On the Degeneracy of Random Expressions Specified by Systems of Combinatorial Equations

Florent Koechlin[1], Cyril Nicaud[1] and Pablo Rotondo[2]

[1] LIGM, Univ Gustave Eiffel, CNRS, ENPC, firstname.name@u-pem.fr
[2] LITIS, Université de Rouen, pablo.rotondo@univ-rouen.fr

Abstract. We consider general expressions, which are trees whose nodes are labeled with operators, that represent syntactic descriptions of formulas. We assume that there is an operator that has an absorbing pattern and prove that if we use this property to simplify a uniform random expression with n nodes, then the expected size of the result is bounded by a constant. In our framework, expressions are defined using a combinatorial system, which describes how they are built: one can ensure, for instance, that there are no two consecutive stars in regular expressions. This generalizes a former result where only one equation was allowed, confirming the lack of expressivity of uniform random expressions.

1 Introduction

This article is the sequel of the work started in [10], where we investigate the lack of expressivity of uniform random expressions. In our setting, we use the natural encoding of expressions as trees, which is a convenient way to manipulate them both in theory and in practice. In particular, it allows us to treat many different kinds of expressions at a general level (see Fig. 1 below): regular expressions, arithmetic expressions, boolean formulas, LTL formulas, and so on.

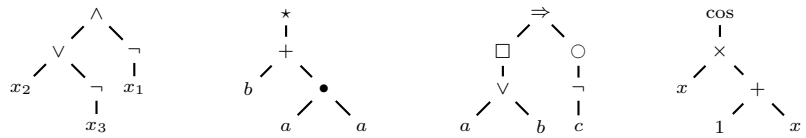

Fig. 1. Four expression trees and their associated formulas. From left to right: a logical formula, a regular expression, an LTL formula and a function.

For this representation, some problems are solved using a simple traversal of the tree: for instance, testing whether the language of a regular expression contains the empty word, or formally differentiating a function. Sometimes however, the tree is not the best way to encode the object it represents in the computer,

and we transform it into an equivalent adequate structure; in the context of formal languages, a regular expression (encoded using a tree) is typically transformed into an automaton, using one of the many known algorithms such as the Thompson construction or the Glushkov automaton.

In our setting, we assume that one wants to estimate the efficiency of an algorithm, or a tool, whose inputs are expressions. The classical theoretical framework consists in analyzing the worst case complexity, but there is often some discrepancy between this measure of efficiency and what is observed in practice. A practical approach consists in using benchmarks to test the tool on real data. But in many contexts, having access to good benchmarks is quite difficult. An alternative to these two solutions is to consider the average complexity of the algorithm, which is sometimes amenable to a mathematical analysis, and which can be studied experimentally, provided we have a random generator at hand. Going that way, we have to choose a probability distribution on size-n inputs, which can be difficult: we would like to study a "realistic" probability distribution that is also mathematically tractable. When no specific random model is available, it is classical to consider the uniform distribution, where all size-n inputs are equally likely. In many frameworks, such as sorting algorithms, studying the uniform distribution yields useful insights on the behavior of the algorithm.

Following this idea, several works have been undertaken on uniform random expressions, in various contexts. Some are done at a general level: the expected height of a uniform random expression [12] always grows in $\Theta(\sqrt{n})$, if we identify common subexpressions then the expected size of the resulting acyclic graph [7] is in $\Theta(\frac{n}{\sqrt{\log n}})$, ... There are also more specific results on the expected size of the automaton built from a uniform random regular expression, using various algorithms [14, 4]. In another setting, the expected cost of the computation of the derivative of a random function was proved to be in $\Theta(n^{3/2})$, both in time and space [8]. There are also a lot of results on random boolean formulas, but the framework is a bit different (for a more detailed account on this topic, we refer the interested reader to Gardy's survey [9]).

In [10], we questioned the model of uniform random expressions. Let us illustrate the main result of [10] on an example, regular expressions on the alphabet $\{a, b\}$. The set $\mathcal{L}_{\mathcal{R}}$ of regular expressions is inductively defined by

$$\mathcal{L}_{\mathcal{R}} = a + b + \varepsilon + \overset{\star}{|}_{\mathcal{L}_{\mathcal{R}}} + \overset{\bullet}{/\!\backslash}_{\mathcal{L}_{\mathcal{R}}\ \mathcal{L}_{\mathcal{R}}} + \overset{+}{/\!\backslash}_{\mathcal{L}_{\mathcal{R}}\ \mathcal{L}_{\mathcal{R}}}. \qquad (\star)$$

The formula above is an equation on trees, where the size of a tree is its number of nodes. In particular a, b and ε represent trees of size 1, reduced to a leaf, labeled accordingly. As one can see from the specification $(\star)$, leaves have labels in $\{a, b, \varepsilon\}$, unary nodes are labeled by $\star$ and binary nodes by either the concatenation $\bullet$ or the union $+$. Observe that the regular expression $\mathcal{P}$ corresponding to $(a+b)^\star$ denotes the regular language $\{a, b\}^\star$ of all possible words. This language is absorbing for the union operation on regular languages. So if we start with a regular expression $\mathcal{R}$ (a tree), identify every occurrence of the pattern $\mathcal{P}$ (a subtree), then rewrite the tree (bottom-up) by using inductively the simplifications

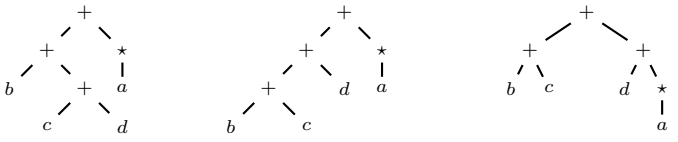

Fig. 2. Three regular expression trees whose denoted languages are equal because of the associativity of the union.

