Representing prefix and border tables: results on enumeration

Julien Clément, Laura Giambruno

Under consideration for publication in Math. Struct. in Comp. Science

Representing prefix and border tables: results on enumeration

Julien Clément, Laura Giambruno

Received February 2014

For some text algorithms, the real measure for the complexity analysis is not the string itself but its structure stored in its prefix table or equivalently border table. In this paper we define the combinatorial class of prefix lists, namely a sequence of integers together with their size, and an injection ψ from the class of prefix tables to the class of prefix lists. We call a *valid* prefix list the image by ψ of a prefix table. In particular we describe algorithms converting a prefix/border table to a prefix list and inverse linear algorithms from computing from a prefix list $L = \psi(P)$ two words respectively in a minimal size alphabet and on a maximal size alphabet with P as prefix table. We then give a new upper bound on the number of prefix tables for strings of length n (on any alphabet) which is of order $(1+\varphi)^n$ (with $\varphi = \frac{1+\sqrt{5}}{2}$ the golden mean) and also present a corresponding lower bound.

Contents

1. Introduction

Prefix tables and border tables are equivalent structures representing the overlap in a word. In particular the prefix table of a string w reports for each position i the length of

the longest substring of w that begins at i and matches a prefix of w, while the border table of the string records for each position the maximal length of prefixes of the string w ending at that position. Indeed two strings have the same border table if and only if they have the same prefix table (Crochemore *et al.* 2007; Bland *et al.* 2013).

These tables are used in algorithms on words to design classical efficient string-matching algorithms (like Morris-Pratt and Knuth-Morris-Pratt algorithms) and are essential for this type of applications (Gusfield 1997; Crochemore *et al.* 2007). We remark that for these classical algorithms the string itself is not considered but rather its structure, meaning that two strings with the same prefix or border table are treated in the same manner. The study of these tables has become topical and is still today an object of interest investigated by various researchers. For instance recent articles (Franek *et al.* 2002; Duval *et al.* 2005; Clément *et al.* 2009; Duval *et al.*) focus on the problem of validating prefix and border tables, that is the problem of checking if an integer array is either the prefix or the border table of at least one string.

In this paper[†]. we are interested in the enumeration of these structures of a given length — note that one can associate an infinite number of words on an infinite alphabet to a given prefix/border table (Crochemore *et al.* 2007). We think that this kind of study is fundamental in order to better understand these tables and string-matching algorithms. We moreover think that that it is an important step to perform average-case studies on algorithms manipulating these structures.

In a previous paper (Moore *et al.*) Moore et al. represented distinct border tables by canonical strings and gave results on generation and enumeration on these string for bounded and unbounded alphabets. Some of these results were reformulated in (Duval *et al.*) using automata-theoretic methods. Note that different words on a binary alphabet have distinct prefix/border tables. This gives us a trivial lower bound of 2^{n-1} (since exchanging the two letters of the alphabet does not change tables).

Here we are interested in giving better estimates on the number p_n of prefix/border tables of words of a given length n, that those known in literature.

We define the combinatorial class of *prefix lists*, where a prefix list $L = [\ell_1, \ldots, \ell_k]$ is a finite sequence of non-negative integers. Then we constructively define an injection ψ from the set of prefix tables to the set of prefix lists. Since the application is only injective we define *valid prefix lists* as prefix lists that are images of prefix tables under ψ. We moreover describe two "inverse" linear algorithms that associate a valid prefix list $L = \psi(P)$ with two words whose prefix table is P, one on a minimal size alphabet and the other on a maximal size alphabet. This result confirms the idea that prefix lists represent a more concise representation for prefix tables.

We then deduce a *new upper bound and a new lower bound on the number p_n of prefix tables* (see Table 1 for the first numerical values) for strings of length n or, equivalently, on the number of border tables of length n.

Let $\varphi = \frac{1}{2}(1 + \sqrt{5}) \approx 1.618$ be the golden mean and let, for any finite set S, $Card(S)$ denote the cardinality of S, then we have:

[†] A preliminary version of this paper appeared in (Clément *et al.* 2013).

Proposition 1.1 (Upper bound). The number of prefix tables p_n is asymptotically bounded from above by the quantity $\frac{1}{2}\left(1+\frac{\sqrt{5}}{5}\right)\ (1+\varphi)^n + o(1)$.

Proposition 1.2 (Lower bound). For any $\varepsilon > 0$ there exists a family of prefix tables $(\mathcal{L}_n)_{n\geq 0}$ such that $\mathrm{Card}(\mathcal{L}_n) = \Omega((1+\varphi-\varepsilon)^n)$.

The paper is organized in the following way. In Section 2 we introduce definitions regarding prefix and border tables and we state the equivalence between prefix and border tables. In Subsection 2.2 we show and analyze results on the enumeration of border tables as written in (Moore *et al.*). We then define the combinatorial class of prefix lists. In Section 3 we establish an injection between prefix tables and prefix lists. In particular, we present algorithms for constructing from either a prefix or a border table a prefix list. Conversely we show two algorithms associating to a prefix list two words having the same prefix table, respectively on a minimal size alphabet and on a maximal size alphabet. In Section 4 we count, by using combinatorial analytical methods, the number of prefix lists of a given size and we deduce the upper bound, as stated in Proposition 1.1. In Section 5, we exhibit a family of languages in bijection with prefix tables. By proving a result on the enumeration of these families and by using combinatorial techniques, we get the proof of Proposition 1.2. Finally we conclude the paper presenting some open problems.

2. Preliminaries

Let A be an ordered alphabet. For each $i > 0$, we denote by $\alpha(i)$ the i-th element of A. A *word* w (also called *string*) of length $|w| = n$ is a finite sequence $w[0]w[1]\dots w[n-1] = w[0\,..\,n-1]$ of letters of A. The language of all words over A is A^*, and A^+ is the set of nonempty words. The prefix (resp. suffix) of length ℓ, $0 \leq \ell \leq n$, of w is the word $u = w[0\,..\,\ell-1]$ (resp. $u = w[n-\ell\,..\,n-1]$)[‡]. A border u of w is a word that is both a prefix and a suffix of w and distinct from w itself. We define $\mathrm{bord}(w)$ as the set of all non-empty borders of w.

2.1. *Prefix and border tables*

Definition 2.1 (Prefix table). The prefix table Pref_w of a word $w \in A^+$ of length n, is the table of size n defined, for $0 \leq i < n$, by

$$\mathrm{Pref}_w[i] = \mathrm{lcp}(w, w[i\,..\,n-1]),$$

where lcp denotes the maximal length of common prefixes of the two words.

Another well-known structure used to represent the correlation structure of a string is the border table of a word.

[‡] With the convention that whenever $\ell = 0$, $u = \varepsilon$.

Definition 2.2 (Border table). The border table Border_w of a word $w \in A^+$ of length n, is the table of size n defined, for $0 \leq i < n$, by

$$\mathrm{Border}_w[i] = \max\{|u| \ \mid \ u \text{ is a border of } w[0\,..\,i]\},$$

Example. Let w be the word `abaababa`. We have the following representations for the prefix and border tables of w (see also Table 2).

i	0	1	2	3	4	5	6	7
$w[i]$	a	b	a	a	b	a	b	a
$\mathrm{Pref}_w[i]$	8	0	1	3	0	3	0	1
$\mathrm{Border}_w[i]$	0	0	1	1	2	3	2	3

These structures (border and prefix tables) are in fact equivalent; actually the following proposition states a fact discussed in (Crochemore *et al.* 2007) and recently deepened in (Bland *et al.* 2013), where linear time conversion algorithms are given. In the following we provide a proof of this equivalence as elements of this proof will be used in the next section.

