AofA'12 DMTCS proc. **AQ**, 2012, 83–96

Enumeration and Random Generation of Concurrent Computations†

Olivier Bodini[1]‡, Antoine Genitrini[2] and Frédéric Peschanski[2]§

[1]*Laboratoire d'Informatique de Paris-Nord, CNRS UMR 7030 - Institut Galilée - Université Paris-Nord, France.*
[2]*Laboratoire d'Informatique de Paris 6, CNRS UMR 7606 and Université Pierre et Marie Curie, Paris, France.*

In this paper, we study the shuffle operator on concurrent processes (represented as trees) using analytic combinatorics tools. As a first result, we show that the mean width of shuffle trees is exponentially smaller than the worst case upper-bound. We also study the expected size (in total number of nodes) of shuffle trees. We notice, rather unexpectedly, that only a small ratio of all nodes do not belong to the last two levels. We also provide a precise characterization of what "exponential growth" means in the case of the shuffle on trees. Two practical outcomes of our quantitative study are presented: (1) a linear-time algorithm to compute the probability of a concurrent run prefix, and (2) an efficient algorithm for uniform random generation of concurrent runs.

Keywords: Concurrency theory. Analytic combinatorics. Shuffle. Random generation. Linear extension.

1 Introduction

Much of *concurrency theory* is articulated around a simple *shuffling* (or interleaving) operator [Mil80]. The basic underlying idea is that two independent processes running in parallel, commonly denoted $P \parallel\!| \, Q$, can be faithfully simulated by the shuffling of their computations. We denote $a.P$ (resp. $b.Q$) a sequential process that first executes an *atomic action* a (resp. b) and then continue as a process P (resp. Q). The shuffle law then states $a.P \parallel\!| \, b.Q = a.(P \parallel\!| \, b.Q) + b.(a.P \parallel\!| \, Q)$ where $+$ is interpreted as a branching operator. In analytic combinatorics, the shuffle on words has already been intensively studied [MZ08, GDG+08, DPRS10]. The shuffle on words can be seen as a specific case of the interleaving of processes (for processes of the form $(a_1 \dots a_n) \parallel\!| \, (b_1 \dots b_m)$).

The shuffle on trees can also be seen as a simplistic form of sequential scheduling, finding admissible *concurrent runs* based on precedence constraints. The latter corresponds to tree-structured partial orders, thus connecting our study to ordered set theory [BW91], in particular to the problem of counting the *linear extensions* of (tree-structured) partial orders (i.e. the set of compatible total orders).

†This research was supported by the A.N.R. project *MAGNUM*, ANR 2010-BLAN-0204.
‡Email: `Olivier.Bodini@lipn.univ-paris13.fr`
§Email: `{Antoine.Genitrini, Frederic.Peschanski}@lip6.fr`

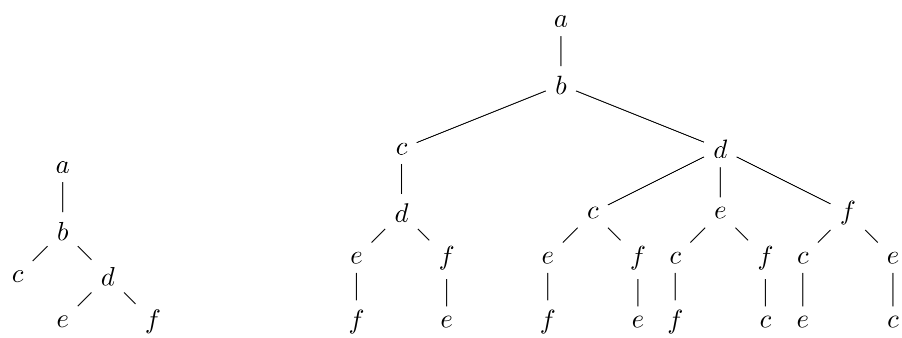


Fig. 1: A process tree (left) and the corresponding shuffle tree (right)

Another approach of this operator is taken in algebraic combinatorics [BFLR11], and specially in the context of partly commutative algebras [DHNT11]. The shuffle is often seen in this case as a binary operator that decomposes in separate left and right shuffles.

Thus shuffle operator is the principal source of *combinatorial explosion* when analyzing concurrent processes, e.g. for *model checking* [CGP99]. Despite the depth and richness of concurrency theory, there are very few (if none, yet) fine-grained analyzes of quantitative properties of this phenomenon, at least from the point of view of analytic combinatorics.

The present paper is organized as follows. In Section 2 we define the shuffle operator on trees and study its basic structural properties. In Section 3 we provide a quantitative study of the shuffle. As a first result, we show that the mean width of shuffle trees is exponentially smaller than the worst case upper-bound, which is reached. This is detailed in Section 3.1. In Section 3.2 we study the expected size (in total number of nodes) of shuffle trees. We notice, rather unexpectedly, that only a small ratio (≈ 0.26) of all nodes do not belong to the last two levels. To address the combinatorial explosion problem, we study the growth of shuffle trees in Section 3.3. This allows us to provide a precise characterization of what "exponential growth" means in our case. Section 4 describes two practical outcomes of our quantitative study: (1) a linear-time algorithm to compute the probability of a concurrent run prefix, and (2) an efficient algorithm for uniform random generation of concurrent runs. The first algorithm can also be used to compute the linear extension of tree-like partial orders, for which the "best" known algorithm performs in quadratic time [Atk90]. The second algorithm has sub-quadratic average complexity, which is to relate to the exponential complexity of generating computations by explicit construction of the shuffle trees.

