Uniform Random Expressions Lack Expressivity

Florent Koechlin, Cyril Nicaud, Pablo Rotondo

Uniform Random Expressions Lack Expressivity

Florent Koechlin
Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE, UPEM, France
florent.koechlin@u-pem.fr

Cyril Nicaud
Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE, UPEM, France
cyril.nicaud@u-pem.fr

Pablo Rotondo
Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE, UPEM, France
pablo.rotondo@u-pem.fr

Abstract

In this article, we question the relevance of uniform random models for algorithms that use expressions as inputs. Using a general framework to describe expressions, we prove that if there is a subexpression that is absorbing for a given operator, then, after repeatedly applying the induced simplification to a uniform random expression of size n, we obtain an equivalent expression of constant expected size. This proves that uniform random expressions lack expressivity, as soon as there is an absorbing pattern. For instance, $(a+b)^\star$ is absorbing for the union for regular expressions on $\{a,b\}$, hence random regular expressions can be drastically reduced using the induced simplification.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics

Keywords and phrases Random expressions, simplification algorithms, analytic combinatorics

Digital Object Identifier 10.4230/LIPIcs.MFCS.2019.51

1 Introduction

Although the classical paradigm to analyze the efficiency of an algorithm is to study its worst case complexity, in many situations it is not a relevant way to describe what is observed in practice, when one actually implements and tests the algorithm. This is typically the case when worst case instances are very unlikely, and when the algorithm behaves well on most inputs. A natural way to deal with this issue is to set a probability distribution on the inputs of size n, for each n, and to study the average running time of the algorithm when the inputs are randomly chosen following the prescribed distribution.[1] The textbook example is QuickSort algorithm with a simple choice of pivot, which runs in $\mathcal{O}(n \log n)$ expected time and $\mathcal{O}(n^2)$ in the worst case. But the average case analysis raises the question of the input distribution, which is usually a difficult one. The result on QuickSort holds when the input is a permutation taken uniformly at random, but are real-life arrays uniformly shuffled? Probably not, whatever "real-life" means. For QuickSort, one can randomize the choice of the pivot to ensure the $\mathcal{O}(n \log n)$ expected running time, which is equivalent to uniformly shuffling the input. This is probably not a good idea in practice though, since we lose the property that some input can be often partially sorted: popular programming languages as Python or Java now use the recent TimSort algorithm that takes advantage of the presortedness of the input.

[1] There are other relevant theoretical alternatives, as the *smoothed analysis* of algorithms, but we will only consider average case analysis in this article.

Still, if one wants to test the efficiency of an implementation of an algorithm, or to compare two tools, and if there are not sufficiently many benchmarks available, doing the analysis on random inputs sounds reasonable. In addition, using the uniform distribution has several advantages: it ensures a lot of diversity on the generated input (as they are all equally likely) and there are several very generic methods to perform the random generation efficiently: the two main ones being the *recursive method* [20] and *Boltzmann samplers* [7]; both directly translate a combinatorial description of the inputs into a uniform random generator. And also, beyond the experiments, the theoretical average case analysis of the algorithm can be done using techniques from discrete probabilities or from analytic combinatorics.

In this article we focus on algorithms whose inputs are formulas, encoded by expressions. Informally, an expression is a tree, whose internal nodes are labelled with operators, and whose leaves are labelled with variables or constants. Such trees are very natural ways to represent formulas in many different contexts; several examples are given in Fig. 1. Observe that the equivalent reverse polish notation of the expression can be obtained by a simple tree traversal.

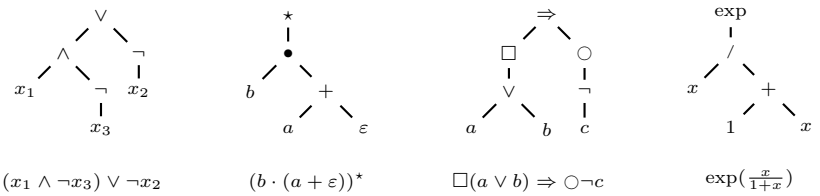


Figure 1 Four expression trees and their associated formulas. From left to right: a logical formula, a regular expression, an LTL formula and a function.

We define the set of expressions by the finite sets of operators of any given degree: for regular expressions on an alphabet $\{a, b\}$, we consider that there are three kinds of leaves $\{a, b, \varepsilon\}$, one unary internal node $\{\star\}$ for the Kleene star and two binary internal nodes $\{\bullet, +\}$ for the concatenation and the union. We define the *size* of an expression as its number of nodes (including leaves). For the set of regular expressions $\mathcal{L}_R$, we have the following formal inductive description:

$$\mathcal{L}_R = a + b + \varepsilon + \overset{\star}{|}_{\mathcal{L}_R} + \overset{\bullet}{/\backslash}_{\mathcal{L}_R\ \mathcal{L}_R} + \overset{+}{/\backslash}_{\mathcal{L}_R\ \mathcal{L}_R}. \tag{1}$$

Observe that if we consider plane trees (where the children of a node are ordered, as when they are drawn on a plan), Eq. (1) is an unambiguous description of the expressions in $\mathcal{L}_R$. It is, of course, not an unambiguous description of the associated languages, since a given language admits several representations using expressions. It is clear that the other expressions of Fig. 1 also have a specification that is similar to Eq. (1). Efficient methods were developed in the field of analytic combinatorics to deal with that kind of combinatorial inductive descriptions. In a nutshell, we first associate the formal power series $L(z) := \sum_{n\geq 0} \ell_n z^n$, where ℓ_n is the number of expressions of size n. Eq. (1) translates into an equation on $L(z)$:

$$L(z) = z\phi(L(z)), \text{ with } \phi(X) = 3 + X + 2X^2. \tag{2}$$

Then we see $L(z)$ as a function from $\mathbb{C}$ to $\mathbb{C}$, which is analytic at 0, and use its functional equation Eq. (2) to derive a lot of information on ℓ_n, typically asymptotic equivalents. When considering the uniform distribution, one can easily go further and introduce formal parameters in the specification (for instance, the number of unary operators), which can also

be interpreted in the analytic world, and directly obtain statistics of interest (for instance, the asymptotic equivalent of the expected number of unary operators). Combinatorial structures whose generating series satisfy an implicit functional equation $L(z) = z\phi(L(z))$ are called *simple varieties of trees* (there are some analytic conditions on ϕ, which are always satisfied in the sequel), and we refer the reader to the textbook on analytic combinatorics [9, VII.3.] for further information. For the remainder of this introduction, the reader has just to keep in mind that the technical framework used to deal with uniform random expressions is well established.