Proposition 2.3. (Crochemore *et al.* 2007) Two strings have the same border table if and only if they have the same prefix table.

Sketch of proof.

Let w be a word in A^+ of length $|w| = n > 0$. We can relate the border table Border_w to the prefix table Pref_w.
For a position i in w, $0 < i < |w|$, let

$$I(i) = \{j \ \mid \ 0 < j \leq i \text{ and } j + \mathrm{Pref}_w[j] - 1 \geq i\}, \tag{1}$$

and by convention, we pose $I(0) = \emptyset$. The elements in $I(i)$ represent the positions $0 < j \leq i$ for which the longest common prefixes between w and $w[j\,..\,n-1]$ overlap position i in w. We remark that we must take j strictly positive in (1), otherwise, since $\mathrm{Pref}_w[0] = |w|$, we would have $0 \in I(i)$ for all positions $i > 0$. On the previous example we get

i	0	1	2	3	4	5	6	7
$w[i]$	a	b	a	a	b	a	b	a
$I(i)$	$\emptyset$	$\emptyset$	$\{2\}$	$\{3\}$	$\{3\}$	$\{3,5\}$	$\{5\}$	$\{5,7\}$

Then we have

$$\mathrm{Border}_w[i] = \begin{cases} 0 & \text{if } I(i) = \emptyset, \\ i - \min I(i) + 1 & \text{otherwise.} \end{cases} \tag{2}$$

Indeed we remark $I(i) = \emptyset$ if and only if $\mathrm{Border}_w[i] = 0$ and, when non empty, $\min I(i)$ stores the starting position of the longest suffix of $w[0\,..\,i]$ which is also a prefix of w, hence $i - \min I(i) + 1$ is the length of the longest border of $w[0\,..\,i]$.

Conversely, given the border table Border_w of w, we define the prefix table Pref_w in the following way. First we set $\mathrm{Pref}_w[0] = |w|$. Then let $i > 0$ be a position in w and let

$$I'(i) = \{j \ \mid \ i \leq j < |w| \text{ and } w[i\,..\,j] \in \mathrm{bord}(w[0\,..\,j])\}.$$

We have

$$\mathrm{Pref}_w[i] = \begin{cases} 0 & \text{if } I'(i) = \emptyset, \\ \max(I'(i)) - i + 1 & \text{otherwise.} \end{cases} \tag{3}$$

From (2) and (3) it follows that two words have the same border table if and only if they have the same prefix table. Indeed by definition of the set I two words having the same prefix table have also the same border tables. The converse is also true since if two words w and w' of length n admit the same border table, then for $0 \leq i \leq j < n$, $w[i\,..\,j] \in \mathrm{bord}(w[0\,..\,j))$ if and only if $w'[i\,..\,j] \in \mathrm{bord}(w'[0\,..\,j))$. □

Recent literature has focused on the problem of validating prefix and border tables and, in the case of a valid table, providing the canonical word associated with it that is the smallest in lexicographic order (Franek *et al.* 2002; Clément *et al.* 2009).

2.2. *On enumeration: previous work*

In this paper we are interested in the computation of the number p_n of distinct prefix tables of length n. As seen in Proposition 2.3 such a number is also the number of distinct border tables of length n. One can pose the problem of enumerating prefix and border tables on a finite or unbounded/infinite alphabet. In order to study p_n it is of interest to enumerate $p_{n,k}$, the number of prefix tables for words of size n with an alphabet of size k which cannot be obtained using a smaller alphabet.

In Table 1 we give some experimental results and in Table 2 we give the nine distinct prefix/border tables for words of length 4 together with the minimal corresponding word for lexicographical order (named canonical words in the literature (Moore *et al.*)).

Previous work in (Moore *et al.*) focused on counting distinct strings of length n with respect to their prefix/border tables: an upper bound is given in the form

$$b_n = \textstyle\sum_{k=1}^{k^*} \left\{ {n-2^{k-1}+k \atop k} \right\}, \tag{4}$$

where $\left\{ {m \atop j} \right\}$ denotes the Stirling numbers of second kind (the number of partitions of m into j nonempty parts), and $k^* = \lceil \log_2(n+1) \rceil$. The quantity k^* is the minimal number of distinct letters in order to obtain all possible prefix tables of size n.

Numerically it is clear that b_n is far from being a tight approximation of the number p_n of prefix tables of size n. One can indeed prove that $b_n \gg p_n$. We formalise this fact in the following combinatorial lemma.

Lemma 2.4. Let $\alpha_n \to \infty$ with $\alpha_n = O(n^c)$ for some $0 < c < 1$, then one has

$$\left\{ {n \atop \alpha_n} \right\} \sim \frac{(\alpha_n)^n}{\alpha_n!} \sim \sqrt{2\pi}(\alpha_n)^{n-\alpha_n+1/2} e^{\alpha_n}.$$

Proof. The proof of this lemma relies on the fact that mappings from $\{1, \ldots, n\}$ to $\{1, \ldots, \alpha_n\}$ (related to Stirling number of second kind) are, if α_n is small enough, almost always surjective. (The original idea of the proof is due to Cyril Nicaud (Nicaud 2010)). Indeed, let $S(n, m)$ be the set of surjections from $\{1, \ldots, n\}$ to $\{1, \ldots, m\}$. It is well

n	$p_{n,1}$	$p_{n,2}$	$p_{n,3}$	$p_{n,4}$	$p_{n,5}$	$p_{n,6}$	p_n
1	1						1
2	1	1					2
3	1	3					4
4	1	7	1				9
5	1	15	4				20
6	1	31	15				47
7	1	63	46				110
8	1	127	134	1			263
9	1	255	370	4			630
10	1	511	997	16			1525
11	1	1023	2625	52			3701
12	1	2047	6824	162			9034
13	1	4095	17,544	500			22,140
14	1	8191	44,801	1467			54,460
15	1	16,383	113,775	4180			134,339
16	1	32,767	287,928	11,742	1		332,439
17	1	65,535	726,729	32,466	4		824,735
18	1	131,071	1,831,335	88,884	16		2,051,307
19	1	262,144	4,610,078	241,023	52		5,113,298
20	1	524,287	11,599,589	649,022	168		12,773,067
21	1	1,048,575	29,182,347	1,736,614	504		31,968,041
22	1	2,097,151	73,430,919	4,623,344	1486		80,152,901
23	1	4,194,303	184,845,142	12,253,644	4248		201,297,338
24	1	8,388,607	465,567,693	32,356,073	11,983		506,324,357
25	1	16,777,215	1,173,418,456	85,156,997	33,242		1,275,385,911
26	1	33,554,431	2,959,762,252	223,493,213	91,297		3,216,901,194
27	1	67,108,863	7,471,688,677	585,104,586	248,196		8,124,150,323
28	1	134,217,727	18,877,965,663	1,528,508,811	669,799		20,541,362,001
29	1	268,435,455	47,739,117,581	3,985,452,962	1,795,120		51,994,801,119
30	1	536,870,911	120,831,350,575	10,374,418,698	4,784,707		131,747,424,892
31	1	1,073,741,823	306,104,380,017	26,965,612,590	12,689,612		334,156,424,043
32	1	2,147,483,647	776,139,381,391	69,999,199,986	33,513,035	1	848,319,578,061
33	1	4,294,967,295	1,969,623,334,609	181,500,343,408	88,172,789	4	2,155,506,818,106

