On the State Complexity of Square Root and Substitution for Subword-Closed Languages

Jérôme Guyot[1]

DER Informatique, Univ. Paris-Saclay, ENS Paris-Saclay, Gif-sur-Yvette, France
jerome.guyot@ens-paris-saclay.fr

Abstract. This paper investigates the state complexities of subword-closed and superword-closed languages, comparing them to regular languages. We focus on the square root operator and the substitution operator. We establish an exponential lower bound for the n-th roots of superword-closed languages. For subword-closed languages we analyze in detail a specific instance of the square root problem for which a quadratic complexity is proven. For the substitution operator, we show an exponential lower bound for the general substitution. In the case of singular substitution, we show a quadratic upper bound when the languages are subword closed and based on disjoint alphabets. We conjecture a quadratic upper bound in the general case of singular substitution with subword-closed languages and prove it in the case where the substitution applies on a directed language.

Introduction

State complexity. The number of states of the canonical automaton recognizing a regular language L is known as its state complexity, denoted $\kappa(L)$. It is a common measure of the complexity of regular languages. Finite state automaton are often used as data structure: the size of the automata thus becomes an important parameter in the complexity analysis of some algorithms.

For an operation or a function f on regular languages, the natural question is "*how does $\kappa(f(L))$ relate to $\kappa(L)$?*" This leads to the definition of *the state complexity of f* as the function $\phi_f : \mathbb{N} \to \mathbb{N}$ such that $\phi_f(n)$ is the maximum state complexity of $f(L)$ with L having state complexity at most n. This notion can be extended to functions having multiple arguments, for example the state complexity of intersection would be given by $\phi_\cap(n_1, n_2)$. This area of automata theory already has a rich literature and the recent survey [14] describes the known results for a wide range of operations and classes of languages. As it can be difficult to find the exact complexity of some $f(L)$ or to give a uniform formula for the function ϕ_f, the goal is often to obtain bounds on the complexity of $f(L)$ and on ϕ_f. This induces a classification of the operations on regular languages and finite automata based on the growth of ϕ_f.

State complexity of subregular classes. It is often interesting to measure the state complexity of a function f when we restrict its argument to a subregular class. As some applications only focus on a subclass of automata it becomes natural to study state complexity on this restricted domain. For example, computational linguistics use automata to encode lexicons that are always finite languages: they are considered in [12] while their complement, cofinite languages, are considered in [4]. Linguistics are also interested in locally-testable languages [19], and other areas like genomics or databases or pattern matching have their own subclasses of interest.

The study of state complexity restricted to such subclasses has recently become quite active after Brzozowski et al. initiated a systematic study of state complexity on various fundamental classes of subregular languages [11,6,9,10].

Subword-closed and superword-closed languages. In the context of computer-aided verification, several algorithms for program verification use well-quasi-ordered data domains [13,1,5] and in particular, using Higman's lemma, words ordered by the subword, or subsequence, ordering.[1] These algorithms handle subword- and superword-closed languages.

[1] Here the terminology is not fixed. Some authors use "subword" for a factor, and use "scattered subword" for what we call a subword (we follow [24]). Superword-closed languages are sometimes called "shuffle ideals" ([18]) and even "all-sided ideals" ([10]). In [15] "ideals" denote the directed subword-closed languages, while superword-closed languages are called "filters".

The state complexity of subword- and superword-closed languages has not been analyzed extensively.[2] The main results are due to Brzozowski et al. who considered subword-closed languages in [9] and superword-closed languages in [10]: these works actually consider several subregular classes at once and only focus on the main operations: boolean combinations (with $\oplus$ denoting the symmetric difference), concatenation, iteration and mirror (denoted L^R). However, there exist other interesting operations to consider as they also preserve the subword/superword closedness such as the shuffle.

The following tables give the known bounds on the state complexity of a few operations.

State complexity for subword-closed languages

Operation	Upper Bound	Tightness requirement	References
$L \cap K$	$mn - (m + n - 2)$	$\lvert\Sigma\rvert \geq 2$	[9] Theorem 2
$L \cup K$	mn	$\lvert\Sigma\rvert \geq 4$	[9] Theorem 2
$L \setminus K$	$mn - (n - 1)$	$\lvert\Sigma\rvert \geq 4$	[9] Theorem 2
$L \oplus K$	mn	$\lvert\Sigma\rvert \geq 2$	[9] Theorem 2
$L \cdot K$	$m + n - 1$	$\lvert\Sigma\rvert \geq 2$	[9] Theorem 3
L^* (and L^+)	2	$\lvert\Sigma\rvert \geq 2$	[9] Theorem 4
L^R	$2^{n-2} + 1$	$\lvert\Sigma\rvert \geq 2n$	[9] Theorem 5
L^k	$k(n - 1) + 1$	$\lvert\Sigma\rvert \geq 2$	[20] Theorem 9

State complexity for superword-closed languages

Operation	Upper Bound	Tightness requirement	References
$L \cap K$	mn	$\lvert\Sigma\rvert \geq 2$	[10] Theorem 7
$L \cup K$	$mn - (m + n - 2)$	$\lvert\Sigma\rvert \geq 2$	[7] Theorem 5
$L \setminus K$	$mn - (m - 1)$	$\lvert\Sigma\rvert \geq 2$	[7] Theorem 5
$L \oplus K$	mn	$\lvert\Sigma\rvert \geq 2$	[10] Theorem 7
$L \cdot K$	$m + n - 1$	$\lvert\Sigma\rvert \geq 1$	[10] Theorem 9
L^* (and L^+)	$n + 1$	$\lvert\Sigma\rvert \geq 2$	[10] Theorem 10
L^R	$2^{n-2} + 1$	$\lvert\Sigma\rvert \geq 2n - 4$	[10] Theorem 11
L^k	$k(n - 1) + 1$	$\lvert\Sigma\rvert \geq 1$	our Proposition C.1

Our contribution. We are interested in completing the picture and consider the state complexity of other operations on subword-closed or superword-closed languages.

In the following sections, we focus on two operators: the n^{th}-root operators and the substitution operator. For the root operators, we show that they have exponential complexity even when restricted to superword-closed languages. For the subword closed languages, when $L = \downarrow(\{w\})$ is the downward closure of a word w it seems that there is

[2] For these languages the most studied question is to obtain them by taking subword- or superword-closure of arbitrary regular languages [18,16,17,23,21] and even of nonregular languages [3].

a quadratic upper bound, we do not know if this extends to the general case of subword-closed languages. For the substitution operator $L^{a\leftarrow K}$, we show a quadratic upper bound when L and K are subword-closed and based on disjoint alphabets and conjecture a quadratic upper bound when L and K are subword-closed. Finally we proved a quadratic upper bound in the case where L is directed (without any hypothesis on the alphabets).

This work contributes to understanding the state complexities of subregular languages. It was done in the context of an initiation to research project at ENS Paris-Saclay. I warmly thank Philippe Schnoebelen for his valuable help and dedication to the project. I also thank Maelle Gautrin and Simon Corbard for initiating the research on the substitution operator.

1 Preliminaries

In this work we assume that the reader is familiar with finite automaton and regular languages. We assume a finite alphabet $\Sigma = \{a, b, \dots\}$, use ϵ to denote the empty word, and write concatenation either $u \cdot v$ or uv. We write $\Sigma(u)$ for the set of letters occurring in a word u, and $|u|_a$ for the number of occurrences of letter a in u.

As we are going to work with subword closed languages that we call here downward closed languages, we must define the notion of subwords.

Definition 1.1. *Let $x, y \in \Sigma^*$, x is a subword of y, denoted $x \preccurlyeq y$, if and only if y can be written as $y = u_1x_1 \cdots u_nx_nu_{n+1}$ for some factorization $x = x_1 \cdots x_n$ of x and. some words $u_0, \dots, u_n$, $u_i \in \Sigma^*$.*

Example 1.2. $\mathtt{abba}$ is a subword of $\mathtt{accbebda}$.

Once we have the notion of subwords, we can take the subwords of a language L, this is called the downward closure of L.

Definition 1.3. *Let $L \subseteq \Sigma^*$, $\downarrow(L) = \{x \in \Sigma^* \mid \exists\ y \in L,\ x \preccurlyeq y\}$ is the* downward closure *of L. L is said downward closed if and only if $\downarrow(L) = L$.*

Example 1.4. $\downarrow(\mathtt{a}^m) = \{\mathtt{a}^i \mid 0 \leq i \leq m\}$ and $\{\epsilon, \mathtt{a}, \mathtt{b}, \mathtt{ab}\} = \downarrow(ab)$ are downward closed.

There is a symmetric notion to downward closed languages. The idea is not to take the subwords of words of L but to take the words such that the words of L are subwords of them : superwords of L. This gives the upward closure of L.

Definition 1.5. *Let $L \subseteq \Sigma^*$, $\uparrow(L) = \{x \in \Sigma^* \mid \exists\ y \in L,\ y \preccurlyeq x\}$ denotes the* upward closure *of L. L is said upward closed if and only if $\uparrow(L) = L$.*

Example 1.6. With $\Sigma = \{\mathtt{a}\}$, $\uparrow(\mathtt{a}^m) = \mathtt{a}^m\mathtt{a}^*$ and $\mathtt{a}^m\mathtt{a}^*$ is upward closed.

Remark 1.7. When taking the upward closure we need to specify the alphabet we are using. This is seen in the equality $\uparrow(L) = L \sqcup \Sigma^*$ linking superwords and shuffle.[3] The alphabet will always be clear from the context when we talk of upward closed languages or upward closures.

Remark 1.8. The downward and upward closures verify the Kuratowski closure axioms :

Let $L, K \subseteq \Sigma^*$

- $\downarrow(\emptyset) = \emptyset$ and $\uparrow(\emptyset) = \emptyset$.
- $L \subseteq \downarrow(L)$ and $L \subseteq \uparrow(L)$.
- $\downarrow(\downarrow(L)) = \downarrow(L)$ and $\uparrow(\uparrow(L)) = \uparrow(L)$.
- $\downarrow(L \cup K) = \downarrow(L) \cup \downarrow(K)$ and $\uparrow(L \cup K) = \uparrow(L) \cup \uparrow(K)$.

The reason why we present the downward closedness and the upward closedness as dual notions comes from the following fact :

Fact 1.9 *L is downward closed if and only if its complement $\bar{L}$, i.e., $\Sigma^* \setminus L$, is upward closed. Equivalently, L is upward closed if and only if $\bar{L}$ is downward closed.*

Another important fact is Haines Theorem: if $L \subseteq \Sigma^*$ is downward closed or upward closed then it is regular.

Let us now define the state complexity of a regular language.

Definition 1.10. *Let $L \subseteq \Sigma^*$ be a regular language. We write $sc(L)$ for the number of states of the canonical automaton recognizing L and call it the* state complexity *of L.*

Remark 1.11. The minimal DFA is just the canonical DFA with unproductive transitions and states removed.