As can be expected at this point of the discussion, there are a lot of results in the literature on simple varieties of trees and on specific random expressions. They exhibit some kind of "universal" behavior: for instance, the number of such trees always grows in $\ell_n \sim \alpha\beta^n n^{-3/2}$, for some computable constants α and β [16]. Their expected height is asymptotically equivalent to $\gamma\sqrt{n}$, for some computable γ [8]. In [10], the authors proved that if a uniform random expression is compressed by identifying common subexpressions (changing the tree into a directed acyclic graph), the expected size of the result is asymptotically equivalent to $\frac{\delta n}{\sqrt{\log n}}$. It was also established [11] that computing the derivative of a random function, defined by a tree as in Fig. 1, costs $\Theta(n^{3/2})$ in expected time and space. Concerning regular expressions, the second author studied a standard construction that builds an ε-free non-deterministic automaton recognizing the same language [18]: he proved that there is an expected linear number of transitions in the resulting automaton, although it is $\Theta(n^2)$ in the worst case. This result was subsequently refined and adapted to various other similar constructions by Moreira, Reis and their co-authors (see [2] for a survey). A lot of efforts were also made for Boolean formulas, in slightly different settings: we consider the set $\mathcal{F}_k$ of the 2^{2^k} Boolean formulas on k variables. For a suitable set of logical operators, for instance $\{\vee, \wedge\}$, and for any $n \geq 1$, we consider the probability $p_n(f)$ that a random Boolean expression represents the Boolean function $f \in \mathcal{F}_k$; then we let n tend to infinity, and it can be proved that $p_n(f)$ has a limit $p(f)$, and that p is a probability distribution on $\mathcal{F}_k$. So the distribution is different, and more theoretical than what is usually used for testing implementations. But, since it relies on uniform random expressions, that kind of distributions has a lot of similarities with uniform random expressions (see [12] for a survey). While investigating this distribution on Boolean formulas, its lack of complexity was discovered: with high probability the formulas obtained are very simple [13, 14]. In fact, there is an older result in Nguyên Thê PhD's dissertation [17, Ch 4.4], where he considered and/or Boolean expressions whose leaves are either `true`, `false` or a literal; he studied the impact of inductively applying 8 different reduction rules, such as $\texttt{true} \vee \phi \to \texttt{true}$, on uniform random expressions of size n, and proved that the expected size of the simplified expression tends to a constant as n tends to infinity. This was a very negative result for uniformly chosen random Boolean expressions (with $\{\vee, \wedge\}$ operators and $\{\texttt{true}, \texttt{false}, x\}$ leaves), as most of them can be drastically simplified using simple rules.

In this article, we aim at extending Nguyên Thê's result to much more general families of expressions, proving its universality in the context of simple varieties of trees. Informally, our main result is the following:

▶ **Theorem.** *Consider a simple variety of expressions, where there is a tree pattern $\mathcal{P}$ that is an absorbing element for one of the operators $\circledast$. We consider the simplification algorithm that consists in inductively changing a $\circledast$-node by $\mathcal{P}$ whenever one of its children can be simplified into $\mathcal{P}$. Then the expected size of the simplification of a uniform random expression of size n tends to a constant as n tends to infinity.*

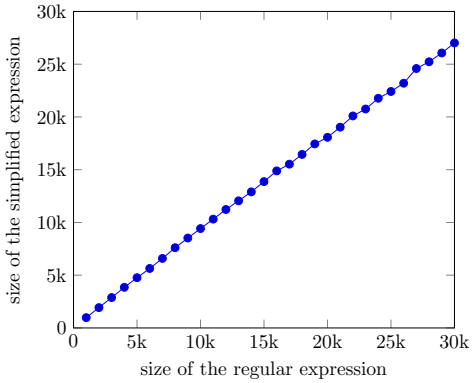


Figure 2 Experimental study of the size of a random uniform regular expression after applying the simplification rules $E + \mathcal{P} \rightsquigarrow \mathcal{P}$ and $\mathcal{P} + E \rightsquigarrow \mathcal{P}$.

As an example, the tree pattern $\mathcal{P}$ corresponding to $(a + b)^\star$ is absorbing for the union $+$ of regular expressions: for any regular expression E, one can simplify $E + \mathcal{P}$ or $\mathcal{P} + E$ into $\mathcal{P}$ without changing the described language. Hence, applying this simple simplification rule to a very large random regular expression results in an expression of constant expected size. Also, the tree pattern associated with $x_1 \vee \neg x_1$ is absorbing for $\vee$, since it evaluates to `true`: random Boolean expressions are also drastically simplified using this transformation rule. It is the same for LTL formulas, and for functions: if there is the constant 0 (or a way to produce it with a tree pattern) and the multiplication, then the same phenomenon occurs. In fact, for most natural notions of expressions, there are absorbing patterns for some operators, and uniform random expressions lack expressivity, since they can be simplified into much smaller equivalent expressions with high probability. This is a universal and negative result on uniform random expressions, which is worth knowing when designing empirical tests. Indeed, comparing two tools on uniform random expressions probably consists in simply comparing their efficiency for simplifying the input (preprocessing step). It also questions the relevance of further theoretical studies on that kind of models. The consequences are discussed in details in Section 4.

To conclude this introduction, we would like to mention that our result is not always easy to observe experimentally. We implemented uniform random samplers for regular expressions, and computed (experimentally) the expected size when using the fact that $(a + b)^\star$ is absorbing for $+$. The results are depicted on Fig. 2. It is not clear that the curve converges to some constant, as stated by our theorem. In fact, if we compute the value of the limit (we used SageMath software for the computations) we obtain a bit more than 3 624 217, a huge constant. This value can be considerably reduced if we introduce several more rewriting rules (see Section 4 for the details).

2 Definitions and settings

2.1 Combinatorial constructions and analytic combinatorics

In this article, the proofs rely on mathematical techniques developed in the framework of analytic combinatorics [9]. In this section, we recall the basic tools of this theory that will be used in the sequel (more advanced results will be introduced when needed). A *combinatorial class* is a set $\mathcal{C}$ equipped with a notion of *size* $|\cdot|$ (a mapping from $\mathcal{C}$ to $\mathbb{Z}_{\geq 0}$) such that for any $n \in \mathbb{Z}_{\geq 0}$ there are finitely many elements of $\mathcal{C}$ of size n. Let $\mathcal{C}_n = \{C \in \mathcal{C} : |C| = n\}$

and let $c_n = |C_n|$ be its cardinality. The *ordinary generating series* (OGS for short) of the combinatorial class $\mathcal{C}$ is the formal power series $C(z) := \sum_{C\in\mathcal{C}} z^{|C|} = \sum_{\geq 0} c_n z^n$. Let $[z^n]C(z) := c_n$ denote the n-th coefficient of $C(z)$.