Table 1. *First values: p_n is the total number of prefix tables for strings of size n, $p_{n,k}$ is the number of prefix tables for strings of size n with an alphabet of size k which cannot be obtained using a smaller alphabet.*

Prefix tables	Border tables	Canonical words
$[4,3,2,1]$	$[0,1,2,3]$	`aaaa`
$[4,2,1,0]$	$[0,1,2,0]$	`aaab`
$[4,1,0,1]$	$[0,1,0,1]$	`aaba`
$[4,1,0,0]$	$[0,1,0,0]$	`aabb`
$[4,0,1,1]$	$[0,0,1,1]$	`abaa`
$[4,0,2,0]$	$[0,0,1,2]$	`abab`
$[4,0,1,0]$	$[0,0,1,0]$	`abac`
$[4,0,0,1]$	$[0,0,0,1]$	`abba`
$[4,0,0,0]$	$[0,0,0,0]$	`abbb`

Table 2. *The nine distinct prefix/border tables for words of length 4 (as counted in Table 1) are listed together with their minimal corresponding words (named canonical words in the literature) for lexicographical order.*

known that

$$\mathrm{Card}(S(n,m)) = m! \left\{ {n \atop m} \right\}, \tag{5}$$

where $\left\{ {n \atop m} \right\}$ denotes the number of partitions of $\{1,\ldots,n\}$ into m blocks: each block of the partition is the preimage of an integer $i \in \{1,\ldots,m\}$, but doing this we have ordered the blocks hence the term $m!$ arises.

The number of surjections from $\{1,\ldots,n\}$ to $\{1,\ldots,m\}$ is trivially bounded from above by the cardinality of the set $M(n,m)$ of mappings from $\{1,\ldots,n\}$ to $\{1,\ldots,m\}$, so we also have

$$\mathrm{Card}(S(n,m)) \leq \mathrm{Card}(M(n,m)) = m^n. \tag{6}$$

Let $S_i(n,m)$ be the set of mappings which are not surjective because i has no preimage. We can write

$$M(n,m) = S(n,m) \cup \big(\cup_{i\in\{1,\ldots,m\}} S_i(n,m) \big).$$

The union of the S_i's sets in this last equation is not disjoint, however it is sufficient to give a lower bound

$$m^n \leq \mathrm{Card}(S(n,m)) + \sum_{i=1}^{m} \mathrm{Card}(S_i(n,m)) = \mathrm{Card}(S(n,m)) + m(m-1)^n,$$

since $\mathrm{Card}(S_i(n,m)) = (m-1)^n$ for any i. Finally we have

$$\mathrm{Card}(S(n,m) \geq m^n - m(m-1)^n = m^n \left(1 - m(1 - \frac{1}{m}) \right). \tag{7}$$

Putting $m = \alpha_n$ for some sequence $\alpha_n \to \infty$ such that $\alpha_n = O(n^c)$ for some constant $c \in]0,1[$, a simple computation proves using (6) and (7) that as n tends to ∞

$$\mathrm{Card}(S(n,\alpha_n)) \sim (\alpha_n)^n.$$

Hence by (5), we have proved that for α_n such that $\alpha_n \to \infty$ and $\alpha_n = O(n^c)$ for a constant $c \in]0,1[$

$$\left\{ {n \atop \alpha_n} \right\} \sim \frac{(\alpha_n)^n}{\alpha_n!}.$$

An application of the usual Stirling formula yields the final result of the lemma. □

The quantity b_n in Equation (4) is at least of order $\left\{ {cn \atop d \log n} \right\}$ for some positive constants c and d (considering just one term of the sum, for instance with $k = k^* - 2$, in (4)). Thus Lemma 2.4 applied to $\left\{ {cn \atop d \log n} \right\}$ suffices to prove that $\log b_n$ is at least of order $n \log\log n$ (posing $N = cn$ and $\alpha_N = d \log(N/c)$ and only considering the term $(\alpha_N)^{N-\alpha_N}$ of the approximation). Hence we have:

Corollary 2.5. We have $\frac{1}{n} \log b_n = \Omega(\log\log n)$.

In Section 4 we improve the bound in (4), yielding the result of Proposition 1.1.

2.3. *Prefix lists*

The information in a prefix table is somewhat redundant since we do not need to use all values in the table to build a corresponding word. For instance, from the prefix table P having the first four entries $8, 0, 1, 3$ we can build the prefix $abaaba = a \cdot b \cdot aba$ of length 6 of a word associated to P. We see that the next entry of the prefix table (which would be 0) can be deduced by the previous ones. Therefore we introduce prefix lists which are more concise than prefix tables and sufficient to reconstruct such a word. We first define the combinatorial class of prefix lists as it follows:

Definition 2.6. We define a prefix list $L = [\ell_1, \ldots, \ell_k]$ as a finite sequence of positive integers together with a *size* defined for a list as $\|L\| = \sum_{i=1}^{k} \|\ell_i\|$, where the size $\|i\|$ is i if $i > 0$ and 1 if $i = 0$.

As will become clear later, this particular size corresponds to the size of a word built from the prefix list.

Let $\mathcal{P}$ denote the set of prefix tables and $\mathcal{L}$ the set of prefix lists. In the following section we define an injection $\psi : \mathcal{P} \longrightarrow \mathcal{L}$ in a constructive manner. We define *valid* prefix lists as:

Definition 2.7. Let L be a prefix list. We say that L is *valid* if $L = \psi(P)$ for a prefix table $P \in \mathcal{P}$.

3. Algorithms : from tables to lists and vice versa

In this section we define an injection from the set $\mathcal{P}$ of prefix tables to the set $\mathcal{L}$ of prefix lists in a constructive manner by defining a quadratic algorithm `PrefixToList` that associates to a prefix table a prefix list. We could of course provide a linear algorithm to perform the same task. However the aim here is to present a simple algorithm and not an efficient one. We also note that such an injection can be reformulated in the context of border tables: in Subsection 3.2 we give a linear algorithm `BorderToList`, equivalent to `PrefixToList`, computing from a border table a prefix list.

In Subsection 3.3 we then describe two "inverse" linear algorithms associating a prefix list $L = \psi(P)$ with two words w whose prefix table is P. We give more precisely two algorithms `ListToMaxWord` and `ListToMinWord` computing respectively the word w on a minimal and on a maximal size alphabet.

Remark 3.1. *(*On complexity). We remark that most algorithms manipulating prefix and border tables work in linear time. In particular given a word there are linear time algorithms for the computation of the associated border and prefix tables. Moreover in (Crochemore *et al.* 2007) and recently in (Bland *et al.* 2013) linear-time conversion algorithms are given and the validation algorithms for both tables (Clément *et al.* 2009; Duval *et al.*) work in linear-time. Thus, quite generally, there would be a trivial way to implement the algorithms in this section by using already known algorithms and auxiliary structures for this purpose.

