RANDOM SAMPLING
FROM BOLTZMANN PRINCIPLES

PHILIPPE DUCHON, PHILIPPE FLAJOLET, GUY LOUCHARD, GILLES SCHAEFFER

ABSTRACT. This note proposes a new framework for random generation of combinatorial configurations based on what we call Boltzmann models. The idea is to perform random generation of possibly complex structured objects by placing an appropriate measure spread over the whole of a combinatorial class. The resulting algorithms often operate in linear time. They can be implemented easily within a computer algebra system, be analysed mathematically with great precision, and, when suitably tuned, tend to be very efficient in practice.

1. INTRODUCTION

The problem considered here is that of generating samples of structured combinatorial objects of a certain size. In the usual setting of combinatorics, the objects should be drawn *uniformly* at random from the family of all objects of the same size. This is the objective of packages available in some computer algebra systems, like Combstruct (under MAPLE) and CS (under MUPAD). Such general-purpose packages however involve a costly preprocessing phase as they necessitate the maintenance of large tables of multiprecision integers. Accordingly, their use in extensive simulations on processors with clock speed around a GigaHertz is usually limited to sizes of a few thousand. The situation can be somewhat remedied by developing carefully crafted floating-point algorithms, but such implementations remain essentially of a nonlinear (usually quadratic) time complexity.

In this note, we present a surprisingly simple approach to the random generation of many types of combinatorial objects. The basic principle is to *relax* the constraint of generating objects of a strictly fixed size, and prefer to draw objects with a somewhat randomly fluctuating size. The algorithms developed make use of a *continuous* control parameter $x > 0$. As we shall see, normally, one can *tune* the value of x in order to draw objects of a size in some vicinity of a target size n. The time complexity of the resulting algorithms is linear (requiring also only $O(1)$ storage). A "tolerance" of, say, a few percents on size of the object produced is likely to cater for most practical simulation needs. If the tuning mentioned above is not sufficient, one can always pile up a rejection method to either refine the size of the element drawn or even aim at the exact target n. The resulting algorithms still tend to be competitive, if not superior, when compared to existing combinatorial methods. Accordingly, objects with sizes easily in the range of millions can be generated uniformly at random, in the great many cases where the method applies.

The ideas presented here draw their origin from many sources. First what has come to be known as the "recursive method" method originating with Nijenhuis and Wilf and formalized by Flajolet, Zimmermann, and Van Cutsem [10] served as a key conceptual guide. Ideas from a statistical physics point of view on combinatorics, of which great use was made by Vershik and his collaborators [2, 31], then provided important intuition regarding the new class of algorithms for random generation that is presented here. Another important ingredient is the collection of rejection algorithms developed by Duchon, Louchard, and Schaeffer for certain

Corresponding author: Philippe.Flajolet@inria.fr. Version of Nov. 11, 2001. Supported in part by the ALCOM-FT Project IST-1999-14186 of the E.U.

types of trees, polyominos, and planar maps [6, 21, 26]. There are also similarities with the technique of "shifting the mean" (see Greene and Knuth's book [13, p. 78–80]) and we have benefited from discussions with Alain Denise on these aspects. Finally, the principles of analytic combinatorics (see [9]) provide essential clues for deciding situations in which the algorithms are likely to be efficient.

We consider a class $\mathcal{C}$ of combinatorial objects of sorts, with $|\cdot|$ the size function from $\mathcal{C}$ to $\mathbb{Z}_{\geq 0}$. By $\mathcal{C}_n$ is meant the subclass of $\mathcal{C}$ comprising all the objects in $\mathcal{C}$ having size n. Each $\mathcal{C}_n$ is assumed to be finite. One may think of binary words (with size defined as length), permutations, graphs and trees of various types (with size defined as number of vertices), and so on. Any set $\mathcal{C}$ endowed with a size function and satisfying the finiteness axiom will henceforth be called a *combinatorial class*.

The *uniform probability distribution* over $\mathcal{C}_n$ assigns to each $\gamma \in \mathcal{C}_n$ the probability $\mathbb{P}_{\mathcal{C}_n}\{\gamma\} = 1/C_n.$, with $C_n := \operatorname{card}(\mathcal{C}_n)$. Our primary objective is to draw uniformly at random from the class $\mathcal{C}_n$; this is called *uniform exact-size* random generation. On many occasions, we shall also consider random generation from "neighbouring classes", $\mathcal{C}_N$ where N may not be totally under control, but should still be in the vicinity of n, namely, $(1-\varepsilon)n \leq N \leq (1+\varepsilon)$, for some "*tolerance*" factor ε; this is called *uniform approximate-size* random generation.

2. Boltzmann Models

Definition 1. *The* Boltzmann models *of parameter x exist in two brands, the ordinary version and the exponential version. They assign to any object $\gamma \in \mathcal{C}$ the following probability:*

$$\text{Ordinary/Unlabelled case:} \qquad \mathbb{P}_x(\gamma) = \frac{1}{C(x)} \cdot x^{|\gamma|} \quad \text{with} \quad C(x) = \sum_{\gamma \in \mathcal{C}} x^{|\gamma|},$$

$$\text{Exponential/Labelled case:} \qquad \mathbb{P}_x(\gamma) = \frac{1}{\widehat{C}(x)} \cdot \frac{x^{|\gamma|}}{|\gamma|!} \quad \text{with} \quad \widehat{C}(x) = \sum_{\gamma \in \mathcal{C}} \frac{x^{|\gamma|}}{|\gamma|!}.$$

A Boltzmann generator (or sampler) $\Gamma C(x)$ *for a class $\mathcal{C}$ is a process that produces objects from C according to a Boltzmann model.*

The normalization coefficients are nothing but the counting generating functions of ordinary type (OGF) for $C(x)$ and exponential type (EGF) for $\widehat{C}(x)$. Only *coherent* values of x defined to be such that $0 < x < \rho_C$ (or $\rho_{\widehat{C}}$), with ρ_f the radius of convergence of f are to be considered.

