On the Average Size of Glushkov's Automata

Cyril Nicaud

On the Average Size of Glushkov's Automata

Cyril Nicaud[1]

LIGM, UMR CNRS 8049, Université Paris Est, 77454 Marne-la-Vallée, France
nicaud@univ-mlv.fr

Abstract. Glushkov's algorithm builds an ε-free nondeterministic automaton from a given regular expression. In the worst case, its number of states is linear and its number of transitions is quadratic in the size of the expression. We show in this paper that in average, the number of transitions is linear.

1 Introduction

Kleene's Theorem states that regular expressions and automata describe the same objects, regular languages. They both are finite objects which encode infinite sets of words and they play a key role in formal language theory as well as in its many fields of application.

One can often take advantage of having this two different representations, the algorithmic problems of transforming one representation into another are therefore fundamental. They have been widely studied and are still improved nowadays, as one can see in a recent survey by J. Sakarovitch [1].

In this paper we focus on one particular such transformation, Glushkov's algorithm [2], an algorithm that builds an ε-free nondeterministic automaton from a given regular expression. Starting from a regular expression with n letters, it builds an automaton with $n+1$ states and $\mathcal{O}(n^2)$ transitions. In [3], J. Hromkovič, S. Seibert and T. Wilke proposed a variation on this algorithm where the produced automaton has $\mathcal{O}(n \log^2 n)$ states. They also proved a lower bound of $\Omega(n \log n)$ states for this general problem. Based on this work, C. Hagenah and A. Muscholl proposed in [4] an algorithm of time complexity $\mathcal{O}(n \log^2 n)$ to achieve the construction. See also [5, 6] for some efficient related algorithms.

Our contribution is to give an average case analysis of the size of Glushkov's automata. Using the framework of analytic combinatorics, we prove that for the uniform distribution of regular expressions, the average number of transitions is linear.

The use of generating functions and complex analysis have proved to be useful in average case analysis of algorithms [7]. The methodology we shall use here can be summarized as follows:

1. Find an unambiguous specification of the objects, which can be recursive.
2. Transform the specification into a functional equation for the associated generating function.
3. Analyze the dominant singularities of the generating function to obtain the needed asymptotics.

The paper is organized as follows. In Section 2, we recall the basic definitions and present some analytic tools. In Section 3, we study the generating function associated to regular expressions. Section 4 is devoted to our main result. Finally, a short discussion about the distribution is presented in Section 5.

Note that due to the lack of space, we can not show all the details of the computations, but most of them are easily done with the help of a computer-algebra system.

2 Preliminaries

2.1 Automata and Regular Expressions

An *automaton* defined on a finite alphabet A is a tuple (A, Q, T, I, F) where Q is the finite *set of states*, $T \subset Q \times A \times Q$ is the set of *transitions*, $I \subset Q$ is the set of *initial states* and $F \subset Q$ is the set of *final states*. We refer the reader unfamiliar with basic notions of automata and regular languages to [8] for definitions and fundamental results.

The set of nonempty *regular expressions* $\mathcal{R}$ on a finite alphabet A is a set of words on the alphabet $\{\varepsilon, \cdot, *, \cup, (,)\} \cup A$ defined inductively by: $\varepsilon \in \mathcal{R}$, $A \subset \mathcal{R}$, $(R)* \in \mathcal{R}$ for all $R \in \mathcal{R}$, $(R_1 \cdot R_2)$ and $(R_1 \cup R_2)$ for every $R_1, R_2 \in \mathcal{R}$. A language defined on A is *denoted* by a regular expression of $\mathcal{R}$ when it is exactly the set of words obtained by interpreting each symbol $*$, $\cdot$ or $\cup$ as the corresponding regular operation on sets of words. Let $L(R)$ be the language denoted by $R \in \mathcal{R}$. For convenience, we shall freely remove parenthesis symbols that are not needed in an element of $\mathcal{R}$. It is also often useful to see regular expressions as trees, with this equivalent inductive definition:

$$\begin{cases} \varepsilon \in \mathcal{R} & \\ a \in \mathcal{R} & \forall a \in A \\ \underset{R}{\overset{*}{|}} \in \mathcal{R} & \forall R \in \mathcal{R} \\ \underset{R_1\ R_2}{\overset{\cup}{\wedge}} \in \mathcal{R} & \forall R_1, R_2 \in \mathcal{R} \\ \underset{R_1\ R_2}{\overset{\bullet}{\wedge}} \in \mathcal{R} & \forall R_1, R_2 \in \mathcal{R} \end{cases} \tag{1}$$

The *size* of an element of $\mathcal{R}$ is the number of nodes in its tree representation:

$$|\varepsilon| = |a| = 1; \qquad \left|\underset{R}{\overset{*}{|}}\right| = |R| + 1; \qquad \left|\underset{R_1\ R_2}{\overset{\cup}{\wedge}}\right| = \left|\underset{R_1\ R_2}{\overset{\bullet}{\wedge}}\right| = |R_1| + |R_2| + 1$$

Note that one usually add to $\mathcal{R}$ the symbol $\emptyset$ that denotes the empty language. For technical reasons it is slightly more convenient in this paper to work on nonempty regular languages.