However we think that it is important to search algorithms directly manipulating

prefix lists. We want to study more deeply these structures and finely understand their properties. Full combinatorial characterizations would also be useful for enumeration.

3.1. *From prefix tables to prefix lists*

We define constructively an injection ψ from $\mathcal{P}$ to $\mathcal{L}$ with the help of the algorithm `PrefixToList` processing the input in a "right-to-left manner". Intuitively, the following algorithm scans the prefix table from right to left, starts with the last position $i = n-1$ and gets from the prefix table the length ℓ of the leftmost longest common proper prefix which overlaps the current position i, or sets $\ell = 0$ if there is no such prefix. This length is inserted at the beginning of the list and the position i is updated to the position immediately before the prefix (if it exists) or just one position before (if it is not the case). The algorithm stops when the first position $i = 0$ is attained.

Algorithm 1: `PrefixToList`$(P = P[0\,..\,n-1])$

```
1  L ← [ ]
2  i ← n − 1
3  while i > 0 do
4  |  I ← {j | 0 < j ≤ i and j + P[j] − 1 ≥ i}
5  |  if I = ∅ then
6  |  |  (ℓ, i) ← (0, i − 1)
7  |  else
8  |  |  (ℓ, i) ← (i − min(I) + 1, min(I) − 1)
9  |  L ← [ℓ] · L /* the integer ℓ is prefixed to the list L   */
10 |
11 return L
```

For each position i in P, the elements in I represent, as in the proof of Proposition 2.3, positions less than or equal to i, such that the longest common proper prefix with w starting at these positions overlap position i.

Definition 3.2. For a given prefix table P in $\mathcal{P}$, we define $\psi(P)$ as the prefix list obtained by executing the algorithm `PrefixToList` on P.

Example 3.3. Let w be the word `abaababa`. We have the following representation for the prefix table of w.

i	0	1	2	3	4	5	6	7
$w[i]$	a	b	a	a	b	a	b	a
$\mathrm{Pref}_w[i]$	8	0	1	3	0	3	0	1

For this table we get the associated prefix list $L = [0, 1, 2, 3]$. In fact, executing the algorithm `PrefixToList` to Pref_w, we start with $i = 7$ and we get that the set of starting indexes of prefixes overlapping i is $I = \{5, 7\}$. Thus $\ell = i - \min(I) + 1 = 3$, the length of the overlapping prefix until i, is appended to $L = [\,]$. Now i is initialised to 4 the position before $\min(I) = 5$. Next we have $I = \{3\}$ and so $\ell = 2$ is prefixed to $L = [3]$ and $i := 2$. Again $I = \{2\}$, $\ell = 1$ and $i := 1$. Now $I = \emptyset$, thus $\ell = 0$, $i := 0$ and the algorithm stops.

Consider now the time and space complexity of the `PrefixToList` algorithm:

Proposition 3.4. The algorithm `PrefixToList` can be implemented in $O(n^2)$ time on an input prefix table of length n without using auxiliary memory.

Proof. For each i the computation of I in line 4 has $O(i) = O(n)$ time complexity. For the main loop we have n instructions in the worst case, that is the case of the prefix table $P = [8, 0, 0, 0, 0, 0, 0, 0]$. Thus the time complexity is in $O(n^2)$. For what concerns space complexity, if $I \neq \emptyset$ then we just need constant space in order to preserve the minimum in I. Thus we do not have to use auxiliary memory. □

Remark 3.5. At first view, it would be more intuitive to define prefix lists with an algorithm visiting the prefix table from left to right. However, the construction of "prefix list" from left to right fails to define an injection from prefix tables to prefix lists (which is our goal for finding an upper bound). For instance, let $P = [8, 0, 1, 3, 0, 3, 0, 1]$ be a valid prefix table, as in Example 3.3, and $P' = [8, 0, 1, 3, 0, 1, 0, 1]$ be a valid prefix table associated with w', then one has

i	0	1	2	3	4	5	6	7
$w'[i]$	a	b	a	a	b	a	c	a
$\mathrm{Pref}_{w'}[i]$	8	0	1	3	0	1	0	1

Since the same list $L = [0, 1, 3, 0, 1]$ is associated with both P and P' then the correspondence between prefix tables and these lists cannot be injective.

3.2. *From border tables to prefix lists*

In order to prove the injection we define the function ψ in terms of border tables: we define another function ψ' from the set of border tables to the set of prefix lists. First consider the following algorithm `BorderToList` which associates a border table to a prefix list:

Algorithm 2: `BorderToList`$(B = B[0\,..\,n-1])$

```
1  L ← [ ]
2  i ← n − 1
3  while i > 0 do
4  |  ℓ ← B[i]
5  |  if B[i] = 0 then
6  |  |  i ← i − 1
7  |  else
8  |  |  i ← i − B[i]
9  |  L ← [ℓ] · L
10 Return L
```

Definition 3.6. For a given border table B we define $\psi'(B)$ as the prefix list obtained

by executing the algorithm `BorderToList` on B. By letting $\ell = B[n-1]$ we have

$$\psi'(B) = \begin{cases} \psi'(B[0\,..\,n-1-\ell]) \cdot [\ell], & \text{if } \ell > 0; \\ \psi'(B[0\,..\,n-2]) \cdot [\ell], & \text{if } \ell = 0. \end{cases}$$

The functions ψ and ψ' applied on equivalent border and prefix tables give rise to the same prefix lists:

Proposition 3.7. Let B be a border table of a word w and P be the prefix table of w. Then we have that $\psi(P) = \psi'(B)$.

Proof. For a given position i in w, let $I = \{j \mid 0 \leq j \leq i \text{ and } j + P[j] - 1 \geq i\}$ as defined in the proof of Proposition 2.3, equation 1, and in the algorithm `PrefixToList` for the computation of $\psi(P)$. By the conversion rules from prefix table to border table (see equation 2 in the proof of Proposition 2.3), we have that

$$B[i] = \begin{cases} 0, & \text{if } I = \emptyset; \\ i - \min(I) + 1, & \text{if } I \neq \emptyset. \end{cases}.$$

Thus if $I = \emptyset$ then the value $\ell = 0 = B[i]$ is inserted at the beginning of L in the same way for both algorithms `PrefixToList` and `BorderToList`. If $I \neq \emptyset$ then $\ell = i - \min(I) + 1 = B[i]$ is inserted at the beginning of the list L for both algorithms. Then i is decremented in the same way for both the algorithms. □

Thus, for a given border table B, there exist $0 \leq i_1 \leq \cdots \leq i_r = n-1$, such that $\psi'(B) = L = [B[i_1], \ldots, B[i_r]]$ and $i_j = i_{j+1} - B[i_{j+1}]$.

Example 3.8. Let w be the word `abaababa`. The following table shows its border table Border_w for all values of i.

i	0	1	2	3	4	5	6	7
$w[i]$	a	b	a	a	b	a	b	a
$\text{Border}_w[i]$	0	0	1	1	2	3	2	3

The associated prefix list is $L = [0, 1, 2, 3] = [B[1], B[2], B[4], B[7]]$.

It is easy to see that the time complexity of the `BorderToList` algorithm is linear:

Proposition 3.9. (Time Complexity) The algorithm `BorderToList` can be implemented in $O(n)$ time on an input border table of length n without using auxiliary memory.