However, reasoning on automaton is not always the easiest way to write formal proofs. For this we will often prefer using the formalism of left-quotients that we will just call *quotients*. For a language L we let $\mathcal{R}(L)$ denote the set of quotients of L and we write $\kappa(L)$ for $|\mathcal{R}(L)|$, i.e., the number of different quotients of L.

Theorem 1.12 ([7]). *For L a regular language, $sc(L) = \kappa(L)$.*

Remark 1.13. Historically, this theorem was proven by Nerode and Myhill in 1957.

The notation $\mathcal{R}(L)$ should be read as $\mathcal{R}_\Sigma(L)$ since the set of quotients depends on the alphabet. Again we leave the alphabet implicit when there is no ambiguity.

Example 1.14. $\kappa(\{\mathtt{a}^m\}) = m + 2$ as the quotients are $\{\mathtt{a}^i \mid 0 \leq i \leq m\}$ and $\emptyset$.

Let us introduce the notion of state complexity through a less trivial example. For this we prefer the notation L/x (over $x^{-1}L$) for a left quotient. This operation has a counterpart in the derivatives of regular expressions, which we denote similarly with e/x.

[3] We do not recall the definition of the shuffle, denoted $L \sqcup L'$, of languages. It can be found, e.g., in [10] or [18].

Lemma 1.15. *For any word w, one has $\kappa(\downarrow(w)) = |w| + 2$.*

Proof. We show that the quotients of $\downarrow(w)$ are exactly the empty set and the $\downarrow(v)$ for all v that are suffix of w, thus $\kappa(\downarrow(w)) = |w| + 2$.

Let $x \in \Sigma^*$, if x is not a subword of w then $\downarrow(w)/x = \emptyset$. If $x \preccurlyeq w$, let w_1 be the smallest prefix of w such that $x \preccurlyeq w_1$ and write $w = w_1 w_2$, then $\downarrow(w)/x = \downarrow(w_2)$. In fact, for any subword y of w_2, $xy \preccurlyeq w$. And for y such that $xy \preccurlyeq w$, then we can factorize w in $w = w_1 w_2$ such that $x \preccurlyeq w_1$ and $y \preccurlyeq w_2$. □

To understand better how we compute quotients of downward closed languages we introduce the following lemma. This is an essential property that will be often used when studying the structure of quotients.

Lemma 1.16. *Let L be a downward closed language and x, y two words with $x \preccurlyeq y$. Then $L/y \subseteq L/x$.*

Proof. Let $w \in L/y$, then we get by definition $yw \in L$. However, as $x \preccurlyeq y$, we also have $xw \preccurlyeq yw$. As L is downward closed, this implies $xw \in L$, hence $w \in L/x$. □

Remark 1.17. $\emptyset$ is a quotient of L if and only if $L \neq \Sigma^*$.

When it is too hard to exactly find the set of quotients, we can use dividing sets to obtain lower bounds on the state complexity of the language.

Definition 1.18. *Let L a language and $\mathcal{F}$ a set of words, $\mathcal{F}$ is a dividing set for L iff $\forall\, x \neq y \in \mathcal{F}, L/x \neq L/y$.*

Remark 1.19. By definition of a dividing set we have that if $\mathcal{F}$ is a dividing set of L then $\kappa(L) \geq |\mathcal{F}|$.

2 Root operators

In this section we will be working on the root operators, that are defined as the n^{th}-root and the union of those roots. Formally:

Definition 2.1. *Let $L \subset \Sigma^*$, and $k \in \mathbb{N}_{>0}$, we let*

$$\sqrt[k]{L} = \{x \mid x^k \in L\}\,, \qquad \sqrt[*]{L} = \bigcup_{k \in \mathbb{N}_{>0}} \sqrt[k]{L}\,.$$

When $k = 2$ we may just write $\sqrt{L}$.

As recalled in the introduction, the root operators preserve regularity and downward/upward closedness. It is known that the state complexity of these operations are exponential in the general case of regular languages [22]. We will prove exponential lower bounds in the case of upward closed languages. For downward closed languages, we will only focus on the square root and study the case of $L = \downarrow(w)$ for w a word to conjecture a quadratic upper bound.

2.1 Upward closed languages

Let us first show that the root operator preserves upward closedness.

Proposition 2.2. *Let L be a upward closed language, then $\sqrt[k]{L}$ and $\sqrt[*]{L}$ are upward closed for $k \in \mathbb{N}_{>0}$.*

Proof. Let $k \in \mathbb{N}$, let $x \preccurlyeq y \in \sqrt[k]{L}$. Then $x^k \in L$ and $x^k \preccurlyeq y^k$. As L is upward closed, $y^k \in L$. Thus $y \in \sqrt[k]{L}$.

If $x \preccurlyeq y \in \sqrt[*]{L}$, then there exists $k \in \mathbb{N}$ such that $x \preccurlyeq y \in \sqrt[k]{L}$ and we conclude as before. □

Let Σ_n be an alphabet having at least n distinct letters $a_1, \dots, a_n$. We denote by V_n the n letter word of length n : $V_n = \mathtt{a}_1 \cdots \mathtt{a}_n$. For example, $V_4 = \mathtt{abcd}$.

Proposition 2.3. $\kappa(\sqrt[k]{\uparrow(V_n)}) \geq 2^n$ *when* $k \in \mathbb{N}_{>0}$ *and* $\kappa(\sqrt[*]{\uparrow(V_n)}) \geq 2^n$.

Proof. Let v, w be two subwords of V_n such that $v \neq w$. Then there is a letter of V_n that is either in v and not in w or in w and not in v. Without loss of generality, let us assume the letter a is in v but not in w.

Then let x be V_n with a removed. Thus, for any $k \geq 1$, V_n is a subword of $(vx)^k$ but not of $(wx)^k$. Thus, $vx \in \sqrt[k]{\uparrow(V_n)}$ but $wx \notin \sqrt[k]{\uparrow(V_n)}$. Hence, $\sqrt[k]{\uparrow(V_n)}/v \neq \sqrt[k]{\uparrow(V_n)}/w$ since one contains x but not the other. Finally, as there are 2^n distinct subwords of V_n, there has to be at least 2^n distinct quotients in $\mathcal{R}(\sqrt[k]{\uparrow(V_n)})$.

Now for $\kappa(\sqrt[*]{\uparrow(V_n)})$, as $wx \notin \sqrt[k]{\uparrow(V_n)}$ for all $k \in \mathbb{N}_{>0}$, $\mathcal{R}(\sqrt[*]{\uparrow(V_n)})$ contains at least 2^n quotients. □

This shows a lower bound in 2^n, however experimentally for the square root, we have values close to 3^{n-1}. In the appendix, we improve Proposition 2.3 and give in Proposition A.4 a lower bound in $\mathcal{O}(2.41^n)$. Furthermore, we give in Definition A.6 a direct characterization of $\sqrt{\uparrow(V_n)}$ by describing its minimal elements (i.e. its generators).

2.2 Downward closed languages

Let us first show that the root operator preserves downward closedness.

Proposition 2.4. *Let L be a downward closed language, then $\forall k \in \mathbb{N}_{>0}$, $\sqrt[k]{L}$ and $\sqrt[*]{L}$ are downward closed.*

Proof. Let $k \in \mathbb{N}_{>0}$, let $y \preccurlyeq x \in \sqrt[k]{L}$. Then $y^k \preccurlyeq x^k \in L$. As L is downward closed, $y^k \in L$. Thus $y \in \sqrt[k]{L}$.

If $y \preccurlyeq x \in \sqrt[*]{L}$, then there exists $k \in \mathbb{N}$ such that $y \preccurlyeq x \in \sqrt[k]{L}$ and we conclude as before. □

Let $V_n = a_1 a_2 \cdots a_n$ be like in Section 2.1 and define $W_n = V_n^3$. For example $W_2 =$ `abababab`. To understand better what could be the square root of a downward closure, we study the example of $\sqrt{\downarrow(W_n)}$.

Recall that the conjugates of u are all the words of the form $v'v$ for some factorization vv' of u. E.g., the conjugates of `babar` are {`babar`, `abarb`, `barba`, `arbab`, `rbaba`}.

Proposition 2.5. *$x \in \sqrt{\downarrow(W_n)}$ iff x is a subword of a conjugate of V_n.*

Proof. Let us proceed by double implications,

Let us start with the easy case, let x be a subword of a conjugate $v'v$ of $V_n = vv'$. Then xx is a subword of $v'vv'v$ which is a subword of $vv'vv'vv' = W_n$. Thus $x \in \sqrt{\downarrow(W_n)}$.

Now for the $\Rightarrow$ direction, let $x \in \sqrt{\downarrow(W_n)}$, if $x = \epsilon$ then it is a subword of any conjugate. Otherwise, let us isolate the first letter of $x = ay$ and factorize $V_n = vav'$ with no a occurring in v: we get $xx = ayay \preccurlyeq vav'vav'vav'$. There are two cases : either the first x embeds in the $vav'v$ prefix, or the second x embeds in the $v'vav$ suffix. In the 1st case $x \preccurlyeq vav'v$ entails $x \preccurlyeq av'v$ since v has no a and we are done since $av'v$ is a conjugate of V_n. In the second case $x \preccurlyeq v'vav$ entails $x \preccurlyeq av$ since $v'v$ has no a and we are done again. □

Proposition 2.6. $\kappa(\sqrt{\downarrow(W_n)}) = n^2 - n + 3$.

Proof. Using Proposition 2.5, we have that $x \in \sqrt{\downarrow(W_n)}$ iff x is a subword of a conjugate of V_n. Let us compute $\kappa(\sqrt{\downarrow(W_n)})$ by listing and counting its quotients.

Let us write $L = \sqrt{\downarrow(W_n)}$. First, if $x = \epsilon$, $L/x = L$. If x contains two times the letter a, then $L/x = \emptyset$. And if x is a conjugate of V_n, $L/x = \epsilon$.

There only remain the cases where x is a strict non-empty subword of a conjugate of V_n. We claim that all these x yield different L/x quotients. Let us write $x = a_{i_1} ... a_{i_m}$ with $1 \leq m \leq n$. Let us denote by $v[i, j]$ the factor of V_n starting by a_i and finishing by a_j, $1 \leq i \leq j \leq n$. We get that the conjugates of V_n are of the form $v[i, n]v[1, i-1]$. Thus, by Proposition 2.5 we have $L = \sum_{1 \leq i \leq n} \downarrow(v[i, n]v[1, i-1])$. Then we get that if $i_1 < i_m < n$,

$$\begin{aligned} L/x &= \bigcup_{1 \leq i \leq n} \downarrow(v_i^n v_1^{i-1})/(a_{i_1} ... a_{i_m}) \\ &= \bigcup_{1 \leq i \leq i_1} \downarrow(v_{i_{m+1}}^n v_1^{i-1}) \\ &= \downarrow(v_{i_{m+1}}^n v_1^{i_1-1}) \end{aligned}$$

Furthermore, using the same equations, if $1 \leq i_m \leq i_1 - 1$, $L/x = \downarrow v_{i_{m+1}}^{i_1-1}$. Thus we get a different quotient, different from L, $\{\epsilon\}$ or $\emptyset$, for each pair $(i_1, i_m) \in \{1, ..., n\} \times \{i +$

$1[n], ..., i + (n-1)[n]\}$ where $x[n]$ denotes x mod n. Thus, we end up with $n^2 - n + 3$ different quotients. □

We implemented an algorithm that given a language returns the minimal DFA of the square root of its downward closure. In practice as it can be quite slow we used it on words of length smaller than 12. Based on the results, it seems that W_n gives the largest state complexity for words of length $3n$. In the following, we try to understand this by studying the largest state complexity of $\sqrt{\downarrow(w)}$ for w word of length n.