2.2 Glushkov's Automaton

Let m be the number of letter symbols in R, for $R \in \mathcal{R}$. We consider the expression $\tilde{R}$ obtained from R by distinguishing the letters with subscripts in

$\{1, \cdots, m\}$, marking them from left to right on its string representation, or equivalently using depth-first order on its tree representation. For instance $R = b^* \cdot (a \cup b \cdot b)^*$ is changed into $\tilde{R} = b_1^* \cdot (a_2 \cup b_3 \cdot b_4)^*$. We denote by $pos(R)$ the set of subscripted letters in $\tilde{R}$: $pos(R) = \{b_1, a_2, b_3, b_4\}$ in the example. We also denote by ν the function from $pos(R)$ to A that removes the subscripts, $\nu(a_2) = a$ for instance.

Let $First(R)$ and $Last(R)$ be the sets defined by

$$First(R) = \{\alpha \in pos(R) \mid \exists u \in L(\tilde{R}),\ u \text{ starts with the letter } \alpha\}$$
$$Last(R) = \{\alpha \in pos(R) \mid \exists u \in L(\tilde{R}),\ u \text{ ends with the letter } \alpha\}$$

And for any letter α in $pos(R)$, the set $follow(R, \alpha)$ is defined by

$$follow(R, \alpha) = \{\beta \in pos(R) \mid \exists u \in L(\tilde{R}),\ \alpha\beta \text{ is a factor of } u\}$$

The *Glushkov's automaton* of R, also called the *position automaton*, is the automaton $\mathcal{A}_R = (A, Q, T, \{i\}, F)$ with $Q = pos(R) \cup \{i\}$, $F = Last(R) \cup \{i\}$ if $\varepsilon \in L(R)$ and $F = Last(R)$ otherwise, and $T = \{(i, \nu(\alpha), \alpha) \mid \alpha \in First(R)\} \cup \{(\alpha, \nu(\beta), \beta) \mid \beta \in follow(R, \alpha)\}$. This classical construction provides an automaton that recognizes $L(R)$.

Let $Edges(R)$ be the set of pairs $(\alpha, \beta) \in pos(R)^2$ such that $\beta \in follow(\alpha)$. The number of transitions in $\mathcal{A}_R$ is thus $|First(R)| + |Edges(R)|$. The set $Edges(R)$ can also be defined inductively as follows.

$$\begin{cases} Edges(\varepsilon) = 0 \\ Edges(a) = 0 \\ Edges\left(\begin{smallmatrix} * \\ | \\ R \end{smallmatrix}\right) = Edges(R) \cup Last(R) \times First(R) \\ Edges\left(\begin{smallmatrix} & \cup & \\ / & & \backslash \\ R_1 & & R_2 \end{smallmatrix}\right) = Edges(R_1) \cup Edges(R_2) \\ Edges\left(\begin{smallmatrix} & \bullet & \\ / & & \backslash \\ R_1 & & R_2 \end{smallmatrix}\right) = Edges(R_1) \cup Edges(R_2) \cup Last(R_1) \times First(R_2) \end{cases} \tag{2}$$

2.3 Generating Functions

A *combinatorial class* $\mathcal{C}$ is a set of objects with a size function $|\cdot|$ from $\mathcal{C}$ to $\mathbb{N}$ such that, for any $n \in \mathbb{N}$, the number c_n of objects of size n in $\mathcal{C}$ is finite. The generating function $C(z)$ of a combinatorial class $\mathcal{C}$ is the formal power series

$$C(z) = \sum_{C \in \mathcal{C}} z^{|C|} = \sum_{n \geq 0} c_n z^n$$

We also denote by $[z^n]C(z) = c_n$ the coefficient of z^n in $C(z)$.

Let $\mathcal{C}$ be a combinatorial class of generating function $C(z)$ and let $f : \mathcal{C} \to \mathbb{R}$ be a mapping from this class to $\mathbb{R}$. The *cost generating function* $F(z)$ of $\mathcal{C}$ associated to f is

$$F(z) = \sum_{C \in \mathcal{C}} f(C) z^{|C|} = \sum_{n \geq 0} f_n z^n, \text{ with } f_n = \sum_{\substack{C \in \mathcal{C} \\ |C| = n}} f(C)$$

For a given n, the average value of f for the uniform distribution on the elements of size n of $\mathcal{C}$ is therefore

$$\mu_n(\mathcal{C}, f) = \frac{[z^n]F(z)}{[z^n]C(z)}$$

The following lemma, though trivial, will be useful throughout this article.