2 Concurrent computations and shuffle trees

As a starting point, we consider a process specification as a set of (abstract) atomic actions (denoted $a, b, \ldots$) related by precedence constraints (tree-like partial orders), and reflect upon the associated *admissible computations*, i.e. the sequences of atomic actions that satisfy the specification. In the remainder of the paper such admissible computation will be called a (concurrent) *run*.

On the left of Fig. 1 we depict a process specification - or *process tree*. It says that the root action a must be executed first and then b. There is no relation between c and d, and e, f may only happen after

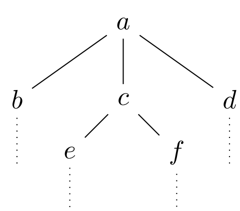
d. This corresponds to a syntactic process $P = a.b.(c \mid\mid\mid d.(e \mid\mid\mid f))$. On the right we build an object that we will call a *shuffle tree* consisting of all the admissible runs for the specification, with the sharing of common prefixes - it is thus a kind of *prefix tree*. Each branch of the shuffle tree is a run (admissible computation) of the given specification, e.g. the leftmost branch yields the run $\langle a, b, c, d, e, f\rangle$. Moreover, the shuffle is complete in the sense that it covers all the runs of the process specification.

More formally we consider process specifications as general plane rooted trees (denoted $T, T_1, T_2, \ldots$) with uninterpreted distinct labels $t, t_1, t_2, \ldots$ corresponding to the actions of the associated process specification. We denote by $T(\alpha)$ the subtree of T rooted in node α. The **size** of a tree T (or of a shuffle tree) is its total number of nodes, and is denoted $|T|$. The **degree** of a node corresponds to the number of its children. The **height** of a tree corresponds to number of nodes minus 1 of its longest branch from root to leaf. The **width** of a tree corresponds to the maximal number of nodes present in the same level.

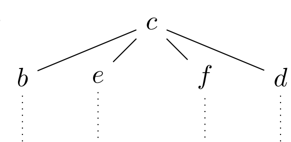
In order to build shuffle trees, we use an elementary operation of *child contraction*.

Definition 1 *Let T be a tree and $v_1, \ldots, v_r$ be the root-labels of the children of the root. For $i \in \{1, \ldots, r\}$, the i-**contraction** of T is the tree with root v_i and whose children are, from left to right, $T(v_1), \ldots, T(v_{i-1}), T(v_{i_1}), \ldots, T(v_{i_m}), T(v_{i+1}), \ldots, T(v_r)$ (where $v_{i_1}, \ldots, v_{i_m}$ are the root-labels of the children of $T(v_i)$). We denote by $T \lhd\ i$ the i-contraction of T.*

For example, if T is



then $T \lhd 2$ is

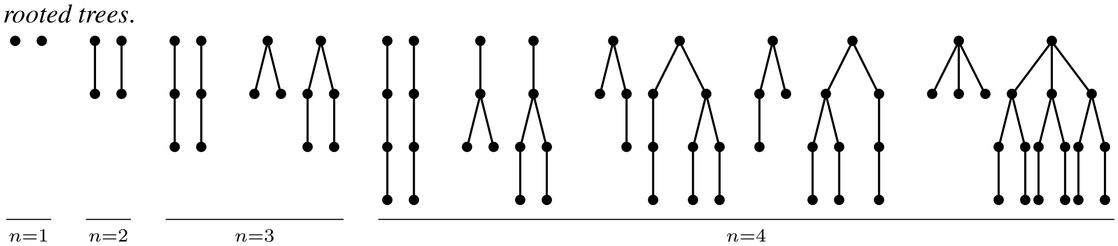


Note that the root (here a) is replaced by the label of the root of the i-th child (here c). Now, the shuffling operation follows a straightforward recursive scheme.

Definition 2 *Let T be a process tree, then its **shuffle tree** $\text{SHUF}(T)$ is defined inductively as:*

- *if T is a leaf, then $\text{SHUF}(T)$ is T*
- *if T has root t and r children ($r \in \mathbb{N} \setminus \{0\}$), then $\text{SHUF}(T)$ is the tree with root t and children, from left to right,* $\text{SHUF}(T \lhd 1), \ldots, \text{SHUF}(T \lhd r)$

The mapping between process trees and shuffle trees is clearly one-to-one. Below we start enumerating the process trees (by size n) together with their shuffle tree. First, we notice that shuffle trees are *plane rooted trees*.



Definition 3 *Let $C(z)$ be the generating function enumerating the combinatorial class $\mathcal{C}$ of plane trees with respect to the number of nodes.*

As a basic recall of *analytic combinatorics* and statement of our conventions, we remind that for a combinatorial class $\mathcal{C}$, we define its counting sequence C_n consisting of the number of objects of $\mathcal{C}$ of size n. This sequence is linked to a formal power series $C(z)$ such that $C(z) = \sum_{n\geq 0} C_n z^n$. We denote by $[z^n]C(z) = C_n$ the n-th coefficient of $C(z)$. Analogous writing conventions will be used for all combinatorial classes.