$\overset{+}{\underset{\mathcal{X}\ \mathcal{P}}{/\backslash}} \to \mathcal{P}$ and $\overset{+}{\underset{\mathcal{P}\ \mathcal{X}}{/\backslash}} \to \mathcal{P}$, this results in a *simplified tree* $\sigma(\mathcal{R})$ that denotes the same regular language. Of course, other simplifications could be considered, but we focus on this particular one. The theorem we proved in [10] implies that if one takes uniformly at random a regular expression of size n and applies this simplification algorithm, then the expected size of the resulting equivalent expression tends to a constant! It means that the uniform distribution on regular expressions produces a degenerated distribution on regular languages. More generally, we proved that: *For every class of expressions that admits a specification similar to Eq.* $(\star)$ *and such that there is an absorbing pattern for some of the operations, the expected size of the simplification of a uniform random expression of size* n *tends to a constant as* n *tends to infinity.*[3] This negative result is quite general, as most examples of expressions have an absorbing pattern: for instance $x \wedge \neg x$ is always `false`, and therefore absorbing for $\wedge$.

The statement of the main theorem of [10] is general, as it can be used to discard the uniform distribution for expressions defined inductively as in Eq. $(\star)$. However it is limited to that kind of simple specifications. And if we take a closer look at the regular expressions from $\mathcal{L}_\mathcal{R}$, we observe that nothing prevents, for instance, useless sequences of nested stars as in $(((a + bb)^\star)^\star)^\star$. It is natural to wonder whether the result of [10] still holds when we forbid two consecutive stars in the specification. We could also use the associativity of the union to prevent different representations of the same language, as depicted in Fig. 2, or many other properties, to try to reduce the redundancy at the combinatorial level.

This is the question we investigate in this article: does the degeneracy phenomenon of [10] still hold for more advanced combinatorial specifications? More precisely, we now consider specifications made using a system of (inductive) combinatorial equations, instead of only one as in Eq. $(\star)$. For instance, we can forbid consecutive stars using the combinatorial system:

$$\begin{cases} \mathcal{L}_\mathcal{R} = \overset{\star}{|}_{\mathcal{S}} + \mathcal{S}, \\ \mathcal{S} = a + b + \varepsilon + \overset{+}{\underset{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}}{/\backslash}} + \overset{\bullet}{\underset{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}}{/\backslash}}. \end{cases} \qquad (\star\star)$$

[3] The idea behind our work comes from a very specific analysis of and/or formulas established in Nguyên Thê PhD's dissertation [13, Ch 4.4].

The associativity of the union (Fig. 2) can be taken into account by preventing the right child of any +-node from being also labeled by +. Clearly, systems cannot be used for forbidding intricated patterns, but they still greatly enrich the families of expressions we can deal with. Moreover that kind of systems, which has strong similarities with context-free grammars, is amenable to analytic techniques as we will see in the sequel; this was for instance used by Lee and Shallit to estimate the number of regular languages in [11].

Our contributions can be described as follows. We consider expressions defined by systems of combinatorial equations (instead of just one equation), and establish a similar degeneracy result: if there is an absorbing pattern, then the expected reduced size of a uniform random expression of size n is upper bounded by a constant as n tends to infinity.[4] Hence, even if we use the system to remove redundancy from the specification (e.g., by forbidding consecutive stars), uniform random expressions still lack expressivity. Technically, we once again rely on the framework of analytic combinatorics for our proofs. However, the generalization to systems induces two main difficulties. First, we are not dealing with the well-known *varieties of simple trees* anymore [6, VII.3], so we have to rely on much more advanced techniques of analytic combinatorics; this is sketched in Section 5. Second, some work is required on the specification itself, to identify suitable hypotheses for our theorem; for instance, it is easy from the specification to prevent the absorbing pattern from appearing as a subtree at all, in which case our statement does not hold anymore, since there are no simplifications taking place.

Due to the lack of space, the analytic proofs are only sketched or omitted in this extended abstract: we chose to focus on the discussion on combinatorial systems (Section 3) and on the presentation of our framework (Section 4).

2 Basic Definitions

For a given positive integer n, $[n] = \{1, \dots, n\}$ denotes the set of the first n positive integers. If E is a finite set, $|E|$ denotes its cardinality.

A *combinatorial class* is a set $\mathcal{C}$ equipped with a *size function* $|\cdot|$ from $\mathcal{C}$ to $\mathbb{N}$ (the size of $C \in \mathcal{C}$ is $|C|$) such that for any $n \in \mathbb{N}$, the set $\mathcal{C}_n$ of size-n elements of $\mathcal{C}$ is finite. Let $C_n = |\mathcal{C}_n|$, the *generating series* $C(z)$ of $\mathcal{C}$ is the formal power series defined by

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

Generating series are tools of choice to study combinatorial objects. When their radius of convergence is not zero, they can be viewed as analytic function from $\mathbb{C}$ to $\mathbb{C}$, and very useful theorems have been developed in the field of analytic combinatorics [6] to, for instance, easily obtain an asymptotic equivalent to C_n. We rely on that kind of techniques in Section 5 to prove our main theorem.

[4] The result holds for natural yet technical conditions on the system.

