HAL
archives-ouvertes.fr

Random generation of combinatorial structures: Boltzmann samplers and beyond

Philippe Duchon

▶ To cite this version:

Philippe Duchon. Random generation of combinatorial structures: Boltzmann samplers and beyond. Winter Simulation Conference, Dec 2011, Phoenix, United States. hal-00654267

HAL Id: hal-00654267
https://hal.inria.fr/hal-00654267

Submitted on 21 Dec 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

RANDOM GENERATION OF COMBINATORIAL STRUCTURES: BOLTZMANN SAMPLERS AND BEYOND

Philippe Duchon

LaBRI, INRIA/Université de Bordeaux
351, cours de la Libération
33405 Talence cedex, FRANCE

This paper is dedicated to Philippe Flajolet (1948-2011) – an outstanding mathematician
and computer scientist, and a great person to work with.

ABSTRACT

The Boltzmann model for the random generation of "decomposable" combinatorial structures is a set of techniques that allows for efficient random sampling algorithms for a large class of families of discrete objects. The usual requirement of sampling uniformly from the set of objects of a given size is somehow relaxed, though uniformity among objects of each size is still ensured. Generating functions, rather than the enumeration sequences they are based on, are the crucial ingredient.

We give a brief description of the general theory, as well as a number of newer developments.

1 INTRODUCTION

1.1 Random Generation of Combinatorial Structures: a Quick Overview

Random generation is often used as a tool for exploration ("what do large objects of this type look like?"), or to provide large datasets for software and algorithm testing.

Usually, one defines a combinatorial class as a finite or countable family $\mathscr{C}$ of discrete "objects", equipped with a "size" function from $\mathscr{C}$ to the natural numbers, such that for each natural number n, the subset $\mathscr{C}_n$ of all objects with size n is finite.

A *uniform* random generator for the class $\mathscr{C}$ is then a randomized algorithm that takes as input an integer n, and outputs an element randomly selected uniformly from $\mathscr{C}_n$. The efficiency of the generator is typically measured in terms of its expected time and space complexities, expressed as a function of the size n.

Random generation methods tend to fall into one of a small number of classes. *Ad hoc* methods rely on precise combinatorial properties of the considered class; a fine example is provided by Rémy's algorithm [14] for the random generation of plane binary trees. *Markov chain* methods rely on the simulation of a Markov chain whose states are the objects one wants to sample from, and that converges to the uniform (or another suitably chosen) distribution; careful analysis of the convergence speed makes it possible to run the chain for a fixed number of steps and obtain an almost-uniform generator, or more sophisticated techniques such as Coupling from the Past [13] can be used for exact sampling from the stationary distribution.

Decomposition methods apply for classes where objects are, informally, "made up" of smaller objects, be they from the same or another class that is itself decomposable. A prime example is that of *plane binary trees*, that is, rooted trees where each internal node has exactly two children, one of which is distinguished as the *left* child while the other is the *right* child. In this case, a plane binary tree is either made up of a single root-leaf, or of a root and left and right subtrees, both of which can be any plane binary trees.

The first systematic example of a decomposition method for random generation is the so-called *recursive method* [11], where decompositions are used to obtain recurrences satisfied by the counting sequences. These counting sequences are then used, together with the decomposition rules themselves, to guide the random generation algorithm. In the above example of plane binary trees, the counting sequence (the well-known Catalan numbers) is used to determine the probability $p_{n,k}$ that a uniform random tree with n internal nodes has a left subtree with k internal nodes, and to sample K from this distribution; then the random generator is recursively called with sizes K and $n-1-K$ to obtain the left and right subtrees (which are independent conditioned on their respective sizes); the resulting tree is then uniform.

1.2 The Boltzmann Method

The *Boltzmann method*, as introduced in [7, 8], is another decomposition-based method that can be applied to roughly the same combinatorial classes as the recursive method. We will give a precise description of the method in the next section. For this introduction, we will simply describe the crucial ingredients.

Where the recursive method uses counting sequences for the random generation algorithms, and generating functions are mostly a tool to compute the counting sequences, the Boltzmann method uses the generating functions, viewed as analytic functions of a real variable (and, in practice, the values of the generating functions) in the random generation algorithms – thus reducing the need for precomputation to a small number of real constants. The idea is to "relax" the requirement for a uniform sampler (which outputs a uniform random structure among those of the target size n) into allowing structures of all sizes, while keeping uniformity among all objects of each

individual size. By choosing the "right" distribution on sizes, independence among substructures is introduced, which results in very simple and efficient algorithms.

1.3 Outline of the Paper

Sections 2-4 make up the bulk of what can be termed the "Boltzmann method". Section 2 gives a general description of the Boltzmann method. Section 3 lists a number of constructions which can be used to define specifications of combinatorial classes for which Boltzmann samplers can be automatically compiled from the specification. Section 4 sums up various results on the complexities of the random sampling algorithms. Section 5 describes how Boltzmann samplers can be used to get closer to the classical model of uniform, fixed-size random generation.

Section 6 deals with the question of how one can effectively obtain the real constants used in Boltzmann samplers, and Section 7 describes results that draw on the principles of the method without exactly fitting in it.

The list of references does not attempt to give a complete list of articles using the ideas exposed here. The interested reader will find more examples in the bibliography of [2], even though it is not limited to references about the Boltzmann method.

