasise this is a loop). Now the path is rewritten as $((q_0, \mathtt{a}_1 \cdots \mathtt{a}_h)^\circ, \mathtt{a}_{h+1}, q_{h+1}, \ldots, \mathtt{a}_\ell, q_\ell)$. If q_0 appears only once in π, then the path is rewritten as $((q_0, \varepsilon)^\circ, \mathtt{a}_1, q_1, \ldots, \mathtt{a}_\ell, q_\ell)$.
- Now, we repeat the procedure for the path $(q_{h+1}, \mathtt{a}_{h+2}, \ldots, \mathtt{a}_\ell, q_\ell)$ (respectively, if q_0 appeared only once on π, for the path $(q_1, \ldots, \mathtt{a}_\ell, q_\ell)$).

After completing this process, one obtains a *normal-form representation* of the path π as $((q_0', \alpha_1)^\circ, \mathtt{a}_1', (q_1', \alpha_2)^\circ, \mathtt{a}_2', \ldots, \mathtt{a}_r', (q_r', \alpha_{r+1})^\circ)$, with $r \leq n$, as the states $q_0' = q_0, q_1', \ldots, q_r'$ are pairwise distinct; moreover, $\alpha_i \in \Sigma^*$ for all $i \in [r]$.

Let us come back now to the case when there exists no state $q \in Q$ such that there is a path in $\mathcal{A}$ from q to q labelled with a word which contains all letters of Σ. Consider now all accepting paths $((q_0', \alpha_1)^\circ, \mathtt{a}_1', (q_1', \alpha_2)^\circ, \mathtt{a}_2', \ldots, \mathtt{a}_r', (q_r', \alpha_{r+1})^\circ)$ (given in normal-form representation) where the states $q_0', \ldots, q_r'$ and the letters $\mathtt{a}_1', \ldots, \mathtt{a}_r'$ are fixed, and the loops $\alpha_1, \ldots, \alpha_r$ are variable. We are interested in how we can, for $i \in [r+1]$, choose $\alpha_i \in V_{q_{i-1}}^*$ in order to maximise the universality index of the resulting word. We claim that it is enough to take $\alpha_i = \beta_{q_{i-1}}^2$ (where the words β_q were defined above). Indeed, this holds because the arch decomposition of the word labelling the path $((q_0', \alpha_1)^\circ, \mathtt{a}_1', (q_1', \alpha_2)^\circ, \mathtt{a}_2', \ldots, \mathtt{a}_r', (q_r', \alpha_{r+1})^\circ)$ is done greedily from left to right, and we are thus interested in packing as many arches as possible in each of the prefixes $((q_0', \alpha_1)^\circ, \mathtt{a}_1', (q_1', \alpha_2)^\circ, \mathtt{a}_2', \ldots, \mathtt{a}_i', (q_i', \alpha_{i+1})^\circ)$, for all i. So, we begin by noting that the alphabet of the word labelling $((q_0', \alpha_1)^\circ)$ is always included in $V_{q_0'}$, and is indeed equal to $V_{q_0'}$ for, e.g., $\alpha_1 = \beta_{q_0'}^2$. Now, let us consider $i \geq 1$ and the path $((q_0', \alpha_1)^\circ, \mathtt{a}_1', (q_1', \alpha_2)^\circ, \mathtt{a}_2', \ldots, \mathtt{a}_i', (q_i', \alpha_{i+1})^\circ)$. We note that, as we do not have 1-universal loops, the loop $(q_i', \alpha_{i+1})^\circ$ can at most complete the final (and previously incomplete) arch of $((q_0', \alpha_1)^\circ, \mathtt{a}_1', (q_1', \alpha_2)^\circ, \mathtt{a}_2', \ldots, \mathtt{a}_{i-1}', (q_{i-1}', \alpha_i)^\circ)$; after this potential completion, we can only try to add as many new letters as we can in the last arch. So, we will define α_{i+1} as a power of the word $\beta_{q_i'}$, that labels the loop containing q_i' which adds the most new letters to the constructed word, and moreover it is enough to only use $\beta_{q_i'}$ twice in this power (i.e., $\alpha_{i+1} = \beta_{q_i'}^2$). To see that this holds, note that if the rest of the word labelling the path $((q_0', \alpha_1)^\circ, \mathtt{a}_1', (q_1', \alpha_2)^\circ, \mathtt{a}_2', \ldots, \mathtt{a}_{i-1}', (q_{i-1}', \alpha_i)^\circ)$ is completed to a new arch by a repetition $\beta_{q_i'}^f$ for some $f > 2$, then this is done by the first $\beta_{q_i'}$ from this repetition. Then, the suffix $\beta_{q_i'}^{f-1}$ adds as many new letters to the alphabet of the rest of the resulting word as a single occurrence of $\beta_{q_i'}$. This completes the proof of our claim.

In conclusion: among all accepting paths $((q_0', \alpha_1)^\circ, \mathtt{a}_1', (q_1', \alpha_2)^\circ, \mathtt{a}_2', \ldots, \mathtt{a}_r', (q_r', \alpha_{r+1})^\circ)$ (given in normal-form representation) the path $((q_0', \beta_{q_0'}^2)^\circ, \mathtt{a}_1', (q_1', \beta_{q_1'}^2)^\circ, \mathtt{a}_2', \ldots, \mathtt{a}_r', (q_r', \beta_{q_r'}^2)^\circ)$ has the highest universality index.

Based on these observations, we can now state the main idea of our algorithm. We first

preprocess the input automaton to get rid of the states which are not accessible and of the states which are not co-accessible. Then, we check whether there exists a state $q \in Q$ such that there is a path in $\mathcal{A}$ from q to q labelled with a word which contains all letters of Σ; if yes, we simply decide that there exists k-universal accepted words, for any k. If no such state exists, we compute, by dynamic programming, the path $((q'_0, \beta^2_{q'_0})^\circ, \mathtt{a}'_1, (q'_1, \beta^2_{q'_1})^\circ, \mathtt{a}'_2, \ldots, \mathtt{a}'_r, (q'_r, \beta^2_{q'_r})^\circ)$ with the highest universality index, for all states $q_r \in Q$.

In the following we describe this algorithm in more details, by highlighting its main phases.

Preprocessing: We determine the accessible states of $\mathcal{A}$ by running a graph traversal algorithm on $\mathcal{A}$ starting from the state q_0. The not-accessible states are removed. We determine the co-accessible states of $\mathcal{A}$ by running a graph traversal algorithm on the graph of $\mathcal{A}$, with the direction of all edges inverted, starting from the final states. The not-co-accessible states of $\mathcal{A}$ are removed. The complexity of this step is $O(n^3)$ in the worst case. Note that for this step it is enough to consider a simplified form of the graph associated to $\mathcal{A}$, where we only see, for each two states, if there is at least one edge between them or not; as such, the total size of the graph $\mathcal{A}$ is $O(n^2)$). From now on, we assume that all states of $\mathcal{A}$ are both accessible and co-accessible.