Lemma 1. *Let $\mathcal{A}$ and $\mathcal{B}$ be two combinatorial classes. Let $f : \mathcal{A} \to \mathbb{R}$ and $g : \mathcal{B} \to \mathbb{R}$ be two mappings from $\mathcal{A}$ and $\mathcal{B}$ to $\mathbb{R}$. The following property holds.*

$$\sum_{A\in\mathcal{A}} \sum_{B\in\mathcal{B}} (f(A) + g(B))z^{|A|}z^{|B|} = F(z)B(z) + A(z)G(z)$$

where $A(z)$, $B(z)$, $F(z)$ and $G(z)$ respectively denote the generating functions of $\mathcal{A}$ and $\mathcal{B}$ and the cost generating functions associated to f and g.

2.4 Symbolic Methods and Transfer Theorem

The *symbolic methods* were introduced in [9]. When they can be applied, they directly and almost automatically build the generating functions associated to combinatorial classes. The idea is to avoid recurrence formulas on the number of objects by providing a dictionary that maps combinatorial constructions on classes into constructions on generating functions. This dictionary covers a lot of used constructions, and one can easily use it to describe combinatorial classes such as trees, permutations, set partitions, integer partitions, random mappings, etc. therefore obtaining directly their generating functions.

In this article we shall only use a small part of this dictionary, the most basic one. If $\mathcal{A} = \mathcal{B} \cup \mathcal{C}$ are combinatorial classes such that $\mathcal{B}$ and $\mathcal{C}$ are disjoint, it is direct to prove that the associated generating functions $A(z)$, $B(z)$ and $C(z)$ satisfy $A(z) = B(z) + C(z)$. Moreover if $\mathcal{A} = \mathcal{B} \times \mathcal{C}$, one can see that $A(z) = B(z)C(z)$. We shall only need this two constructions here, but in this framework, one can directly derive the generating functions of sequences of elements in $\mathcal{A}$, sets of elements in $\mathcal{A}$, and so on. We refer the reader to P. Flajolet and R. Sedgewick's book for more informations about this topic [7].

Once the generating function is known, either in close or implicit form, several theorems exist to compute asymptotic estimations of its coefficients. This theorems mainly use the theory of complex analysis, seeing generating functions as analytic functions from $\mathbb{C}$ to $\mathbb{C}$. The main idea is that the asymptotics of the coefficients of a generating function can be obtained by studying it around its dominant singularities (its singularities of smallest moduli). Informally, given a generating function $A(z)$ of unique dominant singularity ρ, the transfer theorem states under some analytic conditions that if $A(z) \sim B(z)$ as $z \to \rho$, then $[z^n]A(z) \sim [z^n]B(z)$, for some useful functions $B(z)$ whose coefficient asymptotics are well-known.

We use the notations of [7] to give a formal description of the theorem. Let $R > 1$ and $0 < \phi < \pi/2$ be two real numbers, the domain $\Delta(\phi, R)$ is

$$\Delta(\phi, R) = \{z \in \mathbb{C} \mid |z| < R,\ z \neq 1 \text{ and } |Arg(z-1)| > \phi\}$$

A domain is a Δ-*domain at* 1 if it is a $\Delta(\phi, R)$ for some R and ϕ. For a given complex number $\zeta \neq 0$, a Δ-*domain at* ζ is the image by the mapping $z \mapsto \zeta z$ of a Δ-domain at 1. A function is Δ-*analytic* if it is analytic in some Δ-domain.

Theorem 1 (part of Transfer Theorem). *Let α and β be real number and let $f(z)$ be a function that is Δ-analytic that satisfies, on the intersection of a neighborhood of* 1 *and its Δ-domain, the condition*

$$f(z) = o\left((1-z)^{-\alpha}\left(\log \frac{1}{1-z}\right)^{\beta}\right)$$

then $[z^n]f(z) = o(n^{\alpha-1}(\log n)^{\beta})$.

Recall that Pringsheim's Theorem states that if $f(z)$ is representable at the origin by a series expansion with nonnegative coefficients, one of its dominant singularities, if any, is on $\mathbb{R}^+$. This theorem is useful as generating functions always have nonnegative coefficients.

2.5 Analytic Tools for This Paper

The generating functions we shall study in this paper always have a unique dominant singularity, which is therefore in $\mathbb{R}^+$ by Pringsheim's theorem. For any fixed alphabet size k, the dominant singularity of each generating function will always be the same ρ_k. Moreover, as they are all made of polynomials, quotients and square roots, their analysis have a lot of similarities. In particular, all generating functions satisfy one of the two conditions of the following proposition.