All proofs and most technical details have been purposefully omitted; the algorithms in Section 3 have been included mostly to demonstrate their simplicity. We have made the choice of not detailing any of the many examples that could be given; the interested reader will find many such examples, including pictures of large random structures, in the original papers.

2 THE BOLTZMANN METHOD: GENERAL DESCRIPTION

The Boltzmann method can be used in two flavors, the *ordinary* (or unlabelled) and *exponential* (labelled) variants.

Throughout the paper, we use the word *structure* (or $\mathscr{C}$-structure) to indicate an element of a combinatorial class $\mathscr{C}$. No particular assumption is ever made on the nature of such structures, though classical examples tend to come from discrete mathematics or theoretical computer science: words over some finite alphabet, sequences, various flavors of trees, etc.

2.1 Combinatorial Classes and Products

Let $\mathscr{C}$ be some combinatorial class. The size of an object $c \in \mathscr{C}$ will be noted $|c|$. For any integer n, let c_n denote the number of objects in $\mathscr{C}$ with size n. The ordinary (resp. exponential) *generating function* for $\mathscr{C}$ is

$$C(z) = \sum_n c_n z^n, \quad \text{resp. } \tilde{C}(z) = \sum_n c_n \frac{z^n}{n!};$$

it is always assumed that the considered generating function has positive radius of convergence ρ, *i.e.* that $\overline{\lim} c_n^{1/n} < \infty$ (resp., $\overline{\lim}(c_n/n!)^{1/n} < \infty$).

For $0 < x < \rho$, the (normal, resp. exponential) *Boltzmann distribution over $\mathscr{C}$ for parameter x* is the probability distribution defined, for any $c \in \mathscr{C}$, by

$$\mathbb{P}_x(c) = \frac{x^{|c|}}{C(x)}, \text{ resp. } \mathbb{P}_x(c) = \frac{x^{|c|}}{n!\tilde{C}(x)}.$$

These distributions give positive probability to all objects in the class, with the property that two objects with the same size have the same probability. They are, of course, not the only probability distributions with this property; their interest lies mostly in their relationship with two common constructions in the combinatorial world: the normal and labelled products.

Given two classes $\mathscr{A}$ and $\mathscr{B}$, their normal product is just their Cartesian product $\mathscr{C} = \mathscr{A} \times \mathscr{B}$, with size defined additively by $|(a,b)| = |a| + |b|$ for $(a,b) \in \mathscr{A} \times \mathscr{B}$.

To define the labelled product, one has to assume that structures are made up of both an unlabelled structure c and a "labelling", a permutation on a set whose is that of c. Think of a structure c as being formed of $|c|$ basic "atoms", each of which receives a distinct label from $[[1,|c|]] = \{i \in \mathbb{Z} : 1 \leq i \leq |c|\}$. By a slight abuse of notation, we identify these atoms with the integers 1 to $|c|$, so that the labellings are just permutations $\sigma \in \mathscr{S}_{|c|}$. For each unlabelled structure c, the set of admissible permutations may be a strict subset of $\mathscr{S}_{|c|}$.

Then, the labelled product of two labelled structures (a,σ_a) and (b,σ_b) is defined as (a,b), with admissible labelings obtained by taking all partitions of $[[1,|a|+|b|]]$ into two parts A and B of respective sizes $|a|$ and $|b|$, and, for each partition, taking the one permutation $\sigma \in \mathscr{S}_{|a|+|b|}$ where all entries in A are in the same respective order as that of σ_a, and all entries in B are in the same respective order as that of σ_b (that is, if $(x,y) \in A^2$, then $\sigma_a(x) < \sigma_b(y)$ iff $\sigma(x) < \sigma(y)$, and similarly for B). As a result, (a,σ_a) and (b,σ_b) have $\binom{|a|+|b|}{|a|}$ different structures in their labelled product. The labelled product of two classes is defined as the set of all labelled products of structures in the two original classes, with size again defined additively.

The first important property is as follows: if two unlabelled (resp., labelled) classes $\mathscr{A}$ and $\mathscr{B}$ have generating functions $A(z)$ and $B(z)$ (resp., exponential generating functions $\tilde{A}(z)$ and $\tilde{B}(z)$), then their normal (resp. labelled) product has generating function $C(z) = A(z)B(z)$ (resp., $\tilde{C}(z) = \tilde{A}(z)\tilde{B}(z)$).

An immediate, and most useful, consequence, valid under both models, is this: *if $\mathscr{C} = \mathscr{A} \times \mathscr{B}$, then taking the product of two independent $\mathscr{A}$-structure and $\mathscr{B}$-structure, each following the Boltzmann distribution with parameter x, results in a $\mathscr{C}$-structure under the Boltzmann distribution with parameter x.* For labelled structures, it is implied that one selects a uniform random set to define the permutation in the product.

This "independence under substructures" property, in turn, has interesting practical consequences, in that it makes it possible to describe, for a number of classical combinatorial constructions, systematic ways to produce efficient algorithms to sample from the Boltzmann distribution for classes that are entirely described (possibly in a recursive way) with them. This is the topic of Section 3.

2.2 The Boltzmann Method

By a *Boltzmann sampler* for a combinatorial class $\mathscr{C}$, we mean a randomized algorithm $\Gamma\mathscr{C}$ that takes as input a real parameter x and outputs a random $\mathscr{C}$-structure under the Boltzmann distribution with parameter x. Our overall goal is to create efficient Boltzmann samplers for as many combinatorial classes as possible.