Universal Loops: In this step, we determine whether there exists a state q of $\mathcal{A}$ such that there is a path in $\mathcal{A}$ from q to q labelled with a word which contains all letters of Σ. This is done as follows. For each state $q \in Q$, we run the following process. We define L to be a queue, initially containing the element $(q, \emptyset)$, and S be a set which is initially empty. As long as L is not empty, we pop the first element from L, let it be (q', R'), and update S to $S \cup \{(q', R')\}$. Now, for all transitions leaving q of the form $q'' \in \delta(q', \mathtt{a})$, if $(q'', R' \cup \{\mathtt{a}\}) \notin S$, we insert $(q'', R' \cup \{\mathtt{a}\})$ into L. When L is empty, we are done, and we simply check if (q, Σ) is contained in S. If (q, Σ) is contained in S then there exists a path from q to q which is 1-universal. Otherwise, there is no such path. Moreover, we also compute and store the set V_q of maximal cardinality for which (q, V_q) belongs to S.

After running the above process for all states in Q, if we have identified a state q for which there exists a path from q to q which is 1-universal, then we simply conclude that $\mathcal{A}$ accepts words whose universality index is arbitrarily large. Otherwise, we continue with the next phase.

The complexity of this phase is $O^*(n^3 2^\sigma)$, as the numbers of transitions leaving a state q is upper bounded by $n\sigma$.

No Universal Loop: In this case, we will use a $n \times 2^\sigma$ matrix $M[\cdot][\cdot]$ (and an auxiliary matrix M' of the same size). Our algorithm has at most n iterations, and we will maintain the property that, before iteration $\ell+1$ (for $\ell \geq 0$), $M[q'_r][V]$ is the maximum among the number of arches of some path with the normal-form $((q'_0, \beta^2_{q'_0})^\circ, \mathtt{a}'_1, (q'_1, \beta^2_{q'_1})^\circ, \mathtt{a}'_2, \ldots, \mathtt{a}'_r, (q'_r, \beta^2_{q'_r})^\circ)$, with $q'_0 = q_0$ and $r \leq \ell$, such that the rest $\mathrm{r}(u)$ of the word u labelling this path has the alphabet V, or $M[q'_r][V] = -1$ if no such path exists.

To begin with, before the first iteration of the loop, we initialize $M[q][V] = -1$ for all q and V. Then we set $M[q_0][V_{q_0}] = 0$.

In the ℓ^{th} iteration, we copy M into the matrix M'. We then go through all q and V and, if $M[q][V] \neq -1$, we do the following: for all letters $a \in \Sigma$, for all $q' \in \delta(q, a)$, set $V' = V \cup \{a\}$ and $\delta = 0$. If $V' = \Sigma$, we reset $V' = \emptyset$ and set $\delta = 1$. Then, we compute $V'' = V' \cup V_{q'}$. If $V'' = \Sigma$ (note that this can only happen when $V' \neq \emptyset$) we reset $V'' = \emptyset$ and set $\delta = 1$. Finally, we compute $V''' = V'' \cup V_{q'}$ and set $M'[q][V'''] = \max\{M'[q'][V''], M[q][V] + \delta\}$.

We stop this process after n iterations.

Clearly, before the first iteration of our algorithm, the property we intend to maintain holds. Then, in the i^{th} iteration, we try to extend the already constructed path ending in state q, with rest V, by reading first one letter and reaching state q', and then by reading $\beta^2_{q'}$, and compute the universality index of this resulting path. So, the property is maintained in the ℓ^{th} iteration.

The complexity of this phase is $O^*(n^3 2^\sigma)$, if all the above steps are implemented naïvely.

Now, the highest universality index of a word accepted by $\mathcal{A}$ is the largest value $M[q][V]$ for $q \in F$, $V \subseteq \Sigma$.

Conclusion: The algorithm consisting of the three phases above decides whether $\mathcal{A}$ accepts words whose universality index is arbitrarily large, or, if this is not the case, computes the largest universality index of a word accepted by $\mathcal{A}$. Its time complexity is $O^*(n^3 2^\sigma)$. ◀

As a corollary of Lemma 19, we get the following theorem.

▶ **Theorem 20.** *For a given NFA $\mathcal{A}$ with n states and $|\Sigma| = \sigma$ and a natural number $k \in \mathbb{N}$, we can decide k-ESU in $O^*(n^3 2^\sigma)$ time.*

Proof. We first use Lemma 19 to test whether $\mathcal{A}$ accepts words whose universality index is arbitrarily large. If yes, then the answer for the given instance of k-ESU is positive. If $\mathcal{A}$ does not accept words whose universality index is arbitrarily large, we compute the largest universality index ℓ of a word accepted by $\mathcal{A}$. If $\ell \geq k$, then the answer for the given instance of k-ESU is positive. Otherwise, the answer for the given instance of k-ESU is negative. ◀

Next, we approach the k-ASU problem. Once again, we show a preliminary lemma.

▶ **Lemma 21.** *For a given NFA $\mathcal{A}$ with n states and $|\Sigma| = \sigma$, we can compute in $O^*(n^3 2^\sigma)$ time the smallest universality index of a word accepted by $\mathcal{A}$.*

Proof. Clearly, the set of words accepted by $\mathcal{A}$ which have the smallest universality index (among all words accepted by $\mathcal{A}$) includes a word which is the label of a simple path in $\mathcal{A}$. So, to solve the problem stated in this lemma it is enough to consider only words which are the label of simple paths in $\mathcal{A}$. The length of these words is upper bounded by n. This also shows that their universality index is upper bound by n (so, as a consequence, this universality index and the numbers smaller than it fit in one memory word).

The solution of our problem is done by a dynamic programming algorithm. We define two $n \times 2^\sigma$ matrices M, M' where initially $M[q][V] = M'[q][V] = \infty$ for $V \subseteq \Sigma$. Also, we use a set $L = \emptyset$. Further, set $M[q_0][\emptyset] = 0$ and insert $(q_0, \emptyset)$ in L.

We will maintain the property that before the $(r+1)^{\text{th}}$ iteration of our algorithm (for $r \geq 0$) $M[q][S]$ stores the minimal number of arches of the word $w_{r,q}$ of length at most r which connects q_0 and q such that $\text{alph}(\text{r}(w_{r,q})) = S$. This property clearly holds before the first iteration of the algorithm. Now, we perform the following steps

```
for ℓ = 1 to n
  for all (q, S) ∈ L
    for all a ∈ Σ and q' ∈ δ(q, a)
      if S ∪ {a} = Σ
        set M'[q'][∅] = M[q][S] + 1 and insert (q', ∅) in L
      else
        set M'[q'][S ∪ {a}] = M[q][S] and insert (q', S ∪ {a}) in L
for all q ∈ Q and V ⊆ Σ
  set M[q][V] = min{M[q][V], M'[q][V]}
```

In the iteration of the outmost for-loop for $\ell = r$ we actually consider, for each pair (q, S), the word v of length at most $r-1$ which connects q_0 and q, such that v has a minimum number of arches and $\text{alph}(\text{r}(v)) = S$ (the relevant information about this path, i.e., its rest and the number of arches is stored in M), and try to extend this path by one letter $\mathtt{a}$. This leads to another pair $(q', S \cup \{\mathtt{a}\})$, and we simply check if this is the word v' with a minimum number of arches, of length at most r which connects q_0 and q', such that $\text{alph}(\text{r}(v'))$ is either $\emptyset$, when $\Sigma = S \cup \{\mathtt{a}\}$, or $S \cup \{\mathtt{a}\}$ otherwise (see also Lemma 25, which explains how a path is extended). This implies that the property which we wanted to maintain holds before the iteration of the loop for $\ell = r + 1$.