Let $\mathcal{A}$ and $\mathcal{B}$ be two combinatorial classes, and let $A(z)$ and $B(z)$ be their OGS, respectively. If $\mathcal{A}$ and $\mathcal{B}$ are disjoint, then the OGS of their union is simply $A(z) + B(z)$. Moreover, if we define the size of a pair (α, β) in $\mathcal{A} \times \mathcal{B}$ as the sum of the sizes of α and β, then $\mathcal{A} \times \mathcal{B}$ is a combinatorial class with OGS $A(z)B(z)$, as a simple computation shows. There are many other set constructions on combinatorial classes that directly translate on OGS, but we will not need them in the sequel (see [9, Part A]).

Let us take an example that will be used throughout this article, the combinatorial set $\mathcal{L}_R$ of regular expressions on the alphabet $\{a, b\}$. Its elements are plane rooted trees, whose nodes (internal and leaves) are labelled: leaves are labelled with a, b or ε, unary nodes are labelled with $\star$, and binary nodes with $\bullet$ or $+$. Rooted means that there is a root (contrary to the graph-theory notion of tree), and plane means that the order of the children matters: $\overset{+}{\underset{a}{/}}\!\!\overset{}{\underset{b}{\backslash}}$ and $\overset{+}{\underset{b}{/}}\!\!\overset{}{\underset{a}{\backslash}}$ are different elements of $\mathcal{L}_R$. Encoding trees as tuples (for instance $(\bullet, \texttt{left}, \texttt{right})$ for a tree of root $\bullet$ and left and right children `left` and `right`), we have the identity:

$$\mathcal{L}_R = \{a, b, \varepsilon\} \cup \{\star\} \times \mathcal{L}_R \cup \{\bullet, +\} \times \mathcal{L}_R \times \mathcal{L}_R, \tag{3}$$

which we rewrite as in Eq. (1) for readability: $\mathcal{L}_R = a + b + \varepsilon + \overset{\star}{\underset{\mathcal{L}_R}{|}} + \overset{\bullet}{\underset{\mathcal{L}_R}{/}}\!\!\overset{}{\underset{\mathcal{L}_R}{\backslash}} + \overset{+}{\underset{\mathcal{L}_R}{/}}\!\!\overset{}{\underset{\mathcal{L}_R}{\backslash}}$. From this we directly obtain the following functional equation for the associated OGS $L_R(z)$:

$$L_R(z) = 3z + zL_R(z) + 2zL_R(z)^2 = z\phi(L_R(z)), \text{ with } \phi(y) = 3 + y + 2y^2.$$

Going further, one can also encode statistics on the elements of a combinatorial class into generating series. Let ξ be a mapping from a combinatorial class $\mathcal{C}$ to $\mathbb{Z}_{\geq 0}$. We are interested in the double sequence $c_{n,k}$ that counts the number of elements $C \in \mathcal{C}_n$ such that $\xi(C) = k$. The associated bivariate generating series (BGS for short) is by definition:[2]

$$C(z, u) := \sum_{C\in\mathcal{C}} z^{|C|} u^{\xi(C)} = \sum_{n,k\geq 0} c_{n,k} z^n u^k.$$

When the statistics ξ is *additive*, which means that it is defined on pairs by $\xi((\alpha, \beta)) = \xi(\alpha) + \xi(\beta)$, then we also have a generic dictionary to translate combinatorial constructions into BGS directly (see [9, Ch III]). This is for instance the case if ξ is the number of a-leaves in a regular expression. Moreover, formal differentiation with respect to u allows us to compute the expected value of ξ for the uniform distribution: by definition, the expectation of ξ for objects of size n is $\mathbb{E}_n[\xi] = \frac{1}{|\mathcal{C}_n|} \sum_{C\in\mathcal{C}_n} \xi(C)$. But observe that

$$\frac{d}{du}C(z, u) = \frac{d}{du} \sum_{C\in\mathcal{C}} z^{|C|} u^{\xi(C)} = \sum_{n\geq 0} \left(\sum_{C\in\mathcal{C}_n} \xi(C) u^{\xi(C)-1} \right) z^n.$$

Hence, if we extract the n-th coefficient[3] in z, and set $u = 1$, we have c_n times the expectation. We thus get the following formula to compute the expectation for the uniform distribution:

$$\mathbb{E}_n[\xi] = \frac{[z^n] \frac{d}{du} C(z, u)\Big|_{u=1}}{[z^n]C(z, 1)}. \tag{4}$$

[2] Observe that $C(z, 1) = C(z)$.
[3] Recall that it is exactly what the operator $[z^n]$ does.

Higher moments are obtained similarly, by differentiating several times with respect to u.

Let us continue our example on regular expressions over a binary alphabet by studying the statistics ξ associated with the number of unary operators $\star$ in an expression. The technique consists in *marking* the $\star$-nodes (we use a circle, symbolically) and in changing the marks into the formal variable u when translating the combinatorial specification:

$$\mathcal{L}_R = a + b + \varepsilon + \overset{\circledast}{|}_{\mathcal{L}_R} + \overset{\bullet}{/\!\backslash}_{\mathcal{L}_R\ \mathcal{L}_R} + \overset{+}{/\!\backslash}_{\mathcal{L}_R\ \mathcal{L}_R} \longrightarrow L_R(z,u) = 3z + zuL_R(z,u) + 2zL_R(z,u)^2.$$

After differentiating with respect to u and setting $u = 1$ we get:

$$\frac{d}{du}L_R(z,u)\Big|_{u=1} = \frac{zL_R(z)}{1 - z - 4zL_R(z)}. \tag{5}$$

At this point, the values of interests are hidden as coefficients of formal power series, and we need to extract or estimate them. This is where we use complex analysis: when their radius of convergence at 0 is not zero, the formal power series are seen as functions from $\mathbb{C}$ to $\mathbb{C}$. Using theorems developed in the field, this proves very useful to obtain an asymptotic equivalent of the n-th coefficient. We will not state formally the theorems here, because there are some analytic conditions on the series that must be satisfied and that are a bit too long for this extended abstract (we refer to the textbook [9] for all the details). What is needed to know is the following: since we are doing combinatorics, the coefficients in our series $C(z)$ are non-negative reals. Hence, Pringsheim's theorem applies and yields that, if finite, the radius of convergence ρ of $C(z)$ is a singularity. If there are no other singularity of modulus ρ then, together with other analytic conditions, the behavior of the function $C(z)$ as $z \to \rho$ determines the asymptotic equivalent of $[z^n]C(z)$ as $n \to \infty$. For instance, the *Transfer Theorem* [9, Ch VI.3] states that, under analytic conditions, if $C(z) \sim_{z\to\rho} \lambda(1 - z/\rho)^{-\alpha}$ with $\alpha \notin \{0, -1, -2, \ldots\}$, then $[z^n]C(z) \sim \lambda\rho^{-n}n^{\alpha-1}/\Gamma(\alpha)$, where Γ is Euler's gamma-function, the generalization of factorial.