Given a description of a combinatorial class $\mathscr{C}$ from which we would like to obtain "large random structures" (of size n, ideally), the Boltzmann method can be summarized as follows:

1. Find out if our class can be specified (up to a reasonably simple size-preserving bijection) from the constructions in Section 3. If not, try to extend the expressive power of the method by adding new constructions. If this does not work, maybe the Boltzmann method is not the best choice after all.

2. Use the techniques of the Purple Book [10] to locate the "dominant" singularities (those of smallest modulus, which govern the asymptotics of the counting coefficients; since the generating functions have nonnegative coefficients, at least one such singularity lies on the positive real axis) of the generating functions in our specification, and possibly an estimate of the value we should give parameter x to give expected size n to $\mathscr{C}$-structures under the Boltzmann distribution.

3. Compute approximations (to roughly $\Theta(\log n)$ digits) of the values at x of all involved generating functions, possibly using the combinatorial oracle of Section 6.

4. Use the patterns in Section 3, together with our specification, to write a Boltzmann sampling program.

5. Optionally, add a rejection scheme to obtain samples with size in $[(1-\varepsilon)n, (1+\varepsilon)n]$, or even of exact size n (more costly).

Alternatively, step 2 can be replaced by experimentation using the other steps.

2.3 Choosing the Parameter and Tuning for Size

As said above, the Boltzmann model uses a real parameter that may be chosen arbitrarily inside the radius of convergence of the generating function for the class under consideration. This parameter x governs the distribution of sizes of the random structures. In typical applications, one would like to obtain *large* structures, that is, there is an ideal value n for sizes. We now briefly turn to the question of picking an appropriate value of x for a target n.

For a given x, the probability that a Boltzmann-distributed $\mathscr{A}$-structure will have size n is given by

$$p_n = p_n(x) = \frac{a_n x^n}{A(x)};$$

multiplying by n, and summing over all values of n, we obtain the *expected* size

$$N = N_{\mathscr{A}}(x) = \frac{xA'(x)}{A(x)},$$

where A' is just the derivative of the generating function A (which can be formally defined as $A'(x) = \sum_n n a_n x^{n-1}$; this corresponds to the usual derivative inside the radius of convergence of A).

Except in degenerate cases, $N_{\mathscr{A}}$ is a strictly increasing and convex function on the interval $[0, \rho)$, and the equation $N(x) = n$ has at most one solution; if, as is often the case, $A'(x)$ goes to infinity as x goes to ρ, the equation has a unique solution x_n for each integer n. Setting x to this value x_n in the Boltzmann samplers from the previous section results in a sampling algorithm that produces structures of expected size n. Interestingly enough, the same equation also describes the value of x that maximizes the probability p_n that the output structure will have size n.

When the generating function is known exactly, one can solve for the exact value of x_n. In many situations of interest, the generating function is known only through an equation that it satisfies (this is typically the case when the class is defined recursively, as described in the next section). In this case, it is often possible to use the techniques of analytic combinatorics [10] to derive precise asymptotic information about the generating function and obtain a precise estimate of x_n.

In some cases, an attractive alternative is to use the singularity $x = \rho$; although it often implies an infinite expected size, a simple adaptation of the sampling algorithms makes this a very viable choice. This will be described in more detail in Section 5.

3 BASIC CONSTRUCTIONS AND BOLTZMANN SAMPLERS

In this section, we describe a number of constructions that can be used to describe more complex classes from simpler ones, and, for each construction, the corresponding combination algorithm that allows one to build a Boltzmann sampler for the new class using Boltzmann samplers for the classes involved in the description. The constructions described here allow one to describe combinatorial classes that are close to those of the theory of combinatorial species [1].

When nothing is specified, these constructions apply to both labelled and unlabelled structures; in the labelled case, it is silently assumed that one performs a label redistribution as in the case of the labelled product.

The initial constructions were described in [8]; later additions are credited individually.

In all cases, the construction is translated into an expression for the generating function of the new class in terms of the previous one; this in turns gives a simple construction for the sampling algorithm, where "substructures" are independent. Many algorithms can be expressed in the form "Draw integer k from discrete distribution $\mu(x)$, then let γ receive the concatenation of k independent calls to generator $\Gamma\mathscr{A}(x)$"; we abbreviate this as

$$\gamma \leftarrow [\Gamma\mu(x) \implies \Gamma\mathscr{A}(x)].$$

We also use samplers for a few standard distributions: Bernoulli with success probability x ($\mathrm{Bern}(x)$), geometric (with support $\mathbb{N}$) with parameter x ($\mathrm{Geom}(x)$), and Poisson with rate x ($\mathrm{Poiss}(x)$). A subscript condition on these samplers means a condition-

ing on the output, which can be achieved by rejecting outputs until the condition is met.

3.1 Finite Sets

Finite (typically small) sets do not require an elaborate theory to produce sampling algorithms, and are included to serve as elementary bricks for more complex constructions.

One typically defines an "empty structure" class $\mathscr{E}$, containing a single structure of size zero that we denote as **1**, and an "atom" class $\mathscr{Z}$, containing a single structure Z of size 1. Occasionally, one may use a number of different atom "types", which will then be written Z_a, Z_b, and so on.