The time complexity of this algorithm is $O^*(n^3 2^\sigma)$.

Now, the smallest universality index of a word accepted by $\mathcal{A}$ is simply the minimum entry $M[q][V]$, for a final state q. ◀

▶ **Theorem 22.** *For a given NFA $\mathcal{A}$ with n states, and input alphabet of size σ, and a natural number k, k-ASU is decidable in $O^*(n^3 2^\sigma)$ time.*

Proof. We first use Lemma 21 to compute ℓ, the smallest universality index of a word accepted by A. If $\ell < k$, then the answer for the given instance of k-ASU is negative. Otherwise, the answer is positive. ◀

After the submission of this conference paper, we realized that the algorithmic problems solved in Lemma 21 and Theorem 22 can actually be solved, in fact, in polynomial time. These results will be included in an extended version of this paper.

We conclude this section by showing that k-ESU is actually NP-complete, and it is NP-hard even for $k = 1$. Clearly, in the light of our previous results from Theorem 20, for this problem to be NP-hard we need to consider the case of an input alphabet $\sigma \in \Omega(\log n)$.

▶ Remark 23. Note that an automaton accepts a word w which is k-universal for $k > n = |Q|$ iff there exists a state q, which is both accessible and co-accessible, and a loop from q to q labelled with a ℓ-universal word, for some $\ell \geq 1$. Indeed, for the left-to-right implication, we look at the states q_i reached by reading the first i arches of the word w, for $i \in [k]$. Clearly, there will be some $i < j$ such that $q_i = q_j$, and the subpath between q_i and q_j on the path labelled with w is ℓ-universal for some $\ell \geq 1$. The other implication is immediate.

▶ **Theorem 24.** *k-ESU is NP-complete.*

Proof. We begin by showing that this problem is in NP.

To solve k-ESU in NP-time, we first check if the automaton $\mathcal{A}$ contains a state q, which is both accessible and co-accessible, and a path from q to q labelled with a x-universal word. First, we traverse (deterministically) the graph of $\mathcal{A}$ from q_0 to determine the accessible states, and also from the final states using the inverted edges to determine the co-accessible states. For each of these states, we non-deterministically guess a 1-universal word of length at most $n\sigma$ and see if this takes us from q to q (by Lemma 18 we have that, if there exists a 1-universal word accepted by an altered version of the automaton $\mathcal{A}$, where q is the unique initial and final state, then there exists one such word of length at most $n\sigma$). If we successfully guessed this word, and $k > n$, then we simply accept the input automaton; otherwise, we reject. If $k \leq n$ and no such altered autonaton exists, then follwoing Lemma 18 any k-subsequence universal word must have length at most $kn\sigma$. Therefore, we can check every word in $\Sigma^{kn\sigma}$ to determine if any word is both k-subsequence universal and accepted by $\mathcal{A}$. If such a word is found we accept the input automaton, otherwise we reject.

The fact that the problem is NP-hard follows from the proof of Theorem 3 from [29]. For completeness (as the respective proof is not given in the accessible version of the paper), we sketch here a reduction from the Hamiltonian Path Problem. The high level idea behind this

reduction is to take a graph $G = (V, E)$ containing n vertices $v_1, \ldots, v_n$ and construct an automaton $\mathcal{A}$ containing $n^2 + 2$ states, with a unique starting state q_0, a unique failure state q_f, and a set of n^2 states labeled $q_{i,j}$ for every $i, j \in [1, n]$. The state $q_{i,j}$ is used to represent visiting the vertex v_j at the i^{th} step of some path in G. With this in mind, a transition exists from $q_{i,j}$ to $q_{\ell,k}$ if and only if $\ell = i + 1$, and (j, k) is an edge in G. In order to map the paths accepted by this automaton directly to the paths in G, each transition is labelled by the index k corresponding to the end state of the edge, i.e., the transition between $q_{i,j}$ and $q_{i+1,k}$ is labelled by k. With this construction, the path in $\mathcal{A}$ labelled by the word w corresponds directly to some path of length $|w|$ in G.

As a Hamiltonian path must have length exactly n, for every $j \in [1, n]$ the state $q_{n,j}$ is marked as an accepting state, and every transition from $q_{n,j}$ leads to the failure state q_f. From this construction, any 1-universal word must correspond exactly to some permutation of the alphabet $[n]$, representing a Hamiltonian path in G. In the other direction if no such word exists, then there does not exist any such path. ◀

As far as k-ASU is concerned, from Lemma 16, and the other results presented here, we can only infer that this problem is in coNP. However, as mentioned above, more recent results show that it can actually be solved in polynomial time.

4 Counting and Ranking

In this section we discuss the problems of counting and ranking efficiently k-universal words from a regular language, given as a DFA, or k-universal paths in the case when the respective language is given as an NFA. Note, that there is a one-to-one correspondence between paths and words in the case of DFAs, so, for the sake of simplicity, from now on we will simply talk about counting and ranking paths accepted by the finite automata we are given as input.

Here, we define the problem of *counting* and *ranking* problems, for a given automaton $\mathcal{A}$, and universality index k. The counting problem is defined as the problem of determining the number of k-universal words accepted by $\mathcal{A}$, equivalent to determining the size of $\mathrm{Univ}_{L(\mathcal{A}),k}$. For a given length $m \in \mathbb{N}$, the problem of counting the number of k-universal words of length exactly (respectively, at most) m is the problem of determining the size of $\mathrm{Univ}_{\mathcal{A}_m,k}$ (respectively $\mathrm{Univ}_{\mathcal{A}_{\leq m},k}$. The *rank* of a word $w \in \mathcal{A}_n$ is the number of k-universal words accepted by $\mathcal{A}$ that are smaller than w, i.e. the size of $\{v < w \mid v \in \mathrm{Univ}_{L(\mathcal{A}),k}\}$. For a given length m, the *rank* of w within the set of k-universal words of length exactly (respectively, at most) m is the problem of determining the size of the set $|\{v < w \mid v \in \mathrm{Univ}_{\mathcal{A}_m,k}\}|$ (respectively, $|\{v < w \mid v \in \mathrm{Univ}_{\mathcal{A}_{\leq m},k}\}|$).