Back to our example, as a polynomial equation of degree two, Eq. (2) can be solved in $L_R(z)$. Only one of the two solutions is combinatorially meaningful (the other one has negative coefficients). The result is amenable to the Transfer Theorem, with $\rho_R = \frac{2\sqrt{6}-1}{23}$ and an exponent $-\alpha = -1/2$ near ρ_R, yielding $[z^n]L_R(z) \sim \lambda\rho_R^{-n}n^{-3/2}$, for some computable $\lambda > 0$. Similarly, we can estimate the expected number of $\star$-nodes using Eq. (4) and applying the Transfer Theorem to Eq. (5). This routine exercise yields that the expected number of $\star$-nodes is asymptotically equivalent to γn, for some computable positive constant γ.

Observe that the latter result has a direct consequence on the algorithm applying the simplification pattern $(\mathcal{T}^\star)^\star \rightsquigarrow \mathcal{T}^\star$: since it cannot remove more nodes than there are stars in the expression, the expected size after simplification is in $\Theta(n)$. In fact, we can easily be more precise using our tools, as stated in the following proposition.

▶ **Proposition 1.** *Consider the algorithm that simplifies consecutive stars in a regular expression. For the uniform distribution on $\mathcal{L}_R$, the expected size after reduction is asymptotically equivalent to κn, with $\kappa = \frac{4}{529}(126 + \sqrt{6}) \approx 0.97$.*

Hence, this simplification rule is not powerful enough to demonstrate the lack of expressivity.

2.2 Combinatorial expressions: simple varieties of trees

As announced above, our combinatorial model for expressions is the notion of simple varieties of trees [9, VII.3]. We consider expressions as a combinatorial class of $\mathcal{L}$ defined in terms of a sequence of sets of labels $\mathcal{A} = (\mathcal{A}_i)_{i\in\mathbb{Z}_{\geq 0}}$ indexed by their arity (or degree): $f \in \mathcal{A}_i$ has

arity i and labels nodes with i children. For instance, the set of regular expressions over the alphabet $\{a, b\}$, can be described in this formalism using $\mathcal{A}_0 = \{\varepsilon, a, b\}$, $\mathcal{A}_1 = \{\star\}$, and $\mathcal{A}_2 = \{+, \bullet\}$, the other $\mathcal{A}_i$'s being empty for $i \geq 3$.

To avoid trivially uninteresting cases, we define combinatorial expressions as follows:

▶ **Definition 2.** *A set of* combinatorial expressions *of labels $\mathcal{A} = (\mathcal{A}_i)_{i\in\mathbb{Z}_{\geq 0}}$ is a combinatorial set of rooted plane trees whose nodes of arity i are labelled with elements of a finite set $\mathcal{A}_i$, under the additional conditions that $\mathcal{A}_0 \neq \emptyset$ and that there exists $i \geq 2$ such that $\mathcal{A}_i \neq \emptyset$. Its* characteristic series *is the formal power series $\phi(y) = \sum_{i\geq 0} \phi_i z^i$, where $\phi_i = |\mathcal{A}_i|$.*

Note that there can be infinitely many non-empty $\mathcal{A}_i$ and that a label can be in several $\mathcal{A}_i$'s: we can have the union label in $\mathcal{A}_i$ for any $i \geq 2$, to take its associativity into account.

If $\mathcal{L}$ is a set of combinatorial expressions of characteristic series $\phi(y)$, then we have the fundamental formal equation for its OGS $L(z)$:

$$L(z) = z\,\phi(L(z))\,. \tag{6}$$

In our running example, rational expressions, ϕ is a quadratic polynomial and Eq. (6) can be solved explicitly. But usually this is not the case. Fortunately, we can still work out the asymptotic equivalents directly from the functional equation Eq. (6), using implicit functions. Several conditions, gathered under the name of *Smooth inverse-function schema for trees* [9, Definition VII.3], need to be checked in order to apply this fundamental result:

▶ **Definition 3** (Smooth inverse-function schema for trees [9])**.** *A function $L(z)$ satisfying the functional equation $L(z) = z\phi(L(z))$ is said to belong to the smooth inverse-function schema if $L(z)$ is analytic at $z = 0$, $\phi(y)$ is analytic at $y = 0$, $\phi(0) \neq 0$, the coefficients of ϕ are non-negative, ϕ is not linear, and finally there exists a solution τ of the equation $\phi(\tau) - \tau\phi'(\tau) = 0$, with $0 < \tau < \rho_\phi$ where ρ_ϕ is the radius of convergence of ϕ.*

Let us check this definition when $\mathcal{L}$ is a set of combinatorial expressions of characteristic series $\phi(y)$. The fact that $\phi(y)$ is analytic at 0 means that there are not too many operators of arity i (typically, not super-exponential in i), which is indeed the case for all the examples of Fig. 1. The other conditions are satisfied by definition, except the last one. The solution τ always exists, but we have to ensure that it is smaller than the radius of convergence of ϕ. The following elementary observation is quite useful:

▶ Remark 4. If ϕ is a polynomial, i.e. only finitely many $\mathcal{A}_i$'s are non-empty, then a set of expressions of characteristic series ϕ belongs to the smooth inverse-function schema.

Consider a variation on regular expressions $\mathcal{L}'_R$, where $\mathcal{A}_0 = \{\varepsilon, a, b\}$, $\mathcal{A}_1 = \{\star\}$, $\mathcal{A}_2 = \{+, \bullet\}$, and $\mathcal{A}_i = \{+\}$ for any $i \geq 3$. Its characteristic series is $\phi(y) = 3+y+2y^2+\frac{y^3}{1-y} = \frac{1}{1-y}+y^2+2$. Its radius of convergence is $\rho_\phi = 1$, and the positive solution of $\phi(\tau) - \tau\phi'(\tau) = 0$ is $\frac{\sqrt{5}-1}{2} \approx 0.618$. Then the characteristic series belong to the smooth inverse-function schema.