We remind the reader that for the function $C(z)$ enumerating plane trees then C_n corresponds to the *Catalan numbers* (indeed, shifted by one). For further reference, we give the asymptotic approximation of the Catalan numbers (obtained by an approximation of $n!$ as in e.g. [Com74, P. 267]):

Fact 4 $C_n = \frac{4^{n-1}}{\sqrt{\pi n^3}} \left(1 + \frac{3}{8n} + \frac{25}{128n^2} + \frac{105}{1024n^3} + \frac{1659}{32768n^4} + \mathcal{O}(1/n^5)\right)$.

Perhaps the most important observation is that the shuffle trees are *balanced*, and even more importantly that the height of the shuffle tree is $n-1$ where n is the size of the associated process tree. This is obvious since each branch of a shuffle tree corresponds to a complete traversal of the process tree. Thus there are as many shuffle trees of height $n-1$ as there are plane trees of size n (as counted by C_n above).

A further basic observation is that the contraction operator (cf. Definition 1) ensures that the number of nodes at a given level of a shuffle tree is smaller or equal to the number of nodes at the next level. Thus, the width of the shuffle tree corresponds to its number of leaves.

The following observation bears witness to the high level of redundancy exhibited by shuffle trees.

Proposition 5 *The knowledge of a single branch of a shuffle tree is sufficient to recover the corresponding process tree.*

To go slightly further into the details, we may indeed exhibit a familiy of inverse functions from singled-out shuffle tree branches to process trees. These inverse functions exploit the concept of a *degree-sequence*, defined as follows.

Definition 6 *A **degree-sequence** $(u_p)_{p\in\{1,\ldots,n\}}$ is a sequence of non-negative integers of length n satisfies:*

$$u_1 > 0; \qquad \forall p > 1,\ u_p \geqslant u_{p-1} - 1; \qquad u_n \text{ is the single term equal to 0.}$$

The degrees of the nodes from the root to a leaf in any branch of a shuffle tree is a degree sequence.

Proposition 7 *Let (u_p) be a degree-sequence of length n that is linked to the leftmost branch of a shuffle tree S. Let us define the new sequence (v_p) such that:*

$$v_1 = a_1 \qquad \forall p > 1,\ v_p = u_p - u_{p-1} + 1.$$

We build a tree T of size n such that the sequence (v_n) corresponds to the degrees of each node of the tree, ordered by the prefix traversal. The shuffle of T is the tree S.

A *crucial* remark on this proposition is that we only considered the leftmost branch of the shuffle tree to construct the corresponding degree-sequence, from which we recover the initial tree. We can show, in fact, that the initial tree can be recovered by considering *any* of the branches of its shuffle tree, not just the leftmost one. Each branch corresponds to a degree sequence visiting the nodes of the initial tree by a specific traversal. For example, if the leftmost branch encodes the prefix traversal; the rightmost branch enumerates its mirror. Last but not least, the set of degree-sequences of length n is only of cardinality C_n (hence, degree-sequences are also in bijection with binary trees). This confirms that shuffle tree structures are highly *symmetrical* because many branches must be defined by the same degree-sequence.

3 Quantitative study of shuffle trees

Our quantitative study begins (in Section 3.1) by measuring the number of different concurrent runs for process trees of size n. This indeed boils down to measuring the *mean width* of the associated shuffle trees. To understand the degree of *sharing* in shuffle trees, we provide a much more involved study of their expected size in Section 3.2. This, together with basic facts, sketches some *typical measures* of shuffle trees, as summarized in Fig. 2.

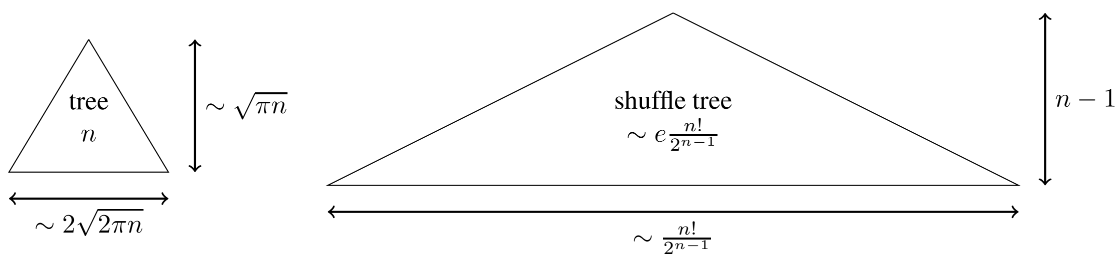


Fig. 2: Some typical measures of a tree of size n (left) and its shuffle tree of height $n-1$ (right).

The shuffling operator induces an exponential growth when considering process trees of increasing sizes. Section 3.3 offers some quantitative arguments relative to the infamous (and imprecise) *combinatorial explosion* problem [CGP99].

3.1 Mean width of shuffle trees

The mean values for rooted plane trees have been studied in [dBKR72] for the average height and in [DG04] for the average width. Shuffle trees are however much more constrained than in the general case, as highlighted by our main result.

Theorem 8 *The mean number of concurrent runs built on process trees of size n is:*

$$\bar{W}_n = \frac{n!}{2^{n-1}} \sim_{n\to\infty} \frac{n!}{2^{n-1}} \sim_{n\to\infty} 2\sqrt{2\pi n}\left(\frac{n}{2e}\right)^n .$$

Let us first observe that this mean value is much smaller than the reached upper bound equal to $(n-1)!$. The proof of Theorem 8 relies on a one-to-one correspondance between the shuffle trees and the *increasing trees* [Drm09].