Recall that we are operating on a DFA or NFA $\mathcal{A}$ with n states and an alphabet of size $\sigma > 1$ (the case $\sigma = 1$ is trivial). We are also given as input the natural numbers k and m in binary representation: we are interested in counting (ranking) the k-universal words of length m contained in $L(\mathcal{A})$. It is important to know that these problems are only interesting for $k \leq m/\sigma$; otherwise, there are no m-length k-universal words. However, an additional difficulty related to this problem, compared to the case of the decision problems discussed in Section 3, is that we need to do arithmetics with large numbers. In general, if M is an upper bound on the value of the numbers that we need to process and ω is the size of the memory word of our model, then each arithmetic operation requires at most $O(\frac{\log M}{\omega})$ time to complete. In the cases we approach here, $M \leq (n\sigma)^{m+1}$ (a crude upper bound on the total number of paths of length at most m in $\mathcal{A}$), so each arithmetic operation requires at most $O(\frac{(m+1)\log(n\sigma)}{\omega})$, that is $O^*(m)$ time.

This section is laid out as follows. First, we consider the problems of counting the number of k-universal accepting paths of the finite automaton $\mathcal{A}$. For a given $m \in \mathbb{N}$ this is split into two cases, counting the number of k-universal accepting paths of $\mathcal{A}$ of length exactly m, and counting the number of k-universal accepting paths of $\mathcal{A}$ of length at most m. We show that both can be computed in $O^*(m^2n^2k2^\sigma)$ time. Additionally, we show that the number of k-universal accepting paths of $\mathcal{A}$ can be computed in $O^*(n^4k^22^\sigma)$ time.

We extend our counting results to the ranking setting, showing that a path π (respectively, word w) can be ranked within the set of k-universal paths (respectively, words) of length exactly m accepted by the NFA (respectively, DFA) $\mathcal{A}$ in $O^*(m^2n^2k2^\sigma)$ time, of length at most m in $O^*(m^2n^2k2^\sigma)$ time, and of any length in $O^*(n^4k^22^\sigma)$ time.

The main tool used in this section is the $n \times (m+1) \times k \times 2^\sigma$ size table T, which we refer to as the *path table of length* m for a given $m \in \mathbb{N}_0$. Each entry in the table T is indexed by a state $q \in Q$, a length $\ell \in [0, m]$, the number of arches $c \in [0, k-1]$, and a subset of symbols $\mathcal{R} \subset \Sigma$. The entry $T[q, \ell, c, \mathcal{R}]$ contains the number of ℓ-length paths starting at the state q_0 and ending in the state q such that the words induced by the paths contain each c arches, and the alphabet of the rest is $\mathcal{R}$. Thus, in the context of the arch factorisation, we are interested in all words belonging to paths in $\mathcal{A}$ which are *not yet* k-universal. Formally, we have $T[q, \ell, c, \mathcal{R}] = \sum_{w \in \Sigma^\ell, \mathrm{alph}(\mathrm{r}(w)) = \mathcal{R}, \iota(w) = c} |\mathcal{P}(w, q)|$, where $\mathcal{P}(w, q)$ is the set of paths from q_0 to q in $\mathcal{A}$ labelled by w. Note, that for DFAs, we have $|\mathcal{P}(w, q)| = 1$. The words which have at least k arches in their arch factorisation are captured in the auxiliary $n \times m+1$ size table $U[q, \ell]$, which we refer to as the *universal words table*. Each entry in U is indexed by a state $q \in Q$ and length $\ell \in [0, m]$ with the entry $U[q, \ell]$ containing the number of ℓ-length paths ending at q which are k-universal, i.e., $\iota(w) \geq k$.

The remainder of this section provides the combinatorial and technical tools needed to construct the tables T and U. Theorems 29 and 34, and Corollary 31 summarise the main complexity results of this section.

▶ **Lemma 25.** *Let π be an $(\ell-1)$-length path in the NFA $\mathcal{A}$ ending at q and corresponding to the word w_π, such that $\iota(w_\pi) = c$ and $\mathrm{alph}(\mathrm{r}(w_\pi)) = \mathcal{R}$. Then the word $w_{\pi'}$ corresponding to the path π' formed by following a transition labelled $\mathtt{x} \in \Sigma$ from q either:*

- *has an empty rest ($\mathrm{r}(w_{\pi'}) = \varepsilon$) if $\mathcal{R} \cup \{\mathtt{x}\} = \Sigma$ and hence $\iota(w_{\pi'}) = c+1$, or*
- *has a rest equal to $\mathcal{R} \cup \{\mathtt{x}\}$ ($\mathrm{alph}(\mathrm{r}(w_{\pi'})) = \mathcal{R} \cup \{\mathtt{x}\}$) if $\mathcal{R} \cup \{\mathtt{x}\} \subsetneq \Sigma$ and hence $\iota(w_{\pi'}) = c$.*

Proof. In the first case, if $\mathrm{alph}(\mathrm{r}(w_\pi)) = \Sigma \setminus \{\mathtt{x}\}$ then $\mathrm{r}(w_\pi)\mathtt{x}$ is an arch, as it contains every symbol from Σ at least once, and hence the arch factorisation of $w_{\pi'}$ contains $c+1$ arches, and an empty rest, i.e. $\iota(w_{\pi'}) = c+1$ and $\mathrm{r}(w_{\pi'}) = \varepsilon$. Otherwise, $\mathrm{r}(w_\pi)\mathtt{x}$ contains the set of letters $\mathcal{R} \cup \{\mathtt{x}\}$, and does not complete a new arch. Hence the arch factorisation of $w_{\pi'}$ contains c arches and the rest contains the letters $\mathcal{R} \cup \{\mathtt{x}\}$, i.e., $\iota(w_{\pi'}) = c$ and $\mathrm{alph}(\mathrm{r}(w_{\pi'})) = \mathrm{alph}(w_\pi) \cup \{\mathtt{x}\}$. ◀

Lemma 25 provides the outline of the dynamic programming approach used to compute the table T. Starting with the 0-length path corresponding to the empty word ε, the value of $T[q, \ell, c, \mathcal{R}]$ is computed from the values of $T[q', \ell-1, c', \mathcal{R}']$, allowing an efficient computation of the table. Corollary 26 rewrites this in terms of computing the number of paths of length ℓ ending at state q corresponding to words with c arches and the set of symbols $\mathcal{R}$ of w's rest. The proof is analogous to the one of Lemma 25.

▶ **Corollary 26.** *Let π be an ℓ-length path in the NFA $\mathcal{A}$ with $\iota(w_\pi) = c$, and $\mathrm{alph}(\mathrm{r}(w_\pi)) = \mathcal{R}$. Now, we distinguish the cases whether $\mathcal{R}$ is empty:*
$\mathcal{R} = \emptyset$: $\iota(w_\pi[1, \ell-1]) = c-1$ and $\mathrm{alph}(\mathrm{r}(w_\pi[1, \ell-1])) = \Sigma \setminus \{w_\pi[\ell]\}$,

$\mathcal{R} \neq \emptyset$: $\iota(w_\pi[1, \ell-1]) = c$ *and either* $\mathrm{alph}(\mathrm{r}(w_\pi[1, \ell-1])) = \mathrm{alph}(r(w_\pi)) \setminus \{w_\pi[\ell]\}$ *or* $\mathrm{alph}(\mathrm{r}(w_\pi[1, \ell-1])) = \mathrm{alph}(\mathrm{r}(w_\pi))$. *In this case we also have* $\mathcal{R} = \mathrm{alph}(r(w_\pi[1, \ell-1])) \cup \{w_\pi[\ell]\}$.

