Random Sampling from Boltzmann Principles

Philippe Duchon[1], Philippe Flajolet[2], Guy Louchard[3], and Gilles Schaeffer[4]

[1] Université Bordeaux I, 351 Cours de la Libération, F-33405 Talence, France
[2] Algorithms Project, INRIA-Rocquencourt, F-78153 Le Chesnay, France
[3] Université Libre de Bruxelles, Département d'informatique, Boulevard du Triomphe, B-1050 Bruxelles, Belgium
[4] Adage Group, LORIA, F-54000 Villers-les-Nancy, France

Abstract. This extended abstract proposes a surprisingly simple framework for the random generation of combinatorial configurations based on *Boltzmann models*. Random generation of possibly complex structured objects is performed by placing an appropriate measure spread over the whole of a combinatorial class. The resulting algorithms can be implemented easily within a computer algebra system, be analysed mathematically with great precision, and, when suitably tuned, tend to be efficient in practice, as they often operate in linear time.

1 Introduction

In this text, *Boltzmann models* are proposed as a framework for the random generation of structured combinatorial configurations, like words, trees, permutations, constrained graphs, and so on. A Boltzmann model relative to a combinatorial class $\mathcal{C}$ depends on a control parameter $x > 0$ and places an appropriate measure that is spread over the whole of $\mathcal{C}$. Random objets under a Boltzmann model then have a fluctuating size, but objects with the same size invariably occur with the same probability. In particular, a *Boltzmann sampler* (i.e., a random generator that obeys a Boltzmann model), with the size of its output conditioned to be a fixed value n, draws *uniformly* at random an object of size n.

As we demonstrate in this article, Boltzmann samplers can be derived systematically (and simply) for classes that are specified in terms of a basic collection of general-purpose combinatorial constructions. These constructions are precisely the ones that surface recurrently in modern theories of combinatorial analysis; see, e.g., [2, 7, 9] and references therein. As a consequence, one obtains with surprising ease Boltzmann samplers covering an extremely wide range of combinatorial types.

Fixed-size generation is the standard paradigm in the random generation of combinatorial structures, and a vast literature exists on the subject. There, either specific bijections are exploited or general combinatorial decompositions are put to use in order to generate objects at random based on possibility counts—this has come to be known as the "recursive method" originating with Nijenhuis and Wilf [12] and formalized by Flajolet, Zimmermann, and Van Cutsem in [8]. In contrast, the basic principle of Boltzmann sampling is to *relax* the constraint of generating objects of a strictly fixed size, and prefer to draw objects with a randomly varying size. As we shall see, normally, one can *tune* the value of the control parameter x in order to favour objects of a size in the vicinity of a target value n. If needed, one can pile up a filter that rejects objects whose size is out of range. In this way, Boltzmann samplers may also serve for approximate-size as well as exact-size random generation.

We propose Boltzmann samplers as an attractive alternative to standard combinatorial generators based on the recursive method and implemented in packages like Combstruct (under the computer algebra system MAPLE) and CS (under MUPAD). Boltzmann algorithms are expected to be competitive when compared to many existing combinatorial methods: they only necessitate a small *fixed* number of multiprecision constants that are fairly easy to compute; when suitably optimized, they operate in low polynomial time—often even in linear time. Accordingly, uniform generation of objects with sizes in the range of millions is becoming a possibility, whenever the approach is applicable.

2 Boltzmann models and generators

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.*

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

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

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

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

The normalization coefficients are nothing but the counting generating functions of ordinary type (OGF) for $C(x) := \sum_n C_n x^n$ and exponential type (EGF) for $\widehat{C}(x) := \sum_n C_n x^n/n!$. 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* $e^{-\beta E}$. (There, β is an inverse temperature.) 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. 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. (This terminology is standard in combinatorial enumeration and graph theory [2,7,9].)

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.

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. By construction the probability of drawing an object of size n is $\mathbb{P}_x(N = n) = C_n x^n / C(x)$. Consequently, the probability generating function of N is $C(xz)/C(x)$ and the result follows.