Definition 9 *An **increasing tree** is a labelled plane rooted tree such that the sequence of labels along any branch starting at the root is increasing.*

For example, to label the tree of Fig. 1, a must take the label 1, b must take the label 2. Then we can choose for the node c: its label belongs to $\{3,4,5,6\}$ but then it induces constraints on the other nodes. Finally, we have 6! labelled trees derived from this tree, but only 8 of them correspond to increasing trees.

Lemma 10 *Let T be a process tree. The number of concurrent runs associated to T corresponds to the number of increasing trees whose structure is the unlabelled tree T.*

This is simply by observing that each labelling of an increasing tree corresponds to a distinct way to traverse the process tree, i.e. a distinct branch of the corresponding shuffle tree.

Fact 11 Hook length in trees [Knu98, p.67].
Let T be a unlabelled tree. The number ℓ_T of increasing trees built on T equals:

$$\ell_T = \frac{|T|!}{\prod_{S \textit{ subtree of } T} |S|}.$$

Corollary 12 *The number of concurrent runs of a process tree T is the number ℓ_T.*

The proof is obvious by using Lemma 10.

Proof for Theorem 8: The mean number is obtained by dividing by C_n the number of runs build on trees of size n. General increasing trees are satisfying the following specification (using the classical boxed product $^{\square}\star$ see [FS09, p.139] for details): $\mathcal{G} = \mathcal{Z}^{\square} \star \textsc{Seq}\ \mathcal{G}$. So, let $G(z)$ denote the exponential generating function enumerating $\mathcal{G}$, where $\mathcal{Z}$ marks all the nodes. The function $G(z)$ is the unique solution analytic in zero of $G'(z) - G'(z) \cdot G(z) - 1 = 0$ with the condition $G(0) = 0$. Consequently, $G(z) = 1 - \sqrt{1-2z}$. So, the number of increasing trees of size n is $[z^n]G(z) = 1 \cdot 3 \cdot \ldots \cdot (2n-3)$. Using Lemma 10, we get the exact result and the Stirling formula gives the asymptotic behaviour. $\square$

3.2 Expected size of shuffle trees

Our goal here is to compute the arithmetic average size of shuffle trees induced by process trees of size n. The result can be obtained by a generalization of the hook length formula to work on *admissible cuts* instead of just subtrees. Such objects are classical in algebraic combinatorics, see in [CK98] for example.

Definition 13 *Let T be a tree of size n ($n > 1$). An* ***admissible cut*** *of T of size $n-1$ is a tree obtained by removing one leaf from T. The set of all the admissible cuts of T is obtained recursively.*

As a remark we add that a tree T of size n is itself an admissible cut of T of size n.

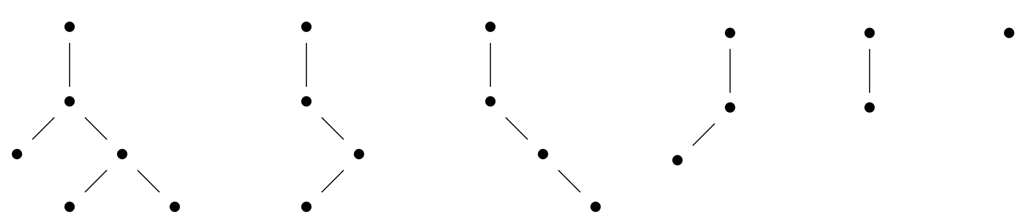


Fig. 3: Some admissible cuts of the tree from Fig. 1 (with actions omitted).

Theorem 14 *Let T be a process tree. The size n_T of the shuffle tree built on T satisfies:*

$$n_T = \sum_{S \text{ admissible cut of } T} \ell_S.$$

Proof: This direct results from the following observations. First, for every node of the process tree T, let say at level i, there is a one-to-one correspondence between the branch ending at this node in the shuffle tree and one run of an admissible cut of T of size $i+1$. Thus, the number of nodes of the shuffle tree corresponds to the sum on all admissible cuts of their number of runs. $\square$

Working with number n_T is non-trivial since the admissible cuts, unlike subtrees, do not follow a simple recursive scheme. We first characterize the mean number $\bar{m}_n$ of recursive calls to ℓ_S required to compute n_T.

Observation 15 *The mean number $\bar{m}_n$ of admissible cuts of trees of size n is:*

$$\bar{m}_n = \frac{-1}{2} + \sum_{l=1}^{n} \sum_{\substack{r_1,\dots,r_l \in \mathbb{N} \\ r_1+\cdots+r_l=n}} \frac{C_l}{4^l C_n} \prod_{i=1}^{l} \gamma_{r_i} \sim_{n\to\infty} \frac{1}{\sqrt{15}} \left(\frac{5}{4}\right)^{2n}, \text{ where } \gamma_1 = 6 \text{ and } \forall k > 1,\ \gamma_k = C_k.$$

Proof sketch: Let us denote by $M(z)$ the ordinary generating function enumerating the multiset of admissible cuts of all trees. More precisely $M(z) = \sum_{n\in\mathbb{N}} M_n z^n$ where $M_n = \sum_{T;|T|=n} \sum_{S \text{ admissible cut of } T} 1$. The tag $\mathcal{Z}$ marks the nodes of the tree carrying the admissible cut. The generating function $C(z)$ enumerates all trees. The specification of $\mathcal{M}$ is $\mathcal{Z} \times \textsc{Seq}(\mathcal{M} \cup \mathcal{C})$. Once this specification is given, the rest of the proof is very classical in analytic combinatorics. $\square$

This observation can be combined together with the notion of increasing trees to give the specification of *increasing admissible cuts*, which in turn allows us to compute n_T properly.