Proposition 1. *Let $f(z)$ be a function that is Δ-analytic at $\rho \in \mathcal{R}^+$.*

1. *If on the intersection of a neighborhood of ρ and its Δ-domain,*
$$f(z) = a - b\sqrt{1-z/\rho} + o(\sqrt{1-z/\rho}),\ \text{with } a, b \in \mathbb{R},\ b \neq 0$$
then $[z^n]f(z) \sim \frac{b}{2\sqrt{\pi}}\rho^{-n}n^{-3/2}$.
2. *If on the intersection of a neighborhood of ρ and its Δ-domain,*
$$f(z) = \frac{a}{\sqrt{1-z/\rho}} + o\left(\frac{1}{\sqrt{1-z/\rho}}\right),\ \text{with } a \in \mathbb{R},\ a \neq 0$$
then $[z^n]f(z) \sim \frac{a}{\sqrt{\pi}}\rho^{-n}n^{-1/2}$.

This proposition is a direct consequence of Theorem 1, using classical formulas for the asymptotic of the coefficients of $z \mapsto \sqrt{1-z}$ and $z \mapsto (1-z)^{-1/2}$. If $f(z)$ is a function which is Δ-analytic at $\rho \in \mathbb{R}^+$, we say that f *satisfies the property Π_1 or Π_2 at ρ* if it satisfies the first or second condition of Proposition 1 respectively.

The computations involved here can be technical, but can be done almost automatically (with the help of computer algebra software for the most complex ones). It is always straightforward to check that this functions satisfy the analytic conditions of Proposition 1. The value of a (and b) can also be computed, and will be some functions of the size of the alphabet. In the rest of the paper, due to the lack of space, most details of the computations will be omitted.

3 Generating Functions for Regular Expressions

In this section we apply the framework of analytic combinatorics to the classes of regular expressions we are studying. From now on, the alphabet considered is $A = \{a_1, \dots, a_k\}$, where $k \geq 1$ is a positive integer.

3.1 General Regular Expressions

From Equation (1), one can use the symbolic method to obtain the following specification

$$\mathcal{R} = \varepsilon + a_1 + \cdots + a_k + \begin{array}{c} * \\ | \\ \mathcal{R} \end{array} + \begin{array}{c} \cup \\ / \backslash \\ \mathcal{R}\;\mathcal{R} \end{array} + \begin{array}{c} \bullet \\ / \backslash \\ \mathcal{R}\;\mathcal{R} \end{array}$$

And therefore, using the dictionary, one obtain the following equation for $R(z)$, the generating function associated to $\mathcal{R}$:

$$R(z) = (k+1)z + zR(z) + 2zR^2(z)$$

From which one can derive the following expression for $R(z)$, using the fact that its Taylor coefficients are nonnegative:

$$R(z) = \frac{1 - z - \sqrt{\Delta_k(z)}}{4z}, \text{ with } \Delta_k(z) = 1 - 2z - (7+8k)z^2$$

The unique dominant singularity of $R(z)$ is $\rho_k = \frac{2\sqrt{2k+2}-1}{7+8k}$, and around ρ_k, one has the following expansion of $R(z)$

$$R(z) = \frac{1-\rho_k}{4\rho_k} - \frac{1}{4\rho_k}\sqrt{\Delta_k(z)} + o\left(\left(1 - \frac{z}{\rho_k}\right)^{\frac{1}{2}}\right)$$

$$R(z) = \frac{\sqrt{2k+2}}{2} - \frac{\sqrt{2(1-\rho_k)}}{4\rho_k}\left(1 - \frac{z}{\rho_k}\right)^{\frac{1}{2}} + o\left(\left(1 - \frac{z}{\rho_k}\right)^{\frac{1}{2}}\right)$$

therefore, using Proposition 1, one can obtain an asymptotic equivalent to the number of nonempty regular expressions.

Lemma 2. *The number of elements of size n in $\mathcal{R}$ is asymptotically equivalent to $C_k \rho_k^{-n} n^{-3/2}$, with $C_k = \frac{\sqrt{2(1-\rho_k)}}{8\rho_k\sqrt{\pi}}$.*

Note that if some generating function $f(z)$ has a unique dominant singularity on ρ_k where we can apply Proposition 1, and satisfies as $z \to \rho_k$:

$$f(z) = a - \frac{b}{4\rho_k}\sqrt{\Delta_k(z)} + o(\sqrt{1 - z/\rho_k})$$

then $[z^n]f(z) \sim b\,[z^n]R(z)$. Therefore, for our computations, it is often more convenient to obtain developments in terms of $\frac{1}{4\rho_k}\sqrt{\Delta_k(z)}$ instead of $\sqrt{1 - z/\rho_k}$. The techniques are of course the same, since they only differ by a multiplicative constant as $z \to \rho_k$.