In the next two sections (Sections 3 and 4), we develop a collection of rules by which one can assemble Boltzmann generators from simpler ones. The combinatorial classes considered are built on a small set of constructions that have a wide expressive power. The language in which classes are specified is in essence the same as the one underlying the recursive method [6,8]: it consists of the constructions of union, product, sequence, set, and cycle. For each allowable class, a Boltzmann sampler can be built in an entirely systematic manner.

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* $(\times)$, 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 $C(x) = A(x) + B(x)$. The Boltzmann model corresponding to $C(x)$ is then a mixture of the models associated to $A(x)$ and $B(x)$, with the probability of selecting a particular γ in $\mathcal{C}$ being $\mathbb{P}(\gamma \in \mathcal{A}) = A(x)/C(x)$, $\mathbb{P}(\gamma \in \mathcal{B}) = A(x)/C(x)$. Let us be given a generator for a Bernoulli variable $\mathrm{Bern}\,(p)$ defined as follows: $\mathrm{Bern}\,(p) = 1$ with probability p; $\mathrm{Bern}\,(p) = 0$ with probability $1 - p$; a sampler ΓC given ΓA and ΓB is simply obtained by

```
function $\Gamma C(x : \text{real})$; let $p_A := A(x)/(A(x) + B(x))$;
if $\mathrm{Bern}\,(p_A)$ then $\mathrm{return}(\Gamma A(x))$ else $\mathrm{return}(\Gamma B(x))$ fi.
```

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, one finds $C(x) = A(x) \cdot B(x)$. 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 $\Gamma C(x : \text{real})$; $\mathrm{return}(\langle \Gamma A(x), \Gamma B(x) \rangle)$ $\{\text{independent calls}\}$.
```

Sequences. Write $\mathcal{C} = \mathfrak{S}(\mathcal{A})$ if $\mathcal{C}$ is composed of all the finite sequences of elements of $\mathcal{A}$. The sequence class $\mathcal{C}$ is also the solution to the symbolic equation $\mathcal{C} = \mathbf{1} + \mathcal{A}\mathcal{C}$ (with $\mathbf{1}$ the empty sequence), which only involves unions and products. Consequently, one has $C(x) = (1 - A(x))^{-1}$. This gives rise to a recursive generator for sequences. Once recursion is unwound, the resulting generator assumes a particularly simple form:

```
function $\Gamma C(x : \text{real})$; let $A(x)$ be the value of the OGF of $\mathcal{A}$;
draw $K$ according to Geometric$(A(x))$;
return the $K$-tuple $\langle \Gamma A(x), \Gamma A(x), ..., \Gamma A(x) \rangle$ $\{\text{independent calls}\}$.
```

Finite sets. There finally remains to discuss initialization (when and how do we stop?). 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 explicitly by the definition of the Boltzmann model.

Proposition 2. *Define as* specifiable *an unlabelled class that can be specified (in a possibly recursive way) from finite sets by means of disjoint unions, cartesian products, and the sequence construction. Let C be an unlabelled specifiable class. 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. The set $\mathcal{W}$ of all words is expressible by a regular expression written in our notation $\mathcal{W} = \mathfrak{S}(b) \times \mathfrak{S}\,(a\mathfrak{S}(a)b\mathfrak{S}(b)) \times \mathfrak{S}(a)$. This expresses the fact that any word has a "core" formed with blocks of a's and blocks of b's in alternation that is bordered by a header of b's and a trailer of a's. The decomposition serves for $\mathcal{R}$: e.g., 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 $\mathcal{R}$ of the following form: two generators produce sequences of a's or b's according to a truncated geometric law; a generator for the product $\mathcal{C} := (\mathfrak{S}_{1\,..\,m}(a)\mathfrak{S}_{1\,..\,m}(b))$ is built according to the product rule; a generator for the "core" sequence $\mathcal{D} := \mathfrak{S}(\mathcal{C})$ is constructed according to the sequence rule. The generator assembled *automatically* from the general rules is then

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

Example 2. Trees (rooted, plane). Take first 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, $B(x) = (1 - \sqrt{1-4x^2})/(2x)$. 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 (subcritical) branching process with probabilities $x/B(x)$, $xB(x)^2/B(x)$.* Unbalanced 2-3 trees are similarly produced from $\mathcal{U} = \mathcal{Z} + \mathcal{U}^2 + \mathcal{U}^3$, unary-binary trees from $\mathcal{V} = \mathcal{Z}(\mathbf{1} + \mathcal{V} + \mathcal{V}^2)$, etc.

Example 3. Secondary structures. This example is inspired by the works of Waterman *et al.*, themselves motivated by the problem of enumerating secondary RNA structures. 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; see, e.g., [7]. 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. From the definition, a binomial convolution $C_n = \sum_{k=0}^{n} \binom{n}{k} A_k B_{n-k}$, 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.

Labelled product. The cartesian product construction adapts to this case: in order to produce an element from $\mathcal{C} = \mathcal{A} \star \mathcal{B}$, simply produce an independent pair by the cartesian product rule, but using the EGF values $\widehat{A}(x), \widehat{B}(x)$.

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 $\widehat{C}(x) = (1 - \widehat{A}(x))^{-1}$ holds, and the sequence construction of the generator ΓC from ΓA given in Section 3 and based on the geometric law 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 $\mathfrak{S}\{\mathcal{A}\}$ by the relation 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 $\widehat{C}(x) = \sum_{k \geq 0} \widehat{A}(x)^k / k! = e^{\widehat{A}(x)}$, where the factor $1/k!$ "kills" the order present in sequences. A moment of reflection shows that, under the exponential Boltzmann model, the probability for a set in $\mathcal{C}$ to have k components is $e^{-\widehat{A}(x)} \widehat{A}(x)^k / k!$, 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 $\Gamma C(x$ : real); let $\widehat{A}(x)$ be the value of the EGF of $\mathcal{A}$;
draw $K$ according to Poisson($\widehat{A}(x)$);
return the $K$-tuple $\langle \Gamma A(x), \Gamma A(x), ..., \Gamma A(x) \rangle$ {independent calls}.
```

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 $\widehat{C}(x) = \sum_{k \geq 0} \widehat{A}(x)^k / k = \log(1 - \widehat{A}(x))^{-1}$. The log-law (also known as "logarithmic series distribution") of rate $\lambda < 1$, is defined by $\mathbb{P}(X = k) = (-\log(1 - \lambda))^{-1} \lambda^k / k$. 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.

Proposition 3. *Define as* specifiable *a labelled class that can be specified (in a possibly recursive way) from finite sets by means of disjoint unions, cartesian products, as well as sequence, set and cycle constructions. Let $\mathcal{C}$ be a labelled specifiable class. Let x be a "coherent" parameter in $(0, \rho_{\widehat{C}})$, and let ΓC be the generator compiled from the definition of $\mathcal{C}$ by means of the five rules above. Then ΓC correctly draws elements from $\mathcal{C}$ according to the exponential Boltzmann model. It halts with probability 1 and 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. 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, $\mathcal{Z} = \{1\}$. 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, a random generator is 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, each conditioned to be at least 1.*

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 $K \in \text{Geom}\,(e^x-1)$ and then launches K Poisson generators.

Example 6. Cycles in permutations. This corresponds to $\mathcal{P} = \mathfrak{P}(\mathfrak{C}_{\geq 1}(\mathcal{Z}))$ and is obtained by a (Poisson$\circ$Log) process. (This example is loosely related to the Shepp–Lloyd model 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 $\exp(z/(1-z))$. The random generation algorithm is a compound of the form (Poisson$\circ$Geometric), with appropriate parameters. (See A000262 in Sloane's encyclopedia [14].)

5 The realization of Boltzmann samplers

In this section, we examine the way Boltzmann sampling can be implemented and sketch a discussion of complexity issues involved. In this abstract, only the *real-arithmetic model* ($\mathbb{R}$) is considered. There, what is assumed to be given is a random-access machine with unit cost for (exact) real arithmetic operations and elementary transcendental functions over the real numbers.