If $C(z) = \sum_{n\geq 0} C_n z^n$ is a formal power series, let $[z^n]C(z)$ denote its n-th coefficient C_n. Let ξ be a *parameter* on the combinatorial class $\mathcal{C}$, that is, a mapping from $\mathcal{C}$ to $\mathbb{N}$. Typically, ξ stands for some statistic on the objects of $\mathcal{C}$: the number of cycles in a permutation, the number of leaves in a tree, ... We define the *bivariate generating series* $C(z,u)$ associated with $\mathcal{C}$ and ξ by:

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

where $C_{n,k}$ is the number of size-n elements C of $\mathcal{C}$ such that $\xi(C) = k$. In particular, $C(z) = C(z,1)$. Bivariate generating series are useful to obtain information on ξ, such as its expectation or higher moments. Indeed, if $\mathbb{E}_n[\xi]$ denotes the expectation of ξ for the uniform distribution on $\mathcal{C}_n$, i.e. where all the elements of size n are equally likely, a direct computation yields:

$$\mathbb{E}_n[\xi] = \frac{[z^n]\partial_u C(z,u)\big|_{u=1}}{[z^n]C(z)}, \tag{1}$$

where $\partial_u C(z,u)\big|_{u=1}$ consists in first differentiating $C(z,u)$ with respect to u, and then setting $u = 1$. Hence, if we have an expression for $C(z,u)$ we can estimate $\mathbb{E}_n[\xi]$ if we can estimate the coefficients of the series in Eq. (1).

In the sequel, the combinatorial objects we study are trees, and we will have methods to compute the generating series directly from their specifications. Then, powerful theorems from analytic combinatorics will be used to estimate the expectation, using Eq. (1). So we delay the automatic construction and the analytic treatment to their respective sections.

3 Combinatorial Systems of Trees

3.1 Definition of combinatorial expressions and of systems

In the sequel the only combinatorial objects we consider are *plane trees*. These are trees embedded in the plane, which means that the order of the children matters: the two trees and  are different. Every node is labeled by an element in a set of symbols and the *size* of a tree is its number of nodes.

More formally, let S be a finite set, whose elements are *operator symbols*, and let a be a mapping from S to $\mathbb{N}$. The value $a(s)$ is called the *arity*[5] of the operator s. An *expression over* S is a plane tree where each node of arity i is labeled by an element $s \in S$ such that $a(s) = i$ (leaves' symbols have arity 0).

Example 1. In Fig. 1, the first tree is an expression over $S = \{\wedge, \vee, \neg, x_1, x_2, x_3\}$ with $a(\wedge) = a(\vee) = 2$, $a(\neg) = 1$ and $a(x_1) = a(x_2) = a(x_3) = 0$.

[5] We do not use the term *degree*, because if the tree is viewed as a graph, the degree of a node is its arity plus one (except for the root).

An *incomplete expression over* S is an expression where (possibly) some leaves are labeled with a new symbol $\Box$ of arity 0. Informally, such a tree represents part of an expression, where the $\Box$-nodes need to be completed by being substituted by an expression. An incomplete expression with no $\Box$-leaf is called a *complete* expression, or just an expression. If T is an incomplete expression over S, its *arity* $a(T)$ is its number of $\Box$-leaves. It is consistent with the definition of the arity of a symbol, by viewing a symbol s of arity $a(s)$ as an incomplete expression made of a root labeled by s with $a(s)$ $\Box$-children: $\wedge$ is viewed as $\overset{\wedge}{\underset{\Box\ \Box}{/\backslash}}$. Let $\mathcal{T}_\Box(S)$ and $\mathcal{T}(S)$ be the set of incomplete and complete expressions over S.

If T is an incomplete expression over S of arity t, and $T_1, \ldots, T_t$ are expressions over S, we denote by $T[T_1, \ldots, T_t]$ the expression obtained by substituting the i-th $\Box$-leaf in depth-first order by T_i, for $i \in [t]$. This notation is generalized to sets of expressions: if $\mathcal{T}_1, \ldots, \mathcal{T}_t$ are sets of expressions then $T[\mathcal{T}_1, \ldots, \mathcal{T}_t] = \{T[T_1, \ldots, T_t] : T_1 \in \mathcal{T}_1, \ldots, T_t \in \mathcal{T}_t\}$.

A *rule* of dimension $m \geq 1$ over S is an incomplete expression $T \in \mathcal{T}_\Box(S)$ where each $\Box$-node is replaced by an integer of $[m]$. Alternatively, a rule can be seen as a tuple $\mathcal{M} = (T, i_1, \ldots, i_t)$, where T is an incomplete expression of arity t and $i_1, \ldots, i_t$ are the values placed in its $\Box$-leaves in depth-first order. The arity $a(\mathcal{M})$ of a rule $\mathcal{M}$ is the arity of its incomplete expression, and $\textsc{ind}(\mathcal{M}) = (i_1, \ldots, i_t)$ is the tuple of integer values obtained by a depth-first traversal of $\mathcal{M}$. A *combinatorial system of trees* $\mathcal{E} = \{E_1, \ldots, E_m\}$ of dimension m over S is a system of m set equations of complete trees in $\mathcal{T}(S)$: each E_i is a non-empty finite set of rules over S, and the system in variables $\mathcal{L}_1, \ldots, \mathcal{L}_m$ is:

$$\begin{cases} \mathcal{L}_1 = \displaystyle\bigcup_{(T, i_1, \ldots, i_t) \in E_1} T[\mathcal{L}_{i_1}, \ldots, \mathcal{L}_{i_t}] \\ \quad \vdots \\ \mathcal{L}_m = \displaystyle\bigcup_{(T, i_1, \ldots, i_t) \in E_m} T[\mathcal{L}_{i_1}, \ldots, \mathcal{L}_{i_t}]. \end{cases} \tag{2}$$

Example 2. To specify the system given in Eq. ($\star\star$) using our formalism, we have $m = 2$. Its tuples representation is: $E_1 = \left\{ \left(\overset{\star}{\underset{\Box}{|}}, 2 \right), (\Box, 2) \right\}$, and $E_2 = \left\{ \left(\overset{\bullet}{\underset{\Box\ \Box}{/\backslash}}, 1, 1 \right), \left(\overset{+}{\underset{\Box\ \Box}{/\backslash}}, 1, 1 \right), (a), (b), (\varepsilon) \right\}$, and its equivalent tree representation is $E_1 = \left\{ \overset{\star}{\underset{2}{|}} \right\}$, and $E_2 = \left\{ \overset{\bullet}{\underset{1\ 1}{/\backslash}}, \overset{+}{\underset{1\ 1}{/\backslash}}, a, b, \varepsilon \right\}$, which corresponds to Eq. ($\star\star$) with $\mathcal{L}_\mathcal{R} = \mathcal{L}_1$ and $\mathcal{S} = \mathcal{L}_2$. In practice, we prefer descriptions as in Eq. ($\star\star$), which are easier to read, but they are all equivalent.