3.2 Regular Expressions of Languages Containing ε

Denote by $\mathcal{R}_\varepsilon$ and $\mathcal{R}_{\overline{\varepsilon}}$ the regular expressions whose associated languages respectively recognize and do not recognize the empty word. A specification of $\mathcal{R}_\varepsilon$ is the following:

$$\mathcal{R}_\varepsilon = \varepsilon + \underset{\mathcal{R}}{\overset{*}{|}} + \underset{\mathcal{R}_\varepsilon\ \mathcal{R}}{\overset{\cup}{\wedge}} + \underset{\mathcal{R}_{\overline{\varepsilon}}\ \mathcal{R}_\varepsilon}{\overset{\cup}{\wedge}} + \underset{\mathcal{R}_\varepsilon\ \mathcal{R}_\varepsilon}{\overset{\bullet}{\wedge}} \tag{3}$$

From which one can obtain the following expression:

$$R_\varepsilon(z) = \frac{z + zR(z)}{1 - 2zR(z)}$$

The dominant singularity of $R_\varepsilon(z)$ is also ρ_k and it satisfies Π_1 at ρ_k:

$$R_\varepsilon(z) = \frac{2k + \sqrt{2k+2}}{4k+2} - \frac{1 - 2k + 4k\sqrt{2k+2}}{(2k+1)^2}\,\frac{\sqrt{\Delta_k(z)}}{4\rho_k} + o\left(\left(1 - \frac{z}{\rho_k}\right)^{\frac{1}{2}}\right)$$

Therefore, using Proposition 1 one can obtain an asymptotic equivalent to its coefficients:

$$[z^n]R_\varepsilon(z) \sim D_k[z^n]R(z), \text{ with } D_k = \frac{1 - 2k + 4k\sqrt{2k+2}}{(2k+1)^2} \tag{4}$$

Note that, as a consequence, the ratio of regular expressions whose denoted language contains the empty word is asymptotically D_k. For a two-letters alphabet, the value of D_k is approximatively $D_2 \approx 0.664$.

4 Main Result

This section is devoted to the proof of our main theorem:

Theorem 2. *The average number of transitions of the Glushkov's automaton associated to a regular expression of size n, for the uniform distribution, is in $\Theta(n)$.*

4.1 Lower Bound

First remark that if an expression $R \in \mathcal{R}$ contains m letters, then its Glushkov's automaton has $m+1$ states. Moreover, since it is accessible by construction, it has at least m transitions. For $R \in \mathcal{R}$, let $\ell(R)$ be the number of letters in R. Let $L(z)$ be the cost generating function of the number of letters in an element R of $\mathcal{R}$:

$$L(z) = \sum_{R \in \mathcal{R}} \ell(R) z^{|R|}$$

From Equation (1) and Lemma 1, we have

$$L(z) = kz + zL(z) + 2zL(z)R(z) + 2zL(z)R(z)$$

Hence, using the fact that $1 - z - 4zR(z) = \sqrt{\varDelta_k(z)}$:

$$L(z) = \frac{kz}{\sqrt{\varDelta_k(z)}}$$

Therefore, $L(z)$ has a unique dominant singularity at ρ_k and satisfy $\varPi_2$ at ρ_k. By Proposition 1,

$$[z^n]L(z) \sim \frac{k\rho_k}{\sqrt{2\pi(1-\rho_k)}}\rho_k^{-n} n^{-1/2}$$

and the average number of letters in an element of size n of $\mathcal{R}$ is equivalent to $\frac{4k\rho_k^2}{1-\rho_k}n$ as $n \to \infty$. For $k = 2$, an approximation of $\frac{4k\rho_k^2}{1-\rho_k}$ is 0.408. We conclude that in average, the number of transitions in the Glushkov's automaton of an element of size n of $\mathcal{R}$ is in $\varOmega(n)$.

4.2 Outline of the Proof of the Upper Bound

In the following subsections we establish some results used to compute an upper bound for the average number of transitions in a Glushkov's automaton. As stated before, the size of the Glushkov's automaton associated to a nonempty regular expression $R \in \mathcal{R}$ is equal to the number of elements in $First(R)$ plus the number of elements in $Edges(R)$. In the next subsection we prove that in average, the size of $First(R)$ tends towards a constant. For the remaining part, we actually compute an upper bound of the size of $Edges(R)$, not its exact average cardinality. Define the function $e : \mathcal{R} \longrightarrow \mathbb{N}$ by:

$$\begin{cases} e(\varepsilon) = 0 & \\ e(a) = 0 & \forall a \in A \\ e\left(\begin{smallmatrix} * \\ | \\ R \end{smallmatrix}\right) = e(R) + |Last(R)| \cdot |First(R)| & \forall R \in \mathcal{R} \\ e\left(\begin{smallmatrix} & \cup & \\ & / \backslash & \\ R_1 & & R_2 \end{smallmatrix}\right) = e(R_1) + e(R_2) & \forall R_1, R_2 \in \mathcal{R} \\ e\left(\begin{smallmatrix} & \bullet & \\ & / \backslash & \\ R_1 & & R_2 \end{smallmatrix}\right) = e(R_1) + e(R_2) + |Last(R_1)| \cdot |First(R_2)| & \forall R_1, R_2 \in \mathcal{R} \end{cases} \quad (5)$$

Clearly $e(R)$ is an upper bound of $|Edges(R)|$, since its definition is obtained from the one of $Edges$, Equation (2), and the inequality $|X \cup Y| \leq |X| + |Y|$.