Now, we formally establish the dynamic programming approach to calculate the value of $T[q, \ell, c, \mathcal{R}]$ from the already computed cells of the table. For better readability, we use $\mathcal{P}[q, \ell, c, \mathcal{R}]$ as the set of all ℓ-length paths from q_0 to q where the associated word has c arches and the alphabet of its rest is $\mathcal{R}$. Notice that we have $|\mathcal{P}[q, \ell, c, \mathcal{R}]| = T[q, \ell, c, \mathcal{R}]$.

▶ **Lemma 27.** *Let $\mathcal{A}$ be an NFA and assume that $T[q_0, 0, 0, \emptyset]$ is given. Notice that given $q \in Q$, the combination $(q', \mathtt{x}) \in Q \times \Sigma$ only contributes to $T[q, \ell, c, \mathcal{R}]$ if we have $q' \in \Delta(q, \mathtt{x})$. Thus, we have for all $\ell \geq 1$*

$$T[q, \ell, c, \mathcal{R}] = \sum_{\substack{\mathtt{x} \in \Sigma, \\ q' \in \Delta(q, \mathtt{x})}} \begin{cases} 0 & \text{if } \mathtt{x} \notin \mathcal{R} \text{ and } \mathcal{R} \neq \emptyset, \\ 0 & \text{if } \mathcal{R} = \emptyset, c = 0, \\ T[q', \ell-1, c-1, \Sigma \setminus \{\mathtt{x}\}] & \text{if } \mathcal{R} = \emptyset, c > 0, \\ T[q', \ell-1, c, \mathcal{R} \setminus \{\mathtt{x}\}] + T[q', \ell-1, c, \mathcal{R}] & \text{if } \mathcal{R} \neq \emptyset, x \in \mathcal{R}. \end{cases}$$

Proof. From Lemma 25, for every path $\pi \in \mathcal{P}[q, \ell-1, c, \mathcal{R}]$, corresponding to the word w_π, and symbol $\mathtt{x}$, there exists some ℓ-length path π' ending at some state $q' \in \delta(q, x)$ corresponding to the word $w_{\pi'} = w_\pi \mathtt{x}$ such that:

- $\pi' \in \mathcal{P}[q', \ell, c+1, \emptyset]$, if $\mathcal{R} = \Sigma \setminus \{\mathtt{x}\}$, or
- $\pi' \in \mathcal{P}[q', \ell, c, \mathcal{R} \cup \{x\}]$ if $\mathcal{R} \neq \Sigma \setminus \{\mathtt{x}\}$.

In the other direction, from Lemma 26, every path $\pi' \in \mathcal{P}[q, \ell, c, \mathcal{R}]$ corresponding to the word $w_{\pi'}$ must contain an $(\ell-1)$-length prefix corresponding to a word with either:

- $c-1$ arches, and the set of symbols $\Sigma \setminus \{\mathtt{x}\}$ in the rest of the word, if $\mathcal{R} = \emptyset$, or
- c arches, and rest of the word containing the set of symbols $\mathcal{R}$ or $\mathcal{R} \setminus \{\mathtt{x}\}$, if $\mathcal{R} \neq \emptyset$.

Therefore, if $\mathcal{R} = \emptyset$, the size of $\mathcal{P}[q, \ell, c, \mathcal{R}]$ is equal to $\sum_{x \in \Sigma} \sum_{q' \in \Delta(q, \mathtt{x})} T[q', \ell-1, c-1, \Sigma \setminus \{\mathtt{x}\}]$. Similarly, if $\mathcal{R} \neq \emptyset$, the size of $\mathcal{P}[q, \ell, c, \mathcal{R}]$ is equal to $\sum_{x \in \mathcal{R}} \sum_{q' \in \Delta(q, \mathtt{x})} T[q', \ell-1, c, \mathcal{R} \setminus \{x\}] + T[q', \ell-1, c, \mathcal{R}]$.

Note that if $\mathcal{R} \neq \emptyset$, and $\mathtt{x} \notin \mathcal{R}$ for some $\mathtt{x} \in \Sigma$, there is no path in $\mathcal{P}[q, \ell, c, \mathcal{R}]$ where the final transition is labelled $\mathtt{x}$. Similarly , $\mathcal{P}[q, \ell, 0, \emptyset] = \emptyset$ for any $\ell \geq 1$. Otherwise, one of the two above cases must apply, depending on the value of $\mathcal{R}$. This concludes the proof. ◀

Following Lemma 27, the table T can be constructed via dynamic programming. As a base case, note that the only length 0 path in this automaton starting at q_0 is the empty path, which must also end at state q_0 and contains 0 arches and no symbols in the alphabet of the rest. Therefore, $T[q, 0, c, \mathcal{R}]$ is set to 0 for every $q \in Q, \mathcal{R} \subsetneq \Sigma$ and $c \in [0, k-1]$ other than $T[q_0, 0, 0, \emptyset]$, which is set to 1. We now use T to construct U.

▶ **Corollary 28.** *We have* $U[q, \ell] = \sum_{\mathtt{x} \in \Sigma, q' \in \Delta(q, \mathtt{x})} U[q', \ell-1] + T[q', \ell-1, k-1, \Sigma \setminus \{\mathtt{x}\}]$.

Proof. The correctness of this construction follows from Lemma 27. ◀

Note that the total number of k-universal accepting paths of the automaton $\mathcal{A}$ is given by $\sum_{q \in F} U[q, m]$. Using the tables T and U, the total number of k-universal paths and words of length m can be computed in $O^*(m^2 n^2 k 2^\sigma)$.

▶ **Theorem 29.** *The number of k-universal accepting paths of length m of an NFA $\mathcal{A}$, for $k \leq m/\sigma$, can be computed in $O^*(m^2 n^2 k 2^\sigma)$ time. For $k > m/\sigma$, this number is 0.*

Proof. Note that, as explained before, all arithmetic operations require $O(m)$ time in our computational model (we operate with numbers of value at most $(n\sigma)^m$). Using the table U, the number of k-universal words of length m can be counted by computing the sum $\sum_{q\in F} U[q,m]$, which takes $O^*(mn)$ time. To construct the table U, it is nessesary first to construct the table T. Assuming that the value of $T[q',\ell-1,c',\mathcal{R}']$ has been precomputed for every $q'\in Q, c'\in[1,k-1]$ and $\mathcal{R}'\subsetneq\Sigma$, the value of $T[q,\ell,c,\mathcal{R}]$ can be computed in $O^*(mn)$ time. As there are n values of $q\in Q$, 2^σ values of $\mathcal{R}\subset\Sigma$, and k values of $c\in[0,k-1]$, the value of $T[q,\ell,c,\mathcal{R}]$ can be computed for every $q\in Q$, $c\in[0,k-1]$, and $\mathcal{R}\subset\Sigma$ in $O^*(mn^2k2^\sigma)$ time. As there are $m+1$ values of $\ell\in[0,m]$, the total time of constructing T is $O^*(m^2n^2k2^\sigma)$.