Theorem 16 *The mean size $\bar{S}_n$ of a shuffle tree induced by a tree of size n follows the linear recurrence:*

$$\begin{aligned}(2n^4 + 12n^3 + 22n^2 + 12n)\bar{S}_n + (-4n^4 - 32n^3 - 87n^2 - 87n - 18)\bar{S}_{n+1} \\ + (2n^4 + 24n^3 + 85n^2 + 106n + 39)\bar{S}_{n+2} + (-4n^3 - 20n^2 - 31n - 15)\bar{S}_{n+3} = 0,\end{aligned}$$

with the initial conditions: $\bar{S}_0 = 0, \bar{S}_1 = 1$ and $\bar{S}_2 = 2$. Consequently,

$$\bar{S}_n \sim_{n\to\infty} e\frac{n!}{2^{n-1}} \sim_{n\to\infty} 2e\sqrt{2\pi n}\left(\frac{n}{2e}\right)^n.$$

Proof sketch: As for the expected number of admissible cuts, we are going to deal with specifications and generating functions. First of all, let us explain the following bivariate (in $\mathcal{Z}$,$\mathcal{U}$) specification for $\mathcal{M}$: $\mathcal{U} \times \mathcal{Z} \times \textsc{Seq}(\mathcal{M} \cup \mathcal{C})$. The tag $\mathcal{Z}$ (resp. $\mathcal{U}$) marks the nodes of the tree carrying the admissible cut (resp. of the admissible cut). Now, observing Theorem 14, we see that we are not interested in admissible cuts but in increasing admissible cuts. So, let us consider the specification $\mathcal{S} = \mathcal{U}^{\square_\mathcal{U}} \star \mathcal{Z} \times \textsc{Seq}(\mathcal{S} \cup \mathcal{C})$ where the boxed product operates on $\mathcal{U}$. Its associated generating function

$S(z,u)$ verifies $S(z,u) = \int_{v=0}^{u} \frac{z}{1-S(z,v)-C(z)} dv$. So, $S(z,u) = \sum_{n,k\in\mathbb{N}} S_{n,k} z^n u^k / k!$ where $S_{n,k}$ is $\sum_{T;|T|=n} \sum_{S \text{ admissible cut of size } k \text{ of } T} \ell_S$. Now, this differential equation can be easily integrated:

$$S(z,u) = 1/2 + 1/2\sqrt{1-4z} - 1/2\sqrt{2+2\sqrt{1-4z}-4z-8uz}.$$

By substituting u^k by $k!$ in the series $S(z,u)$, we obtain the generating function $S(z)$ for the size of the shuffle trees. That is to say, $S(z) = \sum_{n\in\mathbb{N}} S_n z^n$ where $S_n = \sum_{T;|T|=n} \sum_{S \text{ admissible cut of } T} \ell_S$. This substitution can be done using a gamma transformation. So, we finally obtain:

$$S(z) = \int_{u=0}^{\infty} S(z,u)\exp(-u) du.$$

At this stage, our goal is to analyze the asymptotic of the coefficients of $S(z)$. Due to the unusual expression of $S(z)$, this analysis requires some attention. We proceed according to the following plan which is based on multivariate holonomic closure [Lip89]:

(i) As $S(z,u)$ is algebraic, we may characterize explicitly the minimal polynomial $P \in \mathbb{C}[z,u][X]$ such that $P(S(z,u)) = 0$.

(ii) As algebraic functions are holonomic, one can give a partial differential equation for $S(z,u)$ using P.

(iii) The combinatorial Laplace transform allows to obtain a partial differential equation for $\hat{S}(z,u) = \int_{v=0}^{\infty} S(z,uv)\exp(-v)dv$.

(iv) Using the holonomic stability under partial evaluation, a differential equation for $S(z)$ is obtained.

This calculation has been done with a computer algebra system (see [SZ94], the package Gfun of Maple) and produced very huge equations (including more than one thousand terms). However following this technical but conceptually straightforward calculation, we finally reach a relatively simple linear differential equation for $S(z)$:

$$\left(20z^2-14z+2\right)S(z) + \left(-48z^4-8z^3+18z^2-3z\right)S'(z) + \left(-192z^5+112z^4-24z^3+2z^2\right)S''(z) + \left(-64z^6+32z^5-4z^4\right)S'''(z) = 4z^2 - z,$$

with the initial conditions $S(0) = 0, S'(0) = 1$ and $S''(0) = 4$. From this differential equation, we deduce a P-recurrence for the coefficients S_n:

$$16n(1-4n^2)S_n + 4(8n^3+28n^2+18n+3)S_{n+1} - 2(2n^3+18n^2+31n+13)S_{n+2} + (2n^2+7n+5)S_{n+3} = 0,$$

with the initial conditions $S_0 = 0, S_1 = 1$ and $S_2 = 2$.
Now, consider the series of the mean size of the shuffle tree $\bar{S}(z)$ whose coefficients are $\bar{S}_n = S_n/C_n$. The function $\bar{S}(z)$ is also holonomic and yields the following differential equation:

$$\left(10z^2+7z+6\right)\bar{S}(z) + \left(48z^4-28z^3-11z^2-7z\right)\bar{S}'(z) + \left(72z^5-91z^4+27z^3+4z^2\right)\bar{S}''(z) + \left(24z^6-40z^5+20z^4-4z^3\right)\bar{S}'''(z) + \left(2z^7-4z^6+2z^5\right)\bar{S}''''(z) + 4z^2 + z = 0$$

with $\bar{S}(0) = 0, \bar{S}'(0) = 1$ and $\bar{S}''(0) = 4$. And now, the coefficients $\bar{S}_n$ follow the recurrence:

$$2n(n^3 + 6n^2 + 11n + 6)\bar{S}_n - (4n^4 + 32n^3 + 87n^2 + 87n + 18)\bar{S}_{n+1} + (2n^4 + 24n^3 + 85n^2 + 106n + 39)\bar{S}_{n+2} - (4n^3 + 20n^2 + 31n + 15)\bar{S}_{n+3} = 0,$$

with the initial conditions $\bar{S}_0 = 0, \bar{S}_1 = 1$ and $\bar{S}_2 = 2$.
This achieves the first part of the theorem.

For the asymptotic part of the theorem, we introduce new auxiliary generating functions which are more tractable than $S(z)$. For that purpose, recall that the total number of leaves in the shuffle trees induced by process trees of size n (which is also the number of increasing trees of size n) is equal to $n!/2^{n-1}$. So, it is natural to study the series $R(z)$ with general terms $\bar{S}_n 2^{n-1}/n!$ which is also holonomic and verifies:

$$-2\left(10z^2 + 7z + 3\right)R(z) + \left(-16z^4 + 32z^3 + 18z^2 + 7z\right)R'(z) + 4\left(4z^4 - 6z^3 - z^2\right)R''(z) + 4\left(-z^4 + z^3\right)R'''(z) = 4z^2 + z,$$

with the initial conditions $R(0) = 0, R'(0) = 1$, $R''(0) = 4$. The coefficients R_n follow the recurrence:

$$-16nR_n + 4(4n^2+12n+3)R_{n+1} - 2(2n^3+18n^2+31n+13)R_{n+2} + (4n^3+20n^2+31n+15)R_{n+3} = 0,$$

with $R_0 = 0, R_1 = 1$ and $R_2 = 2$. Now, we can easily prove that this recurrence is convergent. Indeed, the recurrence is non-negative and asymptotically decreasing, just by observing that $R_{n+3} - R_{n+2} = \left(\frac{4}{n} + \mathcal{O}(\frac{1}{n^2})\right)(R_{n+2} - R_{n+1}) + \mathcal{O}(\frac{1}{n^2})R_n$ implies that for n sufficiently large the difference is always negative.

By calculating the first terms of the series (using a computer algebra software), we were surprised to see that the series seems to converge to $\exp(1)$. In order to prove it, we need to make some another technical calculation following the plan below:

(i) We are going to compare R_n with another P-recursive sequence that converges to $\exp(1)$. For that, we consider the holonomic function $f(z) = \exp(z)/(1-z)$ whose coefficients tend to $\exp(1)$.

(ii) We find a differential equation for $R'(z) - f'(z)$ (it is easy thanks to holonomicity).

(iii) The sequence $n(R_n - f_n)$ tends to a constant (because it is non-negative and asymptotically decreasing). So, we can conclude that R_n tends to $\exp(1)$ (as f_n). $\square$

The structure of shuffle trees is such that both last levels contain the same number of nodes and this number corresponds to the width of the shuffle tree. So by observing both average width and size of the shuffle trees, we notice that only a ratio $1 - 2/e \approx 0.26$ of all nodes do not belong to the last two levels.

3.3 Analysis of growth

To analyse quantitatively the growth between the process trees and their shuffle trees, we measure the typical explosion of the number of concurrent runs induced by large trees of size n. Although the arithmetic mean is the usual way to measure in average, in our case it is more natural to compute the geometric mean since the ratio between these quantities is exponential.

Theorem 17 *The geometric mean number of concurrent runs built on process trees of size n is:*

$$\bar{\Gamma}_n = \prod_{k=2}^{n-1} k^{1-\frac{n+1-k}{2}\frac{C_k C_{n-k+1}}{C_n}} \sim_{n\to\infty} \sqrt{2\pi} n^n \exp\left(-(1+2L(1/4))n + \sqrt{\pi n} + L(1/4)\right),$$

where $L\,(1/4) = \sum_{n>1} \log n \cdot C_n \cdot 4^{-n} \approx \mathbf{0.5790439}217 \pm 5 \cdot 10^{-9}$.

The explosion is less important than the one that one could have conjectured with the arithmetic mean, but is however very large. For both means, the result is far from the upper bound $(n-1)!$. Concerning the constant, the exact digits of the constant are written in bold type. It has been computed by a computer algebra software, in the following way: (i) The first $5 \cdot 10^4$ terms have been computed with the exact value of C_n; (ii) The terms from $5 \cdot 10^4$ to $3.3 \cdot 10^5$ have been calculated with an approximation of C_n based on the four first terms in its asymptotic decomposition (see Fact 4); (iii) The queue of the series has been approximated by two Riemann sums. In order to obtain one digit more in the approximation of $L(1/4)$, one should do the second approximation for terms from $5 \cdot 10^4$ to $1.7 \cdot 10^6$.