The name "Boltzmann model" comes from the great statistical physicist Boltzmann whose works (together with those of Gibbs) led to enounce the following principle: *Statistical mechanical configurations of energy equal to E in a system have a probability of occurrence proportional to $e^{-\beta E}$.* If one identifies size of a combinatorial configuration with energy of a thermodynamical system and sets $x = e^{-\beta}$, then what we term the ordinary Boltzmann models become the true model of statistical mechanics. The counting generating function in the combinatorial world then coincides with the normalization constant in the statistical mechanics world where it is known as the *partition function* and is often denoted by Z. Under this perhaps artificial dictionary, Boltzmann models and random combinatorics become united.

For reasons which will become apparent, we have also defined the exponential Boltzmann model. These are appropriate for handling *labelled* combinatorial structures while the ordinary models are to be used for *unlabelled* combinatorial models[1]. An obvious consequence of the definition is then: *The Boltzmann model of parameter x conditioned by the fact that the size of the object drawn equals n coincides with the* uniform *model on $\mathcal{C}_n$.* This gives rise to a very simple random generator sampling uniformly over $\mathcal{C}_n$, written $\mu C(n)$, once a random sampler $\Gamma C(x)$ for the Boltzmann model $\mathcal{C}$ with parameter x is assumed to be given:

```
function uC(n : integer size);
{Uses a Boltzmann generator ''gC''}
    choose an x such that 0<x<rho;
    repeat
       gamma := gC(x);
    until |gamma| = n;  end.
```

This is plainly based on rejection, a supple technique of subtle effectiveness in many contexts [4]. Random generation of "*approximate size*" is also easily obtained by suitably weakening the halting condition of the repeat loop (thereby increasing the probability of success).

Precisely, the size of the resulting object under a Boltzmann model is a random variable denoted throughout by N, whose law is quantified by the following lemma. (See, e.g., Huang's book [16] for similar calculations from the statistical mechanics angle.)

Proposition 1. *The random size of the object produced under the ordinary Boltzmann model of parameter x satisfies*

$$\mathbb{E}_x(N) = x\frac{C'(x)}{C(x)}, \qquad \mathbb{E}_x(N^2) = \frac{x^2C''(x) + xC'(x)}{C(x)}. \tag{1}$$

Proof. The probability generating function of N is $\sum_n \mathbb{P}_x(N = n)z^n = \dfrac{C(xz)}{C(x)}$. □

Henceforth, we shall examine conditions that make the Botzmann generator an efficient alternative to existing random generators. The main questions are:

— How can one assemble Boltzmann generators from simpler ones so as to develop a general methodology for the random generation of structurally complex objects?

— How should one control the continuous parameter x so as to bring the random size N "close" to the target value n, thereby increasing the efficiency of the rejection method μC?

The first question is a critical design issue and a determinant of *correctness*; the second one quantifies *efficiency.* We examine them in sequence assuming an *infinite precision model over the reals*, for the time being (see Sec. 5.3 for a discussion).

[1]In the unlabelled universe, all elementary components of objects ("atoms") are indistinguishable, while in the labelled universe, they are all distinguished from one another by bearing a distinctive mark, say one of the integers between 1 and n if the object considered has size n. Permutations written as sequences of distinct integers are typical labelled objects while words over a binary alphabet appear as typical unlabelled objects made of "anonymous" letters, say $\{a, b\}$ for a binary alphabet. This terminology is standard in combinatorial enumeration and graph theory (see, e.g., the books of Goulden-Jackson, Harary-Palmer, Stanley [11, 14, 30] or [9]).

3. Ordinary Boltzmann Generators

A *combinatorial construction* builds a new class $\mathcal{C}$ from structurally simpler classes $\mathcal{A}, \mathcal{B}$, in such a way that $\mathcal{C}_n$ is determined from objects in $\{\mathcal{A}_j\}_{j=0}^n, \{\mathcal{B}_j\}_{j=0}^n$. Constructions considered here are disjoint *union* (+), cartesian *product* (×), and *sequence* formation ($\mathfrak{S}$). We define these in turn and concurrently build the corresponding Boltzmann sampler ΓC for the composite class $\mathcal{C}$, given random generators $\Gamma A, \Gamma B$ for the ingredients and assuming the values of intervening generating functions $A(x), B(x)$ at x to be known exactly.