3.2 Disjoint Union

The most basic construction is that of *disjoint union*: if $\mathscr{A}$ and $\mathscr{B}$ are disjoint classes, then their union $\mathscr{C} = \mathscr{A} \cup \mathscr{B}$ (with size inherited from the original class) is a new class, whose generating function is simply $C(z) = A(z) + B(z)$.

Algorithm $\Gamma[\mathscr{A} \cup \mathscr{B}]$
if Bern$(A(x)/(A(x)+B(x)))$ **then**
  Return $\Gamma\mathscr{A}(x)$
else
  Return $\Gamma\mathscr{B}(x)$
end if

3.3 Product

The product construction, being of fundamental importance, has been described in the previous section.

Algorithm $\Gamma[\mathscr{A} \times \mathscr{B}]$
Return $(\Gamma\mathscr{A}(x), \Gamma\mathscr{B}(x))$

3.4 Sequence

If $\mathscr{A}$ is a class with no structures of size 0, $\mathscr{C} = \mathrm{Seq}(\mathscr{A})$ is the set of sequences $(A_1, \ldots, A_k)$, for arbitrary $k \geq 0$, with $A_i \in \mathscr{A}$, and size defined additively by

$$|(A_1, \ldots, A_k)| = |A_1| + \cdots + |A_k|.$$

The generating function for $\mathscr{C}$ is the *pseudo-inverse* of that of $\mathscr{A}$,

$$C(z) = \frac{1}{1 - A(z)},$$

and the corresponding Boltzmann sampler for $\mathscr{C}$ is as follows:

Algorithm $\Gamma\mathrm{Seq}(\mathscr{A})$
$\gamma \leftarrow [\mathrm{Geom}(A(x)) \implies \Gamma\mathscr{A}(x)]$
Return γ

3.5 Cycle (labelled)

If $\mathscr{A}$ is a class with no structures of size 0, $\mathscr{C} = \mathrm{Cycle}(\mathscr{A})$ is the set of *cycles* of $\mathscr{A}$-structures, that is, sequences defined up to a circular permutation of the component $\mathscr{A}$-structures.

Working with labelled structures means that each sequence of k structures has exactly $k-1$ other structures that correspond to the same cycle. Consequently, the generating function is

$$\tilde{C}(z) = \sum_{k\geq 1} \frac{\tilde{A}^k(z)}{k} = -\log(1-\tilde{A}(z)),$$

and the corresponding Boltzmann sampler uses the "logarithmic" distribution $\mu_x(k) = \frac{x^k}{k|\log(1-x)|}$ ($k \geq 1$), provided by sampler $\mathrm{Loga}()$:

Algorithm $\Gamma\mathrm{Cycle}(\mathscr{A})$
$\gamma \leftarrow \left[\mathrm{Loga}(\tilde{A}(x)) \implies \Gamma\mathscr{A}(x)\right]$
Return γ

3.6 Set

If $\mathscr{A}$ is a labelled class with no structures of size 0, $\mathscr{C} = \mathrm{Set}(\mathscr{A})$ is the class of *sets* of $\mathscr{A}$-structures, that is, (possibly empty) sequences up to an arbitrary permutation of component structures.

The corresponding generating function is

$$\tilde{C}(z) = \sum_{k\geq 0} \frac{\tilde{A}^k(z)}{k!} = \exp(\tilde{A}(z)),$$

and the distribution for the number of components is the *Poisson* distribution:

Algorithm $\Gamma[\mathrm{Set}(\mathscr{A})]$
$\gamma \leftarrow \left[\mathrm{Poiss}(\tilde{A}(x)) \implies \Gamma\mathscr{A}(x)\right]$
Return γ

A set construction for unlabelled structures was introduced in [9]. It is based on the multiset construction described next, and requires more elaborate manipulations.

3.7 Multiset (unlabelled)

Boltzmann samplers for the Multiset construction were introduced in [9].

If $\mathscr{A}$ is an unlabelled class with no structures of size 0, $\mathscr{C} = \mathrm{MSet}(\mathscr{A})$ is the class of all *multisets* of $\mathscr{A}$-structures - sets with possible repetitions.

The corresponding generating function is

$$C(z) = \exp\left(\sum_{k\geq 1} \frac{1}{k} A(z^k)\right).$$

The Boltzmann sampler uses a $\mathrm{MaxIndex}(A,x)$ subroutine, which samples from the discrete distribution defined by

$$\mathbb{P}_{A,x}(K \leq k) = \frac{1}{C(x)} \prod_{j\leq k} \exp\left(\frac{1}{j} A(x^j)\right).$$

The Boltzmann sampler itself is then as follows:

Algorithm $\Gamma\mathrm{MSet}(\mathscr{A})$
$\gamma \leftarrow \emptyset$
$k \leftarrow \mathrm{MaxIndex}(A,x)$
for j from 1 to $k-1$ **do**
  $\gamma \leftarrow \gamma \cup \left[\mathrm{Poiss}(A(x^j)/j) \implies j \text{ copies of } \Gamma\mathscr{A}(x^j)\right]$
end for
$\gamma \leftarrow \gamma \cup \left[\mathrm{Poiss}_{\geq 1}(A(x^k)/k) \implies k \text{ copies of } \Gamma\mathscr{A}(x^k)\right]$.
Return γ

(It should be noted that k in the above algorithm is *not* the maximum number of repetitions of a structure in the output, but only a lower bound: all calls to the sampler $\Gamma\mathscr{A}$ are independent, so that a structure may be output more than once and obtain larger multiplicity.)