The following theorem is a key property in analytic combinatorics for the functions belonging to the smooth inverse-function schema. It is stated when $\phi(y)$ is *aperiodic*, i.e. when $\phi(y) = \psi(y^d)$ implies $d = 1$. This is not a real restriction as shown in [9, Ex. VI.17], and this implies that $L(z)$ is also aperiodic and then has a unique dominant singularity.

▶ **Theorem 5** (see [5, 9, 16])**.** *Let $L(z)$ that belongs to the smooth inverse-function schema with $L(z) = z\phi(L(z))$. Assume that $\phi(y)$ is aperiodic, then $L(z)$ has a unique dominant singularity $\rho_L = \tau/\phi(\tau)$. Moreover, there exist two analytic functions $g(z)$ and $h(z)$ analytic at $z = \rho_L$, such that, around $z = \rho_L$ we have:*

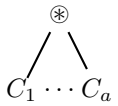
$$L(z) = g(z) - h(z)\sqrt{1 - z/\rho_L}, \text{ with } h(\rho_L) \neq 0. \tag{7}$$

Moreover, we have $[z^n]L(z) \sim C_L \frac{\rho_L^{-n}}{n^{3/2}}$, for some computable positive constant C_L.

3 Main results

3.1 Simplification using an absorbing pattern: main result

Let $\mathcal{L}$ be a set of combinatorial expressions of sets of labels $(\mathcal{A}_i)_{i\geq 0}$. Let $\mathcal{P}$ be an element of $\mathcal{L}$ and let $\circledast$ be an element of $\mathcal{A}_a$ for some $a \geq 2$. If $L \in \mathcal{L}$, the *simplification of L following the absorbing pattern $\mathcal{P}$ for $\circledast$* is the element $S \in \mathcal{L}$ obtained by applying bottom-up the following rewriting rule:



$\rightsquigarrow \mathcal{P}$, whenever $C_i = \mathcal{P}$ for some $i \in \{1, \ldots, a\}$.

More formally, the simplification $s(L, \mathcal{P}, \circledast)$ of L following the absorbing pattern $\mathcal{P}$ for $\circledast$ is inductively defined by: $s(L, \mathcal{P}, \circledast) = L$ if L has size 1, $s(L, \mathcal{P}, \circledast) = \mathcal{P}$ if the root of L is $\circledast$ and at least one of its children C is such that $s(C, \mathcal{P}, \circledast) = \mathcal{P}$, and

$$s(L, \mathcal{P}, \circledast) = (\oplus, s(C_1, \mathcal{P}, \circledast), \ldots, s(C_d, \mathcal{P}, \circledast)), \text{ otherwise for } L = (\oplus, C_1, \ldots, C_d).$$

Of course, this simplification is purely syntactic, but the idea behind is that, when interpreted, the pattern $\mathcal{P}$ represents an absorbing element for the operator $\circledast$, hence the name. For instance, the pattern $\mathcal{P} = (a + b)^\star$ and operator $\circledast = +$ correspond to the fact that for languages, $\mathcal{P}$ represents $\Sigma^\star$, which is absorbing for the union $+$.

Let $\sigma_{\mathcal{P},\circledast}(L) = |s(L, \mathcal{P}, \circledast)|$ denote the size of the simplification of L, or just $\sigma(L)$ when there is no ambiguity about the $\mathcal{P}$ and the $\circledast$ considered. Our main result concerns the random variable σ when L is taken uniformly at random in $\mathcal{L}_n$.

▶ **Theorem 6.** *Let $\mathcal{L}$ be a set of combinatorial expressions whose OGS $L(z)$ belongs to the smooth inverse-function schema $L(z) = \phi(L(z))$, with ϕ aperiodic. Let $\mathcal{P} \in \mathcal{L}$ and let $\circledast$ be an operator of arity at least 2 of $\mathcal{L}$. Then the expected size of the simplification of a uniform random expression of size n in $\mathcal{L}$ following the absorbing pattern $\mathcal{P}$ for $\circledast$ tends to a constant as n tends to infinity: $\lim_{n\to\infty} \mathbb{E}_n[\sigma] = \delta$, for some positive δ. Furthermore, all moments of σ also converge: for every $i \in \mathbb{Z}_{\geq 1}$, $\lim_{n\to\infty} \mathbb{E}_n[\sigma^i] = \delta_i$ for some positive δ_i.*

Observe that it is not because the expectation converges that the moments converge too. Moreover, convergence of the moments has an implication in the analysis of algorithm, as stated in the following corollary.

▶ **Corollary 7.** *Let $\mathcal{L}$ be a set of expressions that satisfies the conditions of Theorem 6. Consider a polynomial time algorithm working on elements of $\mathcal{L}$. If we first simplify the expression before running the algorithm, then the process takes $\mathcal{O}(1)$ expected time. The simplification itself is done in linear expected time.*

Hence, the automaton construction studied in [18] is useless in the uniform random setting, since the output automaton has size polynomial in the input in the worst case. It is thus of expected constant size if we first simplify the input, according to Corollary 7.

We fix $\mathcal{L}$, ϕ, $\mathcal{P}$ and $\circledast$ for the remainder of Section 3, which is devoted to the proof of our main theorem, i.e., Theorem 6. A discussion of its consequences can be found in Section 4.

3.2 Completely reducible expression trees

An expression tree $L \in \mathcal{L}$ is *completely reducible* when $s(L, \mathcal{P}, \circledast) = \mathcal{P}$. We first study the set $\mathcal{R} \subset \mathcal{L}$ of all the completely reducible expression trees and denote by $R(z)$ its OGS.

Let $p = |\mathcal{P}|$ and let a be the arity of $\circledast$. An element of $\mathcal{R}$ is either $\mathcal{P}$ itself, or a tree of root $\circledast$ such that at least one of its subexpression is completely reducible. Hence, translating into OGS, we have the following equation for $R(z)$:

$$R(z) = z^p + z\left((L(z))^a - (L(z) - R(z))^a\right) . \tag{8}$$

We can also easily establish the following property:

▶ **Lemma 8.** *The radii of convergence of $R(z)$ and $L(z)$ are equal. Moreover, there exists $N_R \in \mathbb{Z}_{\geq 0}$ such that for every $n \geq N_R$, $[z^n]R(z) > 0$.*

We need to be more precise, as a fine grain description of $R(z)$ is an important ingredient of our proof of Theorem 6. We use an advanced analytic result established[4] by Drmota [5, Th. 1 and Prop. 3]. His theorem provides a precise estimate of the solutions of strongly connected systems of equations involving analytic functions around their (then) shared dominant singularity.