Disjoint union. Write $\mathcal{C} = \mathcal{A} + \mathcal{B}$ if $\mathcal{C}$ is the union of disjoint copies of $\mathcal{A}$ and $\mathcal{B}$, while size on $\mathcal{C}$ is inherited from $\mathcal{A}, \mathcal{B}$. One has

$$C(x) = A(x) + B(x). \tag{2}$$

The Boltzmann model corresponding to $C(x)$ is then a mixture of the models associated to $A(x)$ and $B(x)$, with the probabilities of selection of a γ in $\mathcal{C}$ being

$$\mathbb{P}(\gamma \in \mathcal{A}) = \frac{A(x)}{C(x)}, \qquad \mathbb{P}(\gamma \in \mathcal{B}) = \frac{A(x)}{C(x)}.$$

This gives rise to a random generator ΓC given ΓA and ΓB:

```
function gC(x : real); {generates C = A + B}
Uses generators gA(x), gB(x), and values of the OGFs A(x), B(x).
   pA := A(x)/(A(x)+B(x)); draw U uniformly in [0,1];
   if U<pA then return gA(x) else return gB(x) fi;  end.
```

Cartesian Product. Write $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ if $\mathcal{C}$ is the set of ordered pairs from $\mathcal{A}$ and $\mathcal{B}$, and size on $\mathcal{C}$ is inherited additively from $\mathcal{A}, \mathcal{B}$. For generating functions

$$C(x) = A(x) \cdot B(x) \quad \text{since} \quad C(x) = \sum_{\langle \alpha, \beta \rangle \in \mathcal{A} \times \mathcal{B}} x^{|\alpha| + |\beta|}. \tag{3}$$

A simple reasoning shows that a random element of $C(x)$ is then obtained by forming a pair $\langle \alpha, \beta \rangle$ with α, β drawn *independently* from the Boltzmann models $A(x), B(x)$, respectively.

```
function gC(x : real); {generates C = A x B}
     return the independent pair <gA(x),gB(x)> fi;  end.
```

Sequences. Write $\mathcal{C} = \mathfrak{S}(\mathcal{A})$ if $\mathcal{C}$ is composed of all the finite sequences of elements of $\mathcal{A}$ (with size of a sequence additively inherited from sizes of components). The sequence class $\mathcal{C}$ is also the solution to the symbolic equation $C = \mathbf{1} + AC$ (with $\mathbf{1}$ the empty sequence), which only involves unions and products. Consequently,

$$C(x) = \frac{1}{1 - A(x)}. \tag{4}$$

This gives rise to two logically equivalent designs for a ΓC generator:

(i) the recursive generator: Draw $\mathbf{1}$ with probability $1/C(x)$ and stop; else, draw $\alpha \in \mathcal{A}$ under $A(x)$ and, recursively, $\gamma \in \mathcal{C}$ according to $C(x)$.

(ii) the geometric generator: draw K randomly according to the *geometric* distribution of parameter $A(x)$; then draw $(\alpha_1 \ldots, \alpha_K)$ under $A(x)$.

The recursive generator for sequences is built from first principles (union and product rules). It might in principle loop for ever. However, a simple analysis shows that it stops with probability 1, the expected number of recursive calls being finite and equal to $(1 - A(x))^{-1}$. The geometric generator unwinds the recursion of the basic recursive generator.

There finally remains to discuss two more points: initialization (when and how do we stop?) and recursion. Clearly if $\mathcal{C}$ is finite (and in practice small), one can generate a random element of $\mathcal{C}$ by selecting it according to the finite probability distribution given by the Boltzmann model. Next, as suggested by the sequence construction, recursively defined classes admit generators that will call themselves recursively. It can be proved that under a natural combinatorial condition called "well-foundedness"[2], all generators constructed in this way terminate with probability 1 and in finite expected time.

Theorem 1. *Consider an unlabelled class $\mathcal{C}$ defined (in a possibly recursive way) from finite sets by means of disjoint unions, cartesian products, and sequences. Let x be a "coherent" parameter in $(0, \rho_C)$, and let ΓC be the generator compiled from the definition of $\mathcal{C}$ by means of the three rules above. Then ΓC correctly draws elements from $\mathcal{C}$ according to the ordinary Boltzmann model. It halts with probability 1 and in finite expected time.*

EXAMPLE 1. *Words without long runs.* Consider the collection $\mathcal{R}$ of binary words over the alphabet $\mathcal{A} = \{a, b\}$ such that they never have more than m consecutive occurrences of any letter (such things are also called "runs" and intervene at many places in statistics). It is not *a priori* obvious how to generate a random word[3] in $\mathcal{W}$ of length n. Any word decomposes into a sequence of alternations like

$$(aa\cdots a \mid bb\cdots b)\,(aa\cdots a \mid bb\cdots b)\ \cdots (aa\cdots a \mid bb\cdots b),$$

possibly prefixed with a header of b's and postfixed with a trailer of a's. In symbols, the set $\mathcal{W}$ of all words is expressible by a regular expression written in our notation

$$\mathcal{W} = \mathfrak{S}(b) \times \mathfrak{S}\,(a\mathfrak{S}(a)b\mathfrak{S}(b)) \times \mathfrak{S}(a).$$

The decomposition was customized to serve for $\mathcal{R}$: simply replace any internal $a\mathfrak{S}(a)$ by $\mathfrak{S}_{1\,..\,m}(a)$ and any $b\mathfrak{S}(b)$ by $\mathfrak{S}_{1\,..\,m}(b)$, where $\mathfrak{S}_{1\,..\,m}$ means a sequence of between 1 and m elements. The composition rules given above give rise to a generator for $\mathcal{R}$ that has the following form: two generators that produce sequences of a's or b's according to a truncated geometric law; a generator for the product $\mathcal{C} := (\mathfrak{S}_{1\,..\,m}(a)\mathfrak{S}_{1\,..\,m}(b))$ that is built according to the product rule; a generator for the sequence $\mathcal{D} := \mathfrak{S}(\mathcal{C})$ constructed according to the sequence rule. The generator finally assembled *automatically* is (with more or less obvious notations)

$$\underset{\le m}{\mathrm{Geom}}(x; b)\left\{\ \mathrm{Geom}\left[\tfrac{x^2(1-x^m)^2}{(1-x)^2}\right] \circ \left\langle \underset{1\,..\,m}{\mathrm{Geom}}(x; a),\ \underset{1\,..\,m}{\mathrm{Geom}}(x; b)\right\rangle\ \right\} \underset{\le m}{\mathrm{Geom}}(x; a),$$

which takes care of headers and trailers. $\square$

---

[2]A specification by means of constructors is well-founded if it builds objects from smaller ones. A condition when no recursion is involved is that the sequence (and, for exponential Boltzmann models below, set, and cycle) operations are never applied to classes that contain objects of size 0. For recursive structures this is a testable property akin to "properness" in the theory of context-free grammars. This condition also guarantees that the equations defining generating function equations are well-posed and contracting for all coherent values of x. See [8] on this topic.