Let $E(z)$ and $F(z)$ be the cost generating functions of e and $|First(\cdot)|$ respectively, and let $P(z)$ be the following generating function

$$P(z) = \sum_{R \in \mathcal{R}} |First(R)| \cdot |Last(R)| z^{|R|}$$

Note that, by symmetry, $F(z)$ is also the cost generating function of `last`, $F(z) = \sum_{R \in \mathcal{R}} \texttt{last}(R) z^{|R|}$.

Using Equation (5) and Lemma 1, one has

$$E(z) = zE(z) + zP(z) + 2zR(z)E(z) + 2zR(z)E(z) + zF(z)^2$$

hence

$$(1 - z - 4zR(z))E(z) = zP(z) + zF(z)^2$$

and as $1 - z - 4zR(z) = \sqrt{\Delta_k(z)}$, we finally obtain

$$E(z) = \frac{zP(z) + zF(z)^2}{\sqrt{\Delta_k(z)}} \tag{6}$$

At this point we need more informations about $P(z)$ and $F(z)$ in order to conclude that $[z^n]E(z) = n\,\mathcal{O}([z^n]\mathcal{R}(z))$. They will be obtained in the next subsections.

4.3 The Cost Generating Function $F(z)$

For an element $R \in \mathcal{R}$, let $\texttt{first}(R)$ and $\texttt{last}(R)$ denote respectively the cardinalities of the sets $First(R)$ and $Last(R)$. We analyze the average value of $\texttt{first}(R)$, for a regular expression $R \in \mathcal{R}$ of size n. $F(z)$ is the cost generating function of $\texttt{first}$, and we also consider its restrictions $F_\varepsilon(z)$ to $\mathcal{R}_\varepsilon$ and $F_{\overline{\varepsilon}}(z)$ to $\mathcal{R}_{\overline{\varepsilon}}$:

$$F_\varepsilon(z) = \sum_{R\in\mathcal{R}_\varepsilon} \texttt{first}(R)z^{|R|}; \quad F_{\overline{\varepsilon}}(z) = \sum_{R\in\mathcal{R}_{\overline{\varepsilon}}} \texttt{first}(R)z^{|R|}$$

For a given $R \in \mathcal{R}$, the value of $\texttt{first}(R)$ can be computed inductively as

$$\begin{cases} \texttt{first}(\varepsilon) = 0; & \\ \texttt{first}(a) = 1 & \forall a \in A \\ \texttt{first}\left(\overset{*}{\underset{R}{|}}\right) = \texttt{first}(R) & \forall R \in \mathcal{R} \\ \texttt{first}\left(\overset{\cup}{\underset{R_1\ R_2}{/\backslash}}\right) = \texttt{first}(R_1) + \texttt{first}(R_2) & \forall R_1, R_2 \in \mathcal{R} \\ \texttt{first}\left(\overset{\bullet}{\underset{R_1\ R_2}{/\backslash}}\right) = \texttt{first}(R_1) + \texttt{first}(R_2) & \forall R_1 \in \mathcal{R}_\varepsilon,\ \forall R_2 \in \mathcal{R} \\ \texttt{first}\left(\overset{\bullet}{\underset{R_1\ R_2}{/\backslash}}\right) = \texttt{first}(R_1) & \forall R_1 \in \mathcal{R}_{\overline{\varepsilon}},\ \forall R_2 \in \mathcal{R} \end{cases} \tag{7}$$

We consider the following specification of $\mathcal{R}$

$$\mathcal{R} = \varepsilon + a_1 + \cdots + a_k + \overset{*}{\underset{\mathcal{R}}{|}} + \overset{\cup}{\underset{\mathcal{R}\ \mathcal{R}}{/\backslash}} + \overset{\bullet}{\underset{\mathcal{R}_\varepsilon\ \mathcal{R}}{/\backslash}} + \overset{\bullet}{\underset{\mathcal{R}_{\overline{\varepsilon}}\ \mathcal{R}}{/\backslash}}$$

This specification can be transformed into a functional equation on cost generating functions using Lemma 1 and Equation (7). For instance, the construction $\overset{\bullet}{\underset{\mathcal{R}_\varepsilon\ \mathcal{R}}{/\backslash}}$ produces the term:

$$z \sum_{R_1\in\mathcal{R}_\varepsilon} \sum_{R_2\in\mathcal{R}} (\texttt{first}(R_1) + \texttt{first}(R_2))z^{|R_1|}z^{|R_2|} = zR_\varepsilon(z)F(z) + zF_\varepsilon(z)R(z)$$

All computations and simplifications done, we obtain:

$$F(z) = \frac{kz}{1 - z - 3zR(z) - zR_\varepsilon(z)}$$

As $R(z)$ and $R_\varepsilon(z)$ are known, it is straightforward to obtain the expansion of $F(z)$ near its dominant singularity ρ_k and to check that it satisfies Π_1 at ρ_k:

$$F(z) = E_k - F_k \frac{\sqrt{\Delta_k(z)}}{4\rho_k} + o\left(\left(1 - \frac{z}{\rho_k}\right)^{\frac{1}{2}}\right) \tag{8}$$

with $E_k = 1 + \sqrt{2k+2}$ and $F_k = \frac{12+6k+8\sqrt{2k+2}}{k}$. Hence, using Proposition 1, one can prove the following proposition.