3.2 Generating series

If the system is not ambiguous, that is, if $\mathcal{L}_1, \ldots, \mathcal{L}_m$ is the[6] solution of the system and every tree in every $\mathcal{L}_i$ can be uniquely built from the specification,

[6] In all generalities, there can be several solutions to a system, but the conditions we will add prevent this from happening.

then the system can be directly translated into a system of equations on the generating series. This is a direct application of the *symbolic method* in analytic combinatorics [6, Part A] and we get the system

$$\begin{cases} L_1(z) = \displaystyle\sum_{(T,i_1,\dots,i_{a(T)})\in E_1} z^{|T|-a(T)} L_{i_1}(z)\cdots L_{i_{a(T)}}(z) \\ \qquad\vdots \\ L_m(z) = \displaystyle\sum_{(T,i_1,\dots,i_{a(T)})\in E_m} z^{|T|-a(T)} L_{i_1}(z)\cdots L_{i_{a(T)}}(z). \end{cases} \tag{3}$$

where $L_i(z)$ is the generating series of $\mathcal{L}_i$. If the system is ambiguous, the $L_i(z)$'s still have a meaning: each expression of $\mathcal{L}_i$ accounts for the number of ways it can be derived from the system. When the system is unambiguous, there is only one way to derive each expression, and $L_i(z)$ is the generating series of $\mathcal{L}_i$.

3.3 Designing practical combinatorial systems of trees

Systems of trees such as Eq. (2) are not always well-founded. Sometimes they are, but still contain unnecessary equations. It is not the topic of this article to fully characterize when a system is correct, but we nonetheless need sufficient conditions to ensure that our results hold: in this section, we just present examples to underline some bad properties that might happen. For a more detailed account on combinatorial systems, the reader is referred to [1, 8, 16].

Ambiguity. As mentioned above, the system can be ambiguous, in which case the combinatorial system cannot directly be translated into a system of generating series. This is a case for instance for the following system

$$\begin{cases} \mathcal{L}_1 = a + \overset{\star}{\underset{\mathcal{L}_1}{|}} + \overset{\star}{\underset{\mathcal{L}_2}{|}} \\ \mathcal{L}_2 = \overset{\star}{\underset{\mathcal{L}_1}{|}} + a + b + \varepsilon, \end{cases}$$

as the expression $\overset{\star}{\underset{a}{|}}$ can be produced in two ways for the component $\mathcal{L}_1$.

Empty components. Some specifications produce empty $\mathcal{L}_i$'s. For instance, consider the system $\left\{\mathcal{L}_1 = \overset{\bullet}{\underset{\mathcal{L}_1}{/}}\!\!\underset{\mathcal{L}_2}{\backslash};\ \mathcal{L}_2 = a + b + \varepsilon + \mathcal{L}_1\right\}$: its only solution is $\mathcal{L}_1 = \emptyset$ and $\mathcal{L}_2 = \{a, b, \varepsilon\}$.

Cyclic unit-dependency. The *unit-dependency graph* $\mathcal{G}_\square(\mathcal{E})$ of a system $\mathcal{E}$ is the directed graph of vertex set $[m]$, with an edge $i \to j$ whenever $(\square, j) \in E_i$. Such a rule is called a *unit rule*. It means that $\mathcal{L}_i$ directly depends on $\mathcal{L}_j$. For instance $\mathcal{L}_\mathcal{R}$ directly depends on $\mathcal{S}$ in Eq. ($\star\star$). We can work with systems having unit dependencies, provided the unit-dependency graph is acyclic. If it is not, then the equations forming a cycle are useless or badly defined for our purposes. Consider for instance the system and its unit-dependency graph depicted in Fig. 3.

Fig. 3. The unit-dependency graph is not acyclic, and there are infinitely many ways to derive a from $\mathcal{L}_2$: $\mathcal{L}_2 \to a$, $\mathcal{L}_2 \to \mathcal{L}_1 \to \mathcal{L}_2 \to a \ldots$

Not strongly connected. The *dependency graph* $\mathcal{G}(\mathcal{E})$ of the system $\mathcal{E}$ is the directed graph of vertex set $[m]$, with an edge $i \to j$ whenever there is a rule $\mathcal{M} \in E_i$ such that $j \in \text{IND}(\mathcal{M})$: $\mathcal{L}_i$ depends on $\mathcal{L}_j$ in the specification. Some parts of the system may be unreachable from other parts, which may bring up difficulties. A sufficient condition to prevent this from happening is to ask for the dependency graph to be strongly connected; it is not necessary, but this assumption will also be useful in the proof our main theorem. See Section 6 for a more detailed discussion on non-strongly connected systems. In Fig. 4 is depicted a system and its associated graph.

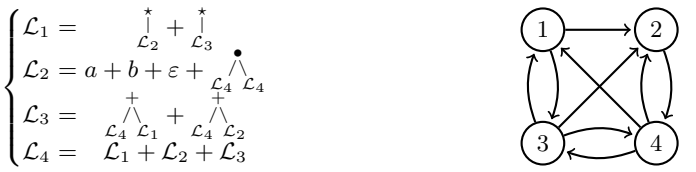

Fig. 4. A system and its associated dependency graph, which is strongly connected.