3.3. *From prefix lists to words*

We now describe two "inverse" and linear algorithms associating a prefix list $L = \psi(P)$ with two words whose prefix table is P, one on a maximal size alphabet and one on a minimal size alphabet. The two algorithms are strictly linked as we will see. In the following we give the common schema `ListToWord` of the algorithms. We give before some definitions, as in (Duval *et al.* 2005).

Let $A = \{\alpha(0), \alpha(1), \ldots, \alpha(i), \ldots\}$ be an infinite alphabet. Let u be a word on A of length n and Border_u the associated border table. For each $i \in \{0\,..\,n-1\}$, let $A'(u,i) = \{u[j] \mid u[0\,..\,j-1] \in \text{bord}(u[0\,..\,i-1])\}$ be the set of symbols extending the borders of $u[0\,..\,i-1]$ in u and let $A(u,i) = \{u[i]\} \cup A'(u,i)$.

We define the `nobord` function applied on a word u as any letter in A not following a border of u in u:

Definition 3.10. The `nobord` function applied on a word u returns any letter in A not contained in $A'(u, |u|-1)$.

We then have the following proposition:

Proposition 3.11. Let u be a word. For each $0 \leq i \leq n-1$, if $u[i] \notin A'(u,i)$ then $\text{Border}_u[i] = 0$.

Proof. It is well known (Crochemore *et al.* 2007) that, for any word va, if va has a non-empty border then it is of the form wa where w is a border of v. Thus if $\text{Border}_u[i] \neq 0$ then $u[i]$ has a border that is a prolongation of a border $u[0\,..\,j-1]$ of $u[0\,..\,i-1]$ and in particular $u[i] = u[j] \in A'(u,i)$. □

That implies the following corollary:

Corollary 3.12. For a word u and for $a = \texttt{nobord}(u)$, the word $u \cdot a$ is such that $\text{Border}_{ua}[|u|] = 0$.

Let the prefix list $L = [\ell_1, \ldots, \ell_m]$ and $n = \|L\|$, for the length $\|\cdot\|$ defined for prefix lists, then the string $w[0\,..\,n]$ on A is computed in the following way.

Algorithm 3: `ListToWord`$(L = [\ell_1, \ldots, \ell_m])$

```
1  w[0] ← α(0)
2  pos ← 1
3  for i ← 1 to m do
4      if ℓ_i > 0 then
5          for j ← 0 to ℓ_i − 1 do
6              w[pos + j] ← w[j]
7          pos ← pos + ℓ_i
8      else
9          w[pos] ← nobord(w[0 .. pos − 1])
10         pos ← pos + 1
11 n ← pos
12 return w[0 .. n]
```

Informally the algorithm proceeds from left to right on the prefix list input $[\ell_1, \ldots, \ell_m]$. It starts with a word reduced to one letter. Then iteratively for $i \in [1\,..\,m]$, if $\ell_i > 0$ the algorithm copies ℓ_i symbols, from the previously constructed word u, at the end of u, otherwise the algorithm introduces a letter in w which induces the empty border. Note that overlapping is allowed since we are building the word from left to right.

One key property is that the word w obtained by this algorithm performed on a valid

prefix list $\psi(P)$ for a prefix table P is such that $\mathrm{Pref}_w = P$. This means that valid prefix lists and prefix tables are equivalent and represent the same information.

Proposition 3.13. Given the valid prefix list $L = \psi(P)$ associated with a prefix table P the word w built by the algorithm `ListToWord` is such that $\mathrm{Pref}_w = P$.

Proof. We prove the proposition on border tables: let $L = \psi(P) = \psi'(B)$. We prove the result by induction on $\|L\|$. If $\|L\| = 0$, that is $L = [\,]$, then w is a letter and $\mathrm{Border}_w = [0] = B$.

Suppose now $\|L\| > 0$ and $L = L' \cdot [\ell]$. We denote by w and w' the words built by the algorithm `ListToWord` on input L and L' respectively. If $\ell = 0$ then by construction $w = w' \cdot x$, where x is a letter obtained by applying `nobord` on w'. By the inductive definition of valid prefix lists there exists a decomposition $B = B' \cdot B[n-1]$ such that $L = \psi(B) = \psi(B') \cdot [0]$. By Corollary 3.12 we have that $B[n-1] = 0$. Thus $L' = \psi(B')$ and, by the inductive hypothesis, $\mathrm{Border}_{w'} = B'$ and $\mathrm{Border}_w = B$. By the inductive definition ofvalid prefix lists there exists a decomposition $B = B' \cdot B''$ such that $L = \psi(B) = \psi(B') \cdot [\ell]$, $B''[\ell - 1] = \ell$ and the length of B'' is equal to ℓ. Thus $L' = \psi(B')$ and, by the inductive hypothesis, $\mathrm{Border}_{w'} = B'$.

By construction $w = w' \cdot v$, where v consists necessarily of the first ℓ symbols (considered eventual overlapping) of w'. In general (see (Duval *et al.*), (Moore *et al.*)), given a border table $B = H \cdot T$, every word with border table H is prolonging to a word with border table B. In our case $B = B' \cdot B''$ and since $B[n-1] = \ell$, the word u prolonging w' consists necessarily of the first ℓ symbols (considering possible self overlap) of w' and is equal necessarily to v. Thus $\mathrm{Border}_w = B$. □

From prefix lists to words on a maximal size alphabet.
Let us define the `newletter` function as a function returning a new letter not used so far.

Definition 3.14. The `newletter` function applied on a word u returns a letter in A not contained in u.

For instance, if we let the `newletter` be the `nobord` function given in Algorithm `ListToWord` then we call the algorithm that we obtain `ListToMaxWord`. This algorithm computes for a prefix list $L = \psi(P)$ the word w on an alphabet of maximal cardinality between the words having the same prefix table P.

Example 3.15. Let $w = abaabbabb$ and let

i	0	1	2	3	4	5	6	7	8
$w[i]$	a	b	a	a	b	b	a	b	b
$\mathrm{Pref}_w[i]$	9	0	1	2	0	0	2	0	0

The associated prefix list is $L = [0, 1, 2, 0, 2, 0]$. Choosing arbitrarily the first letter to be a, one can build $w = \mathtt{a} \cdot \mathtt{b} \cdot \mathtt{a} \cdot \mathtt{ab} \cdot \mathtt{c} \cdot \mathtt{ab} \cdot \mathtt{d}$. A value 0 in the prefix list implies we can choose a new letter (here `b` at the second position, `c` at the 5-th position and d at the 8-th position).

It is easy to prove that:

Proposition 3.16. The word constructed by the algorithm `ListToMaxWord` on a valid prefix list $L = \psi(P)$ has maximal size alphabet among the words whose prefix table is P.

Concerning time and space complexity of the algorithm:

Proposition 3.17. (Time Complexity) The algorithm `ListToMaxWord` can be implemented to run in time $O(n)$ on an input border table of length n without using auxiliary memory.