▶ **Proposition 9.** *The probability that a uniform random expression in $\mathcal{L}_n$ is completely reducible tends to a strictly positive constant γ as $n \to \infty$. Furthermore, locally around $z = \rho$, we have*

$$R(z) = g_R(z) - h_R(z)\sqrt{1 - \frac{z}{\rho}}, \tag{9}$$

where ρ is the radius of convergence of $L(z)$ and $R(z)$, and the functions $g_R(z)$ and $h_R(z)$ are analytic around $z = \rho$, and $h_R(\rho) \neq 0$.

Proof sketch. We define a complementary class $\mathcal{G} := \mathcal{L} \setminus \mathcal{R}$ of elements that do not reduce (completely) to $\mathcal{P}$. Let $G(z)$ denote its OGS. We observe that from Eq. (8) we have

$$\begin{cases} R(z) = & z^p + z\Big((G(z) + R(z))^a - G(z)^a\Big), \\ G(z) = & z\,\underline{\phi}\big(G(z) + R(z)\big) + zG(z)^a, \end{cases} \tag{10}$$

where $\underline{\phi}(x) = \phi(x) - x^a$. We demonstrate, simultaneously, that $R(z)$ and $G(z)$ verify the conclusions of the proposition by applying Drmota's multidimensional version of Theorem 5 (see [5, Thm. 1 and Prop. 3]). Thus we consider the implicit system above, in the variables z, R and G, and prove that it satisfies the conditions of Drmota's Theorem. This will then imply immediately our desired conclusions. We consider the corresponding algebraic system

$$[R, G] = \boldsymbol{F}(z, R, G), \text{ where } \boldsymbol{F}(z, R, G) = \Big[z^p + z\big((G + R)^a - G^a\big), \quad z\,\underline{\phi}\big(G + R\big) + zG^a\Big].$$

Now we need to check that all the hypotheses of Drmota's theorem are satisfied; this includes studying the Jacobian matrix of the system, which is

$$\boldsymbol{J}(z, R, G) = \begin{pmatrix} za(G + R)^{a-1} & za((G + R)^{a-1} - G^{a-1}) \\ z\underline{\phi}'(G + R) & z\underline{\phi}'(G + R) + zaG^{a-1} \end{pmatrix}.$$

Once done, we obtain Eq. (9).

From this formula and the Transfer Theorem, we conclude that $[z^n]R(z)$ satisfies an asymptotic formula similar to the one of $[z^n]L(z)$ in Theorem 5, but with a different multiplicative constant. This yields the existence of γ and thus concludes the proof. ◀

[4] It turns out that Drmota's theorem needs a little rectification, given in [1]. The theorem, among other things, gives a way to localize the dominant singularity, which was proved wrong. Fortunately, we do not need to rely on this to characterize the singularity in our case, so there is no issue for us, as we do not use the incorrect part of this result.

▶ Remark 10. If we had several reduction rules with the same pattern, rather than just one, the conclusions from Proposition 9 remain true. For instance, for regular expressions, we could add the reduction rule $\mathcal{P}^\star \rightsquigarrow \mathcal{P}$ in addition to the fact that $\mathcal{P}$ is absorbing for $+$.

For this we introduce an auxiliary series $A(x) = \sum_{i\geq 1} a_i x^i$ where a_i is the number of operations of arity i having $\mathcal{P}$ as an absorbing element. The system in the proof of Proposition 9 rewrites

$$\begin{cases} R(z) = z^p + z\Big(A(G(z)+R(z)) - A(G(z))\Big)\,, \\ G(z) = z\,\underline{\phi}\big(G(z)+R(z)\big) + zA(G(z))\,, \end{cases} \tag{11}$$

with $\underline{\phi}(x) = \phi(x) - A(x)$. The limit probability γ of Proposition 9 admits an elegant formula

$$\gamma = \frac{A'(\tau_L) - A'(\tau_L - \tau_R)}{\phi'(\tau_L) - A'(\tau_L - \tau_R)}, \text{ where } \tau_L = L(\rho) \text{ and } \tau_R = R(\rho).$$

3.3 Estimating the expectation and the moments

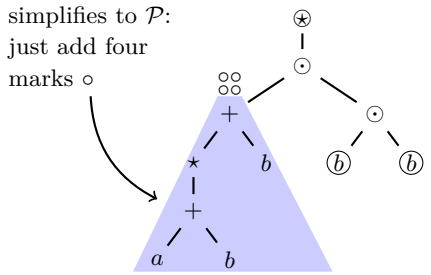
Following the method explained in Section 2.1, we use bivariate generating series to keep track of the size of the reduced expression. We therefore define $L(z,u)$ and $R(z,u)$ by (recall that $\sigma(L)$ is the size of the simplification of L):

$$L(z,u) = \sum_{L\in\mathcal{L}} z^{|L|}u^{\sigma(L)} \text{ and } R(z,u) = \sum_{R\in\mathcal{R}} z^{|R|}u^{\sigma(R)}.$$

Of course, since all elements of $\mathcal{R}$ simplify to $\mathcal{P}$, we have $R(z,u) = u^p R(z)$, with $p = |\mathcal{P}|$.

As in Section 2.1, we want to mark the expressions so that the number of marks is equal to the size of the simplified expression. For this, we put one mark on the nodes that are not simplified, and p marks on the root of a subtree that simplifies to $\mathcal{P}$, to take into account the size of $\mathcal{P}$ once reduced. We just have to reformulate the combinatorial specification and to enrich it with marks. As an example, let us consider regular expressions $\mathcal{L}_R$, with $\mathcal{P} = (a+b)^\star$ and $\circledast = +$. We mark an element $L \in \mathcal{L}_R$ the following way (we denote by $\mathcal{R}_R$ the completely reducible regular expressions):

- If $L \in \mathcal{R}_R$ then we add p marks at the root.
- If the root of L is not $+$, then we mark its children inductively (if any), and mark the root of L.
- If the root of L is $+$ and its children are all not in $\mathcal{R}_R$, then we mark its children inductively, and mark the root of L.



This expression has size 9 after simplificiation.