4 Settings, working hypothesis and simplifications

4.1 Framework

In this section, we describe our framework: we specify the kind of systems we are going to work with, and the settings for describing syntactic simplifications.

Let $\mathcal{E}$ be a combinatorial system of trees over S of dimension m of solution $(\mathcal{L}_1, \ldots, \mathcal{L}_m)$. A set of expressions $\mathcal{L}$ over S is *defined by* $\mathcal{E}$ if there exists a non-empty subset I of $[m]$ such that $\mathcal{L} = \cup_{i \in I} \mathcal{L}_i$.

From now on we assume that we are using a system $\mathcal{E}$ of dimension m over S and that S contains an operator $\circledast$ of arity at least 2. We furthermore assume that there is a complete expression $\mathcal{P}$, such that when interpreted, every expression of root $\circledast$ having $\mathcal{P}$ as a child is equivalent to $\mathcal{P}$: the interpretation of $\mathcal{P}$ is absorbing for the operator associated with $\circledast$. The expression $\mathcal{P}$ is the *absorbing pattern* and $\circledast$ is the *absorbing operator*.

Example 3. Our main example is $\mathcal{L}$ defined by the system of Eq. $(\star\star)$ with $\mathcal{L} = \mathcal{L}_\mathcal{R}$, the regular expressions with no two consecutive stars. As regular expressions, they are interpreted as regular languages. Since the language $(a+b)^\star$ is absorbing

for the union, we set the associated expression as the absorbing pattern $\mathcal{P}$ and the operator symbol $+$ as the absorbing operator.

The *simplification* of a complete expression T is the complete expression $\sigma(T)$ obtained by applying bottom-up the rewritting rule, where a is the arity of $\circledast$:

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

More formally, the simplification $\sigma(T, \mathcal{P}, \circledast)$ of T, or just $\sigma(T)$ when the context is clear, is inductively defined by: $\sigma(T) = T$ if T has size 1 and

$$\sigma((\oplus, C_1, \ldots, C_d)) = \begin{cases} \mathcal{P} & \text{if } \oplus = \circledast \text{ and } \exists i, \sigma(C_i) = \mathcal{P}, \\ (\oplus, \sigma(C_1), \ldots, \sigma(C_d)) & \text{otherwise.} \end{cases}$$

A complete expression T is *fully reducible* when $\sigma(T) = \mathcal{P}$.

We also need some conditions on the system $\mathcal{E}$. Some of them come from the discussion of Section 3.3, others are needed for the techniques from analytic combinatorics used in our proofs. A system $\mathcal{E}$ satisfies the hypothesis (**H**) when:

- (**H**$_1$) The graph $\mathcal{G}(\mathcal{E})$ is strongly connected and $\mathcal{G}_{\square}(\mathcal{E})$ is acyclic.
- (**H**$_2$) The system is *aperiodic*: there exists N such that for all $n \geq N$, there is at least one expression of size n in every coordinate of the solution $(\mathcal{L}_1, \ldots, \mathcal{L}_m)$ of the system.
- (**H**$_3$) For some j, there is a rule $T \in E_j$ of root $\circledast$, having at least two children T' and T'' such that: there is a way to produce a fully reducible expression from T' and $a(T'') \geq 1$.
- (**H**$_4$) The system is *not linear*: there is a rule of arity at least 2.
- (**H**$_5$) The system is *non-ambiguous*: each complete expression can be built in at most one way.

Conditions (**H**$_1$) and (**H**$_5$) were already discussed in Section 3.3. Condition (**H**$_4$) prevents the system from generating only lists (trees whose internal nodes have arity 1), or more generally families that grow linearly (for instance $\mathcal{L} = \mathcal{L} \overset{+}{\wedge} a + b$), which are degenerated. Without Condition (**H**$_3$) the system could be designed in a way that prevents simplifications (in which case our result does not hold, of course). Finally, Condition (**H**$_2$) is necessary to keep the analysis manageable (together with the strong connectivity of $\mathcal{G}(\mathcal{E})$ of Condition (**H**$_1$)).

4.2 Proper systems and system iteration

A combinatorial system of trees $\mathcal{E}$ is *proper* when it contains no unit rules and when the $\square$-leaves of all its rules have depth one (they are children of a root). In this section we establish the following preparatory proposition:

Proposition 1. *If $\mathcal{L}$ is defined by a system $\mathcal{E}$ that satisfies* (**H**)*, then there exists a proper system $\mathcal{E}'$ that satisfies* (**H**) *such that $\mathcal{L}$ is defined by $\mathcal{E}'$.*

Proposition 1 will be important in the sequel, as proper systems are easier to deal with for the analytic analysis. One key tool to prove Proposition 1 is the notion of *system iteration*, which consists in substituting simultaneously every integer-leaf i in each rule by all the rules of E_i. For instance, if we iterate once our recurring system $\{\mathcal{L}_1 = \overset{\star}{\underset{\mathcal{L}_2}{|}} + \mathcal{L}_2;\ \mathcal{L}_2 = a + b + \varepsilon + \overset{+}{\underset{\mathcal{L}_1}{/\backslash}}\overset{}{\underset{\mathcal{L}_1}{}} + \overset{\bullet}{\underset{\mathcal{L}_1}{/\backslash}}\overset{}{\underset{\mathcal{L}_1}{}}\}$, we get[7]