Note that, in contrast to the previous constructions, the value $C(z)$ is expressed not in terms of the value $A(z)$, but of the values of A for a whole geometric sequence of values.

3.8 Recursive Constructions

All of the above constructions can be used *recursively*, *i.e.* a class $\mathscr{C}$ can be defined using one of these constructions on a class that is itself (ultimately) defined in terms of $\mathscr{C}$ itself. Some care must be taken to avoid circular definitions: recursive specifications define structures from smaller structures, possibly of the same type, but not from themselves. Thus, one can define a class $\mathscr{P}$ by $\mathscr{P} = \mathscr{Z} \times \mathrm{Seq}(\mathscr{P})$ (this defines plane trees: a plane tree is composed of a root having an ordered sequence (possibly empty) of children, each the root of a plane tree), but a specification such as $\mathscr{P} = \mathscr{A} \times \mathscr{P}$ would be invalid if class $\mathscr{A}$ contains structures of size 0 (it would then attempt to create an infinite number of $\mathscr{P}$-structures of size 0).

Subject to this "well-foundedness" condition, all the previous constructions can be used recursively - and indeed, in most applications of interest recursivity is used.

Whenever it is the case, the Boltzmann samplers derived from the previous subsections become recursive algorithms, for which termination can only be guaranteed with probability 1 (and in finite expected time; see Section 4).

An effective characterization of this "well-foundedness" condition is given in [12], for specifications that outright forbid structures of size 0.

3.9 Ordered Structures and Differential Operators (labelled)

The constructions described in this subsection appear in [15, 5].

The derivative α' of a labelled combinatorial structure α is obtained by replacing the atom in α having the largest label with a "hole" - this hole holds the place of an atom, but does not contribute to size and does not get a label. Thus, the derivative of a structure of size n has size $n-1$. The derivative of a combinatorial class $\mathscr{A}$ is, of course, the class $\mathscr{C} = \mathscr{A}' = \{\alpha' : \alpha \in \mathscr{A}\}$ of all derivatives of $\mathscr{A}$-structures. The corresponding (exponential) generating functions are related by

$$\begin{aligned} \tilde{C}(z) &= \tilde{A}'(z), \\ \tilde{A}(z) &= a_0 + \int_0^z \tilde{C}(z)dz, \end{aligned}$$

where a_0 is the number of $\mathscr{A}$-structures of size 0.

Derivative classes can be used in recursive constructions, under suitable "well-foundedness" conditions (see [2, 5] for details), to define a class as the solution to a symbolic differential equation. In very rough terms, this corresponds to imposing order conditions on labels. A simple example is provided by the class $\mathscr{T}$ of *decreasing (labelled) binary trees*, that is, labelled binary plane trees where each node (atom) is required to have a larger label than each of its children: the equation reads

$$\begin{aligned} \mathscr{T}' &= \varepsilon \cup \mathscr{T} \times \mathscr{T} \\ \mathscr{T}_0 &= \emptyset. \end{aligned}$$

and should be understood as this: the largest label in a decreasing binary tree has to be at the root, so its derivative will either be a unique object of size zero or equivalent (after relabelling) to a pair of binary trees, each of which has to be decreasing.

Bodini *et al.* [5] describe a generic Boltzmann sampler for a class defined by a first-order differential operator $\mathscr{A}' = \mathscr{F}(\mathscr{Z}, \mathscr{A})$, provided one has a Boltzmann sampler for the class $\mathscr{F}(\mathscr{Z}, \mathscr{A})$ (that is, $\mathscr{F}$ is defined in terms of other classical constructions, and the whole sampler will necessarily be recursive). Like the sampler for multisets, it requires a change of the parameter - here, a random change - for each recursive call.

Given the generating function A and a parameter $0 < x_0 < \rho_A$, one defines a probability density on the interval $[0,1]$ by

$$h_{x_0,A}(u) = \frac{x_0 A'(ux_0)}{A(x_0) - A(0)};$$

if U is a random variable following this distribution, Ux_0 can be interpreted as the result of picking a random point (according to Lebesgue measure) in the domain $0 < y < A(x), 0 < x < x_0$, and keeping the abscissa x.

With this definition, the Boltzmann sampler is as follows:

Algorithm $\Gamma\mathscr{A}$ from $\Gamma\mathscr{A}'$, $\mathscr{A}' = \mathscr{F}(\mathscr{Z}, \mathscr{A})$
if Bern$(A(0)/A(x))$ **then**
  Return a uniform object from $\mathscr{A}_0$
else
  Draw $U \in [0,1]$ following density $h_{x,A}$
  $f \leftarrow \Gamma\mathscr{F}[\mathscr{Z}, \mathscr{A}](Ux)$
  Return object (Z, f) with atom Z having the largest label
end if

3.10 Multivariate Models

So far, we have only considered generating functions with a single variable, which "counts" for the size of the structures; that is, each structure γ in the class contributes a single term $z^{|\gamma|}$ or $z^{|\gamma|}/\gamma!$ to the generating function.