Analogously, the value of $U[q,\ell]$ can be computed in $O^*(mn)$ time, assuming the values of $U[q',\ell-1]$ and $T[q',\ell-1,k-1,\Sigma\setminus\{\mathtt{x}\}]$ have be precomputed from every $q'\in Q$ and $\mathtt{x}\in\Sigma$. Hence U can be constructed from the table T in $O^*(m^2n^2)$ time. By extension the number of k-universal paths of length m accepted by $\mathcal{A}$ computed in $O^*(m^2n^2k2^\sigma)$ time. ◀

▶ **Corollary 30.** *The number of k-universal accepting paths of length at most m of an NFA $\mathcal{A}$, for $k\le m/\sigma$, can be computed in $O^*(m^2n^2k2^\sigma)$ time. For $k>m/\sigma$, this number is* 0*.*

Proof. Observe that the number of k-universal paths of length at most ℓ is equal to $\sum_{q\in F}\sum_{\ell\in[1,m]} U[q,\ell]$. As this summation can be computed in $O^*(mn)$ time once U has been constructed, the total complexity follows from the cost of constructing tables T and U. ◀

Since in a DFA each word is associated with exactly one path, the following holds.

▶ **Corollary 31.** *The number of k-universal words of length either exactly or at most m accepted by a DFA $\mathcal{A}$, for $k\le m/\sigma$, can be computed in $O^*(m^2n^2k2^\sigma)$ time. For $k>m/\sigma$, this number is* 0*.*

We may further generalise this to the problem of finding the number of perfect k-universal words and paths by discarding any k-universal paths with a non-empty rest.

▶ **Corollary 32.** *The number of perfect k-universal accepting paths of length either exactly or at most m of an NFA $\mathcal{A}$, for $k\le m/\sigma$, can be computed in $O^*(m^2n^2k2^\sigma)$ time.*

▶ **Corollary 33.** *The number of perfect k-universal words of length either exactly or at most m accepted by a DFA $\mathcal{A}$, for $k\le m/\sigma$, can be computed in $O^*(m^2n^2k2^\sigma)$ time.*

We now generalise these tools to the problem of counting the total number of k-universal paths accepted by a finite automaton $\mathcal{A}$. The primary challenge of counting the total number of such paths comes from determining if there exists either a finite or an infinite number of k-universal paths accepted by $\mathcal{A}$. By the pumping lemma [26], we have that an automaton $\mathcal{A}$ accepts an infinite number of k-universal words (paths) if and only if $\mathcal{A}$ accepts some k-universal word (path) of length at least $n+1$. Clearly, if $k>n$ we have that $\mathcal{A}$ either accepts an infinite number of k-universal words (paths) or none (and this can be tested as in Lemma 19, in time that does not depend on k). Therefore, using the upper bound on the maximum length of the shortest k-universal word accepted by the automaton $\mathcal{A}$ from Lemma 18, combined with Corollary 31, we now count the total number of k-subsequence universal words accepted by $\mathcal{A}$.

▶ **Theorem 34.** *The total number of k-universal words (resp., paths) accepted by a DFA (resp., NFA) $\mathcal{A}$ can be determined in $O^*(n^4k^22^\sigma)$ time (resp., $O^*(n^4k^32^\sigma)$ time), for $k\le n$. For $k>n$, this number is either* 0 *or* ∞*, and can be determined in $O^*(n^32^\sigma)$ time.*

Proof. We only show this for NFAs, as the argument for DFAs is similar (and uses the previous results corresponding to this class of automata).

Note first that if an automaton accepts some k-universal paths w of length at least $n+1$, then it must accept an infinite number of such words, as the path induced by w in $\mathcal{A}$ must visit some state twice and thus contain a cycle. Therefore, if $\mathcal{A}$ accepts only a finite number of k-universal paths, the total number of paths accepted by $\mathcal{A}$ can be computed in $O^*(n^4k2^\sigma)$ time via Theorem 29 and Corollary 30.

Following the same arguments as in Lemma 18, a k-universal path of length of at least $n+1$ exists if and only if there exists some k-universal word of length between $n+1$ and $kn\sigma$, hence it is sufficient to check if $\mathcal{A}$ accepts some word of length between $n+1$ and $kn\sigma$, which can be achieved in $O^*(n^4k^32^\sigma)$ time. If no k-universal word accepted by $\mathcal{A}$ of length at least $n+1$ exists, then total number of k-universal words accepted by $\mathcal{A}$ is determined by counting the number of k-universal words accepted by $\mathcal{A}$ of length at most n. ◀

We now extend our results of counting to the problem of ranking k-universal words. Note, that these results can be generalised to ranking the k-universal accepted paths, but one needs to define an ordering on the transitions from each state in the automaton. To keep the presentation simple, we will therefore only discuss here about DFAs and words. The main idea is to count the number of k-universal words with a prefix strictly smaller than the prefix of w of the same length; again, each arithmetic operation takes $O(m)$ time in our computational model. This section is laid out as follows. First, we show how to compute the rank of w efficiently within the set $\mathrm{Univ}_{\mathcal{A}_{\leq m},k}$ in $O^*(m^2n^2k2^\sigma)$ time. Secondly, we show that the rank of w can be computed within the set $\mathrm{Univ}_{L(\mathcal{A}),k}$ in $O^*(n^4k^32^\sigma)$ time. As noted above, when discussing counting, these problems only make sense for $k \leq m/\sigma$. This follows from the same arguments used to count the total number of k-universal words accepted by $\mathcal{A}$ laid out in Theorem 34.

The primary tool used in this section is a generalisation of the path table: the *fixed prefix path table of length* m is an $n \times (m+1) \times k \times 2^\sigma$ sized table, defined for a set of prefixes $\mathcal{PR}$ and denoted $T(\mathcal{PR})$. Informally, the table $T(\mathcal{PR})$ is used to count the number of paths with some prefix from the given set $\mathcal{PR}$. Thus, $T(\mathcal{PR})[q,\ell,c,\mathcal{R}]$ stores the number of words $w \in \Sigma^\ell$ associated to a path from q_0 to q, with $\iota(w) = c$, $\mathrm{alph}(r(w)) = \mathcal{R}$, and there exists a $p \in \mathcal{PR}$ such that $p = w[1,|p|]$. The table $U(\mathcal{PR})$ is defined analogously to the table U but again with the additional condition as in $T(\mathcal{PR})$. These tables can be constructed directly using the same techniques introduced for T and U, by initially setting $T(\mathcal{PR})[q_0,0,0,\emptyset]$ to 0 and $T(\mathcal{PR})[\delta(q_0,p),|p|,\iota(p),\mathrm{alph(r(}p\mathrm{))}]$ to 1 for every $p \in \mathcal{PR}$. Similarly, in the special case where there exists some $p \in \mathcal{PR}$ such that $\iota(p) \geq k$, then the value of $U[\delta(q_0,p),|p|]$ is set to 1. We assume, without loss of generality, that no prefix in $\mathcal{PR}$ is also the prefix of some other word $p' \in \mathcal{PR}$. The remaining entries are computed as before. In the following results, we use $\mathcal{PR}(w) = \{w[1,i]\mathtt{x} \mid i \in [0,|w|], \mathtt{x} \in [1, w[i+1]-1]\}$. Note, that the following results hold for both counting words accepted by deterministic automata, and accepting paths in non-deterministic automata.