Definition 2.7. *Let* $\alpha(n) = \max_{|w| \leq n} \kappa(\sqrt{\downarrow(w)})$.

Proposition 2.8. *For all* $n \in \mathbb{N}$ *we have*

- $\alpha(n) \leq \alpha(n+1)$,
- $\alpha(n) < \alpha(n+2)$.

Proof. For the first point, let w be a word reaching the maximum for $n : \kappa(\sqrt{\downarrow(w)}) = \alpha(n)$. Let us consider wa_{n+1}, $\sqrt{\downarrow(wa_{n+1})} = \sqrt{\downarrow(w)}$. This comes from the fact that a letter appearing once does not appear in the square root !

Thus $\kappa(\sqrt{\downarrow(w)}) = \kappa(\sqrt{\downarrow(wa_{n+1})}) \leq \alpha(n+1)$.

For the second point, let w be a word reaching the maximum for n. Let u a word of maximal length in $\sqrt{\downarrow(w)}$. This means that u is a maximal word such that uu is a subword of w. Thus we can factorize w as $w = w_1 w_2$ where $u \preccurlyeq w_1$ and $u \preccurlyeq w_2$. Let us now consider the word $v = w_1 a_{n+1} w_2 a_{n+1}$. We can easily see that ua_{n+1} is a maximal word of $\sqrt{\downarrow v}$, and that $\sqrt{\downarrow(w)} \subsetneq \sqrt{\downarrow(v)}$.

It is important to see that a dividing set of $\sqrt{\downarrow(w)}$ is still a dividing set of $\sqrt{\downarrow(v)}$. To show this, let us take x, y such that $\sqrt{\downarrow(w)}/x \neq \sqrt{\downarrow(w)}/y$. Then there exists a word z made of letter $\{a_1, ..., a_n\}$ such that $xz \in \sqrt{\downarrow(w)}$ but $yz \notin \sqrt{\downarrow(w)}$ (or the other way around, but this case is symmetric and leads to the same conclusion). Seeing that all the new words in $\sqrt{\downarrow(v)}$ are of the form ua_{n+1} for $u \in \sqrt{\downarrow(w)}$, we get that $yz \notin \sqrt{\downarrow(v)}$. This concludes the validity of the dividing set.

Now we can also add a new element to this dividing set of $\sqrt{\downarrow(v)} : ua_{n+1}$. To see this, we can just observe that if $\sqrt{\downarrow(w)}/x = \epsilon$ then $\sqrt{\downarrow(v)}/x = \{\epsilon, a_{n+1}\}$ and $\sqrt{\downarrow(v)}/ua_{n+1} = \{\epsilon\}$. Indeed, if $x \in \sqrt{\downarrow(w)}$, then $xa_{n+1}xa_{n+1} \preccurlyeq w_1 a_{n+1} w_2 a_{n+1} = v$, and the second equality comes from the paragraph above. Thus we can add ua_{n+1} to the dividing set, thus giving that $\alpha(n+2) \geq \alpha(n) + 1 > \alpha(n)$. □

Corollary 2.9. *Let* w *be a word of length* $n+2$*, if* w *has* 2 *letters appearing only once, then* $\kappa(\sqrt{\downarrow(w)}) < \alpha(n+2)$.

Proof. Let a, b be the two letters appearing only once in w and w' be the word w without a and b, we get $|w'| = n$. Let $x \in \sqrt{\downarrow(w)}$ then $xx \preccurlyeq w$. However, as there is only one a and b in w, x cannot contain an a or b. Thus, $x \preccurlyeq w \Leftrightarrow x \preccurlyeq w'$ and $\kappa(\sqrt{\downarrow(w)}) = \kappa(\sqrt{\downarrow(w')}) \leq \alpha(n) < \alpha(n+2)$. □

Let us introduce the notation $c(n) = \lceil \frac{n}{3} \rceil$. Let U_n be defined as $U_n = V_{c(n)} V_{c(n)} V_{n-2c(n)}$. For example, $U_7 = \texttt{abcabca}$ and $U_{11} = \texttt{abcdabcdabc}$.

Conjecture 2.10. $\forall\, n \in \mathbb{N}_{>0},\ \kappa(\sqrt{\downarrow(U_n)}) = \alpha(n) \leq c(n)^2 - c(n) + 3$.

Remark 2.11. Experimentally, we observed that for $n \leq 11$, $\alpha(n) \leq c(n)^2 - c(n) + 3$. The question is now to see if this can extend to larger words and be proven.

3 Substitution operator

Let us now consider the substitution operator. We consider a fixed alphabet $\Sigma = \{a_1, \ldots, a_n\}$.

Definition 3.1. *Let $K_1, \ldots, K_n \subseteq \Gamma^*$ be n languages on an alphabet Γ that may be different from Σ. The K_is define a substitution $\rho : \Sigma^* \to \mathcal{P}(\Gamma^*)$ that associates a Γ-language with every word of Σ^*. Formally $\rho(w)$ is given by*

$$\rho(a_{i_1} a_{i_2} \cdots a_{i_m}) = K_{i_1} \cdot K_{i_2} \cdots K_{i_m}\,, \qquad \rho(\epsilon) = \{\epsilon\}\,.$$

This is then lifted to Σ-languages with

$$\rho(L) = \bigcup_{w \in L} \rho(w)\,.$$

We sometimes write $L^{a_1 \leftarrow K_1, \ldots, a_n \leftarrow K_n}$ instead of $\rho(L)$: this notation is convenient, e.g., when $K_i = \{a_i\}$ for most i's. In such cases we may just write, e.g., $x^{a_3 \leftarrow K_3}$ with the meaning that letter different from a_3 are unchanged by the substitution.

It is well known that if all the K_i are regular then $\rho(L)$ is regular when L is (and we say that ρ is a regular substitution).

Actually, the substitution preserves more : if each $K_i + \epsilon$ is downward closed (for $i = 1, \ldots, n$) then ρ preserves downward closedness (and we say it is a downward closed substitution). By showing this, we can imply in this case the regularity as downward closed languages are regular.

Proposition 3.2. *If $K_1, ..., K_n$ are downward closed then $\rho(L)$ is downward closed (hence also regular).*

Proof. Let $z \in \rho(L)$ then $z \in \rho(x)$ for some $x = a_{i_1} \cdots a_{i_m} \in L$ and z can be factorized as $z = z_1 \cdots z_m$ with $z_i \in K_{a_i}$. If now $y \preccurlyeq z$ then y is some $y_1 \cdots y_m$ with $y_i \preccurlyeq z_i$ for $i = 1, \ldots, m$, so $y_i \in K_{a_i}$ since each K_{a_i} is downward closed. Finally $y \in \rho(x) \subseteq \rho(L)$.

Hence, $\rho(L)$ is downward closed. □

Remark 3.3. We observe that for any substitution where the (K_i) are non-empty languages (not necessarily downward closed) we have $\downarrow\left(L^{a_1 \leftarrow K_1, \ldots, a_n \leftarrow K_n}\right) = \downarrow(L)^{a_1 \leftarrow \downarrow(K_1), \ldots, a_n \leftarrow \downarrow(K_n)}$.

The case of substitutions is well known and studied, and we will see in the next proposition that there is an exponential lower bound for the state complexity of substitutions even in the case of downward closed L and ρ.

Example 3.4. Let $L = \{\epsilon, a_1, ..., a_n\} = \downarrow(\Sigma)$ and, for $i = 1, \dots, n$, $K_i = (\Sigma \setminus \{a_i\})^*$. Write L' for $\rho(L) = \bigcup_i A_i$. Now a word in L' cannot use all letters of Σ so if x is the prefix of some word xy in L' then x can only be continued by some y that does not use all the letters of $\Sigma \setminus \Sigma(x)$. We see that the quotient L'/x is $(\Sigma \setminus \Sigma(x))^*$ so there is a bijection between the subsets of Σ and the quotients of L'. Hence, $\kappa(\rho(L)) = 2^n$.

Proposition 3.5. *Let $\Sigma = \{a_1, ..., a_n\}$, and $A_i = (\Sigma \setminus \{a_i\})^*$. Then, let $A[i, j] = A_i \sqcup \downarrow(a_i^j)$, the words not having a_i^{j+1} as subword. Let $L = \{\epsilon, a_1, ..., a_n\}$, $(m_i)_{1 \le i \le n} \in \mathbb{N}^n$. Let us define $K_i = A[i, m_i]$, then $\kappa(L^{a_1 \leftarrow K_1, ..., a_n \leftarrow K_n}) = \prod_{1 \le i \le n}(m_i + 2) = \prod_{1 \le i \le n} \kappa(K_i)$.*

Proof. Let $u \in \Sigma^*$, $L/u = \{\sum A[i, m_i - j_i] \mid a_i^{j_i}$ subword of u, $1 \le j_i \le m_i + 1\}$ with the convention $A[i, j] = \emptyset$ if $j > i$. Indeed, if u contains j_i occurrences of a_i then we can still have $m_i - j_i$ such occurrences in a suffix while staying in K_i, else we are not in K_i. Thus we can take u such that it has $0 \le j_i \le m_i$ occurrences of a_i and for each value in $\{0, ..., m_i + 1\}^n$ we get a different quotient of K_i, and we do this for each a_i. Thus, there are $\prod_{1 \le i \le n}(m_i + 2)$ such quotients of $L^{a_1 \leftarrow K_1, ..., a_n \leftarrow K_n}$. Finally, K_i has $m_i + 2$ quotients : $\{A[i, j] \mid 0 \le j \le m_i + 1\}$. Hence $\kappa(\rho(L)) = \prod_{1 \le i \le n} \kappa(K_i)$. □

Remark 3.6. We can actually see that Example 3.4 is a particular case of the previous proof with $m_i = 0$ for all i.

As the state complexity of the substitution is exponential in the general case, it can be interesting to know if there exists a restriction for which the state complexity is polynomial. We thus consider singular substitutions, that is to say substitutions $L^{a \leftarrow K}$.