[3]A brutal rejection method based on generating random words and filtering out those that satisfy the condition $\mathcal{R}$ won't work in polynomial time since such words have an exponentially small probability of existing.

Example 2. *Binary trees (rooted, plane).* Take the class $\mathcal{B}$ of binary trees defined by the recursive specification

$$\mathcal{B} = \mathcal{Z} + (\mathcal{Z} \times \mathcal{B} \times \mathcal{B}),$$

where $\mathcal{Z}$ is the class comprising the generic node. The generator ΓZ is deterministic and consists simply of the instruction "output a node" (since $\mathcal{Z}$ is finite and in fact has only one element). The Boltzmann generator ΓB calls ΓZ (and halts) with probability $x/B(x)$ where $B(x)$ is the OGF of binary trees,

$$B(x) = \frac{1 - \sqrt{1 - 4x^2}}{2x}.$$

With the complementary probability corresponding to the strict binary case, it will make a call to ΓZ and two recursive calls to itself. In other words: *the Boltzmann generator for binary trees as constructed automatically from the composition rules produces a random sample of the branching process with probabilities* $(\frac{x}{B(x)}, \frac{xB(x)^2}{B(x)})$. Note that the generator is defined for $x < 1/2$ (the radius of convergence of $B(x)$), in which case the branching process is subcritical, so that the algorithm halts in expected finite time, as it should. Unbalanced 2-3 trees will be similarly produced by $\mathcal{U} = \mathcal{Z} + \mathcal{U}^2 + \mathcal{U}^3$, unary-binary trees by $\mathcal{V} = \mathcal{Z}(\mathbf{1} + \mathcal{V} + \mathcal{V}^2)$, etc. □

Example 3. *Secondary structures.* This example is inspired by Waterman *et al.*, themselves motivated by the problem of enumerating secondary RNA structures [15]. To fix ideas, consider rooted binary trees where edges contain 2 or 3 atoms and leaves ("loops") contain 4 or 5 atoms. A specification is $\mathcal{S} = (\mathcal{Z}^4 + \mathcal{Z}^5) + (\mathcal{Z}^2 + \mathcal{Z}^3)^2 \times (\mathcal{S} \times \mathcal{S})$. A Bernoulli switch will decide whether to halt or not, two independent recursive calls being made in case it is decided to continue, with the algorithm being sugared with suitable Bernoulli draws. The method is clearly universal for this entire class of problems. □

4. Exponential Boltzmann Generators

We consider here *labelled structures* in the precise technical sense of combinatorial theory. A labelled object of size n is then composed of n distinguishable atom, each bearing a distinctive label that is an integer in the interval $[1, n]$. Labelled combinatorial classes can be subjected to the *labelled product* defined as follows: if $\mathcal{A}$ and $\mathcal{B}$ are labelled classes, the product $\mathcal{C} = \mathcal{A} \star \mathcal{B}$ is obtained by forming all ordered pairs $\langle \alpha, \beta \rangle$ with $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{B}$ and relabelling them in all possible order-consistent ways. Straight from the definition, one has a binomial convolution $C_n = \sum_{k=0}^{n} \binom{n}{k} A_k B_{n-k}$, where the binomial takes care of relabellings. In terms of exponential generating functions, this becomes

$$\widehat{C}(z) = \widehat{A}(z) \cdot \widehat{B}(z).$$

Like in the ordinary case, we proceed by assembling Boltzmann generators for structured objects from simpler ones.

Disjoint union. The unlabelled construction carries over verbatim to this case.

Labelled product. The cartesian product construction adapts to this case with minor modifications: to produce an element from $\mathcal{C} = \mathcal{A} \star \mathcal{B}$, simply produce a pair by the cartesian product rule using values $\widehat{A}(x), \widehat{B}(x)$. (Complete by a randomly chosen relabelling if actual values of the labels are needed.)

Sequences. In the labelled universe, $\mathcal{C}$ is the sequence class of $\mathcal{A}$, written $\mathcal{C} = \mathfrak{S}(\mathcal{A})$ iff it is composed of all the sequences of elements from A up to order-consistent relabellings. Then, the EGF relation

$$\widehat{C}(x) = \sum_{k\geq 0} \widehat{A}(x)^k = \frac{1}{1-\widehat{A}(x)}$$

holds, and either of the two constructions of the generator ΓC from ΓA is applicable.

Sets. This is a new construction that we did not consider in the unlabelled case. The class $\mathcal{C}$ is the set-class of $\mathcal{A}$, written $\mathcal{C} = \mathfrak{P}(\mathcal{A})$ ($\mathfrak{P}$ is reminiscent of "powerset") if $\mathcal{C}$ is the quotient of sequences, $\mathcal{B} = \mathfrak{S}(\mathcal{A})/\equiv$, by the relation $\equiv$ that declares two sequences as equivalent if one derives from the other by an arbitrary permutation of the components. It is then easily seen that the EGFs are related by

$$\widehat{C}(x) = \sum_{k\geq 0} \frac{1}{k!}\widehat{A}(x)^k = e^{\widehat{A}(x)},$$

where the factor $1/k!$ "kills" the order present in sequences.