Proof. The instruction in line 9 is executed in constant time with constant memory, since the `newletter` function can be simulated by using a counter each time incremented while introducing a new letter. Since $|L| = n$ then the main loop $3 - 10$ is executed in $O(n)$ and we get the thesis. □

From prefix lists to words on a minimal size alphabet

Algorithm 4: `ListToMinWord`$(L = [\ell_1, \ldots, \ell_m])$

```
1  k[0] ← 0
2  w[0] ← α(0)
3  B[0] ← 0
4  pos ← 1
5  for i ← 1 to m do
6      if ℓ_i > 0 then
7          for j ← 0 to ℓ_i − 1 do
8              w[pos + j] ← w[j]
9              B[pos + j] ← border(B, w[0 .. pos + j − 1], w[pos + j])
10             k[pos + j] ← k[B(pos + j − 1)]
11         pos ← pos + ℓ_i
12     else
13         k[pos] ← 1 + k[B(pos − 1)]
14         w[pos] ← α(k[pos])
15         pos ← pos + 1
16 n ← pos
17 return w[0 .. n]
```

The algorithm computing the word on a minimal size alphabet is a bit more complex. Using the schema of the algorithm `ListToWord`, in order to obtain the desired word, we want to define as function `nobord` on a word u the smallest letter that is not following a border of $u[|u| - 1]$. To do this we will use an auxiliary array $k[0 \,..\, n-1]$ storing, for each prefix u of the word that we are computing, the greatest letter that is following a border of u. More formally, for each word u of length n we define the array $k[0 \,..\, n-1]$ in the following way: for each $i \in \{0, \cdots, n-1\}$, let $k[i] = \mathrm{Card}(A'(u, i))$. Recall that for each $i \in \{0 \,..\, n-1\}$, $A'(u, i) = \{u[j] \mid u[0 \,..\, j-1] \in \mathrm{bord}(u[0 \,..\, i-1])\}$ is the set of symbols extending the borders of $u[0 \,..\, i-1]$ in u.

For computing the array k we will also need to compute on-line, while constructing the word, its border table. Let $\texttt{border}(B, u, a)$ be a the function that applied on a word u, on the border table B of u and on a letter a returns the length of the longest border of ua. Let $L = [\ell_1, \ldots, \ell_m]$ and $n = \|L\|$, for the length $\|\cdot\|$ defined for prefix lists, then the string $w[0\,..\,n]$ is computed in the following way.

Let us note that the function `nobord` is computed in line 14 and involves the array k.

Algorithm 5: $\texttt{border}(B, u, a)$

```
1 i ← B[|u| − 1]
2 while i ≥ 0 and u[i] ≠ a do
3     if i = 0 then
4         i ← −1
5     else
6         i ← B[i − 1]
7 return i + 1
```

Proposition 3.18. The word constructed by the algorithm `ListToMinWord` on a valid prefix list $L = \psi(P)$ is on a minimal size alphabet among the words whose prefix table is P.

The following statements are slight modifications of statements from (Duval *et al.* 2005) and will be useful to prove Proposition 3.18. The following lemma shows how to iteratively compute the set of symbols $A(u, i)$ for a given word u and a position i in u.

Lemma 3.19. (Duval *et al.* 2005) For a given word u and a position i in u, if $\text{Border}_u[i] = 0$ then we have that $A(u, i) = A(u, \text{Border}_u[i-1])$ otherwise $A(u, i) = \{u[i]\} \cup A(u, \text{Border}_u[i-1])$.

The consequent corollaries give the link between the array k and the sets $A(u, i)$:

Corollary 3.20. (Duval *et al.* 2005) Let $u = u[0\,..\,n-1]$ be a word and let $k[0\,..\,n-1]$ be the array computed by the algorithm `ListToMinWord`. Then, for $0 \leq i \leq n-1$ we have $k[i] = \text{Card}(A(u, i))$.

Corollary 3.21. (Duval *et al.* 2005) For every string u and every position i in u, the minimal cardinality of an alphabet necessary to build the prefix $u[0\,..\,i]$ is greater than or equal to $\max\{k[0], k[2], \ldots, k[i]\}$ with $k[0\,..\,n-1]$ the array computed by the algorithm `ListToMinWord`.

By the last Corollary and the following Proposition we get the proof of Proposition 3.18:

Proposition 3.22. (Duval *et al.* 2005) The word w built by the `ListToMinWord` algorithm on L is such that:

1 For $0 \leq i \leq n-1$, $A(w, i) = \{\alpha[0], \alpha[2], \ldots, \alpha[k[i]]\}$.
2 The cardinality of the alphabet for each prefix $w[0\,..\,i]$ is $\max_{0 \leq j \leq i} k[j]$.

More intuitively in lines 13–15 of the algorithm `ListToMinWord` we have that in the case of $\ell_i = 0$, the value k on the current position `pos` is equal to $k[B[\texttt{pos} - 1]] + 1$, where

$k[B[\texttt{pos}-1]]$ represents the number of symbols extending borders of $w[0\,..\,\texttt{pos}-1]$. By the following instruction $w[\texttt{pos}] = \alpha(k[\texttt{pos}])$ we append the smallest letter not following a border of $w[0\,..\,\texttt{pos}-1]$ that is needed. Otherwise no new letter is introduced and k is updated.
Concerning the time complexity of the `ListToMinWord` algorithm we have:

Proposition 3.23. (Time Complexity) The algorithm `ListToMinWord` can be implemented to run in time $O(n)$ on a prefix list of size n.

Proof. We can decompose the complexity of the loop in lines 5–15 as the sum of the complexity C_1 for the instruction 9 and the complexity C_2 for the other instructions. It is proved in [(Crochemore *et al.* 2007), Section 1.6] that C_1 is linear in n. The complexity C_2 is also linear since the instructions involved require constant time. □

3.4. *Injectivity*

Proposition 3.24. The function ψ is injective.

Proof. Let us consider two prefix tables $P \neq P'$ and suppose that $\psi(P) = \psi(P') = L$. By Proposition 3.13 the algorithm performed on L gives a word w such that $\mathrm{Pref}_w = P = P'$. Hence we must have $\psi(P) \neq \psi(P')$. □

Let us remark that the application ψ is not surjective. To a list $[0, 2, 2]$, we can associate a word $w = \mathtt{a}\cdot\mathtt{b}\cdot\mathtt{ab}\cdot\mathtt{ab} = \mathtt{ababab}$ with the prefix table $\mathrm{Pref}_w = [6, 0, 4, 0, 2, 0]$, but we have $\psi(\mathrm{Pref}_w) = [0, 4]$.

4. Upper bound

In this section we prove the upper bound stated in Proposition 1.1.

Prefix lists.
We define the set of prefix lists as a combinatorial class $\mathcal{L}$ of lists of positive integers

$$\mathcal{L} = \textsc{Seq}(\{0, 1, 2, 3, \dots\}), \tag{8}$$

together with a *size* defined for a prefix list $L = [\ell_1, \dots, \ell_k]$ as $\|L\| = \sum_{i=1}^{k} \|\ell_i\|$, where the size $\|i\|$ is i if $i > 0$ and 1 if $i = 0$. It just means that $\|L\| = \sum_{i=1}^{k} \ell_i + \mathrm{Card}\{i \mid \ell_i = 0\}$. The Seq operator applied to a combinatorial class $\mathcal{A}$ corresponds to all finite sequences of elements from $\mathcal{A}$, i.e., $\textsc{Seq}(\mathcal{A}) = \cup_{i=0}^{\infty}\mathcal{A}^i$ (reminiscent of the Kleene star operation for regular languages). By convention $\mathcal{A}^0 = \{\varepsilon\}$.