Let us start with the easiest case, when there exists words u, v such that $L, K = \downarrow(u), \downarrow(v)$.

Proposition 3.7. *Let $L = \downarrow(u)$ and $K = \downarrow(v)$ with u and v two words, and $a \in \Sigma$ be a letter. Then $L^{\{a \leftarrow K\}} = \downarrow(u^{\{a \leftarrow v\}})$ and $\kappa(L^{\{a \leftarrow K\}}) = |u| - |u|_a + |u|_a|v| + 2 \le \kappa(L)\kappa(K)$.*

Proof. Using Remark 3.3 we get that $L^{a \leftarrow K} = \downarrow(u^{a \leftarrow v})$. We then apply Lemma 1.15 and conclude. □

We now want to study the quotients of a more general substitution $L^{a \leftarrow K}$, for this we explain how to inductively compute quotients of substitutions. As in Definition 3.1, we write $\rho(R)$ for $R^{a \leftarrow K}$.

Lemma 3.8. *[8]*

Let L be a language, we denote by $L^\epsilon = \begin{cases} \emptyset, & \text{if } \epsilon \notin L, \\ \{\epsilon\}, & \text{otherwise.} \end{cases}$

Let us introduce the basic rules of the computation of quotients given in [8] :

- $b/a = \begin{cases} \emptyset, & \text{if } b \neq a, \\ \epsilon, & \text{otherwise.} \end{cases}$
- $(L \cup K)/a = L/a \cup K/a$.
- $(L \cdot K)/a = (L/a).K \cup L^\epsilon.(K/a)$.

Remark 3.9. In the following we are going to use downward closed languages and their quotients which are also downward closed. As $L^\epsilon = \{\epsilon\}$ for any non-empty downward closed language L, the concatenation rule becomes in this case :

$$(L \cdot K)/a = (L/a) \cdot K + (K/a) \quad \text{if } L \text{ is non-empty.} \tag{1}$$

This property is essential to understand the structure of the quotients.

Remark 3.10. One could think that the case $K = \emptyset$ might be pathological. However n our setting one can see that, if L is downward closed, then $L^{a\leftarrow\emptyset} = L^{a\leftarrow\epsilon}$.

Now that we have those basic rules for computing quotients, we want to see how we can apply them in our case. For this we need to introduce the SRE formalism (for Simple Regular Expressions) and show how the rules for computing quotients behave on what we call products of atoms.

Definition 3.11 ([2]).
An atom is a (particular case of) regular expression α that is either a letter-atom $a + \epsilon$ with $a \in \Sigma$, or a star-atom B^* with $B \subseteq \Sigma$.

A product of atoms (or product) I is a finite concatenation of atoms : $I = \prod_{1\leq i\leq n} \alpha_i$ where α_i is an atom. It is a regular expression and the empty product denotes ϵ.

An SRE E is a finite sum of products : $E = \sum_{1\leq j\leq m} I_j$. The empty sum denotes ϵ. We denote by $[\![E]\!]$ the language described by E. The SREs form a subclass of regular expressions.

Remark 3.12. Atoms are included in products, that are included in SRE. Furthermore, we might abuse notations and write E when what we mean is $[\![E]\!]$. For example by saying $w \in E$ when w is a word.

The following theorem linking the SREs and downward closed languages justifies this new representation. Recall that, in our context, L is *directed* $\overset{\text{def}}{\Leftrightarrow} \forall x, y \in L : \exists z \in L : x \preccurlyeq z \wedge y \preccurlyeq z$.

Theorem 3.13. *[2,15] A language L on a finite alphabet Σ is downward closed if and only if it can be defined by an SRE.*

Furthermore L is directed if and only if it can be defined by a product.

A corollary is that any downward closed language is a finite union of directed languages.

Now that we introduced the concept of SREs, we need to understand how quotienting behaves on them. For now we will focus on quotients of products of atoms.

Definition 3.14. *Let $I = \prod_{i\leq n} \alpha_i$ be a product of atoms. A suffix product of I is any product $I' = \prod_{i_0\leq i\leq n} \alpha_i$ for some i_0.*

We are now going to use this notion to describe the structure of the quotients of a product.

Lemma 3.15. *Let $I = \prod_{i \leq n} \alpha_i$ be a product, then the quotients of I are exactly the suffix products of I, with also the empty set if $[\![I]\!] \neq \Sigma^*$.*

Proof. Let R be a quotient of I, there is a word x such that $R = I/x$. Let us proceed by induction on the length of x.

If $|x| = 0$, $I/x = I$ and I is a suffix product of itself.

Let $x = x'b$ with $|x'| \geq 0$, and let $I' = I/x'$ be the suffix product we get by the induction hypothesis. If $I' = \emptyset$, then $I/x = I'/b = I' = \emptyset$ and it is valid. Now if I' is non-empty, using the definition of suffix product, $I' = \prod_{j' \leq i} \alpha_i$. Let $\alpha_{i_0} = \min_{j' \leq i}(b \in \alpha_i)$. If such an α_{i_0} does not exist, then $I/b = \emptyset$. Using the rules for the computation of quotient, we have:

- $(a + \epsilon)/b = \begin{cases} \emptyset, & \text{if } b \neq a \\ \{\epsilon\}, & \text{otherwise} \end{cases}$
- $(B^*)/b = \begin{cases} \emptyset, & \text{if } b \notin B \\ B^*, & \text{otherwise} \end{cases}$

Thus, if α_{i_0} exists, then $I'/b = \alpha_{i_0} \prod_{i_0 < i} \alpha_i$ if α_{i_0} is a star-atom and $I'/b = \epsilon \cdot \prod_{i_0 < i} \alpha_i$ if α_{i_0} is a letter atom . Thus I'/b is a suffix product of I', which makes it a suffix product of I.

Finally, if $[\![I]\!] \neq \Sigma^*$ then there exists $w \in \Sigma^*$ such that $w \notin [\![I]\!]$, thus $I/w = \emptyset$. And if $[\![I]\!] = \Sigma^*$ then for all $w \in \Sigma^*$, $I/w = \Sigma^* = I \neq \emptyset$. □

Corollary 3.16. *Let I be a product of atoms, and R_1, R_2 two quotients of I. Then one of the quotient contains the other.*

Having this result for the quotients of products of atoms will make it easier for us to prove that quotients once we apply the substitution can be expressed in a compact form in the case of downward closed language. For now let us show this result for product for atoms.

Lemma 3.17. *Let I be a product of atoms, K a downward closed language and $a, b \in \Sigma$, then* $\rho(I)/b = \begin{cases} \rho(I/b) + (K/b) \cdot \rho(I/a) & \text{if } a \neq b\,, \\ (K/a) \cdot \rho(I/a) & \text{otherwise.} \end{cases}$

Proof. Let us proceed by induction on the length of I. If I is the empty product then both terms give $\emptyset$ hence the equality.

If $I = \alpha \cdot I'$, then $\rho(\alpha \cdot I')/b = (\rho(\alpha)/b) \cdot \rho(I') + \rho(\alpha)^\epsilon \cdot \rho(I')/b$. Now as α and I' are downward closed and $\alpha \neq \emptyset$ we can use Equation (1). Hence, $\rho(I)/b = (\rho(\alpha)/b) \cdot \rho(I') + \rho(I')/b$.

- If $\alpha = (a + \epsilon)$ then $\rho(\alpha)/b = K/b$. If $K/b \neq \emptyset$, then $\rho(I')/b \subseteq (\rho(\alpha)/b) \cdot \rho(I') = (K/b).\rho(I/a)$, thus $\rho(I)/b = (K/b) \cdot \rho(I/a)$. And as $I/b \subseteq I/a$, we get $\rho(I/b) \subseteq \rho(I/a) \subseteq (K/b) \cdot \rho(I/a)$. Hence, $\rho(I)/b = \rho(I/b) + (K/b) \cdot \rho(I/a)$.
  If, on the other hand $K/b = \emptyset$, then $\rho(I)/b = \rho(I')/b = \rho(I'/b) + (K/b) \cdot \rho(I'/a)$ by induction. And as $K/b = \emptyset$ we get $\rho(I)/b = \rho(I')/b = \rho(I'/b)$. Thus if $a \neq b$ implies $I'/b = I/b$ we get $\rho(I'/b) = \rho(I/b) = \rho(I/b) + (K/b) \cdot \rho(I/a)$ as $I = (a + \epsilon)I'$ and $K/b = \emptyset$. On the other hand, if $a = b$, then as $K/a = \emptyset$, $\rho(I)/a = \emptyset = K/a \cdot \rho(I/a)$.

- If $\alpha = (c + \epsilon)$ with $c \neq a$, $\rho(I)/b = (\rho(c + \epsilon) \cdot \rho(I'))/b = ((c + \epsilon) \cdot \rho(I'))/b$. Thus if $c = b$, $\rho(I)/b = \rho(I') + \rho(I'/b) = \rho(I') = \rho(I/b)$. And when $c \neq b$ and $b \neq a$, $\rho(I)/b = \rho(I')/b = \rho(I'/b) + (K/b) \cdot \rho(I'/a)$ by induction. However as $I = (c + \epsilon)I'$, $\rho(I'/b) + (K/b) \cdot \rho(I'/a) = \rho(I/b) + (K/b) \cdot \rho(I/a) = \rho(I)/b$. Finally, if $c \neq b$ and $b = a$, $\rho(I)/a = \rho(I')/a = (K/a) \cdot \rho(I'/a) = (K/a) \cdot \rho(I/a)$.

- If $\alpha = B^*$ with $a \notin B$, then if $b \in B$ we have $B^*/b \neq \emptyset$ which implies $(K/b) \cdot \rho(I/a) \subseteq \rho(I')/b \subseteq (\rho(\alpha)/b) \cdot \rho(I') = (B^*) \cdot \rho(I') = \rho(I/b)$. Thus $\rho(I)/b = \rho(I/b) + (K/b) \cdot \rho(I/a)$.
  When $b \notin B$ and $b \neq a$, $\rho(I)/b = \rho(I')/b = \rho(I'/b) + (K/b) \cdot \rho(I'/a)$ by induction. And as neither a or b is in B it gives $\rho(I'/b) + (K/b) \cdot \rho(I'/a) = \rho(I/b) + (K/b) \cdot \rho(I/a) = \rho(I)/b$. On the other hand, when $b \notin B$ and $b = a$, $\rho(I)/b = \rho(I')/b = (K/a) \cdot \rho(I'/a) = (K/a) \cdot \rho(I/a)$.