A moment of reflection shows that, under the exponential Boltzmann, the probability for a set in $\mathcal{C}$ to have k components in $\mathcal{A}$ is

$$\frac{1}{\widehat{C}(x)}\frac{1}{k!}\widehat{A}(x)^k = e^{-\widehat{A}(x)}\frac{\widehat{A}(x)^k}{k!},$$

that is, a Poisson law of rate $\widehat{A}(x)$. This gives rise to a simple algorithm for generating sets (analogous to the geometric algorithm for sequences):

```
function gC(x : real); {Generates C = Set(A)}
  let A(x) be the value of the EGF of A;
  draw K according to Poisson(A(x));
  return a K-tuple <gA(x),gA(x),...,gA(x)> {independent calls} end.
```

Cycles. This construction, written $\mathcal{C} = \mathfrak{C}(\mathcal{A})$, is defined like sets but with two sequences being identified if one is a cyclic shift of the other. The EGFs satisfy

$$\widehat{C}(x) = \sum_{k\geq 0} \frac{1}{k}\widehat{A}(x)^k = \log\frac{1}{1-\widehat{A}(x)}.$$

The log-law of rate $\lambda < 1$ (an "integral" of the geometric law!) is the law of X with

$$\mathbb{P}(X = k) = \frac{1}{\log(1-\lambda)^{-1}}\frac{\lambda^k}{k}.$$

Then cycles under the exponential Boltzmann model can be drawn like in the case of sets upon replacing the Poisson law by the log-law.

Theorem 2. *The analogue of Theorem 1 holds for labelled classes: labelled classes are correctly generated under the exponential Boltzmann model by the five rules above. The algorithms halt with probability 1, in finite expected time.*

Example 4. *Set partitions.* A set partition of size n is a partition of the integer interval $[1, n]$ into a certain number of nonempty classes, also called blocks, the blocks being by definition unordered between themselves. Let $\mathfrak{P}_{\geq 1}$ represent the powerset construction where the number of components is constrained to be ≥ 1. (This modified construction is easily subjected to random generation by using a truncated Poisson variable K, where K is conditioned to be ≥ 1.) The labelled class of all set partitions is then definable as $\mathcal{S} = \mathfrak{P}(\mathfrak{P}_{\geq 1}(\mathcal{Z}))$, where $\mathcal{Z}$ consists of a single labelled atom, $\mathcal{Z} = \{1\}$. Observe that the EGF of $\mathcal{S}$ is the well-known generating function of the Bell numbers, $\widehat{S}(x) = e^{e^x - 1}$. By the composition rules, one gets a random generator as follows: *Choose the number K of blocks as Poisson($e^x - 1$). Draw K independent copies $X_1, X_2, \ldots, X_K$ from the Poisson law of rate x, but each conditioned to be at least 1.* What this generates is in reality the "shape" of a set partition (the number of blocks (K) and the blocksizes (X_j)), with the "correct" distribution. To complete the task, it suffices to transport this structure on a random permutation of the integers between 1 and N, where $N = X_1 + \cdots + X_K$.

The process distinctly differs from the classical algorithm of Nijenhuis and Wilf [23] that requires tables of large integers. It is loosely related to a continuous model devised by Vershik [31] that can be interpreted as generating random set partitions based on $S(x) = \prod e^{x^j/j!}$, i.e., by ordered block lengths, as a potentially infinite sequence of Poisson variables of parameters $x/1!$, $x^2/2!$, etc. □

Example 5. *Random surjections (or ordered set partitions).* These may be defined as functions from $[1, n]$ to $[1, n]$ such that the image of f is an initial segment of $[1, n]$ (i.e., there are no "gaps"). One has for the class $\mathcal{Q}$ of surjections $\mathcal{Q} = \mathfrak{S}(\mathfrak{P}_{\geq 1}(\mathcal{Z}))$. Thus a random generator for $\mathcal{Q}$ first chooses a number of components given by a geometric law and then launches a number of Poisson generators. □

Set partitions find themselves attached to a compound (Poisson∘Poisson) process, whereas surjections are generated by a compound (Geometric∘Poisson) process (with suitable dependencies on parameters). This reflects the basic combinatorial opposition between freedom and order (for blocks).

Example 6. *Cycles in permutations.* This corresponds to $\mathcal{P} = \mathfrak{P}(\mathfrak{C}_{\geq 1}(\mathcal{Z}))$ and is obtained by a (Poisson∘Log) process. This example is loosely related to the Shepp–Lloyd model [27] that generates permutations by ordered cycle lengths. □

Example 7. *Assemblies of filaments in a liquid.* We may model these as sets of sequences, $\mathcal{F} = \mathfrak{P}(\mathfrak{S}_{\geq 1}(\mathcal{Z}))$. The EGF is $\exp\left(\frac{z}{1-z}\right)$. The random generation algorithm is a compound of the form (Poisson∘Geometric), with appropriate parameters. (See A000262 in Sloane's encyclopedia [28].) □

As these example show, one can *compile* automatically Boltzmann generators from formal specifications ("grammars") describing combinatorial models. As a matter of fact, we plan to develop a general-purpose implemention on top of Maple's Combstruct, as many functionalities are already available there.

5. Efficiency Considerations

First, a note about the efficiency of the random generation algorithms.