▶ **Corollary 35.** *$T(\mathcal{PR})$, $U(\mathcal{PR})$ are constructible for m-length paths in $O^*(m^2n^2k2^\sigma)$ time.*

▶ **Theorem 36.** *The rank of $w \in \mathrm{Univ}_{\mathcal{A}_m,k}$ can be determined in $O^*(m^2n^2k2^\sigma)$ time.*

Proof. Note that a word v is smaller than w (w.r.t. the lexicographical ordering) if and only if v is a prefix of w or they share a common prefix u and $v[|u|+1] < w[|u|+1]$. Therefore, the number of m-length k-universal words starting with some prefix in $\mathcal{PR}(w)$ is given by either $\sum_{q\in F} U(\mathcal{PR}(w))[q,m]$, if w has length at most m, or $1 + \sum_{q\in F} U(\mathcal{PR}(w))[q,m]$

if the $w[1,m]^{\text{th}}$ state of the path associated with w is an accepting state, $|w| > m$, and $\iota(w[1,m]) = k$. As $T(\mathcal{PR}(w))$ can be computed in $O^*(m^2n^2k2^\sigma)$ time, and the above summation completed in $O^*(mn)$ time, the total time complexity of finding the m-length rank of w is $O^*(m^2n^2k2^\sigma)$. ◀

▶ **Corollary 37.** *The rank of $w \in \mathrm{Univ}_{\mathcal{A}_{\leq m},k}$ can be determined in $O^*(m^2n^2k2^\sigma)$ time.*

Proof. Using the table $T(\mathcal{PR}(w))$ as above, the number of words k-universal words of length at most m smaller than w is given by

$$\sum_{i\in[1,m]}\sum_{q\in F} U(\mathcal{PR}(w))[q,i] + \sum_{i\in[1,m]} \begin{cases} 1, & q_{w[1,i]} \in F, \\ 0, & q_{w[1,i]} \notin F. \end{cases}$$

As the table can be constructed in $O^*(m^2n^2k2^\sigma)$ time, and the summation requires at most $O^*(m^2n)$ time, the total complexity of finding the at-most-m-length rank of a word w in $\mathrm{Univ}_{\mathcal{A}_m,k}$ is $O^*(m^2n^2k2^\sigma)$. ◀

▶ **Corollary 38.** *The rank of $w \in \mathrm{Univ}_{L(\mathcal{A}),k}$ can be determined in $O^*(n^4k^32^\sigma)$ time.*

Proof. Following the same arguments as given in Theorem 34, note that there is an infinite number of words smaller than w if and only if there exists some word of length at least $n+1$ with a prefix in $\mathcal{PR}$. The existence of such a word can be determined from the tables $T(\mathcal{PR}(w))$ and $U(\mathcal{PR}(w))$ for paths of length at most $km\sigma$ in $O^*(k^2n^3)$ time. As the tables $T(\mathcal{PR}(w))$ and $U(\mathcal{PR}(w))$ can be constructed for paths of length at most $kn\sigma$ in $O^*(n^4k^32^\sigma)$ time, the total rank of w within $\mathrm{Univ}_{L(\mathcal{A}),k}$ can be computed in $O^*(n^4k^2)$ time. ◀

5 Conclusions

This paper proposed a series of novel algorithmic results and insights regarding the analysis of the sets which can be expressed as the intersection of regular languages and the language of k-universal words over some alphabet. We have introduced two natural notions of k-universality in regular languages, namely existence k-universal languages and universal k-universal languages, and have proposed algorithms for testing whether a regular language is in one of these two classes. While we have a good understanding of the problem of deciding whether a language is defined by an existence k-universal automaton, the exact complexity of the problem of deciding whether a language is universal k-universal remains open. As well as the introduction of these notions, and the study of some decisions problems related to them, we have provided a toolbox for counting and ranking k-universal paths (respectively, words) accepted by a given NFA (respectively, DFA).

We note that using a divide and conquer approach to count paths (with a certain amount of arches, at most m) of length 2^ℓ by combining paths of length $2^{\ell-1}$ (with less arches), one factor m can be reduced to $\log m$ for counting and ranking words of (or of at most) given length m, at the cost of additional complexity in terms of n, k and σ (as, in that case, one would have to allow the existence of prefixes of such paths which are not part of arches, as well as consider the fact that these paths connect arbitrary pairs of states, and may have different counts of arches). This has been left out of the current version to provide a clearer understanding of our main results, and avoid over-complicating the presentation of the paper.

Duncan Adamson's work was funded by the Leverhulme Trust via the Leverhulme Research Centre for Functional Material Design. Tore Koß's work was supported by the DFG project number 389613931. Florin Manea's work was supported by the DFG Heisenberg-project number 466789228.

References

1 D. Adamson. Ranking binary unlabelled necklaces in polynomial time. In *DCFS*, pages 15–29. Springer, 2022.

2 D. Adamson. Ranking and unranking k-subsequence universal words. In Anna Frid and Robert Mercaş, editors, *WORDS*, pages 47–59. Springer Nature Switzerland, 2023.

3 D. Adamson, A. Deligkas, V. V. Gusev, and I. Potapov. Ranking bracelets in polynomial time. *CPM*, pages 4–17, 2021.

4 D. Adamson, M. Kosche, T. Koß, F. Manea, and S. Siemer. Longest common subsequence with gap constraints. In Anna Frid and Robert Mercaş, editors, *WORDS*, pages 60–76, 2023.

5 A. Artikis, A. Margara, M. Ugarte, S. Vansummeren, and M. Weidlich. Complex event recognition languages: Tutorial. In *DEBS*, pages 7–10, 2017.

6 L. Barker, P. Fleischmann, K. Harwardt, F. Manea, and D. Nowotka. Scattered factor-universality of words. In *DLT*, pages 14–28. Springer, 2020.

7 H. Z. Q. Chen, S. Kitaev, T. Mütze, and B. Y. Sun. On universal partial words. *Electronic Notes in Discrete Mathematics*, 61:231–237, 2017.

8 M. Crochemore, C. Hancart, and T. Lecroq. *Algorithms on strings*. Cambridge University Press, 2007.

9 J.D. Day, P. Fleischmann, M. Kosche, T. Koß, F. Manea, and S. Siemer. The edit distance to k-subsequence universality. In *STACS*, volume 187, pages 25:1–25:19, 2021.

10 N. G. de Bruijn. A combinatorial problem. *Koninklijke Nederlandse Akademie v. Wetenschappen*, 49:758–764, 1946.

11 L. Fleischer and M. Kufleitner. Testing simon's congruence. In *MFCS*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

12 P. Fleischmann, S.B. Germann, and D. Nowotka. Scattered factor universality–the power of the remainder. *preprint arXiv:2104.09063 (published at RuFiDim)*, 2021.

13 P. Fleischmann, L. Haschke, A. Huch, A. Mayrock, and D. Nowotka. Nearly k-universal words-investigating a part of simon's congruence. In *DCFS*, pages 57–71, 2022.