$$\begin{cases} \mathcal{L}_1 = \overset{\star}{\underset{a}{|}} + \overset{\star}{\underset{b}{|}} + \overset{\star}{\underset{\varepsilon}{|}} + \overset{\star}{\underset{\mathcal{L}_1}{|}}\overset{+}{\underset{\mathcal{L}_1}{/\backslash}}\overset{}{\underset{\mathcal{L}_1}{}} + \overset{\star}{\underset{\mathcal{L}_1}{|}}\overset{\bullet}{\underset{\mathcal{L}_1}{/\backslash}}\overset{}{\underset{\mathcal{L}_1}{}} + a + b + \varepsilon + \overset{+}{\underset{\mathcal{L}_1}{/\backslash}}\overset{}{\underset{\mathcal{L}_1}{}} + \overset{\bullet}{\underset{\mathcal{L}_1}{/\backslash}}\overset{}{\underset{\mathcal{L}_1}{}} \\ \mathcal{L}_2 = a + b + \varepsilon + \overset{+}{\underset{\mathcal{L}_2}{/\backslash}}\overset{}{\underset{\mathcal{L}_2}{}} + \overset{+}{\underset{\mathcal{L}_2}{/\backslash}}\overset{\star}{\underset{}{}}\overset{}{\underset{\mathcal{L}_2}{|}} + \overset{+}{\underset{\mathcal{L}_2}{/\backslash}}\overset{\star}{\underset{}{}}\overset{}{\underset{\mathcal{L}_2}{|}} + \overset{+}{\underset{\mathcal{L}_2}{/\backslash}}\overset{\star}{\underset{\mathcal{L}_2}{|}}\overset{\star}{\underset{\mathcal{L}_2}{|}} + \overset{\bullet}{\underset{\mathcal{L}_2}{/\backslash}}\overset{}{\underset{\mathcal{L}_2}{}} + \overset{\bullet}{\underset{\mathcal{L}_2}{/\backslash}}\overset{\star}{\underset{}{}}\overset{}{\underset{\mathcal{L}_2}{|}} + \overset{\bullet}{\underset{\mathcal{L}_2}{/\backslash}}\overset{\star}{\underset{}{}}\overset{}{\underset{\mathcal{L}_2}{|}} + \overset{\bullet}{\underset{\mathcal{L}_2}{/\backslash}}\overset{\star}{\underset{\mathcal{L}_2}{|}}\overset{\star}{\underset{\mathcal{L}_2}{|}}. \end{cases}$$

Formally, if we iterate $\mathcal{E} = \{E_1, \ldots, E_m\}$ once, then for all $i \in [m]$ we have

$$\mathcal{L}_i = \bigcup_{(T, i_1, \ldots, i_t) \in E_1} T\left[\bigcup_{(T_1, \mathbf{j_1}) \in E_{i_1}} T_1[\mathcal{L}_{j_{1,1}}, \ldots, \mathcal{L}_{j_{1,t_1}}], \ldots, \bigcup_{(T_t, \mathbf{j_t}) \in E_{i_t}} T_t[\mathcal{L}_{j_{t,1}}, \ldots, \mathcal{L}_{j_{t,t_t}}]\right]$$

where $\mathbf{j_1} = (j_{1,1}, \ldots, j_{1,t_1}), \ldots, \mathbf{j_t} = (j_{t,1}, \ldots, j_{t,t_t})$.
Let $\mathcal{E}^2$ denote the system obtained after iterating $\mathcal{E}$ once; it is called the *system of order 2* (from $\mathcal{E}$). More generally $\mathcal{E}^t$ is the system of order t obtained by iterating $t-1$ times the system $\mathcal{E}$. From the definition we directly get:

Lemma 1. *If $\mathcal{L}$ is defined by a system $\mathcal{E}$, it is also defined by all its iterates $\mathcal{E}^t$. Moreover, if $\mathcal{E}$ satisfies* (**H**)*, every $\mathcal{E}^t$ also satisfies* (**H**)*, except that $\mathcal{G}(\mathcal{E}^t)$ may not be strongly connected.*

We can sketch the proof of Proposition 1 as follows: since $\mathcal{G}_\square(\mathcal{E})$ is acyclic, we can remove all the unit rules by iterating the system sufficiently many times. By Lemma 1, we have to be cautious, and find an order t so that $\mathcal{G}^t$ is strongly connected: a study of the cycle lengths in $\mathcal{G}(\mathcal{E})$ ensures that such a t exists. So $\mathcal{L}$ is defined by $\mathcal{E}^t$, which has no unit rules and which satisfies (**H**). To transform $\mathcal{E}^t$ into an equivalent proper system, we have to increase the dimension to cut the rules as needed. It is better explained on an example:

$$\mathcal{L}_1 = \overset{\star}{\underset{\mathcal{L}_3}{|}} + \overset{\bullet}{\underset{\mathcal{L}_1}{\star/\backslash}}\overset{}{\underset{\mathcal{L}_1}{\mathcal{L}_2}} \quad \to \quad \begin{cases} \mathcal{L}_1 = \overset{\star}{\underset{\mathcal{L}_3}{|}} + \overset{\bullet}{\underset{\mathcal{K}}{/\backslash}}\overset{}{\underset{\mathcal{K}}{\mathcal{L}_2}} \\ \mathcal{K} = \overset{\star}{\underset{\mathcal{L}_1}{|}}. \end{cases}$$

This construction can be systematized. It preserves (**H**) and introduces no unit rules, which concludes the proof sketch.

[7] Observe that the iterated system is not strongly connected anymore. It also yields two ways of defining the set of expressions using only one equation: it is very specific to this example, no such property holds in general.