Proposition 2. *Given a well-founded specification of a combinatorial class $\mathcal{C}$, and given the finite collection of values of the intervening generating functions at a coherent value x, the Boltzmann generator $\Gamma C(x)$ has a complexity in the number of real-arithmetic operations that is linear in the size of its output object.*

Proof. Roughly speaking, given a specification Σ, each object admits a parse tree (or syntax tree) according to Σ. For well-founded specifications, this parse tree is of a size linear in the size of the object produced. It is readily recognized that the overhead in the schemes we have given is bounded from above by a linear function of the sum of all degrees of nodes in the parse tree. The result follows. □

Note. We even believe the mean *bit*-complexity to remain linear for a multitape Turing machine equipped with a source of random bits. (E.g., Knuth-Yao [17] show that one can build in this context a Bernoulli source of parameter $1/\pi$, say.)

5.1. **Controlling Sizes.** Recall that our primary goal was random generation of objects of some fixed size n, or at least in some target interval "close" to n. We use $\mathcal{C}$ like before to denote a class, but let $C(x)$ be its GF either ordinary or exponential depending on context and set $\rho = \rho_C$, the radius of convergence of $C(x)$.

Let $\nu_1(x)$ and $\nu_2(x)$ be the expressions for the moments of size, $\mathbb{E}(N)$ and $E(N^2)$, in terms of $C(x)$. Let also $\sigma(x) = \sqrt{\nu_2(x) - \nu_1(x)^2}$ be the corresponding standard deviation. If $C'(\rho^-) = \infty$, then the Boltzmann model attributes some nonnegligible weight to large structures as $x \to \rho$. We shall call the condition $C'(\rho^-) = +\infty$ the "*mean value condition*". If furthermore $\sigma(x)$ is of an order smaller than $\nu_1(x)$, then a window of small size (in relative terms) will be obtained. We shall call the condition $\sigma(x) = o(\nu_1(x))$, as $x \to \rho^-$, the "*variance condition*". (These two conditions obviously depend on the *singular type* of the generating function $C(x)$. This is an otherwise well researched topic as it is central to asymptotic counting [24].) We have, straight from Chebyshev's inequalities:

Proposition 3. *Let ε be a* fixed *relative tolerance on size. An object from $\mathcal{C}$ of size $N \in [n(1-\varepsilon), n(1+\varepsilon)]$ is to be generated uniformly at random in its size class. If the mean value and the variance conditions are satisfied, then, for any fixed tolerance $\varepsilon > 0$, the rejection method based on Boltzmann sampling succeeds with probability tending to 1 as $n \to \infty$.*

Proof. It suffices to invert the relation

$$n = x\frac{C'(x)}{C(x)}. \tag{5}$$

The solution $x = x_n$ is then employed in a Boltzmann sampler with parameter x. The output will be with high probability sufficiently close to n. ☐

Thus, for finite values of n, we expect under the conditions of Proposition 3 that only one or a few trials will be needed provided the tolerance on size is not too strict. For instance the variance condition *is* satisfied by set partitions ($x_n e^{x_n} = n$, so that $x_n \sim \log n - \log\log n$) and assemblies of filaments (for $x_n \sim 1 - n^{-1/2}$, size has mean n and standard deviation $\approx n^{3/4}$).

5.2. **Singular Boltzmann generators.** Sometimes the variance condition is not satisfied, but subtle phenomena regarding the "physics" of a class and the cost of the partial attempts still render approximate-size generation of linear time complexity. We discuss two *infinite* categories of models, where this interesting situation occurs and it is advantageous to place oneself right at the singularity $x = \rho_C$ (rather than at x_n given by (5)). Hence the name of "singular" generator.

Theorem 3. *Define a sequence construction to be supercritical if $\mathcal{C} = \mathfrak{S}(\mathcal{A})$ and $\rho_A > \rho_C$. Then an adapted singular Boltzmann generator for $\mathcal{C}$ produces a random $\mathcal{C}$ object of size $n + O(1)$ in one trial, with high probability.*

Proof. The notion and properties of supercriticality in this context are borrowed from Soria [29]. The adapted algorithm does simply as follows:

```
repeat: draw in A according to A(rho) until total_size >= n;
```

(Literally taken, the singular Boltzmann would loop for ever.) The isolated polar singularity of $C(x)$ at ρ entails the promised characteristics by virtue of common properties of meromorphic functions. $\square$

This theorem provides a global setting for a variety of *ad hoc* algorithms developed by Louchard [19, 20, 22] in the context of efficient generation of certain brands (directed, convex) of random planar diagrams known as "animals" and "polyominos". Here, it applies for instance to "words without long runs" and "surjections".

For what are structural complex-analytic reasons, the concentration property tends *not* to be satisfied by recursively defined objects. However, the balance of costs and probabilities is then such that, *whereas exact-size random generation is of quadratic complexity, approximate-size generation requires only linear time with "reasonable" constants.* For the statement below, what we name precisely a recursive class is the component $\mathcal{F}_1$ of a system of mutually dependent equations.

$$\{\mathcal{F}_1 = \Psi_1(\mathcal{Z}; \mathcal{F}_1, \ldots, \mathcal{F}_m), \ldots, \mathcal{F}_m = \Psi_m(\mathcal{Z}; \mathcal{F}_1, \ldots, \mathcal{F}_m)\}$$

where the Ψ's are *any* functional term involving *any* constructor defined previously ('$+$', '$\times$' or '$\star$', and $\mathfrak{S}, \mathfrak{P}, \mathfrak{C}$). The system is said to be irreducible if the dependency graph between the $\mathcal{F}_j$ is strongly connected (anybody depends on anybody else).