- Finally, if $\alpha = B^*$ with $a \in B$, then $\rho(\alpha) = (\Sigma(K) \cup (B \setminus \{a\}))^* = B'^*$. If $b \in B'$ and $a \neq b$ then $\rho(I)/b = B'^*\rho(I')$. Now if $b \notin \Sigma(K)$ then $B'^*\rho(I') = \rho(I/b) = \rho(I/b) + (K/b) \cdot \rho(I/a)$. If $b \in \Sigma(K)$, $B'^*\rho(I') = \rho(I/a) = (K/b) \cdot \rho(I/a) = \rho(I/b) + (K/b) \cdot \rho(I/a)$. In the case where $b \in B'$ and $a \neq b$, this means that $a \in \Sigma(K)$. Thus $\rho(I)/a = B'^*\rho(I') = \rho(I/a) = K/a \cdot \rho(I/a)$.
  Now if $b \notin B'$ and $b \neq a$, $\rho(I)/b = \rho(I')/b = \rho(I'/b) + (K/b) \cdot \rho(I'/a)$ by induction. However as $b \notin B$, $\rho(I'/b) = \rho(I/b)$. And as $b \notin B'$, we get $K/b = \emptyset$, thus $(K/b) \cdot \rho(I'/a) = (K/b) \cdot \rho(I/a) = \emptyset$. Hence $\rho(I)/b = \rho(I'/b) = \rho(I/b) = \rho(I/b) + (K/b) \cdot \rho(I/a)$. Finally, if $b \notin B'$ and $b = a$ we get $K/a = \emptyset$ since $a \notin \Sigma(K)$. Thus $\rho(I)/a = \emptyset = K/a \cdot \rho(I/a)$.

  Thus, in the case where $a = b$, we get that $\rho(I)/a = K/a \cdot \rho(I/a)$. And when $a \neq b$ we get $\rho(I)/b = \rho(I/b) + K/b \cdot \rho(I/a)$. □

This result is interesting but as we do not want to limit ourselves to product of atoms, we need to extend it. And as the substitution and the quotient distribute over sums, we can easily extend this lemma to general downward closed languages.

Proposition 3.18. *Let $L, K \subset \Sigma^*$ be languages with L,K downward closed and $a \neq b \in \Sigma$, we compute the quotients of $L^{a \leftarrow K}$ as follows :*

- $\rho(L)/\epsilon = \rho(L)$,
- $\rho(L)/b = \rho(L/b) + K/b \cdot \rho(L/a)$,
- $\rho(L)/a = K/a \cdot \rho(L/a)$.

Proof. Since L is downward closed it can be written as a sum of product $L = \sum_j I_j$,
Let $b \neq a$ a letter.

$$\begin{aligned}\rho(L)/b &= \rho\Big(\sum_j I_j\Big)/b \\ &= \sum_j \rho(I_j)/b \\ &= \sum_j \big(\rho(I_j/b) + (K/b).\rho(I_j/a)\big)\end{aligned}$$

using Lemma 3.17

$$\begin{aligned}&= \sum_j \rho(I_j/b) + \sum_j (K/b).\rho(I_j/a) \\ &= \rho(L/b) + (K/b).\rho(L/a)\ .\end{aligned}$$

A similar computation shows $\rho(L)/a = K/a \cdot \rho(L/a)$. □

Now that we know the structure of the quotients by a letter of the substitution in the case of downward closed languages, we can study the state complexity (i.e., quotients by words).

Proposition 3.19. *Let L, K be downward closed languages such that $\Sigma(L) \cap \Sigma(K) = \emptyset$, with $K \neq \emptyset$. For every R quotient of $\rho(L)$, there exists a pair (P, Q) where $Q \in \mathcal{R}(L)$ and $P \in \mathcal{R}(K) \cup \{\{\epsilon\}\}$, such that $R = P.\rho(Q)$.*

Proof. Let us write Ψ for the function associating, with every quotient R of $\rho(L)$, a pair (P, Q) such that $R = P.\rho(Q)$: we want to show that this function is well defined. For a quotient R of $\rho(L)$, there exists $x \in \Sigma^*$ such that $R = \rho(L)/x$. Let us prove the proposition by induction on the length of x.

If $|x| = 0$, $R = \rho(L) = \{\epsilon\}.\rho(L)$, thus the pair $(L, \{\epsilon\})$ fulfills the claim.

If $x = x'b$ with $|x'| \geq 0$ and b a letter, let us write $R' = \rho(L)/x'$, by induction hypothesis we have that $R' = P'.\rho(Q')$. Using Proposition 3.18 we get that $R = R'/b = (P' \cdot \rho(Q'))/b = P'/b \cdot \rho(Q') + \epsilon \cdot \rho(Q')/b$. However, if $P'/b \neq \emptyset$, as $\rho(Q')/b \subseteq \rho(Q')$ because L is downward closed and $\epsilon \in P'/b$, then $\rho(Q')/b \subseteq P'/b \cdot \rho(Q')$. Thus, if $P'/b \neq \emptyset$, then $R = P'/b \cdot \rho(Q')$.

If $P'/b = \emptyset$ we have $R = \rho(Q')/b$. If $a = b$ then $R = K/a \cdot \rho(Q'/a)$ and this fulfills the claim. On the other hand, if $b \neq a$, $R = \rho(Q'/b) + K/b.\rho(Q'/a)$. This is not of the expected form. However, using the hypothesis that the alphabets for L and K are disjoints, we see that only one of the terms is non-empty. In both cases we have quotients of K or $\{\epsilon\}$ and of L. □

Remark 3.20. If L and K are on the same alphabet and are downward closed, then the function Ψ is not well defined. For example, with $L = \downarrow \texttt{ab} + \downarrow \texttt{ba}$ and $K = \downarrow \texttt{bbc}$, we have that $L^{a \leftarrow K}/b = \downarrow \texttt{bcb} + \downarrow \texttt{bbc}$.

Remark 3.21. Let L, K be downward closed languages, if R quotient of $\rho(L)$ is non-empty, then for all pairs $P, Q \in (\mathcal{R}(K) \cup \{\{\epsilon\}\}) \times \mathcal{R}(L)$ such that $R = P \cdot \rho(Q)$ we have $P \neq \emptyset$ and $Q \neq \emptyset$.

Theorem 3.22. *Let L, K be downward closed languages based on disjoints alphabets. Then $\kappa\left(L^{a\leftarrow K}\right) \leq \kappa(L)\kappa(K)$.*

Proof. Using Proposition 3.19 we know that the function Ψ is well defined when L, K are downward closed languages on disjoints alphabets.

Let us show that Ψ is injective. Take any two quotients $R \neq R'$ of $\rho(L)$, and assume $\Psi(R) = \Psi(R') = (P, Q)$. Then $R = P \cdot \rho(Q) = R'$ which is absurd. Thus Ψ is injective.

The injectivity of Ψ gives the upper bound $\kappa(\rho(L)) \leq |(\mathcal{R}(K) \cup \{\{\epsilon\}\}) \times \mathcal{R}(L)| \leq (\kappa(K)+1)(\kappa(L))$. However, if $L \neq \Sigma(L)^*$ and $K \neq \Sigma(K)^*$, then they both have an empty quotients and the Remark 3.21 applies. Hence all quotients of $\rho(L)$ have their image by Ψ in a set having $(\kappa(K))(\kappa(L)-1)+1$ elements. Thus $\kappa(L^{a\leftarrow K}) \leq (\kappa(L)-1)\kappa(K)+1$.

We now need to deal with the cases where we do not have $\emptyset$ as a quotient in one of the languages. Those cases are easier but the methods are different to reach the upper bounds.

- If $L = \Sigma(L)^*$, $\rho(L) = (\Sigma(L) \setminus \{a\} \cup \Sigma(K))^*$, which is equal to Σ'^* on the result alphabet $\Sigma' = \Sigma(L) \setminus \{a\} \cup \Sigma(K)$. Thus, in this case $\kappa(L^{a\leftarrow K}) = 1 \leq \kappa(L)\kappa(K)$.
- If $K = \Sigma(K)^*$, using Proposition 3.18 when $b \in \Sigma(K)$ gives $\rho(L)/b = K/b \cdot \rho(L/a) = K \cdot \rho(L/a)$, and if $c \notin \Sigma(K)$, $\rho(L)/bc = (K \cdot \rho(L/a))/c = \rho(L/a)/c = \rho(L/ac)$. Thus a word $w \in (\Sigma(L) \cup \Sigma(K))^*$ can be factorized as $w = k_1 \cdot l_1 \cdots k_n \cdot l_n$ with $k_i \in \Sigma(K)^*$ and $l_i \in \Sigma(L)^*$. This gives $\boxed{\rho(L)/w = \rho(L/(al_1 \cdots al_n))}$ if $l_n \neq \epsilon$ and $\boxed{\rho(L)/w = K \cdot \rho(L/(al_1 \cdots l_{n-1}a))}$. Thus $\kappa(\rho(L)) \leq \kappa(L) = \kappa(L)\kappa(K)$.

Thus, in all those cases, $\kappa(L^{a\leftarrow K}) \leq \kappa(L)\kappa(K)$. □

Remark 3.23. One can find in Proposition B.1 the construction of the automaton for $\rho(L)$ based on one for L and one for K. This can give an insight, a more visual argument for the definition of Ψ.

The following corollary is the extension of Theorem 3.22 to a regular substitution.

Corollary 3.24. *Let L and $(K_{a_i})_{a_i \in \Sigma}$ downward closed languages such that all $|\Sigma|+1$ languages have pairwise disjoints alphabets, then we have that*

$$\kappa(L^{a_1\leftarrow K_{a_1},\ldots,a_n\leftarrow K_{a_n}}) \leq \kappa(L) \prod_{1\leq i\leq n} \kappa(K_{a_i})$$

Proof. As the alphabets are pairwise disjoints, we can decompose the substitution as a composition of singular substitutions : $L^{a_1\leftarrow K_{a_1},\ldots,a_n\leftarrow K_{a_n}} = (\ldots(L^{a_1\leftarrow K_{a_1}})^{a_2\leftarrow K_{a_2}}\ldots)^{a_n\leftarrow K_{a_n}}$. Using the iteration of the bound from Theorem 3.22 we have that $\kappa(L^{a_1\leftarrow K_{a_1},\ldots,a_n\leftarrow K_{a_n}}) \leq \kappa(L)\prod_{1\leq i\leq n}\kappa(K_{a_i})$. □

This result gives an exponential bound for the substitution, which hints that even under the restrictive hypothesis of pairwise disjoint alphabets it seems hard to do better than exponential. We proved in Proposition 3.5 an example that reached this bound

without this hypothesis, it could be interesting to try to reach it with the hypothesis on the alphabets for arbitrary $\kappa(K_i)$.

Furthermore, while researching on the singular substitution we could not find a family of downward closed languages $(L_i, K_i)_i$ whose substitutions $(\rho_i(L_i))$ had an exponentially growing state complexity. We thus make the following conjecture :

Conjecture 3.25. If L and K are downward closed languages, then $\kappa(L^{a\leftarrow K}) \leq \kappa(L)\kappa(K)$.

In order to prove Conjecture 3.25 we need to avoid any assumption on the alphabets. Let us first prove that if L is a product of atoms then the quadratic bound holds.