Given a combinatorial class $\mathscr{C}$, one can define an arbitrary number of statistics $s_i : \mathscr{C} \to \mathbb{N}$ $(1 \le i \le k)$, and the corresponding multivariate generating function $C(z, u_1, \ldots, u_k)$ (first as a multivariate formal power series, then as an analytic function) by changing the contribution of each structure $c \in \mathscr{C}$ to $z^{|c|} \prod_{1 \le i \le k} u_i^{s_i(c)}$ (for the rest of this subsection, we assume unlabelled structures), and consequently, a Boltzmann distribution for any tuple $(x, u_1, \ldots, u_k)$ of positive real variables lying inside the convergence domain of the generating function.

When the considered statistics are transmitted additively under the constructions described in this section (say, if there are several types of "atoms", and statistic s_i counts the number of atoms of type i), the Boltzmann samplers can be adapted to this generalized model.

This area has not been explored as extensively as others, probably because a general theory would involve additional technical details. Bodini and Ponty [4] use it to sample from context-free languages with a nonuniform distribution where the frequency of letters is artificially skewed, with an application to "Tetris tesselations" (perfect tilings of a rectangular region with pentominoes) where each piece has the same frequency.

4 ALGORITHM COMPLEXITIES

Each of the individual algorithms in Section 3 has low overhead complexity, but they tend to make possibly unbounded numbers of calls to other algorithms. The general theorem below is a compilation of results from [8] and other papers that extend the expressive power of "specifiable" classes.

Theorem 1 Let $\mathscr{C}$ denote a (labelled or unlabelled) class that can be entirely specified, in a possibly recursive way, with the constructions of Section 3, and let $0 < x < \rho$ be any positive real lying inside the convergence domain for the generating function of $\mathscr{C}$. Then, assuming an oracle that provides values of the relevant generating functions at real values, the algorithm $\Gamma\mathscr{C}$ compiled from the specification by the patterns of Section 3 terminates with probability 1 and in finite expected time, outputs a random $\mathscr{C}$-structure under the Boltzmann distribution with parameter x, and uses a number of

real arithmetic operations that is *linear* in the size of the output.

5 APPROXIMATE AND EXACT SIZE SAMPLERS

The user of random generation algorithm is often used to *exact* random samplers (algorithms that take n as input, and output a uniform random element of the subclass $\mathscr{C}_n$), or perhaps *approximate-size* samplers (algorithms that take two integers $n < N$ as input, and output a random element with size in $[n,N]$ with equal probability for any two elements with the same size). Both types can be obtained by adding a rejection mechanism to Boltzmann samplers, at overall costs that depend on the distribution of sizes under the Boltzmann model.

We denote $\mu_1(x)$ for the expected size, $\mu_2(x)$ for the expected squared size, and $\sigma(x)$ for the standard deviation on size, all as functions of parameter x; analytically,

$$\begin{aligned}\mu_1(x) &= \frac{xC'(x)}{C(x)}\\ \mu_2(x) &= \frac{xC'(x)+x^2C''(x)}{C(x)}\\ \sigma(x) &= \sqrt{\mu_2(x)-\mu_1^2(x)}\end{aligned}$$

5.1 Approximate Size Samplers

Assume we are given a target size n, some tolerance $\varepsilon > 0$, and the value x_n of the parameter that ensures that the expected size of structures is n. We obtain an approximate size sampler with acceptable sizes in $I = I(n,\varepsilon) = ((1-\varepsilon)n,(1+\varepsilon)n)$ by repeatedly using the Boltzmann sampler $\Gamma\mathscr{C}(x_n)$, until a structure whose size lies in I appears.

Since the cost of each call to the Boltzmann sampler is linear in the size of the output by Theorem 1, the expected cost of this approximate size sampler is asymptotic to n times the expected number of calls to the Boltzmann sampler.

In favorable situations, described as "lumpy" distributions in [8] and characterized by $\sigma(x)/\mu_1(x) \to 0$ as $x \to \rho^-$, this expected number of calls goes to 1 as n (the target size) goes to infinity – the probability that the first call will yield a structure whose size is in I is asymptotically 1. More precise information on the distribution of sizes produced by such an approximate-size sampler can be obtained through the asymptotics of the generating function. Typically, for "lumpy" distributions, this size is concentrated around the expected size, and shorter intervals of length $o(n)$ could be used without altering the theoretical results.

In less favorable situations, this success probability goes to a finite positive constant as n goes to infinity, so that the expected cost of the approximate size sampler is still asymptotically linear. In many cases, a size tolerance of, say, 5% around the target size, at the cost of a constant number of rejections, is quite acceptable.

In some even less favorable situations, it may be necessary to change the class by using "pointing" (a combinatorial operation close to derivation, corresponding to distinguishing a single atom in the structure) a finite number of times before one gets to the situation described above.

5.2 Exact Size Samplers

Setting $\varepsilon = 1/n$ in the approximate size samplers of the previous subsection, results in exact size samplers. It should be noted, however, that in most cases, the success probability (the probability, with parameter x_n, of obtaining a structure of size exactly n) is only of order $\Theta(1/n)$, which results in an expected complexity $\Theta(n^2)$ for the exact size sampler.

5.3 Singular Samplers

Whenever the generating function is finite at its dominant singularity ρ, one can define a Boltzmann distribution for $x = \rho$, and the Boltzmann samplers can be used with parameter ρ. This is typically (though not universally) true with recursive specifications, and the most frequent case is for the generating function to have a "square root-type" singularity, *i.e.* $C(z)$ has a singular expansion of the form $C(z) = C(\rho) + a(1 - z/\rho)^{1/2} + o((1 - z/\rho)^{1/2})$ as z approaches ρ.