5 Main result

Our main result establishes the degeneracy of uniform random expressions when there is an absorbing pattern, in our framework:

Theorem 1. *Let $\mathcal{E}$ be a combinatorial system of trees over S, of absorbing operator $\circledast$ and of absorbing pattern $\mathcal{P}$, that satisfies* (**H**). *If $\mathcal{L}$ is defined by $\mathcal{E}$ then there exists a positive constant C such that, for the uniform distribution on size-n expressions in $\mathcal{L}$, the expected size of the simplification of a random expression is smaller than C. Moreover, every moment of order t of this random variable is bounded from above by a constant C_t.*

The remainer of this section is devoted to the proof sketch of the first part of Theorem 1: the expectation of the size after simplification. The moments are handled similarly. Thanks to Proposition 1, we can assume that $\mathcal{E}$ is a proper system. By Condition ($\mathbf{H}_5$), it is non-ambiguous so we can directly obtain a system of equations for the associated generating series, as explained in Section 3.2. From now on, for readability and succinctness, we use the vector notation (with bold characters): $\mathbf{L}(z)$ denotes the vector $(L_1(z), \dots, L_m(z))$, and we rewrite the system of Eq. (3) in the more compact form

$$\mathbf{L}(z) = z\,\phi(z; \mathbf{L}(z)), \tag{4}$$

where $\phi = (\phi_1, \dots, \phi_m)$ and $\phi_i(z; \mathbf{y}) = \sum_{(T, i_1, \dots, i_{a(T)}) \in E_i} z^{|T|-1-a(T)} \prod_{j=1}^{a(T)} y_{i_j}$.

Under this form, and because $\mathcal{E}$ satisfies (**H**), we are in the setting of Drmota's celebrated Theorem for systems of equations (Theorem 2.33 in [5], refined in [3]), which gives the asymptotics of the coefficients of the $L_i(z)$'s. This is stated in Proposition 2 below, where $\mathtt{Jac}_{\mathbf{y}}[\phi](z; \mathbf{y})$ is the Jacobian matrix of the system, which is the $m \times m$ matrix such that $\mathtt{Jac}_{\mathbf{y}}[\phi](z; \mathbf{y})_{i,j} = \partial_{y_j} \phi_i(z; \mathbf{y})$.

Proposition 2. *As $\mathcal{E}$ satisfies* (**H**), *the solution $\mathbf{L}(z)$ of the system of equations* (4) *is such that all its coordinates $L_j(z)$ share the same dominant singularity $\rho \in (0, 1]$, and we have $\tau_j := L_j(\rho) < \infty$. The singularity ρ and $\tau = (\tau_j)_j$ verify the characteristic system $\{\tau = \rho\,\phi(\rho; \tau), \det(\mathtt{Id}_{m\times m} - \rho\,\mathtt{Jac}_{\mathbf{y}}[\phi](\rho; \tau)) = 0\}$. Moreover, for every j, there exist two functions $g_j(z)$ and $h_j(z)$, analytic at $z = \rho$, such that locally around $z = \rho$, with $z \notin [\rho, +\infty)$,*

$$L_j(z) = g_j(z) - h_j(z)\sqrt{1 - z/\rho}\,, \quad \text{with } h_j(\rho) \neq 0\,.$$

Lastly, we have the asymptotics $[z^n]L_j(z) \sim C_j \rho^{-n}/n^{3/2}$ for some positive C_j.

The next step is to introduce the bivariate generating series associated with the size of the simplified expression $\mathbf{L}(z, u) = (L_1(z, u); \dots, L_m(z, u))$. We rely on Eq (1) to estimate the expectation of this statistic for uniform random expressions. Proposition 2 already gives an estimation of the denominator, so we focus on proving that for all $j \in [m]$, $[z^n]\partial_u L_j(z, u) \leq \alpha \rho^{-n} n^{-3/2}$, for some positive α.

For this purpose, let $\mathcal{R}_j$ be the set of fully reducible elements of $\mathcal{L}_j$ and let $\mathcal{G}_j = \mathcal{L}_j \setminus \mathcal{R}_j$. Let $\mathbf{R}(z)$ and $\mathbf{L}(z)$ be the vectors of the generating series $\mathcal{R}_j$ and

$\mathcal{L}_j$, respectively. Let also $\mathbf{R}(z,u)$ and $\mathbf{G}(z,u)$ be the vectors of their associated bivariate generating series, where u accounts for the size of the simplified expression. Of course we have $\mathbf{R}(z,u) = u^p\mathbf{R}(z)$, where $p = |\mathcal{P}|$ is the size of the absorbing pattern. We also split the system ϕ into $\phi = \underline{\phi} + \mathbf{A} + \mathbf{B}$ where: $\underline{\phi}$ use all the rules of ϕ whose root is not $\circledast$ and $\mathbf{B}$ gathers the rules of root $\circledast$ with a constant fully reducible child; if necessary, we iterate the system to ensure that $\mathbf{B}$ is not constant as a function of $\mathbf{y}$. Using marking techniques (see [10] for a detailed presentation on expression simplification) we finally obtain:[8]

$$\mathbf{L}(z,u) = u^p\left(\mathbf{R}(z) - \mathbf{P}(z)\right) + zu\left(\underline{\phi}(zu; \mathbf{L}(z,u)) + \mathbf{A}(zu;\ \mathbf{G}(z,u))\right), \qquad (5)$$

where $\mathbf{P}(z) = (a_1 z^p, \dots, a_m z^p)$, with $a_i = 1$ if $\mathcal{P} \in \mathcal{L}_i$ and 0 otherwise.

At this point, we can differentiate the whole equality with respect to u and set $u = 1$. But we do not have much information on $\mathbf{R}(z)$ and $\mathbf{G}(z)$, so it is not possible to conclude directly. Instead of working directly on $\mathcal{R}$ and $\mathcal{G}$, which may rise some technical difficulties, we exploit the fact that $\mathcal{G}, \mathcal{R} \subseteq \mathcal{L}$ and apply a fixed point iteration: this results in a crucial bound for $[z^n]\partial_u\mathbf{L}(z,u)\big|_{u=1}$ purely in terms of $\mathbf{L}(z)$, which is stated in the following proposition.