Theorem 4. *Consider a combinatorial class defined by a recursive specification that is irreducible. Then the expected cost of an "abortive" singular Boltzman sampler targetted at size n satisfies, for any bounded tolerance $\varepsilon = O(1)$,*

$$\overline{K}_\varepsilon(n) = O\left(\frac{n}{\varepsilon}\right), \qquad \textit{with} \qquad \overline{K}_0(n) \sim 4n^2, \quad \overline{K}_\varepsilon(n) \sim \frac{2}{\varepsilon}n, \tag{6}$$

where the last estimate holds for $\varepsilon = o(1)$ provided $\varepsilon = \Omega(\log n/n)$. In addition, the standard deviation of the cost is of the same order as the mean: $O(n/\varepsilon)$ for bounded ε, and $\sim 4n^2$ for exact-size generation.

Proof. (i) The algorithm: We use a singular Boltzmann generator with parameter ρ, the dominant singularity common to $F_1(z), \ldots, F_m(z)$. Rejections based on size are effected using "abortion" whenever size gets to exceed the allowed upper bound.

(ii) The analysis: the $F_j(x)$ are implicitly defined by an image system $\mathbf{F} = \Psi[\mathbf{F}]$. The Jacobian matrix of Ψ is at least defined near the origin and its spectral radius $\lambda(x)$ is analytic at 0 (by "well-foundedness"). This matrix is Perron-Frobenius by "irreducibility". By the analytic version of the implicit function theorem, ρ is determined by $\lambda(\rho) = 1$. A local analysis (essentially of the Drmota–Lalley–Woods type [5, 18, 32]) establishes that each F_j has a singularity at ρ that is of the square-root type in the complex plane. Thus the coefficients are asymptotically $\sim \lambda\rho^{-n}n^{-3/2}$; see [24]. The cost of Boltzmann sampling is then determined by the bivariate generating function (think of the regular event $(\mathcal{A} \setminus \mathcal{A}_n)^\star \cdot \mathcal{A}_n$)

$$K(x,u) = \left(1 - \frac{1}{A(x)}\left(A^{<n}(xu) + A^{>n}(x)u^n\right)\right)^{-1} \frac{A_n x^n}{A(x)},$$

on which basic asymptotic surgery can be performed at will. $\square$

The rejection algorithm is akin to the "Florentine algorithm" invented by Barcucci–Pinzani–Sprugnoli [1] to generate prefixes of Motzkin words and some directed plane animals. The cost analysis is related to Louchard's work [21]. The method applies to any family of trees, possibly "multitype", under the irreducibility condition. This covers binary trees, unbalanced 2–3 trees, secondary structures, and so on, where it is similar to drawing from a suitably dimensioned critical branching process.

5.3. **Accuracy of Boltzmann generators.** So far we have assumed an infinite precision model when drawing random variables given by exact values of generating functions. In practice, such values will only be known approximately by solving systems of equations numerically. This has however an effect which can be limited if one operates within computer algebra systems, since then arbitrary precision is available. One could use adaptive precision, interval arithmetics, or randomized rounding, as suggested by Denise and Zimmermann [3] in a related context.

Roughly put, say we tolerate a discrepancy of about $\delta = 10^{-16}$ from the uniform distribution under the Boltzmann model. Say we aim at sizes of at most $S_0 = 10^9$ for random generation. Since at most S_0 choices of values of random variables are to be effected, we consider a numerical accuracy $\delta/S_0 = 10^{-25}$ to be acceptable. Allowing for a wide margin of security, we contend that calculations to 50D combined with randomized rounding should suffice for all practical purposes. Thus, the exact arithmetic model assumed over the reals is not so unrealistic!

Example 8. *Coin fountains ($\mathcal{O}$).* These were enumerated by Odlyzko and Wilf [25]. They correspond to Dyck paths (aka Bernoulli excursions) taken according to area (disregarding length). The OGF is the continued fraction $O(z) = 1/(1 - z/(1 - z^2/(1 - z^3/(\cdots))))$. At top level, the singular Boltzmann sampler of Theorem 3 applies (write $\mathcal{O} = \mathfrak{S}(\mathcal{Q})$ and $O(z) = (1 - Q(z))^{-1}$). The root ρ of $Q(z) = 1$ is easily found to 50D: $\rho := 0.5761487691\cdots$. The objects of $\mathcal{Q}$ needed are with high probability of size at most $O(\log n)$, see [12], so that they can be generated by whichever subexponential method is convenient (e.g., Combstruct). The overall (theoretical and practical) complexity is $O(n)$ with *very* low implementation constants. Random generation well in the range of millions is now easy thanks to the singular Boltzmann generator. (Dutour et al. [7] employed the recursive method, but it is limited to sizes in the order of hundreds, perhaps thousands.) □

Continuations. We have work in progress on: generation of unlabelled multisets and cycles; generation of unlabelled powersets (no repetitions allowed); optimizations using the pointing operator; univariate compositions of GFs; the distribution of costs; multivariate extensions. The Boltzmann framework seems to have the flexibility needed for all such extensions to bring tangible gains.