This way of rewriting the decomposition of $\mathcal{L}_R$ is unambiguous except for $\mathcal{P}$ which is in $\mathcal{R}_R$ and which can also be built by the standard induction, so we have to remove it once in the specification. This yields, where $\mathcal{L}_R^\circ$ and $\mathcal{R}_R^\circ$ respectively denote the marked versions of $\mathcal{L}_R$ and $\mathcal{R}_R$, and where $\mathcal{G}_R^\circ := \mathcal{L}_R^\circ \setminus \mathcal{R}_R^\circ$:

$$\mathcal{L}_R^\circ = (\mathcal{R}_R - \mathcal{P})^{\circ\circ}_{\circ\circ} + \textcircled{a} + \textcircled{b} + \textcircled{c} + \overset{\circledast}{|}_{\mathcal{L}_R^\circ} + \overset{\odot}{/\!\backslash}_{\mathcal{L}_R^\circ\ \mathcal{L}_R^\circ} + \overset{\oplus}{/\!\backslash}_{\mathcal{G}_R^\circ\ \mathcal{G}_R^\circ}.$$

From this we get $L_R(z,u) = (R(z) - z^4)u^4 + 3zu + uzL_R(z,u) + zuL_R(z,u)^2 + zuG_R(z,u)^2$.

This construction generalizes easily, and we get the following lemma.

▶ **Lemma 11.** *We have the following equation for the BGS $L(z,u)$ where z counts the size of an expression and u the size of its simplification:*

$$L(z,u) = (R(z) - z^p)u^p + zu\big(\underline{\phi}(L(z,u)) + (L(z,u) - R(z,u))^a\big), \tag{12}$$

where $\underline{\phi}(x) = \phi(x) - x^a$ and where a is the arity of $\circledast$.

At this point, we want to apply Eq. (4) to compute the expectation of σ. Hence, we differentiate with respect to u, set $u = 1$ and re-arrange the expression, obtaining

$$\frac{d}{du}L(z,u)\Big|_{u=1} = \frac{N\big(z, L(z), R(z)\big)}{D\big(z, L(z), R(z), \phi'(L(z))\big)}, \text{ where } N \text{ and } D \text{ are polynomials.} \tag{13}$$

From this, and using an adaptation of another result of Drmota [6, Lemma 2.26] to handle quotients, we get the following lemma.

▶ **Lemma 12.** *There exist two functions $\tilde{g}(z)$ and $\tilde{h}(z)$ that are analytic at $z = \rho$, such that locally around $z = \rho$ we have the expansion $\frac{d}{du}L(z,u)\Big|_{u=1} = \tilde{g}(z) - \tilde{h}(z)\sqrt{1 - z/\rho}$.*

We can now conclude the proof of the first part of Theorem 6: using the Transfer Theorem, we obtain from Lemma 12 that, if $\tilde{h}(\rho) > 0$, then $[z^n]\frac{d}{du}L(z,u)|_{u=1} \sim \frac{\tilde{h}(\rho)}{2\sqrt{\pi}}\rho^{-n}n^{-3/2}$, so that, by Theorem 5, $\mathbb{E}_n[\sigma]$ tends to the constant $\delta = \frac{\tilde{h}(\rho)}{2C_L\sqrt{\pi}}$, since $\rho = \rho_L$. The other cases are not possible, as $\tilde{h}(\rho) < 0$ yields a negative asymptotic equivalent and $\tilde{h}(\rho) = 0$ implies that the expected size of the simplification tends to 0, but it is not possible as it is at least $p > 0$.

▶ Remark 13. Using a computer algebra software, we can deal symbolically with all the formulas obtained in the article. When applied to regular expressions, we obtain that $\delta \approx 3,624,217.39$, which is a prohibitively large number for most simulations, while the proportion of completely reducible expressions tends to $\gamma \approx 0.0016336$.

The proof for higher moments follows a similar principle as for the expectation; when we differentiate several times on u Eq. (12) from Lemma 11, we get the same factor for the highest order derivative, hence the same denominator $D(z, L(z), R(z), \phi'(L(z)))$ as in Eq. (13). As the remaining terms also fulfill a local expansion like the one in Lemma 12, we can conclude exactly as for the expectation.

4 Conclusion

We have seen in this paper that uniform random expressions have to be considered with great care when using them to analyze the efficiency of algorithms or tools. As soon as there is an absorbing pattern, everything becomes mostly trivial, even for polynomial algorithms or constructions, as stated in Corollary 7.

One could alter the specifications to try to obtain a different behavior. For instance, by weighting some of the labels (trying to get less $\star$ in random expressions). This can easily be done directly on the specification, by adding several identical operators or even positive weights on the operators. The random generation can still be done quite efficiently. However, we have not changed the framework, and if we try to specify weighted regular expressions with, for instance:

$$\mathcal{L}_R = 10 \times a + 10 \times b + \varepsilon + \overset{\star}{|}_{\mathcal{L}_R} + 20 \times \overset{\bullet}{/\!\backslash}_{\mathcal{L}_R\ \mathcal{L}_R} + 15 \times \overset{+}{/\!\backslash}_{\mathcal{L}_R\ \mathcal{L}_R}, \tag{14}$$

we can still apply Theorem 6, we just get a larger limit for the expected size.

Another idea is to specify expressions using combinatorial systems. For instance, we can prevent consecutive stars in regular expressions using the system:

$$\mathcal{L}_R = \mathcal{S} + \mathcal{T}; \quad \mathcal{S} = \overset{\star}{|}_{T}; \quad \mathcal{T} = a + b + \varepsilon + \overset{\bullet}{/\!\backslash}_{\mathcal{L}_R\ \mathcal{L}_R} + \overset{+}{/\!\backslash}_{\mathcal{L}_R\ \mathcal{L}_R}.$$

This is not covered by our main theorem, but it is work in progress to prove that the same result holds, using Drmota's tools to handle systems of equations like this one.

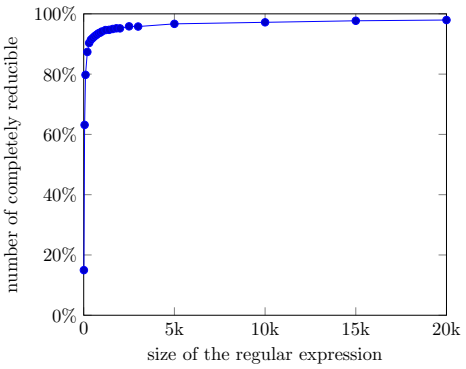


Figure 3 Experimental study of the proportion of completely reducible, for BST-like distribution, with $\mathbb{P}(\star) = \mathbb{P}(\bullet) = \mathbb{P}(+) = \frac{1}{3}$.

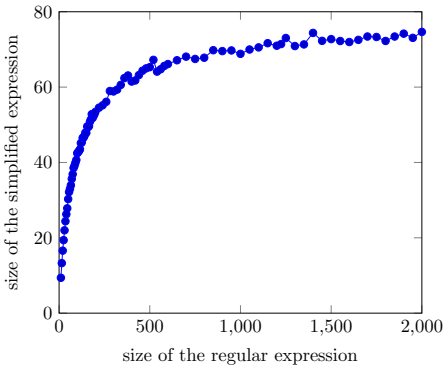


Figure 4 Experimental study of the size of a random uniform regular expression after applying a large set of simplification rules.