In such cases, the expected size for the singular Boltzmann model is infinite. While this offers the best chances of success for the approximate size samplers, it also implies that the expected cost of the same approximate size sampler is infinite, which is unacceptable.

However, on closer examination, this infinite expectation only comes from those (rare: the probability is $\Theta(1/\sqrt{n})$) runs of the sampler where the output size is much larger than the target n. A simple modification of the Boltzmann samplers can thus avoid this higher cost, by keeping track of the number of atoms generated so far and aborting the Boltzmann sampler as soon as the total becomes larger than n; with this modification, expected costs for a square root singularity become $\Theta(n)$ for approximate size with finite ε, and $\Theta(n^{3/2})$ for exact size.

6 GENERATING FUNCTION EVALUATION AND PRECISION

One of the key points of the Boltzmann method, when compared to the recursive method, is that enumeration sequences are replaced by *generating function evaluations*. In many of the constructions of Section 3, each involved generating function needs to be evaluated for the same value of its variable; in a few of them, some have to be evaluated for a deterministic or random sequence of values.

In some cases, the generating functions have closed forms and this evaluation does not lead to special complications, but recursive specifications lead to generating functions that are determined by equations, and the question of determining a good approximation of the required values becomes more troublesome.

Pivoteau, Salvy and Soria [12] provide an efficient solution to this problem, at least for the basic constructions of sums, products, sequences, cycles and sets. The natural idea of iterating the equations provides only slow convergence; the preferred method is based on Newton iteration, which ensures quadratic convergence (asymptotically, distance to the exact solution is squared by each iteration). For a given value of the variable, the generating function equations typically have several real solutions,

only one of which corresponds to the generating function that is the "true" solution; an important result in the above-mentioned paper is that, thanks to the existence of a combinatorial equivalent to the Newton iteration, convergence to the "true" value is ensured.

Also, note that, when applying Newton iteration for generating function evaluation, convergence is significantly faster the further x is from the dominant singularity ρ. While this is good for constructions such as multisets or ordered structures that require the use of sequences of values (for sequences of values of the variable that quickly decrease to zero), it is conversely bad news for classical applications that require very large structures, since this means taking values of x that are very close to ρ. Nevertheless, the experimental results reported in [12], even for very complex specifications, remain within reasonable bounds (the computation time for the oracles of specifications implying 500 equations, for values of the parameter leading to expected structure sizes in the tens of thousands, are of the order of a minute).

Another question that arises naturally is that of the influence of approximations on the final distribution of random samples. Even assuming a "perfect" source of randomness for the simulations, a small error in a generating function value that is used repeatedly by a sampling algorithm might result in a significantly biased distribution of the final samples.

To give an example, suppose a specification involves a disjoint union $\mathscr{C} = \mathscr{A} \cup \mathscr{B}$, and each of $\mathscr{A}$ and $\mathscr{B}$ is defined recursively using $\mathscr{C}$ - this is not an artificial assumption. Assume that, for the value x of the parameter, $A(x)$ is slightly overestimated, and $B(x)$ is slightly underestimated. In this case, each time the sampler for $\mathscr{C}$ is used, it will have a tendency to switch to $\mathscr{A}$ more often than it ideally should; this will result in a distribution that is biased in favor of $\mathscr{C}$-structures that often use $\mathscr{A}$-structures as components.

One possible solution [6] to estimate, and possibly correct, this bias, is to design Boltzmann samplers that not only output a random structure, but also, for each real-valued constant $A \simeq A(x)$ used in the sampling, a "safety interval" $[A^-, A^+]$, with $A^- < A < A^+$, with a precise meaning of "if the sampler had been run with any value in $[A^-, A^+]$ instead of A, the result of the whole computation would have been the same". This is done by studying the small number of discrete distributions one really needs to sample from.

From such "safety interval" samplers, one can derive both a practical and a theoretical result:

- An estimate of the quality of approximation one should have on each involved constant, such that the whole Boltzmann sampler is very unlikely to output any safety interval that does *not* contain the true value of the corresponding generating function. Not surprisingly, $\Theta(\log n)$ bits are enough when the expected size is n.

- A (still hypothetical) construction for a *truly exact* Boltzmann sampler, if one assumes stronger oracles than those provided by [12]. If one assumes oracles that give both an upper an a lower bound for each involved constant, and that the oracles can be called repetitively to decrease the difference between these bounds (say, by a factor of 2 with every iteration), then the "safety interval" approximate

Boltzmann samplers can be modified into exact Boltzmann samplers which will call the oracles a small (logarithmic) number of times on average.

7 NOT-QUITE BOLTZMANN SAMPLERS

Boltzmann samplers share two important and useful properties for the user interested in practical random sampling of large structures: (1) they output structures of random size, with the guarantee that *any two structures of the same size have the same probability of being obtained*, and (2) they are *"stable" under the constructions* of Section 3.

While property (1) is the essential one for direct practical applications, making it possible to use, say, rejection to transform Boltzmann samplers into exact- or approximate-size samplers, property (2) is the one responsible for the wide applicability of the method. Nevertheless, sometimes property (1) alone can be obtained while keeping algorithms of low complexity, typically by using a "nonstandard" final construction using Boltzmann samplers as subroutines.