Proposition 3. *For some $C > 0$, the following coordinate-wise bound holds:*

$$[z^n]\Big\{\partial_u\mathbf{L}(z,u)\big|_{u=1}\Big\} \le C \cdot [z^n]\Big\{\left(\mathtt{Id}_{m\times m} - z\cdot \mathtt{Jac}_{\mathbf{y}}[\underline{\phi} + \mathbf{A}](z;\mathbf{L}(z))\right)^{-1}\cdot \mathbf{L}(z)\Big\}.$$

So we switch to the analysis of the right hand term in the inequality of Proposition 3. Despite its expression, it is easier to study its dominant singularities, and we do so by examining the spectrum of the matrix $J(z) = \mathtt{Jac}_{\mathbf{y}}[\underline{\phi} + \mathbf{A}](z;\mathbf{L}(z))$. This yields the following estimate, which concludes the whole proof:

Proposition 4. *The function $\mathbf{F}\colon z \mapsto (\mathtt{Id} - z\cdot J(z))^{-1}\cdot\mathbf{L}(z)$ has $\rho = \rho_{\mathbf{L}}$, as its dominant singularity. Further, around $z = \rho$ there exist analytic functions $\tilde{g}_j, \tilde{h}_j$ such that $F_j(z) = \tilde{g}_j(z) - \tilde{h}_j(z)\sqrt{1 - z/\rho}$ with $\tilde{h}_j(\rho) \neq 0$. Moreover, we have the asymptotics $[z^n]F_j(z) \sim D_j\rho^{-n}n^{-3/2}$, for some positive D_j.*

6 Conclusion and discussion

To summarize our contributions in one sentence, we proved in this article that even if we use systems to specify them, uniform random expressions lack expressivity as they are drastically simplified as soon as there is an absorbing pattern. This confirms and extends our previous result [10], which holds for much more simple specifications only. It questions the relevance of uniform distributions in this context, both for experiments and for theoretical analysis.

Roughly speaking, the intuition behind the surprising power of this simple simplifications is that, on the one hand the absorbing pattern appears a linear

[8] In fact this is the size of a less effective variation of the simplification algorithm, which is ok for our proof as we are looking for an upper bound.

number of times, while on the other, the shape of uniform trees facilitates the pruning of huge chunks of the expression.

Mathematically speaking, Theorem 1 is not a generalization of the main result of [10]: we proved that the expectation is bounded (and not that it tends to a constant), and we only allowed finitely many rules. Obtaining that the expectation tends to a constant is doable, but technically more difficult; we do not think it is worth the effort, as our result already proves the degeneracy of the distribution. Using infinitely many rules is probably possible, under some analytic conditions, and there are other hypotheses that may be weakened: it is not difficult for instance to ask that the dependency graph has one large strongly connected component (all others having size one)[9], periodicity is also manageable, ... All of these generalizations introduce technical difficulties in the analysis, but we think that in most natural cases, unless we explicitly design the specification to prevent the simplifications from happening sufficiently often, the uniform distribution is degenerated when interpreting the expression: this phenomenon can be considered as inherent in this framework.

In our opinion, instead of generalizing the kind of specification even more, the natural continuation of this work is to investigate non-uniform distributions. The first candidate that comes in mind is what is called BST-like distributions, where the size of the children are distributed as in a binary search tree: that kind of distribution is really used to test algorithms, and it is probably mathematically tractable [15], even if it implies dealing with systems of differential equations.

Acknowledgments. The third author is funded by the Projet RIN Alenor (Regional Project from French Normandy).

References

1. Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling*. Prentice-Hall, Inc., USA, 1972.
2. Cyril Banderier and Michael Drmota. Formulae and asymptotics for coefficients of algebraic functions. *Combinatorics, Probability & Computing*, 24(1):1–53, 2015.
3. Jason P. Bell, Stanley Burris, and Karen A. Yeats. Characteristic points of recursive systems. *Electr. J. Comb.*, 17(1), 2010.
4. Sabine Broda, António Machiavelo, Nelma Moreira, and Rogério Reis. Average size of automata constructions from regular expressions. *Bulletin of the EATCS*, 116, 2015.
5. Michael Drmota. *Random Trees: An Interplay Between Combinatorics and Probability*. Springer Publishing Company, Incorporated, 1st edition, 2009.
6. Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.
7. Philippe Flajolet, Paolo Sipala, and Jean-Marc Steyaert. Analytic variations on the common subexpression problem. In *Automata, Languages and Programming,*

[9] The general case with no constraint on the dependency graph can be really intricate, starting with the asymptotics that may behave differently [2].

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.

8. Philippe Flajolet and Jean-Marc Steyaert. A complexity calculus for recursive tree algorithms. *Mathematical Systems Theory*, 19(4):301–331, 1987.
9. Daniele Gardy. Random boolean expressions. *Discrete Mathematics & Theoretical Computer Science*, DMTCS Proceedings vol. AF, Computational Logic and Applications (CLA '05):1–36, 2005.
10. Florent Koechlin, Cyril Nicaud, and Pablo Rotondo. Uniform random expressions lack expressivity. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 51:1–51:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
11. 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.
12. A Meir and J.W Moon. On an asymptotic method in enumeration. *Journal of Combinatorial Theory, Series A*, 51(1):77 – 89, 1989.
13. Michel Nguyên-Thê. *Distribution of Valuations on Trees.* Theses, Ecole Polytechnique X, February 2004.
14. 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.
15. 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.
16. Carine Pivoteau, Bruno Salvy, and Michèle Soria. Algorithms for combinatorial structures: Well-founded systems and newton iterations. *Journal of Combinatorial Theory, Series A*, 119(8):1711 – 1773, 2012.