Proof sketch for Theorem 17: We will just give some key ideas here. To obtain the geometric mean, we will use another recursive equation for ℓ_T:

$$\ell_T = (|T|-1)! \cdot \left(\prod_{R \text{ child of the root of } T} \frac{\ell_R}{|R|!} \right).$$

Then, by taking the logarithm of the latter sequence, we are able to compute the arithmetic mean value of the logarithm of what we are interested in, so by using the exponential function, we obtain the result. □

4 Application: random generation of concurrent computations

We describe in this section the way one can choose uniformly at random one specific run among those induced by a given process tree. Moreover we present some results that allow us to compute the number of such concurrent runs and also to give the probability of a given prefix run, both in linear time. Fortunately, we do not need to build the shuffle trees to obtain these results. The basic idea is to associate a weight to each node of a process tree, according to the size of its subtree.

Definition 18 *Let T be a process tree and $\sigma = \langle \alpha_1, \ldots, \alpha_p \rangle$ a prefix of a run in* $\textsc{Shuf}(T)$*. Let T_σ be the tree whose root is α_p and its children are all children of nodes $\alpha_1, \ldots, \alpha_p$ (in T) that are distinct from $T(\alpha_1), \ldots, T(\alpha_p)$ and ordered according to the prefix traversal of T.*

For the process tree T of Fig. 1, we obtain $T_{\langle a,b,d \rangle}$ with root d and with three children (from left to right): the leaves c, e and f.

Let T be a process tree. Suppose now that $\sigma_{p+1} = \langle \alpha_1, \ldots, \alpha_p, \alpha_{p+1} \rangle$ and let us denote $\sigma_p = \langle \alpha_1, \ldots, \alpha_p \rangle$. We are interested in the probability of choosing α_{p+1}, once σ_p has already been chosen. To obtain this probability, we need to count how many runs in T_{σ_p} are first running α_{p+1}.

Proposition 19

$$\mathbb{P}(\sigma_{p+1} \mid \sigma_p) = \frac{\ell_{T_{\sigma_{p+1}}}}{\ell_{T_{\sigma_p}}} = \frac{(|T|-p)!}{\prod_{S \textit{ subtree of } T_{\sigma_{p+1}}} |S|} \cdot \frac{\prod_{S \textit{ subtree of } T_{\sigma_p}} |S|}{(|T|-p+1)!} = \frac{|T(\alpha_{p+1})|}{|T|-p+1}.$$

Data: T: a weighted process tree of size n
Result: σ: a run (a list of nodes)
$\sigma := \langle\rangle$
$\mu := \{\!\{a^{\text{weight}(a)}\}\!\}$ # initialize a multiset with the root a with cardinality its weight
for *i from* $n-1$ *to 1 by -1* **do**
  # **invariant**: the total weight (sum of cardinalities) of μ is i
  α :=sample(μ) # sample an action α according to its cardinality in the multiset
  $\sigma := \sigma \cup \{|\sigma| \mapsto \alpha\}$ # append the sampled action to the sequence
  $\mu := \mu \cup \{\!\{\alpha^0\}\!\}$ # α cannot be sampled anymore
  $\mu := \mu \cup \{\!\{\gamma^{\text{weight}(\gamma)} \mid \gamma \text{ a child of } \alpha\}\!\}$ # insert the children of α in the multiset
return σ

Fig. 4: Algorithm: uniform random generation of concurrent runs

The proof directly derives from the Hook length formula (cf. Fact 11).

Corollary 20 *Let T be a process tree and $\sigma = \langle\alpha_1, \dots, \alpha_p\rangle$ be a prefix of a run in* $\text{S}\small{\text{HUF}}(T)$. *For the uniform probability distribution on the set of all concurrent runs, the induced probability on prefixes satisfies $\mathbb{P}(\sigma) = \ell_{T_\sigma}/\ell_T$.*

Corollary 21 *Let T be a process tree. The probability of a prefix run $\sigma = \langle\alpha_1, \dots, \alpha_p\rangle$ in the shuffle tree of T is equal to $\prod_{k=1}^{p} |T(\alpha_k)|/(|T| - k + 1)$.*

Corollaries 20 and 21 give us a simple way to compute ℓ_T, i.e. the number of concurrent runs in the shuffle of process tree T. This is indeed the inverse of the probability of any complete traversal of T. From this we can deduce a linear-time algorithm to compute the linear extensions of T seen as a tree-like partial order, which is clearly an improvement if compared to the quadratic algorithm proposed in [Atk90]. A slight adaptation of the algorithm also offers a way to generate the probability of a prefix concurrent run in linear time.

We are now interested in building uniformly at random one concurrent run for a specified process tree without building its shuffle tree. The main idea is to sample in a multiset containing the actions of the process trees as elements, each one associated to its weight. There are various ways to implement the multiset sampler, see e.g. [WE80]. One idea is to implement a *partial sum tree*, i.e. a balanced binary search tree in which all operations (adding or removing a node) are done in logarithmic time.