A good example of this is the *Hadamard product* of two combinatorial class. If $\mathscr{A}$ and $\mathscr{B}$ are two arbitrary (unlabelled) combinatorial classes, their Hadamard product $\mathscr{C} = \mathscr{A} \odot \mathscr{B}$ is the subset of $\mathscr{A} \times \mathscr{B}$ that only contains pairs $c = (a,b)$ with the same size (with $|c|$ defined to be $|a| = |b|$ instead of the sum for the classical product). The corresponding generating function is none other than the Hadamard product of generating functions,

$$C(z) = A \odot B(z) = \sum_n a_n b_n z^n,$$

with radius of convergence at least the product of radii of $A(z)$ and $B(z)$.

A real Boltzmann sampler for $\mathscr{C}$ can be written easily: one simply checks that, if $x = x_A x_B$ with $x_A < \rho_A$ and $x_B < \rho_B$, the algorithm

repeat
  $\alpha \leftarrow \Gamma\mathscr{A}(x_A)$
  $\beta \leftarrow \Gamma\mathscr{B}(x_B)$
until $|\alpha| = |\beta|$
Return (α, β)

terminates with probability 1 and in finite expected time, and outputs a $\mathscr{C}$ structure under the Boltzmann distribution with parameter x. It is, however, inefficient: each iteration has success probability $C(x)/(A(x_A)B(x_B))$, which can be very small when x is close to the dominant singularity.

Bodini *et al.* [3] use the classical Birthday paradox to devise a more efficient algorithm that preserves property (1), though not the whole Boltzmann distribution: simply alternate (either deterministically or randomly) between $\Gamma\mathscr{A}(x_A)$ and $\Gamma\mathscr{B}(x_B)$, retaining only the first obtained structure of each class and size, until a pair with the same size can be formed.

8 CONCLUSION

Boltzmann samplers are an attractive class of random generation algorithms for classes of combinatorial structures that lend themselves to combinatorial decompositions, al-

lowing for fast (linear-time, or quasi-linear-time) generation of structures with size in the millions for simple classes, and well into the tens of thousands for complex classes.

References

[1] François Bergeron, Gilbert Labelle, and Pierre Leroux. *Combinatorial species and tree-like structures*. Number 67 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, 1998.

[2] Olivier Bodini. *Autour de la génération aléatoire sous modèle de Boltzmann*. Habilitation memoir, Université Pierre et Marie Curie, 2010. in French.

[3] Olivier Bodini, Danièle Gardy, and Olivier Roussel. Boys-and-girls birthdays and Hadamard products. In *Proceedings of Lattice Paths Combinatorics and Applications*, 2010.

[4] Olivier Bodini and Yann Ponty. Multi-dimensional Boltzmann sampling of languages. In *Proceedings of the 21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'10)*, pages 49–64. DMTCS Proceedings, 2010.

[5] Olivier Bodini, Olivier Roussel, and Michèle Soria. Boltzmann samplers for first order differential specifications. *To appear in* Discrete Applied Mathematics, 2011.

[6] Philippe Duchon. Exact Boltzmann samplers from adaptive approximate oracles. In preparation, 2011.

[7] Philippe Duchon, Philippe Flajolet, Guy Louchard, and Gilles Schaeffer. Random sampling from Boltzmann principles. In P. Widmayer, F. Triguero, R. Morales, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *Proceedings of the 29th ICALP*, volume 2380 of *Lecture Notes in Computer Science*, pages 501–513. Springer, 2002.

[8] Philippe Duchon, Philippe Flajolet, Guy Louchard, and Gilles Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, Probability and Computing*, 13(4-5):577–625, 2004.

[9] Philippe Flajolet, Éric Fusy, and Carine Pivoteau. Boltzmann sampling of unlabeled structures. In *Proceedings of ANALCO 07*, pages 201–211, New Orleans, 2007.

[10] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. Cambridge University Press, Cambridge, 2009.

[11] Philippe Flajolet, Paul Zimmermann, and Bernard Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theor. Comput. Sci.*, 132(2):1–35, 1994.

[12] Carine Pivoteau, Bruno Salvy, and MichÈle Soria. Boltzmann oracle for combinatorial systems. In *Algorithms, Trees, Combinatorics and Probabilities*, pages 475 – 488. Discrete Mathematics and Theoretical Computer Science, 2008. Proceedings of the Fifth Colloquium on Mathematics and Computer Science. Blaubeuren, Germany. September 22-26, 2008.

[13] James Gary Propp and David Bruce Wilson. How to get a perfectly random sample from a generic markov chain and generate a random spanning tree of a directed graph. *J. Algorithms*, 27(2):170–217, 1998.

[14] J.-L. Rémy. Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire. *RAIRO Theoretical Informatics and Applications*, 27(2):179–195, 1985.

[15] Olivier Roussel and Michèle Soria. Boltzmann sampling of ordered structures. In *Proceedings of LAGOS 09*, 2009. 9 pages.

AUTHOR BIOGRAPHY

PHILIPPE DUCHON is a Professor in the Computer Science department of Université Bordeaux 1. Before joining the university, he was an Associate Professor in the ENSEIRB engineering school. He is a former student of École Normale Supérieure and received his Ph.D. in Computer Science from Université Bordeaux 1 in 1998. His research interests are in all areas of enumerative combinatorics, random generation algorithms, and the analysis of algorithms for distributed systems. His email address is duchon@labri.fr.