Combinatorial specifications and generating functions.
In order to study a sequence $(a_n)_{n\in N}$, it is now usual (Flajolet *et al.* 2009) to consider its generating function $A(z)$, that is the formal power series defined by $A(z) = \sum_{n\geq 0} a_n z^n = \sum_{\alpha\in\mathcal{A}} z^{\|\alpha\|}$.
In our case, given the combinatorial specification of $\mathcal{L}$, it is easy (Flajolet *et al.* 2009) to compute the generating function $L(z) = \sum_{n\geq 0} \ell_n z^n$ where ℓ_n denotes the numbers of

prefix lists (either valid or invalid) of size n. This is true when specification are unambiguous (in the same way as unambiguity is considered in regular expressions or formal grammars).

Indeed, the general idea is the following: here we first consider a set of atoms $\mathbb{N}$. We need a size $\|\cdot\|$ compatible with the cartesian product and disjoint union, i.e., here for $i \in \mathbb{N}$ the size of atom i is $\|i\| = i$ if $i > 0$ and $\|0\| = 1$. Let us define an empty element ε (the only one with size 0). Then we have the following dictionary for translating directly from combinatorial constructions to generating functions.

Empty element:	$\varepsilon \mapsto 1$
Symbols:	$\alpha \in \mathbb{N} \mapsto z^{\|\alpha\|}$
Disjoint Union:	$\mathcal{A} \cup \mathcal{B} \mapsto A(z) + B(z)$
Sequence product:	$\textsc{Seq}(\mathcal{A}) \mapsto \frac{1}{1-A(z)}$
Cartesian product:	$\mathcal{A} \times \mathcal{B} \mapsto A(z) \times B(z)$

Let $\varphi = \frac{1}{2}(1+\sqrt{5}) \approx 1.618$. With this dictionary and the combinatorial description (8), we get the following result for $\ell_n = [z^n]L(z)$ the number of prefix lists of size n.

Proposition 4.1. The number of both valid and invalid prefix lists of size n is given by

$$\ell_n = \tfrac{1}{2}\left(1+\tfrac{\sqrt{5}}{5}\right)\ \varphi^n + \tfrac{1}{2}\left(1-\tfrac{\sqrt{5}}{5}\right)\ \varphi^{-n} = \tfrac{1}{2}\left(1+\tfrac{\sqrt{5}}{5}\right)\ \varphi^n + o(1).$$

Proof. Let $\mathcal{I} = \{0\} \cup \{1, 2, 3, \dots\}$ then by definition $\mathcal{L} = \textsc{Seq}(\mathcal{I})$. The generating function associated with $\mathcal{I}$ is $I(z) = 2z + z^2 + z^3 + \ldots = z + z\sum_{n\geq 0} z^n = z + \frac{z}{1-z}$. With this dictionary and the combinatorial description we get

$$L(z) = \frac{1}{1-\left(z+\frac{z}{1-z}\right)} = \frac{1-z}{1-3z+z^2}.$$

Since this is a rational function, using decomposition in simple elements we get

$$L(z) = \tfrac{1}{2}\left(1-\tfrac{\sqrt{5}}{5}\right)\tfrac{1}{1-z/\phi} + \tfrac{1}{2}\left(1+\tfrac{\sqrt{5}}{5}\right)\tfrac{1}{1-z/\phi'},$$

where $\phi = \frac{3+\sqrt{5}}{2}$ and $\phi' = \frac{3-\sqrt{5}}{2}$ are the two solutions of $1-3z+z^2$. By the geometric series formula, we have that

$$\ell_n = [z^n]L(z) = \frac{1}{2}\left(1-\frac{\sqrt{5}}{5}\right)\phi^{-n} + \frac{1}{2}\left(1+\frac{\sqrt{5}}{5}\right)\phi'^{-n}.$$

Let $\varphi = \frac{1}{2}(1+\sqrt{5})$ then we have that $\phi = (1+\varphi)$ and $\overline{\phi} = 2-\varphi = \frac{1}{(1+\varphi)}$. We thus obtain

$$\ell_n = [z^n]L(z) = \tfrac{1}{2}\left(1-\tfrac{\sqrt{5}}{5}\right)(1+\varphi)^{-n} + \tfrac{1}{2}\left(1+\tfrac{\sqrt{5}}{5}\right)(1+\varphi)^{n},$$

and the desired result follows. □

The main result on the upper bound (see Proposition 1.1) is a reformulation of the following corollary, which is a consequence of Proposition 3.24.

Corollary 4.2. The number p_n of prefix tables of size n is upper bounded by the number ℓ_{n-1} of prefix lists of size $n-1$.

5. Lower bound

For the lower bound, we exhibit some sets of valid prefix lists such that we are able to count them. We wish these sets to be as large as possible. In this paper, as a first step, our goal is to evaluate the exponential order growth given in Proposition 1.2 rather than to give a precise bound.

The idea for proving Proposition 1.2 is to exhibit a language which maps bijectively to a set of prefix lists, hence maps bijectively to a set of prefix tables. Let us consider, for a fixed k, $\mathcal{L}_k = ab^k \left(ab^{<k}(\varepsilon + cb^*)\right)^*$.

Proposition 5.1. For each $k > 0$, two distinct words in $\mathcal{L}_k$ have distinct prefix tables.

Proof. We prove that the set $\mathcal{L}_k$ is in bijection with a subset of the set of all prefix lists. Then, since a prefix table is associated with a unique prefix list, the desired result immediately follows.

First we prove that prefix lists associated with words in $\mathcal{L}_k$ are concatenations of non negative integers $\ell < k$. Indeed by construction, for any word $u \in \mathcal{L}_k$ we have that the longest border of a prefix of u is of length strictly less than $k+1$. Let us note by $L(u)$ the prefix list associated with u: $L(u) = \psi(\mathrm{Pref}_u)$. Since the elements in $L(u)$ are borders of prefixes of u we get the result.

Let us prove the main statement by contradiction. Let us consider u, v in $\mathcal{L}_k$ such that $u \neq v$ and $L(u) = L(v) = L = [\ell_1, \ldots, \ell_r]$ (for some $r > 0$). The prefix list L induces the same factorization in u and v: the factorization associated to the sequence of lengths $(\ell_1, \ldots, \ell_r)$. Let i be the smallest position in the words such that $u[i] \neq v[i]$ and let ℓ_j such that $\sum_{k=1}^{j-1} \|\ell_k\| < i < \sum_{k=1}^{j} \|\ell_k\|$, where $\|\cdot\|$ is defined as in Definition 2.6. Let $i_1 = \sum_{k=1}^{j-1} \|\ell_k\|$ and $i_2 = \sum_{k=1}^{j} \|\ell_k\|$. If $\ell_j \neq 0$ then $u[i_1+1\,..\,i_2] = ab^{\ell_j - 1} = v[i_1+1\,..\,i_2]$ since $1 \leq \ell_j < k$, that is a contradiction since $u[i] \neq v[i]$. If $\ell_j = 0$ then the longest border of $u[0\,..\,i]$ is the empty word. Then $u[i]$ can be equal either to b or to c. If $u[i] = b$ then, by definition of $\mathcal{L}_k$, $u[i]$ must be preceded by cb^t for some $t \geq 0$. The element $v[i]$, by definition, must be preceded by ab^s for some $s \geq 0$, that is a contradiction since $u[0\,..\,i-1] = v[0\,..\,i-1]$. □

We are now ready to give a sketch of the proof of Proposition 1.2 stating that : for any $\varepsilon > 0$ there exists a family of prefix tables $(\mathcal{L}_n)_{n\geq 0}$ such that $\mathrm{Card}(\mathcal{L}_n) = \Omega((1+\varphi-\varepsilon)^n)$.