Lemma 3.26. *Let I be a product and K a downward closed language, then any quotient of $I^{a\leftarrow K}$ can be written as $P_K.\rho(P_I)$ where P_K is either a quotient of K or $\{\epsilon\}$ and P_I is a quotient of I.*

Proof. Let R be a quotient of $I^{a\leftarrow K}$, $R = I^{a\leftarrow K}/x$. Let us proceed by induction on the length of x.

If $|x| = 0$, $R = I^{a\leftarrow K} = \{\epsilon\}.\rho(I)$ which verifies the claim.

If $x = x'b$, we write $R' = I^{a\leftarrow K}/x' = P'_K.\rho(P'_I)$. We have that $R = R'/b = P'_K/b.\rho(P'_I) + \rho(P'_I)/b$ as P'_K/b is either empty or downward closed thus containing ϵ. If $P'_K/b \neq \emptyset$, we have $R = P'_K/b.\rho(P'_I)$ as $\rho(P'_I)/b \subseteq \rho(P'_I)$.

Now if $P'_K/b = \emptyset$, $R = \rho(P'_I)/b = \rho(P'_I/b) + K/b.\rho(P'_I/a)$. Now as showed in Lemma 3.15, there are I_1, I_2 suffix products of the product of P'_I such that $P'_I/b = I_1$ and $P'_I/a = I_2$. And as showed in Corollary 3.16, one of the quotient contains another.

- If $I_1 \subseteq I_2$, then $\rho(I_1) \subseteq K/b.\rho(I_2)$, thus $R = K/b.\rho(P'_I/a)$.
- If $I_2 \subseteq I_1$ and $I_2 \neq I_1$, then we have that $I_2 = I_1/a$. In fact we have that $I_1 = \prod_{j_1\leq i} \alpha_i$ and $I_2 = \prod_{j_2\leq i} \alpha_i$ with $j_1 < j_2$. And the property is $\alpha_{j_2} = \min_i(a \in \alpha_i)$ thus as $j_1 < j_2$, we have that $\alpha_{j_2} = \min_{j_1\leq i}(a \in \alpha_i)$. Hence, $I_1/a = I_2$. Now $\rho(I_1)/b = \rho(I_1/b) + K/b.\rho(I_1/a) = \rho(I_1/b) + K/b.\rho(I_2)$. Thus $K/b.\rho(I_2) \subseteq \rho(I_1)/b \subseteq \rho(I_1)$. Thus, $R = \rho(I_1) = \{\epsilon\}.\rho(P'_I/b)$. □

Theorem 3.27. *Let I be a product of atoms and K a downward closed language, then $\kappa(I^{a\leftarrow K}) \leq \kappa(K)\kappa(I)$.*

Proof. If we assume that I, K are not Σ^*, using Lemma 3.26 we have that $R = P_K.\rho(P_I) = \emptyset$ iff $P_K = \emptyset$ or $P_I = \emptyset$. Thus, $\kappa(I^{a\leftarrow K}) \leq \kappa(K)(\kappa(I) - 1) + 1 \leq \kappa(K)\kappa(I)$.

Let us now consider the case where I or K is Σ^*.

- If $I = \Sigma(I)^*$ then $I^{a\leftarrow K} = (\Sigma(I)\setminus\{a\}\cup\Sigma(K))^*$. Hence, $\kappa(I^{a\leftarrow K}) = 1 \leq \kappa(I)\kappa(K)$.
- If $K = \Sigma(K)^*$ then if $I/a = \emptyset$, $\kappa(I^{a\leftarrow K}) = \kappa(I)$. Otherwise we are just replacing the a in the atoms by $\Sigma(K)$. Hence $\kappa(I^{a\leftarrow K}) \leq \kappa(I) = \kappa(I)\kappa(K)$. □

3.1 Singular substitution when $K = \downarrow(\Sigma(L))$

As showed in Remark 3.20, there exists singular substitutions having quotients that cannot be written as products of quotients. To understand their apparition, we can study what happens when $K = \downarrow(\Sigma(L))$.

Lemma 3.28. *Let $I = \alpha_0 I'$ be a product that does not contain atoms of the type B^* with $a \in B$, and ρ the substitution of a by $K = \Sigma(I) \cup \epsilon$.*

Then $\mathcal{R}(I) = \{\alpha_0 I'\} \cup \mathcal{R}(I')$. And $\mathcal{R}(\rho(I)) = \{\rho(\alpha_0)\rho(I')\} \cup \mathcal{R}(\rho(I'))$.

Proof. Let us write $I = \prod_{0 \leq i \leq n} \alpha_i$. Then $\mathcal{R}(I) = \{\prod_{j \leq i \leq n} \alpha_i | 0 \leq j \leq n\}$ as the quotients of a product are its suffix products as proved in Lemma 3.15. Thus, $\mathcal{R}(I) = \{\alpha_0 I'\} \cup \mathcal{R}(I')$.

Let us abuse notations and denote by $\mathcal{R}(L) \cdot K$ the set $\{P \cdot K \mid P \in \mathcal{R}(L)\}$. Let us now prove $\mathcal{R}(\rho(I)) = \mathcal{R}(\rho(\alpha_0))\rho(I') \cup \mathcal{R}(\rho(I'))$. Let us proceed by double inclusion.

Let R be a quotient in $\mathcal{R}(\rho(I))$, then $R = \rho(I)/x$ with $x \in \Sigma^*$. Let x' be the longest prefix of x in $\rho(\alpha_0)$, such that $x = x'y$. If $x \neq x'$, $R = \rho(I')/y \in \mathcal{R}(I')$. Otherwise $R = \rho(\alpha_0)/x\ \rho(I') \in \mathcal{R}(\rho(\alpha_0))\rho(I')$.

For the other direction, let $R \in \mathcal{R}(\rho(\alpha_0))\rho(I')$, we have $R = \rho(\alpha_0)/x\ \rho(I')$. Using the proof above, $R = \rho(I)/x \in \mathcal{R}(\rho(I))$. Then, as ρ preserves downward closedness, $\mathcal{R}(\rho(I')) \subseteq \mathcal{R}(\rho(I))$.

Now, let us prove that in our case, $\mathcal{R}(\rho(\alpha_0))\rho(I') \cup \mathcal{R}(\rho(I')) = \rho(\alpha_0)\rho(I') \cup \mathcal{R}(\rho(I'))$.

- If $\alpha_0 = (b+\epsilon)$ with $b \neq a$, then $\mathcal{R}(\rho(\alpha_0))\rho(I') \cup \mathcal{R}(\rho(I')) = \{\emptyset, \rho(I'), (b+\epsilon)\rho(I')\} \cup \mathcal{R}(\rho(I')) = \{(b+\epsilon)\rho(I')\} \cup \mathcal{R}(\rho(I')) = \rho(\alpha_0)\rho(I') \cup \mathcal{R}(\rho(I'))$ as $\emptyset$ and $\rho(I')$ are already in $\mathcal{R}(\rho(I'))$ if $\rho(I') \neq \Sigma^*$, which is verified as we do not have atoms of the form B^* with $a \in B$.

- If $\alpha_0 = (a + \epsilon)$. Then $\mathcal{R}(\rho(\alpha_0))\rho(I') \cup \mathcal{R}(\rho(I')) = \{\emptyset, \rho(I'), K\rho(I')\} \cup \mathcal{R}(\rho(I')) = \{K\rho(I')\} \cup \mathcal{R}(\rho(I')) = \rho(\alpha_0)\rho(I') \cup \mathcal{R}(\rho(I'))$.
- If $\alpha_0 = B^*$ with $a \notin B$, then $\mathcal{R}(\rho(\alpha_0))\rho(I') \cup \mathcal{R}(\rho(I')) = \{\emptyset, B^*\rho(I')\} \cup \mathcal{R}(\rho(I')) = \{B^*\rho(I')\} \cup \mathcal{R}(\rho(I')) = \rho(\alpha_0)\rho(I') \cup \mathcal{R}(\rho(I'))$. □

Proposition 3.29. *Let I be a product that does not have atoms B^* where $a \in B$, and $K = \Sigma(I) \cup \{\epsilon\}$, then $\mathcal{R}(\rho(I)) = \rho(\mathcal{R}(I))$, meaning that ρ and $\mathcal{R}$ commute.*

Proof. As K and I are based on the same alphabet, let us write $\Sigma = \Sigma(I) = \Sigma(K)$. First let us show that the proposition is true for atoms α.

- Let $\alpha = (b + \epsilon)$, when $b \neq a$ or $\alpha = B^*$ when $a \notin B$, then as ρ is the identity on those atoms, the claim ensues. To prove this, we can observe that if an atom does not contain the letter a then its quotients will not contain it either, thus ρ will be the identity on the quotients also.
- Finally, when $\alpha = (a + \epsilon)$, $\mathcal{R}(\rho(a + \epsilon)) = \mathcal{R}(K) = \{K, \epsilon, \emptyset\}$.
  And $\rho(\mathcal{R}(a + \epsilon)) = \rho(\{a + \epsilon, \epsilon, \emptyset\}) = \{K, \epsilon, \emptyset\}$.

Hence, the commutation works on those atoms.

Now let us consider products and prove the proposition by induction on the number of atoms in a product :

If I is the empty product then $\rho(\mathcal{R}(I)) = \emptyset = \mathcal{R}(\rho(I))$.

If $I = \alpha I'$, then we have :

$$\begin{aligned}
\mathcal{R}(\rho(\alpha I')) &= \rho(\alpha)\rho(I') \cup \mathcal{R}(\rho(I')) && \text{using Lemma 3.28}\\
&= \rho(\alpha I') \cup \mathcal{R}(\rho(I')) && \text{using the definition of } \rho\\
&= \rho(\alpha I') \cup \rho(\mathcal{R}(I')) && \text{by induction hypothesis}\\
&= \rho(\alpha I' \cup \mathcal{R}(I')) && \text{using the properties of } \rho\\
&= \rho(\mathcal{R}(\alpha I)) && \text{using Lemma 3.28} \quad \square
\end{aligned}$$

Remark 3.30. The equality $\rho(I/x) = \rho(I)/x$ is false in general, the equality from Proposition 3.29 only happens when considering all the quotients together. For example take $I = \downarrow(\mathtt{abc})$. We have $\rho(I/b) = \downarrow(c)$ and $\rho(I)/b = \downarrow(bc)$. However

$$\mathcal{R}(\rho(I)) = \{\downarrow((\mathtt{a}+\mathtt{b}+\mathtt{c})\mathtt{bc}), \downarrow(\mathtt{bc}), \downarrow(\mathtt{c}), \epsilon, \emptyset\} = \{\rho(\downarrow(\mathtt{abc})), \downarrow(\mathtt{bc}), \downarrow(\mathtt{c}), \epsilon, \emptyset\} = \rho(\mathcal{R}(I)).$$

Remark 3.31. Even if the commutation is not true in the general case, what we really are interested in is the cardinality. Hence, it would be interesting to know if the following inequality holds : let L be a downward closed language, and $K = \Sigma(L) \cup \{\epsilon\}$, then $|\mathcal{R}(\rho(L))| \leq |\rho(\mathcal{R}(L))|$.