Yet a third idea is to completely change the distribution on the set of expressions. The other natural probability on trees is the so-called *BST-like* distribution, obtained on regular expressions as follows to generate an expression of size n: if $n = 1$ return a, b or ε; if $n = 2$ return $a^\star$, $b^\star$ or $\varepsilon^\star$; otherwise, randomly choose an operator in $\{\star, \bullet, +\}$ and if it is binary, choose the size of its left child uniformly in $\{1, \dots, n-2\}$. This distribution is not uniform, but it is often used for testing tools. See [19] for a theoretical study using this distribution on regular expressions. It is not clear if the conclusion of Theorem 6 still holds for these distributions, but we believe that they are not interesting either: on Fig. 3, we experimentally computed the ratio of expressions that can be simplified into $(a + b)^\star$, using a whole set of simplification rules, and obtained that this ratio tends to 1 as n tends to infinity: the situation is even worse than in the uniform case (see also [3, 4]).

If we apply this whole set of simplification rules to uniform random regular expressions, we get the curve depicted in Fig. 4. Apparently, the expected size of the simplified expression tends to a value around 75, which is large, but much smaller than the value 3,624,217 obtained for just using the absorbing pattern.

The real conclusion is that natural distributions on trees seem to be useless when it comes to obtaining good distributions on formulas. At this point, we have no idea on how to produce a good distribution in the general framework, sufficiently simple to have efficient random samplers and to be amenable to mathematical analysis.

References

1 Jason P. Bell, Stanley Burris, and Karen A. Yeats. Characteristic Points of Recursive Systems. *Electr. J. Comb.*, 17(1), 2010.

2 Sabine Broda, António Machiavelo, Nelma Moreira, and Rogério Reis. Average Size of Automata Constructions from Regular Expressions. *Bulletin of the EATCS*, 116, 2015.

3 Nicolas Broutin and Cécile Mailler. And/or trees: A local limit point of view. *Random Struct. Algorithms*, 53(1):15–58, 2018. doi:10.1002/rsa.20758.

4 Brigitte Chauvin, Danièle Gardy, and Cécile Mailler. A sprouting tree model for random boolean functions. *Random Struct. Algorithms*, 47(4):635–662, 2015. doi:10.1002/rsa.20567.

5 Michael Drmota. Systems of functional equations. *Random Struct. Algorithms*, 10(1-2):103–124, 1997.

6 Michael Drmota. *Random Trees: An Interplay Between Combinatorics and Probability*. Springer Publishing Company, Incorporated, 1st edition, 2009.

7 Philippe Duchon, Philippe Flajolet, Guy Louchard, and Gilles Schaeffer. Boltzmann Samplers for the Random Generation of Combinatorial Structures. *Combinatorics, Probability & Computing*, 13(4-5):577–625, 2004.

8 Philippe Flajolet and Andrew M. Odlyzko. The Average Height of Binary Trees and Other Simple Trees. *J. Comput. Syst. Sci.*, 25(2):171–213, 1982. doi:10.1016/0022-0000(82)90004-6.

9 Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009. URL: http://www.cambridge.org/uk/catalogue/catalogue.asp?isbn=9780521898065.

10 Philippe Flajolet, Paolo Sipala, and Jean-Marc Steyaert. Analytic Variations on the Common Subexpression Problem. In Mike Paterson, editor, *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, UK, July 16-20, 1990, Proceedings*, volume 443 of *Lecture Notes in Computer Science*, pages 220–234. Springer, 1990. doi:10.1007/BFb0032034.

11 Philippe Flajolet and Jean-Marc Steyaert. A Complexity Calculus for Recursive Tree Algorithms. *Mathematical Systems Theory*, 19(4):301–331, 1987. doi:10.1007/BF01704918.

12 Danièle Gardy. Random Boolean expressions. In David, René, Gardy, Danièle, Lescanne, Pierre, Zaionc, and Marek, editors, *Computational Logic and Applications, CLA '05*, volume DMTCS Proceedings vol. AF, Computational Logic and Applications (CLA '05) of *DMTCS Proceedings*, pages 1–36, Chambéry, France, 2005. Discrete Mathematics and Theoretical Computer Science. URL: https://hal.inria.fr/hal-01183339.

13 Antoine Genitrini and Bernhard Gittenberger. No Shannon effect on probability distributions on Boolean functions induced by random expressions. In *Discrete Mathematics and Theoretical Computer Science*, pages 303–316. Discrete Mathematics and Theoretical Computer Science, 2010.

14 Antoine Genitrini, Bernhard Gittenberger, and Cécile Mailler. No Shannon effect induced by And/Or trees. In *25th International Meeting on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms*, pages 109–120, 2014.

15 Jonathan Lee and Jeffrey Shallit. Enumerating Regular Expressions and Their Languages. In Michael Domaratzki, Alexander Okhotin, Kai Salomaa, and Sheng Yu, editors, *Implementation and Application of Automata, 9th International Conference, CIAA 2004, Kingston, Canada, July 22-24, 2004*, volume 3317 of *Lecture Notes in Computer Science*, pages 2–22. Springer, 2004.

16 A Meir and J.W Moon. On an asymptotic method in enumeration. *Journal of Combinatorial Theory, Series A*, 51(1):77–89, 1989. doi:10.1016/0097-3165(89)90078-2.

17 Michel Nguyên-Thê. *Distribution of Valuations on Trees*. Theses, Ecole Polytechnique X, February 2004. URL: https://pastel.archives-ouvertes.fr/pastel-00000839.

18 Cyril Nicaud. On the Average Size of Glushkov's Automata. In Adrian-Horia Dediu, Armand-Mihai Ionescu, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications, Third International Conference, LATA 2009, Tarragona, Spain, April 2-8, 2009. Proceedings*, volume 5457 of *Lecture Notes in Computer Science*, pages 626–637. Springer, 2009.

19 Cyril Nicaud, Carine Pivoteau, and Benoît Razet. Average Analysis of Glushkov Automata under a BST-Like Model. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 388–399. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

20 Albert Nijenhuis and Herbert S Wilf. Combinatorial algorithms for computers and calculators. *Computer Science and Applied Mathematics, New York: Academic Press, 1978, 2nd ed.*, 1978.

21 Mike Paterson, editor. *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, UK, July 16-20, 1990, Proceedings*, volume 443 of *Lecture Notes in Computer Science*. Springer, 1990. doi:10.1007/BFb0032016.