Sketch of the proof of Proposition 1.2 First we remark that the regular expression $\mathcal{L}_k = ab^k \left(ab^{<k}(\varepsilon + cb^*)\right)^*$ is unambiguous since in this decomposition letters a and c act as separator to decompose uniquely any word of $\mathcal{L}_k$. For a given k, by using analytic combinatorics, one compute easily (since the regular expression is unambiguous) the generating function $L_k(z)$ for $\mathcal{L}_k$

$$L_k(z) = z^{k+1} \frac{1}{1 - \left(z \frac{1-z^k}{1-z}(1 + \frac{1}{1-z})\right)} = \frac{z^{k+1}(z-1)^2}{1 - 3z + z^2 + z^{k+1}}.$$

We then have to extract coefficients $\ell_{n,k} = [z^n]L_k(z)$ of this rational function. This is done according to general principles (Flajolet *et al.* 2009). We will prove that the number of words of length n in $\mathcal{L}_k$ is

$$\ell_{n,k} := [z^n]L_k(z) \sim C_k \rho_k^{-n}, \tag{9}$$

where C_k is a constant and ρ_k is the smallest real (simple) root of $1 - 3z + z^2 + z^{k+1}$.

To get (9) we prove that $Q_k(z) = 1 - 3z + z^2 + z^{k+1}$ has only a simple root ρ_k in the open unit disc. Then we can express the asymptotic behavior of the coefficient $[z^n]L_k(z)$. Indeed writing $L_k(z) = \frac{P_k(z)}{Q_k(z)}$ as a rational function we have the following expression (see (Flajolet *et al.* 2009))

$$[z^n]L_k(z) = -\frac{P_k(\rho_k)}{Q_k'(\rho_k)}\rho_k^{-(n+1)} + O(1).$$

Thus, providing $P_k(\rho_k) \neq 0$ (which is easily verified here since $P_k(z) = z^{k+1}(z-1)^2$), we obtain (9).

To end the proof, we first recall the classical Rouché theorem (see (Cartan 1985) for instance).

Theorem 5.2 (Rouché's Theorem). Let γ be a simple closed counterclockwise curve. Let f and g be analytic on and inside γ, and let them satisfy the condition

$$|f + g| < |g| \quad \text{on } \gamma.$$

Then f and g have the same number of roots inside γ.

Considering, $f(z) = 1 - 3z + z^2 + z^{k+1}$, $g(z) = 3z$ and γ a circle around $z = 0$ of radius $\frac{1}{2} < r < 1$, we have (or can easily check) that $|f(z) + g(z)| \leq 1 + r^2 + r^{k+1} < 3r = |3z|$. Hence $f(z)$ has only one root of module strictly less than 1. We also know that this root is real (since $Q(0) > 0$ and $Q(1/2) < 0$ for $k > 1$).

Expressing this root $\rho_k = (1+\varphi-\varepsilon_k)^{-1}$ we are thus considering ρ_k as a perturbation of the root $\rho = (1+\varphi)^{-1}$ of $1 - 3z + z^2 = 0$. Solving approximately the perturbed equation when k tends to ∞, we get

$$\varepsilon_k = \tfrac{1}{2}(1 + 3\tfrac{\sqrt{5}}{5})\tfrac{1}{(1+\varphi)^k}(1 + o(1)).$$

This process of reinjecting a solution in order to get better and better approximations is the essence of the so-called *bootstrapping method* (as in (Knuth 1978)). Using the *standard extraction formula* for rational series with a simple pole (see (Flajolet *et al.* 2009)), we obtain the expression (9).

Hence we get that, for any $\varepsilon > 0$, one can fix k such that $\ell_{n,k} = \Omega((\varphi+1-\varepsilon)^n)$ yielding the result of Proposition 1.2. □

This result gives only rough information on the asymptotic of $\ell_{n,k}$. A deep study should be done in order to get better estimates. For instance it is not yet possible to conclude if the number p_n of prefix tables of size n is asymptotically equivalent to $c(1+\varphi)^n$ for some constant c, or even of order $\frac{(1+\varphi)^n}{n^\alpha}$ for some constant $\alpha > 0$.

6. Conclusion

In this paper we have provided some bounds for the number of prefix (or border) tables. The problem of finding an asymptotic equivalent for the number of prefix tables is however still open, and would require a very fine understanding of the autocorrelation structure of words. For this purpose it would be interesting to find characterizations on prefix lists in order to get better bounds. It would be also interesting to study other families of words in bijection with prefix tables to get better lower bounds. It would also be interesting to deal with the problem of enumerating other tables used in string algorithms, like for instance prefix tables associated to indeterminate strings.

Acknowledgements. We would like to thank Maxime Crochemore, Cyril Nicaud and Giuseppina Rindone for helpful discussions.

References

Bland, W., Kucherov, G. and Smyth, W., F. (2013) Prefix Table Construction and Conversion. In *Lecture Notes in Computer Science (LNCS)* **8288**, 41–53. Springer-Verlag.

Cartan, H. (1985) Théorie élémentaire des fonctions analytiques d'une ou plusieurs variables complexes, Hermann.

Clément, J., Crochemore, C. and Rindone, G.(2009) Reverse Engineering Prefix Tables. In *Leibniz International Proceedings in Informatics (LIPIcs)***3**, 289–300. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Clément, J. and Giambruno, L. (2014) On the number of prefix and border tables. In *Lecture Notes in Computer Science (LNCS)* **8392**. Springer-Verlag.

Crochemore, M., Hancart, C. and Lecroq, T. (2007) Algorithms on strings, Cambridge University Press, Cambridge, UK.

Duval, J.- P., Lecroq, T. (2005) Border array on bounded alphabet. *Journal of Automata, Languages and Combinatorics* **10** (1), 51–60.

Duval, J.- P., Lecroq, T. and Lefebvre, A(2009) Efficient validation and construction of border arrays and validation of string matching automata. *RAIRO-Theoretical Informatics and Applications* **43** (2), 281–297.

Flajolet, P. and Sedgewick, R. (2009) Analytic Combinatorics, Cambridge University Press, Cambridge, UK.

Franek, F., Gao, S., Lu, W., Ryan P. J., Smith W. F., Sun Y. and Yang, L.(2002) Verifying a border array in linear time. *Journal on Combinatorial Mathematics and Combinatorial Computing* **42**, 223–236.

Gusfield, D. (1997) Algorithms on strings, trees and sequences: computer science and computational biology, Cambridge University Press, Cambridge, UK.

Knuth, D. E. (1978) The average time for carry propagation. *Indagationes Mathematicae* **40**, 238–242.

Moore, D., Smyth, W., F. and Miller, D.(1999) Counting distinct strings. *Algorithmica* **23** (1), 1–13.

Nicaud, C. (2010) Private communication.