In the case where L is a product and $K = \downarrow(\Sigma(L))$ we are under the assumptions of Theorem 3.27, but we can show that we do not increase the state complexity by another way that follow what we did in this section.

Proposition 3.32. *Let $w \in \Sigma^*$, then $\kappa\left(\downarrow(w)^{a\leftarrow\Sigma(w)}\right) \leq \kappa(\downarrow(w))$.*

Proof. Let us show that every quotient R of $\downarrow(w)^{a\leftarrow\Sigma(w)}$ can be written as $\rho(P)$ with P quotient of $\downarrow(w)$.

Let R be a quotient of $\downarrow(w)^{a\leftarrow\Sigma(w)}$, there exists $x \in \Sigma^*$ such that $R = \downarrow(w)^{a\leftarrow\Sigma(w)}/x$. Let us proceed by induction on the length of x. If $x = \epsilon$, $R = \downarrow(w)^{a\leftarrow\Sigma(w)} = \rho(\downarrow(w))$.

If $x = x'b$ then let $R' = \downarrow(w)^{a\leftarrow\Sigma(w)}/x'$. We have that $R' = \rho(P')$ by induction. Thus $R = R'/b = \rho(P')/b$. Using Proposition 3.18, we get that $R = \rho(P'/b) + \epsilon.\rho(P'/a)$. However as P' is a quotient of $\downarrow(w)$ it can be written as $P' = \downarrow(v)$ with v a suffix of w. Thus if b appears before a in v then $\rho(P'/a) \subseteq \rho(P'/b)$, else $\rho(P'/b) \subseteq \rho(P'/a)$. Thus there exists P a quotient of $\downarrow(w)$ such that $R = \rho(P)$.

Finally, we also have to consider the empty quotient, $\emptyset = \rho(\emptyset)$.

Hence $\kappa(\downarrow(w)^{a\leftarrow\Sigma(w)}) \leq \kappa(\downarrow(w))$. $\square$

4 Conclusion

In this study we considered the state complexity of the n-th root and substitution operations when restricted to upward closed and downward closed languages. These questions have not yet (to the best of our knowledge) been addressed. This can be seen as trying to better understand these two subregular classes of languages. We found that in the case of upward closed languages, the root operations still have an exponential lower bound. We did not conclude for downward closed languages, but conjecture that if the language is made of subwords of a word, then the state complexity will be quadratic for the square root.

We also studied the state complexity of the substitution in the case of downward closed languages. We showed an exponential upper bound in the general case, and when restricting to singular substitutions, we proved a quadratic upper bound when the two languages are on disjoint alphabets and conjecture a quadratic bound in the general case of two downward closed languages. Furthermore, when the language we make the substitution on is directed, we showed a quadratic upper bound. It might be interesting in future studies to answer the conjecture on the singular substitution of downward closed languages and define/study the substitution for upward closed languages.

This works takes part in the more general goal of better understanding upward closed and downward closed languages that appear in practical applications such as automatic verification based on well-quasi-orderings. In this way, it is important to understand better how these classes of languages are structured and how they behave to come up with better data structures allowing to manipulate them more efficiently.

I warmly thank Philippe Schnoebelen again for his advice, feed-backs, suggestions and help in general.

References

1. Abdulla, P.A., Cerāns, K., Jonsson, B., Tsay, Y.K.: Algorithmic analysis of programs with well quasi-ordered domains. Information & Computation **160**, 109–127 (2000)
2. Abdulla, P.A., Collomb-Annichini, A., Bouajjani, A., Jonsson, B.: Using forward reachability analysis for verification of lossy channel systems. Formal Methods in System Design **25**(1), 39–65 (2004)
3. Bachmeier, G., Luttenberger, M., Schlund, M.: Finite automata for the sub- and superword closure of CFLs: Descriptional and computational complexity. In: Proc. 9th Int. Conf. Language and Automata Theory and Applications (LATA 2015). pp. 473–485. Springer (2015)
4. Bassino, F., Giambruno, L., Nicaud, C.: Complexity of operations on cofinite languages. In: Proc. 9th Latin American Symp. Theoretical Informatics (LATIN 2010). Lecture Notes in Computer Science, vol. 6034, pp. 222–233. Springer (2010)
5. Bertrand, N., Schnoebelen, Ph.: Computable fixpoints in well-structured symbolic model checking. Formal Methods in System Design **43**(2), 233–267 (2013)
6. Brzozowski, J., Li, B.: Syntactic complexity of R- and J-trivial regular languages. Int. J. Foundations of Computer Science **25**(07), 807–821 (2014)

7. Brzozowski, J.: Quotient complexity of regular languages. Journal of Automata, Languages and Combinatorics **15**(1–2), 71–89 (2010)
8. Brzozowski, J., Jirásková, G., Li, B.: Quotient complexity of ideal languages. In: Proc. 9th Latin American Symp. Theoretical Informatics (LATIN 2010). pp. 208–221. Springer (2010)
9. Brzozowski, J., Jirásková, G., Zou, C.: Quotient complexity of closed languages. Theory of Computing Systems **54**(2), 277–292 (2014)
10. Brzozowski, J., Jirásková, G., Li, B.: Quotient complexity of ideal languages. Theoretical Computer Science **470**, 36–52 (2013)
11. Brzozowski, J.A., Li, B., Liu, D.: Syntactic complexities of six classes of star-free languages. Journal of Automata, Languages and Combinatorics **17**(2–4), 83–105 (2012)
12. Câmpeanu, C., Culik, K., Salomaa, K., Yu, S.: State complexity of basic operations on finite languages. In: Proc. 4th Int. Workshop Implementing Automata (WIA '99). Lecture Notes in Computer Science, vol. 2214, pp. 60–70. Springer (2001)
13. Finkel, A., Schnoebelen, Ph.: Well-structured transition systems everywhere! Theoretical Computer Science **256**(1–2), 63–92 (2001)
14. Gao, Y., Moreira, N., Reis, R., Yu, S.: A survey on operational state complexity. Journal of Automata, Languages and Combinatorics **21**(4), 251–310 (2017)
15. Goubault-Larrecq, J., Halfon, S., Karandikar, P., Narayan Kumar, K., Schnoebelen, Ph.: The ideal approach to computing closed subsets in well-quasi-orderings. In: Well Quasi-Orders in Computation, Logic, Language and Reasoning, Trends in Logic, vol. 53, chap. 3, pp. 55–105. Springer (2020)
16. Gruber, H., Holzer, M., Kutrib, M.: The size of Higman-Haines sets. Theoretical Computer Science **387**(2), 167–176 (2007)
17. Gruber, H., Holzer, M., Kutrib, M.: More on the size of Higman-Haines sets: Effective constructions. Fundamenta Informaticae **91**(1), 105–121 (2009)
18. Héam, P.C.: On shuffle ideals. RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications **36**(4), 359–384 (2002)
19. Heinz, J.: On the role of locality in learning stress patterns. Phonology **26**(2), 303–351 (2009)
20. Hospodár, M.: Descriptional complexity of power and positive closure on convex languages. In: Proc. 24th Int. Conf. Implementation and Application of Automata (CIAA 2019). pp. 158–170. Springer (2019)
21. Karandikar, P., Niewerth, M., Schnoebelen, Ph.: On the state complexity of closures and interiors of regular languages with subwords and superwords. Theoretical Computer Science **610**, 91–107 (2016)
22. Krawetz, B., Lawrence, J., Shallit, J.: State complexity and the monoid of transformations of a finite set. Int. J. Foundations of Computer Science **16**(3), 547–563 (2005)
23. Okhotin, A.: On the state complexity of scattered substrings and superstrings. Fundamenta Informaticae **99**(3), 325–338 (2010)
24. Sakarovitch, J., Simon, I.: Subwords. In: Lothaire, M. (ed.) Combinatorics on Words, Encyclopedia of Mathematics and Its Applications, vol. 17, chap. 6, pp. 105–142. Cambridge Univ. Press (1983)

A Missing proofs on root operators

An extremely useful tool to obtain lower bounds on the state complexity of languages is the use of dividing sets.

Definition A.1. *Let $L \subseteq \Sigma^*$, then $\mathcal{F} = \{x_1, ..., x_n\}$ is a dividing set for L if and only if for all $x_i \neq x_j$, there exists $z_{i,j} \in \Sigma^*$ such that $x_i z_{i,j} \in L$ and $x_j z_{i,j} \notin L$ or $x_i z_{i,j} \notin L$ and $x_j z_{i,j} \in L$.*

The following fact explains why those dividing sets are useful when studying the state complexity of a language.

Fact A.2 *Let $\mathcal{F}$ be a dividing set of L. Then any DFA recognizing L must have at least $|\mathcal{F}|$ states.*

Proposition A.3. *Let v, w be two words with different alphabets: $\Sigma(v) \neq \Sigma(w)$. Then $\sqrt{\uparrow(V_n)}/v \neq \sqrt{\uparrow(V_n)}/w$.*

Proof. As v and w are based on different alphabets, there is a letter of V_n that is either in v and not in w or in w and not in v. Without loss of generality, let us assume the letter a is in v but not in w.

Then let x be V_n with a removed. Thus, V_n is a subword of $(vx)^k$ but not of $(wx)^k$. Thus, $vx \in \sqrt[k]{\uparrow(V_n)}$ but $wx \notin \sqrt[k]{\uparrow(V_n)}$. Hence, $\sqrt[k]{\uparrow(V_n)}/v \neq \sqrt[k]{\uparrow(V_n)}/w$ since one contains x but not the other. □

Proposition A.4. *Let $\mathcal{F}_n$ be a dividing set of maximal size for $\sqrt{\uparrow(V_n)}$.*
Then $\mathcal{F}_n \bigcup \mathcal{F}_n a_{n+1} \bigcup a_{n+1}(\mathcal{F}_{n-1} \setminus \{V_{n-1}\})a_n$ is a dividing set for $\sqrt{\uparrow(V_{n+1})}$.

Proof. Let u, v in $\mathcal{F}_n \bigcup \mathcal{F}_n a_{n+1}$, If they do not come from the same summand, then their alphabet is not the same as one uses a_{n+1} while the other does not. Thus by Proposition A.3, their quotients are different.

If they come from the same summand, first let us assume they come from $\mathcal{F}_n$. Then there is a word $x \in \Sigma^*$ such that $ux \in L_n$ and $vx \notin L_n$. Thus V_n is a subword of $uxux$ but not of $vxvx$, hence u_{n+1} is a subword of $uxa_{n+1}uxa_{n+1}$ and not of $vxa_{n+1}vxa_{n+1}$. Thus, $L_{n+1}/u \neq L_{n+1}/v$.