Proposition 2. *For any fix integer $k \geq 1$, the average size of* `first` *tends toward a constant $F_k = \frac{12+6k+8\sqrt{2k+2}}{k}$, as n tends toward infinity: $[z^n]F(z) \sim F_k\,[z^n]R(z)$.*

For $k = 2$, an approximation of F_k is $F_2 \approx 21.8$.

4.4 The Cost Generating Function $F_\varepsilon(z)$

Similarly, one can use the specification of $\mathcal{R}_\varepsilon$ of Equation (3) to compute $F_\varepsilon(z)$. We obtain

$$F_\varepsilon(z) = \frac{zF(z) + 2zR_\varepsilon(z)F(z)}{1 - 2zR(z)}$$

Once again, $F_\varepsilon(z)$ satisfies Π_1 at ρ_k with

$$F_\varepsilon(z) = G_k - H_k \frac{\sqrt{\Delta_k(z)}}{4\rho_k} + o(\sqrt{\Delta_k(z)}),\ \text{with } G_k = \frac{4k+1+\sqrt{2k+2}}{2k+1}$$

The expression of $H_k > 0$ in terms of k can also be computed, but is not needed in the following.

4.5 The Cost Generating Function of the Product `first` × `last`

We analyze the average value of the product of $\texttt{first}(R)$ and $\texttt{last}(R)$, for a regular expression $R \in \mathcal{R}$ of size n. We consider the cost generating function $P(z)$ of this product, and its restriction $P_\varepsilon(z)$ to $\mathcal{R}_\varepsilon$ and $P_{\overline{\varepsilon}}(z) = P(z) - P_\varepsilon(z)$ to $\mathcal{R}_{\overline{\varepsilon}}$:

$$P_\varepsilon(z) = \sum_{R \in \mathcal{R}_\varepsilon} \texttt{first}(R) \cdot \texttt{last}(R) z^{|R|}; \quad P_{\overline{\varepsilon}}(z) = \sum_{R \in \mathcal{R}_{\overline{\varepsilon}}} \texttt{first}(R) \cdot \texttt{last}(R) z^{|R|}$$

We use the following specification of $\mathcal{R}$:

$$\mathcal{R} = \varepsilon + a_1 + \cdots + a_k + \begin{array}{c} * \\ | \\ \mathcal{R} \end{array} + \begin{array}{c} \cup \\ /\backslash \\ \mathcal{R}\ \mathcal{R} \end{array} + \begin{array}{c} \bullet \\ /\backslash \\ \mathcal{R}_\varepsilon\ \mathcal{R}_\varepsilon \end{array} + \begin{array}{c} \bullet \\ /\backslash \\ \mathcal{R}_\varepsilon\ \mathcal{R}_{\overline{\varepsilon}} \end{array} + \begin{array}{c} \bullet \\ /\backslash \\ \mathcal{R}_{\overline{\varepsilon}}\ \mathcal{R}_\varepsilon \end{array} + \begin{array}{c} \bullet \\ /\backslash \\ \mathcal{R}_{\overline{\varepsilon}}\ \mathcal{R}_{\overline{\varepsilon}} \end{array}$$

The cost generating function associated to each term of the specification is computed separately. For instance, one has the following cost function for $\underset{\mathcal{R}_\varepsilon\ \mathcal{R}_{\overline{\varepsilon}}}{\overset{\bullet}{\diagup\!\diagdown}}$:

$$\begin{aligned}
\sum_{R_1\in\mathcal{R}_\varepsilon}\sum_{R_2\in\mathcal{R}_{\overline{\varepsilon}}} \texttt{first}\left(\underset{R_1\ R_2}{\overset{\bullet}{\diagup\!\diagdown}}\right)\texttt{last}\left(\underset{R_1\ R_2}{\overset{\bullet}{\diagup\!\diagdown}}\right) z^{1+|R_1|+|R_2|}
&= z\sum_{R_1\in\mathcal{R}_\varepsilon}\sum_{R_2\in\mathcal{R}_{\overline{\varepsilon}}} (\texttt{first}(R_1)+\texttt{first}(R_2))\,\texttt{last}(R_2) z^{|R_1|} z^{|R_2|}\\
&= z\sum_{R_1\in\mathcal{R}_\varepsilon}\sum_{R_2\in\mathcal{R}_{\overline{\varepsilon}}} \texttt{first}(R_1)\texttt{last}(R_2) z^{|R_1|} z^{|R_2|} + zR_\varepsilon(z)P_{\overline{\varepsilon}}(z)\\
&= zF_\varepsilon(z)F_{\overline{\varepsilon}}(z) + zR_\varepsilon(z)P_{\overline{\varepsilon}}(z)
\end{aligned}$$