Let T be a process tree. First by one traversal, we add a label to all nodes of T that corresponds to the size of the subtree rooted in that node. We say that this size corresponds to the weight of each node. We build a list σ, at the end of size n, such that at each step i, we add one action to σ that corresponds to the (i)-th action in our random run. To choose this i-th action, we sample in the multiset of all possible actions available in the considered step. Initially only the root is available (with probability 1 thus cardinality n the size of the process tree T). Then it is added to σ and removed from the multiset. Finally its children are enabled with the weight as cardinality. And we proceed until all actions have been sampled. In the case of the partial sum tree implementation, we have the following complexity results.

Proposition 22 *Let n be the size of the weighted process tree T. To obtain a random execution, we need n random choices of integers and the operations on the multiset is of order $\Theta(n \log n)$ (for the worst case).*

5 Conclusion and perspectives

The quantitative study of the shuffle operator represents a preliminary step towards our goal of investigating concurrency theory from the analysis of algorithms point of view. In the next step, we shall address other typical constructs of formalisms for concurrency, especially non-deterministic choice and *synchronization* [AI07]. There are indeed various forms of synchronization, in general corresponding to reflecting the action labels within the shuffle operator. Other operators, such as *hiding*, also deserve further investigations. A way of generalizing our results would be to take into account non-plane process trees. An approach based on analytic combinatorics would still be possible, albeit more intricate, by considering multisets instead of sequences in the specifications. Another interesting continuation of the work would be to study the compaction of the shuffle trees by identifying common subtrees. This would amount to study the shuffling of process trees up-to *bisimilarity*, the natural notion of equivalence for concurrent processes. Note that our algorithmic framework would not be affected by such study, since they do not require the explicit construction of the shuffle trees (wether compacted or not).

From a broader perspective, we definitely see an interest in reinterpreting *semantic objects* (from logic, programming language theory, concurrency theory, etc.) under the lights of analytics combinatorics tools. Such objects (like shuffle trees) may be quite intricate when considered as combinatorial classes, thus requiring non-trivial techniques. This is highlighted here e.g. by the generalized hook length formula characterizing the expected size of shuffle trees. Conversely, we think it is interesting to know – precisely, not just by intuition – the high-level of sharing and symmetry within shuffle trees. This lead us naturally to practical algorithms, making us confident that real-world applications (in our case, especially related to random testing and probabilistic model-checking) might result from such study.

Acknowledgements. We are grateful to M. Dien and O. Roussel for fruitful remarks about the algorithms.

References

[AI07] L. Aceto and A. Ingólfsdóttir. The saga of the axiomatization of parallel composition. In *CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 2–16. Springer, 2007.

[Atk90] M. D. Atkinson. On computing the number of linear extensions of a tree. *Order*, 7:23–25, 1990.

[BFLR11] A. Boussicault, V. Féray, A. Lascoux, and V. Reiner. Linear extension sums as valuations of cones. *Journal of Algebraic Combinatorics*, 35(4):573–610, 2011.

[BW91] G. Brightwell and P. Winkler. Counting linear extensions is #P-complete. In *STOC*, pages 175–181, 1991.

[CGP99] E.M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT Press, 1999.

[CK98] A. Connes and D. Kreimer. Hopf algebras, renormalization and noncommutative geometry. *Comm. Math. Phys.*, 199(1):203–242, 1998.

[Com74] L. Comtet. *Advanced Combinatorics: The Art of Finite and Infinite Expansions*. Reidel, 1974.

[dBKR72] N.G. de Bruijn, D.E. Knuth, and S.O. Rice. The average height of planted plane trees. *Graph Theory and Computing*, pages 15–22, 1972.

[DG04] M. Drmota and B. Gittenberger. The width of Galton-Watson trees conditioned by the size. *Discrete Math. Theor. Comput. Sci.*, 6(2):387–400 (electronic), 2004.

[DHNT11] G. Duchamp, F. Hivert, J.-C. Novelli, and J.-Y. Thibon. Noncommutative symmetric functions vii: Free quasi-symmetric functions revisited. *Annals of Combinatorics*, 15:655–673, 2011.

[DPRS10] A. Darrasse, K. Panagiotou, O. Roussel, and M. Soria. Boltzmann generation for regular languages with shuffle. In *GASCOM*, 2010.

[Drm09] M. Drmota. *Random trees*. Springer, Vienna-New York, 2009.

[FS09] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[GDG+08] M.-C. Gaudel, A. Denise, S.-D. Gouraud, R. Lassaigne, J. Oudinet, and S. Peyronnet. Coverage-biased random exploration of models. In *ETAPS Workshop on Model Based Testing*, volume 220, pages 3–14. Electr. Notes Theor. Comput. Sci., 2008.

[Knu98] D. E. Knuth. *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.

[Lip89] L. Lipshitz. D-finite power series. *J. Algebra*, 122:353–373, 1989.

[Mil80] Robin Milner. *A Calculus of Communicating Systems*. Springer Verlag, 1980.

[MZ08] M. Mishna and M. Zabrocki. Analytic aspects of the shuffle product. In *STACS*, pages 561–572, 2008.

[SZ94] B. Salvy and P. Zimmerman. Gfun: a maple package for the manipulation of generating and holonomic functions in one variable. *ACM Trans. Math. Softw.*, 20:163–177, June 1994.

[WE80] C. K. Wong and M. C. Easton. An efficient method for weighted sampling without replacement. *SIAM J. Comput.*, 9(1):111–113, 1980.