If they both are from $\mathcal{F}_n a_{n+1}$. Let's write $u = u'a_{n+1}$ and $v = v'a_{n+1}$. There is $x' \in \Sigma^*$ such that V_n is subword of $u'x'u'x'$ but not of $v'x'v'x'$ or the other way around. Thus u_{n+1} is a subword of $u'x'a_{n+1}u'x'a_{n+1}$ but not of $v'x'a_{n+1}v'x'a_{n+1}$ as V_n is not a subword of it and V_n is a subword of V_{n+1}.

Let us consider $x \neq y \in a_{n+1}(\mathcal{F}_{n-1}\setminus\{V_{n-1}\})a_n$. Let us write x and y as $x = a_{n+1}x''a_n$ and $y = a_{n+1}y''a_n$. As $x'', y'' \in \mathcal{F}_{n-1}$, there exists $z \in \Sigma^*$ such that $x''zx''z \in \uparrow(V_{n-1})$ and $y''zy''z \notin \uparrow(V_{n-1})$. Thus $V_{n+1} \preccurlyeq a_{n+1}x''a_nza_na_{n+1}a_{n+1}x''a_nza_na_{n+1}$ but V_{n+1} is not a subword of $a_{n+1}y''a_nza_na_{n+1}a_{n+1}y''a_nza_na_{n+1}$. Thus, x and y give different quotients.

If $x \in a_{n+1}(\mathcal{F}_{n-1} \setminus \{V_{n-1}\})a_n$ and $y \in \mathcal{F}_n$, as x contains a_{n+1} but not y, the proof of Proposition 2.3 entails that their quotients are different.

If $x \in a_{n+1}(\mathcal{F}_{n-1} \setminus \{V_{n-1}\})a_n$ and $y \in \mathcal{F}_n a_{n+1}$, let us write $x = a_{n+1}x''a_n$ and $y = y'a_{n+1}$.

As $x''a_n \in \mathcal{F}_n$, if $x''a_n \neq y'$, there exists $z \in \Sigma^*$ such that $x'zx'z \in \uparrow(V_n)$ and $y'zy'z \notin \uparrow(V_n)$ where $x' = x''a_n$, as $x' \neq y'$. Thus, $V_{n+1} \preccurlyeq a_{n+1}x'za_{n+1}a_{n+1}x'za_{n+1} = (xza_{n+1})^2$ but V_{n+1} is not a subword of $y'a_{n+1}za_{n+1}y'a_{n+1}za_{n+1} = (yza_{n+1})^2$. Thus x and y give different quotients.

If $x' = y'$, then $x = a_{n+1}x''a_n$, the nice thing is that by taking $z = V_{n-1}$ we have $V_n \preccurlyeq x''a_nzx''a_n = x'zx'$ which gives $V_{n+1} \preccurlyeq x''a_na_{n+1}zx''a_na_{n+1} = yzy$ but V_{n+1} is not a subword of $xzx = a_{n+1}x''a_nza_{n+1}x''a_n$, as long as $x'' \neq V_{n-1}$. Thus x, y give different quotients.

Thus, $a_{n+1}(\mathcal{F}_{n-1} \setminus V_{n-1})a_n \bigcup \mathcal{F}_n \bigcup \mathcal{F}_n a_{n+1}$ is a dividing set for $\sqrt{\uparrow(V_{n+1})}$. □

Proposition A.5. *For $n \geq 1$, $\kappa(\sqrt{\uparrow(V_n)}) \geq (\frac{1}{\sqrt{2}} - \frac{1}{4})(\sqrt{2}+1)^n \approx 0.46 \times 2.41^n$.*

Proof. Using Proposition A.4 we have that $a_{n+1}(\mathcal{F}_{n-1} \setminus V_{n-1})a_n \bigcup \mathcal{F}_n \bigcup \mathcal{F}_n a_{n+1}$ is a dividing set for $\sqrt{\uparrow(V_{n+1})}$. Thus for all $n \in \mathbb{N}_{>0}$ we can create dividing sets $\mathcal{F}_n$ of $\sqrt{\uparrow(V_n)}$ verifying $|\mathcal{F}_{n+2}| \geq 2|\mathcal{F}_{n+1}| + |\mathcal{F}_n| - 1$.

Let $v_n = |\mathcal{F}_n| - \frac{1}{2}$, we have the equation : $v_{n+2} \geq 2v_{n+1} + v_n$. Thus $v_n \geq w_n$ where $w_0 = v_0 = -\frac{1}{2}$, $w_1 = v_1 = \frac{3}{2}$ and $w_{n+2} = 2w_{n+1} + w_n$. And using the formula for recurrent sequences of order 2, we get $w_n = (\frac{1}{\sqrt{2}} - \frac{1}{4})(\sqrt{2}+1)^n - (\frac{1}{\sqrt{2}} + \frac{1}{4})(1-\sqrt{2})^n$. Thus asymptotically, $w_n \sim (\frac{1}{\sqrt{2}} - \frac{1}{4})(\sqrt{2}+1)^n$ and $|\mathcal{F}_n| \geq 0.46(2.41)^n$. In fact, by computing the first values and then using the asymptotic analysis, we get $|\mathcal{F}_n| \geq 0.46(2.41)^n$ for $n \geq 1$. □

Definition A.6. *For $w \in \Sigma^*$ we denote with $CS(w)$ the set $\bigcup_{w=tv} t \sqcup\!\sqcup v$. This is the set of words obtained from w by one "cut and shuffle" move.*

Proposition A.7. *Let V_n be a n-letter word of length n, $CS(V_n) = \{$minimal words of $\sqrt{\uparrow(V_n)}\}$ and $\uparrow(CS(V_n)) = \sqrt{\uparrow(V_n)}$.*

Proof. Let u be a word that can be obtained by a cut and shuffle of V_n, $u \in t \sqcup\!\sqcup v$ where $V_n = tv$.

Let us take $r = vt$, we have

$$u^2 \in (t \sqcup\!\sqcup v)(v \sqcup\!\sqcup t) \subseteq V_n \sqcup\!\sqcup r \subseteq \uparrow(V_n)\,.$$

Hence, $u \in \sqrt{\uparrow(V_n)}$, and as $\sqrt{\uparrow(V_n)}$ is upward closed, $\uparrow(CS(V_n)) \subseteq \sqrt{\uparrow(V_n)}$.

Let $u \in \sqrt{\uparrow(V_n)}$ such that $|u| = n$, which implies that u is minimal in $\sqrt{\uparrow(V_n)}$. We have $u^2 \in \uparrow(V_n)$. Let us call u_1 the longest prefix of V_n that is a subword of u. Then as V_n us a subword of u^2, we have that the suffix u_2 such that $V_n = u_1u_2$ is a subword of u too. Finally, as $|u| = |V_n| = |u_1| + |u_2|$ and u_1, u_2 have different letters, we have that $u \in u_1 \sqcup\!\sqcup u_2$.

Hence $u \in CS(V_n)$, and since any upward closed L is $\bigcup_{u \text{ minimal in } L} \uparrow(u)$, we get $\uparrow(CS(V_n)) = \sqrt{\uparrow(V_n)}$. □

Proposition A.8. *Let $V_n = a_1a_2 \cdots a_n$ be word of length n where all letters are different. Then $|CS(V_n)| = 2^n - n$.*

Proof. It is quite easy to see that for $w = tv$ there are $\binom{|w|}{|t|}$ different words in $t \sqcup\!\sqcup v$, but one of them is w. Thus $|CS(V_n)| = \sum_{0 \leq |t| \leq n-1} \binom{n}{|t|} - (n-1) = 2^n - n$. □

B Missing proofs on the substitution operator

Proposition B.1. *The substitution operator preserves regularity.*

Proof. We know this since we proved it preserves the closedness and upward closed/downward closed languages are regular languages, but this is an occasion to explain the classical construction of the automaton for the substitution, which might help to visualize better the situation.

Let us consider the classical construction for the substitution. Let us call $\mathcal{A}_L, \mathcal{A}_K$ the automata for L, K. We modify $\mathcal{A}_L$ in the following way: For every transition $q \xrightarrow{a} q'$ in $\mathcal{A}_L$ that reads the letter a, we insert a copy of $\mathcal{A}_K$ without final states, create an ϵ-transition from q to the initial state of (that copy of) $\mathcal{A}_K$, and from each final state of $\mathcal{A}_K$ we add an ϵ-transitions to q'. Finally, we remove the original $q \xrightarrow{a} q'$ transition, determinize the resulting automaton, and obtain a DFA for the substitution $L^{a \leftarrow K}$.

In order to prove that the construction is correct, let us call $\mathcal{A}$ the automaton before the determinization, and let us show that $L(\mathcal{A}) = L^{a \leftarrow K}$. Since the determinization process preserves the language, we would have the validity of the automaton.

Let $x \in L^{a \leftarrow K}$, there exists $x_L \in L$ such that $x_L = x_1 a x_2 ... x_{n-1} a x_n$ with $x_i \in \Sigma^*$ and $x = x_1 y 1 x_2 ... x_{n-1} y_{n-1} x_n$ with $y_i \in K$. Let $q_0, ...q_r$ be the path of x_L in $\mathcal{A}_L$. Then by inserting the path for y_i in $\mathcal{A}_K$ where there are transitions labeled a we get a path for x in $\mathcal{A}$ using the ϵ-transitions. Thus $L^{a \leftarrow K} \subseteq L(\mathcal{A})$.

Let $x \in L(\mathcal{A})$ then by studying the path that x follows in the automaton, we factorize x under the form $x = x_1 y 1 x_2 ... x_{n-1} y_{n-1} x_n$ with $x_i \in \Sigma^*$ and $y_i \in K$. Furthermore, the construction implies that $x_L = x_1 a x_2 ... x_{n-1} a x_n \in L$ as we inserted $\mathcal{A}_K$ on edges labeled by a. Thus $L(\mathcal{A}) \subseteq L^{a \leftarrow K}$. □

C Missing proofs on iteration operator

Proposition C.1. *In the case of L upward closed, with state complexity n, the upper bound $k(n-1)+1$ on the state complexity of L^k is tight for $|\Sigma| \geq 1$.*

Proof. First of all the bound $k(n-1)+1$ corresponds to the concatenation upper bound applied k times. Thus is it a valid upper bound for L^k. We now have to show that it is actually tight.

Let $L_n = \mathtt{a}^{n-1}\mathtt{a}^*$ be the language of words having at least $n-1$ $\mathtt{a}$, this gives $L_n^k = \mathtt{a}^{(n-1)k}\mathtt{a}^*$. Let $w \in L_n$, and x a word such that $w \preccurlyeq x$, then x has more $\mathtt{a}$'s than w, implying $x \in L_n$. The quotients of L_n are $\{\mathtt{a}^i\mathtt{a}^* \mid 0 \leq i \leq n-1\}$. Hence, L_n has state complexity n. Finally, using the same idea, L_n^k has state complexity $(n-1)k+1$. □