Using the same techniques for the other parts of the expression, we obtain:

$$P(z) = \frac{kz + 3zF(z)^2 + zF_\varepsilon(z)^2}{1 - z - 2zR(z) - 2zR_\varepsilon(z)}$$

From which, using the previous results, we conclude that $P(z)$ satisfies Π_1 at ρ_k and that there exists some real numbers I_k and J_k such that, as $z \to \rho_k$,

$$P(z) = I_k - J_k\sqrt{\Delta_k(z)} + o(\sqrt{\Delta_k(z)})$$

with $I_k = \frac{38k^2+77k+28+(14k^2+45k+20)\sqrt{2k+2}}{2k(k+1)}$ and $J_k > 0$.

4.6 Concluding the Proof of Theorem 2

We can now analyze Equation (6). As both $F(z)$ and $P(z)$ satisfy Π_1, so does $zP(z) + zF(z)^2$, since the constant term of its development near ρ_k is not zero. Hence $E(z)$ satisfies Π_2 at ρ_k and Proposition 1 can be applied:

$$[z^n]E(z) \sim e_k n\,[z^n]R(z)$$

The value of e_k is not needed to prove the theorem, but is still an indication on a bound of the average number of transitions in a Glushkov's automaton:

$$e_k = \frac{32k^4 + 204k^3 + 406k^2 + 306k + 72 + (80k^3 + 230k^2 + 193k + 52)\sqrt{2k+2}}{2k(k+1)(2k+1)(8k+7)}$$

For $k = 2$, an approximation of e_k is $e_2 \approx 6.77$.

The quantity $[z^n]E(z)/[z^n]R(z)$ is an upper bound of the average cardinality of *Edges* and the average cardinality of *First* is bounded by Proposition 2, therefore, by linearity of the average, the average number of transitions in a Glushkov's automaton is in $\mathcal{O}(n)$. This concludes the proof of Theorem 2.

5 Remarks

In this paper we used the specification of regular expressions that directly follows the inductive definition. One could try to use a more precise specification to prevent some useless patterns in regular expressions to appear, as was done for instance in [10] for enumeration purposes. Depending on the chosen such rules, the analysis we conduct here could still be doable, but probably much more complicated. Especially if the specification is not context-free (see [11] for a related analysis). However, we believe that the result would be the same in much cases. Indeed, the proofs of Theorem 2 and Proposition 2 are based on the fact that for a non-negligible proportion of regular expressions, the denoted language does not recognize the empty word, and that the average number of concatenation operators is non-negligible. This must be true for most natural specifications.

Also note that we could have used bivariate generating functions instead of cost generating functions in this paper. The computations would have been the same here, but this approach should be useful if one want to go one step further, and try to obtain some informations not only about the average value, but also about the limit distribution.

The author is supported by the ANR (GAMMA - project BLAN07-2_195422).

References

1. Sakarovitch, J.: The language, the expression, and the (small) automaton. In: CIAA. Volume 3845 of LNCS. (2005) 15–30
2. Glushkov, V.M.: The abstract theory of automata. Russian Math. Surveys **16** (1961) 1–53
3. Hromkovic, J., Seibert, S., Wilke, T.: Translating regular expressions into small epsilon-free nondeterministic finite automata. In: STACS. Volume 1200 of LNCS. (1997) 55–66
4. Hagenah, C., Muscholl, A.: Computing epsilon-free NFA from regular expressions in $O(n \log^2(n))$ time. ITA **34**(4) (2000) 257–278
5. Ilie, L., Yu, S.: Constructing NFA s by optimal use of positions in regular expressions. In: CPM. Volume 2373 of LNCS. (2002) 279–288
6. Champarnaud, J.M., Nicart, F., Ziadi, D.: Computing the follow automaton of an expression. In: CIAA. Volume 3317 of LNCS. (2004) 90–101
7. Flajolet, P., Sedgewick, R.: Analytic Combinatorics. Cambridge University Press (2008)
8. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (1979)
9. Flajolet, P., Odlyzko, A.M.: The average height of binary trees and other simple trees. J. Comput. Syst. Sci. **25**(2) (1982) 171–213
10. Lee, J., Shallit, J.: Enumerating regular expressions and their languages. In: CIAA. Volume 3317 of LNCS. (2004) 2–22
11. Fernández-Camacho, M.I., Steyaert, J.M.: Algebraic simplification in computer algebra: An analysis of bottom-up algorithms. TCS **74**(3) (1990) 273–298