14 P. Fleischmann, J. Höfer, A. Huch, and D. Nowotka. α-β-factorization and the binary case of simon's congruence, 2023. `arXiv:2306.14192`.

15 F. V. Fomin, D. Kratsch, I. Todinca, and Y. Villanger. Exact algorithms for treewidth and minimum fill-in. *SIAM J. Comput.*, 38(3):1058–1079, 2008. `doi:10.1137/050643350`.

16 H. Fredricksen and J. Maiorana. Necklaces of beads in k colors and k-ary de Bruijn sequences. *Discrete Mathematics*, 23(3):207–210, 1978.

17 A. Frochaux and S. Kleest-Meißner. Puzzling over subsequence-query extensions: Disjunction and generalised gaps. In *AMW 2023*, volume 3409 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023.

18 P. Gawrychowski, M. Kosche, T. Koß, F. Manea, and S. Siemer. Efficiently Testing Simon's Congruence. In *STACS*, volume 187, pages 34:1–34:18, 2021.

19 P. Gawrychowski, M. Lange, N. Rampersad, J. O. Shallit, and M. Szykula. Existential length universality. In *Proc. STACS 2020*, volume 154 of *LIPIcs*, pages 16:1–16:14, 2020.

20 E. N. Gilbert and J. Riordan. Symmetry types of periodic sequences. *Illinois Journal of Mathematics*, 5(4):657–665, 1961.

21 B. Goeckner, C. Groothuis, C. Hettle, B. Kell, P. Kirkpatrick, R. Kirsch, and R. W. Solava. Universal partial words over non-binary alphabets. *Theor. Comput. Sci*, 713:56–65, 2018.

22 S. Halfon, P. Schnoebelen, and G. Zetzsche. Decidability, complexity, and expressiveness of first-order logic over the subword ordering. In *LICS*, pages 1–12. IEEE, 2017.

23 R. Han, S. Wang, and X. Gao. Novel algorithms for efficient subsequence searching and mapping in nanopore raw signals towards targeted sequencing. *Bioinformatics*, 36(5):1333–1343, 2020.

24 J.-J. Hebrard. An algorithm for distinguishing efficiently bit-strings by their subsequences. *Theoretical Computer Science*, 82(1):35–49, 1991.

25 M. Holzer and M. Kutrib. Descriptional and computational complexity of finite automata - A survey. *Inf. Comput.*, 209(3):456–470, 2011.

26 J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

27 P. Karandikar, M. Kufleitner, and P. Schnoebelen. On the index of Simon's congruence for piecewise testability. *Inf. Process. Lett.*, 115(4):515–519, 2015.

28 P. Karandikar and P. Schnoebelen. The height of piecewise-testable languages with applications in logical complexity. In *CSL*, 2016.

29 S. Kim, Y. Han, S. Ko, and K. Salomaa. On simon's congruence closure of a string. In *DCFS 2022, Proceedings*, volume 13439 of *Lecture Notes in Computer Science*, pages 127–141. Springer, 2022.

30 S. Kim, Y. Han, S. Ko, and K. Salomaa. On the simon's congruence neighborhood of languages. In *DLT 2023, Proceedings*, volume 13911 of *Lecture Notes in Computer Science*, pages 168–181. Springer, 2023.

31 S. Kim, S. Ko, and Y. Han. Simon's congruence pattern matching. In *ISAAC 2022, Proceedings*, volume 248 of *LIPIcs*, pages 60:1–60:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

32 S. Kleest-Meißner, R. Sattler, M. L. Schmid, N. Schweikardt, and M. Weidlich. Discovering event queries from traces: Laying foundations for subsequence-queries with wildcards and gap-size constraints. In *ICDT 2022, Proceedings*, volume 220 of *LIPIcs*, pages 18:1–18:21, 2022.

33 S. Kleest-Meißner, R. Sattler, M. L. Schmid, N. Schweikardt, and M. Weidlich. Discovering multi-dimensional subsequence queries from traces - from theory to practice. In *BTW 2023, Proceedings*, volume P-331 of *LNI*, pages 511–533, 2023.

34 T. Kociumaka, J. Radoszewski, and W. Rytter. Computing k-th Lyndon word and decoding lexicographically minimal de Bruijn sequence. In *CPM*, pages 202–211. Springer, 2014.

35 M. Kosche, T. Koß, F. Manea, and S. Siemer. Absent subsequences in words. In *RP*, pages 115–131. Springer, 2021.

36 M. Kosche, T. Koß, F. Manea, and S. Siemer. Combinatorial algorithms for subsequence matching: A survey. In Henning Bordihn, Géza Horváth, and György Vaszil, editors, *NCMA*, 2022.

37 M. Krötzsch, T. Masopust, and M. Thomazo. Complexity of universality and related problems for partially ordered NFAs. *Inf. Comput.*, 255:177–192, 2017.

38 M. Lothaire. *Combinatorics on Words*. Cambridge Mathematical Library. Cambridge University Press, 1997.

39 M. H. Martin. A problem in arrangements. *Bull. Amer. Math. Soc.*, 40(12):859–864, December 1934.

40 A. Mateescu, A. Salomaa, and S. Yu. Subword histories and parikh matrices. *Journal of Computer and System Sciences*, 68(1):1–21, 2004.

41 N. Rampersad, J. Shallit, and Z. Xu. The computational complexity of universality problems for prefixes, suffixes, factors, and subwords of regular languages. *Fundam. Inf.*, 116(1-4):223–236, January 2012.

42 C. Savage. A survey of combinatorial gray codes. *SIAM review*, 39(4):605–629, 1997.

43 J. Sawada and A. Williams. Practical algorithms to rank necklaces, Lyndon words, and de Bruijn sequences. *Journal of Discrete Algorithms*, 43:95–110, 2017.

44 P. Schnoebelen and P. Karandikar. The height of piecewise-testable languages and the complexity of the logic of subwords. *Logical Methods in Computer Science*, 15, 2019.

45 P. Schnoebelen and J. Veron. On arch factorization and subword universality for words and compressed words. In *WORDS 2023, Proceedings*, volume 13899 of *Lecture Notes in Computer Science*, pages 274–287. Springer, 2023.

46 A. C. Shaw. Software descriptions with flow expressions. *IEEE Transactions on Software Engineering*, 3:242–254, 1978.

47 R. Shikder, P. Thulasiraman, P. Irani, and P. Hu. An openmp-based tool for finding longest common subsequence in bioinformatics. *BMC research notes*, 12:1–6, 2019.

48 I. Simon. Piecewise testable events. In *Autom. Theor. Form. Lang., 2nd GI Conf.*, volume 33 of *LNCS*, pages 214–222. Springer, 1975.

49 I. Simon. Words distinguished by their subwords. *WORDS*, 27:6–13, 2003.

50 Z. Troniĉek. Common subsequence automaton. In *CIAA*, pages 270–275, 2003.

51 G. Zetzsche. The complexity of downward closure comparisons. In *ICALP*, volume 55, pages 123:1–123:14, 2016.