References

[1] Barcucci, E., Pinzani, R., and Sprugnoli, R. The random generation of directed animals. *Theoretical Computer Science 127*, 2 (1994), 333–350.
[2] Dembo, A., Vershik, A., and Zeitouni, O. Large deviations for integer partitions. *Markov Processes and Related Fields 6*, 2 (2000), 147–179.
[3] Denise, A., and Zimmermann, P. Uniform random generation of decomposable structures using floating-point arithmetic. *Theoretical Computer Science 218*, 2 (1999), 233–248.
[4] Devroye, L. *Non-Uniform Random Variate Generation.* Springer Verlag, 1986.
[5] Drmota, M. Systems of functional equations. *Random Structures & Algorithms 10*, 1–2 (1997), 103–124.
[6] Duchon, P. Relaxed random generation of trees. *Algorithms Seminar*, 05-11-01, 2001.
[7] Dutour, I., and Fédou, J.-M. Object grammars and random generation. *Discrete Mathematics and Theoretical Computer Science 2* (1998), 47–61.
[8] Flajolet, P., Salvy, B., and Zimmermann, P. Automatic average–case analysis of algorithms. *Theoretical Computer Science 79*, 1 (Feb. 1991), 37–109.

[9] Flajolet, P., and Sedgewick, R. *Analytic Combinatorics.* 2001. Book in preparation: Individual chapters are available as INRIA Research Reports 1888, 2026, 2376, 2956, 3162, 4103 and electronically under http://algo.inria.fr/flajolet/Publications/books.html.

[10] Flajolet, P., Zimmerman, P., and Van Cutsem, B. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science 132*, 1-2 (1994), 1–35.

[11] Goulden, I. P., and Jackson, D. M. *Combinatorial Enumeration.* John Wiley, New York, 1983.

[12] Gourdon, X. Largest component in random combinatorial structures. *Discrete Mathematics 180*, 1-3 (1998), 185–209.

[13] Greene, D. H., and Knuth, D. E. *Mathematics for the analysis of algorithms.* Birkhäuser, Boston, 1981.

[14] Harary, F., and Palmer, E. M. *Graphical Enumeration.* Academic Press, 1973.

[15] Howell, J. A., Smith, T. F., and Waterman, M. S. Computation of generating functions for biological molecules. *SIAM Journal on Applied Mathematics 39*, 1 (1980), 119–133.

[16] Huang, K. *Statistical Mechanics*, 2nd ed. John Wiley, 1987.

[17] Knuth, D. E., and Yao, A. C. The complexity of nonuniform random number generation. In *Algorithms and complexity (Proc. Sympos., Carnegie-Mellon Univ., Pittsburgh, Pa., 1976).* Academic Press, New York, 1976, pp. 357–428.

[18] Lalley, S. P. Finite range random walk on free groups and homogeneous trees. *Ann. Probab. 21*, 4 (1993), 2087–2130.

[19] Louchard, G. Probabilistic analysis of some (un)directed animals. *Theoretical Computer Science 159*, 1 (1996), 65–79.

[20] Louchard, G. Probabilistic analysis of column-convex and directed diagonally-convex animals. *Random Structures & Algorithms 11*, 2 (1997), 151–178.

[21] Louchard, G. Asymptotic properties of some underdiagonal walks generation algorithms. *Theoretical Computer Science 218*, 2 (1999), 249–262.

[22] Louchard, G. Probabilistic analysis of column-convex and directed diagonally-convex animals. II. Trajectories and shapes. *Random Structures & Algorithms 15*, 1 (1999), 1–23.

[23] Nijenhuis, A., and Wilf, H. S. *Combinatorial Algorithms*, second ed. Academic Press, 1978.

[24] Odlyzko, A. M. Asymptotic enumeration methods. In *Handbook of Combinatorics*, R. Graham, M. Grötschel, and L. Lovász, Eds., vol. II. Elsevier, Amsterdam, 1995, pp. 1063–1229.

[25] Odlyzko, A. M., and Wilf, H. S. The editor's corner: n coins in a fountain. *American Matematical Monthly 95* (1988), 840–843.

[26] Schaeffer, G. Random sampling of large planar maps and convex polyhedra. In *Proceedings of the thirty-first annual ACM symposium on theory of computing (STOC'99)* (Atlanta, Georgia, May 1999), ACM press, pp. 760–769.

[27] Shepp, L. A., and Lloyd, S. P. Ordered cycle lengths in a random permutation. *Transactions of the American Mathematical Society 121* (1966), 340–357.

[28] Sloane, N. J. A. *The On-Line Encyclopedia of Integer Sequences.* 2000. Published electronically at `http://www.research.att.com/~njas/sequences/`.

[29] Soria-Cousineau, M. *Méthodes d'analyse pour les constructions combinatoires et les algorithmes.* Doctorat ès sciences, Université de Paris–Sud, Orsay, July 1990.

[30] Stanley, R. P. *Enumerative Combinatorics*, vol. II. Cambridge University Press, 1998.

[31] Vershik, A. M. Statistical mechanics of combinatorial partitions, and their limit configurations. *Funktsional'nyĭ Analiz i ego Prilozheniya 30*, 2 (1996), 19–39.

[32] Woods, A. R. Coloring rules for finite trees, and probabilities of monadic second order sentences. *Random Structures Algorithms 10*, 4 (1997), 453–485.

P. Duchon.: Université de Bordeaux I, 351 Cours de la Libération, F-33405 Talence Cedex, France; duchon@labri.u-bordeaux.fr

P. Flajolet: Algorithms Project, INRIA-Rocquencourt, F-78153 Le Chesnay, France; Philippe.Flajolet@inria.fr.

G. Louchard, Université Libre de Bruxelles, Département d'informatique, Boulevard du Triomphe, B-1050 Bruxelles, Belgique; louchard@ulb.ac.be

G. Schaefer: Polka Project, LORIA 615 rue du jardin Botanique, F-54000 Villers-les-Nancy , France. Gilles.Schaeffer@loria.fr