By definition, a Boltzmann sampler requires as input the value of the control parameter x that defines the Boltzmann model of use. As seen in previous sections, it also needs the finite collection of values at x of the generating functions that intervene in a specification. We assume these values to be provided by what we call the (generating function) "*oracle*". Such constants, which need only be precomputed *once*, are likely to be provided by a multiprecision package or a computer algebra system used as coroutine.

First one has to specify fully generators for the probabilistic laws Geom$\,(\lambda)$, Pois$\,(\lambda)$, Loga$\,(\lambda)$, as well as the Bernoulli generator Bern$\,(p)$, where the latter outputs 1 with probability p and 0 otherwise. A random generator 'uniform ()' produces at unit cost a random variable uniformly distributed over the real interval $(0,1)$.

Bernoulli generator. The Bernoulli generator is simply

```
Bern (p) := if uniform () ≤ p then return(1) else return(0) fi.
```

This generator serves in particular to draw from unions of classes.

Geometric, Poisson, and Logarithmic generators. For the remaining laws, we let p_k be the probability that a random variable with the desired distribution has value k, namely,

$$\text{Geom}\,(\lambda) :\ (1-\lambda)\lambda^k; \quad \text{Pois}\,(\lambda) :\ e^{-\lambda}\frac{\lambda^k}{k!}; \quad \text{Loga}\,(\lambda) :\ \frac{1}{\log(1-\lambda)^{-1}}\frac{\lambda^k}{k}.$$

The general scheme that goes well with real-arithmetic models is the *sequential algorithm*:

```
$U :=$ uniform (); $S := 0$; $k := 0$;
while $U < S$ do $S := S + p_k$; $k := k + 1$; od; return($k$).
```

This scheme is nothing but a straightforward implementation based on inversion of distribution functions (see [4, Sec. 2.1]). For the three distributions under consideration, the probabilities p_k can themselves be computed recurrently on the fly. In particular, under the model that has unit cost for real arithmetic operations and functions, the sequential generators have a useful property: *a variable with outcome k is drawn with a number of operations that is $O(k+1)$*. This has immediate consequences for all classes that are specifiable in the sense of Propositions 2 and 3.

Theorem 1. *Consider a specifiable class $\mathcal{C}$, either labelled or unlabelled. Assume as given an oracle that provides the finite collection of* exact *values of the intervening generating functions at a coherent value x. Then, 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.*

The linear complexity in the abstract model $\mathbb{R}$, as expressed in Theorem 1, provides an indication of the broad type of complexity behaviour one may aim for in practice, namely linear-time complexity. For instance, one may realize a Boltzmann sampler by truncating real numbers to some fixed precision, say using floating point numbers represented on 64 bits or 128 bits. The resulting samplers operate in time linear in the size of the output, though they may fail (by lack of digits in values of generating functions) in a small number of cases, and accordingly must deviate (slightly) from uniformity. Pragmatically, such samplers are likely to suffice for most medium-size simulations.

A sensitivity analysis of truncated Boltzmann samplers would be feasible, though rather heavy to carry out. One could even correct perfectly the lack of uniformity by appealing to an adaptive precision strategy based on guaranteed multiprecision floating point arithmetic. (The reader may get a feeling of the type of analysis involved by referring to the papers by Denise, Zimmermann, and Dutour, e.g., [3], where a thorough examination of the recursive method under this angle has been conducted.) In the full paper [5], we shall discuss bit-level implementations of Boltzmann samplers (see Knuth and Yao's insightful work [10] for context), as well as implementation issues raised by the oracle.

6 Exact-size and approximate-size sampling

Our primary objective in this article is the fast random generation of objects of some large size. Two types of constraints on size are considered. In *exact-size* random sampling, objects of $\mathcal{C}$ should be drawn uniformly at random from the subclass $\mathcal{C}_n$ of objects of size *exactly* n. In *approximate-size* random sampling, objects should be drawn with a size in an interval of the form $[n(1-\varepsilon), n(1+\varepsilon)]$, for some quantity $\varepsilon \geq 0$ called the (relative) *tolerance*, with two objects of the same size still being equally likely to occur. The conditions of exact and approximate-size sampling are immediately satisfied if one filters the output a Boltzmann generator by *rejecting* the elements that do not obey the desired size constraint. The main question is when and how can this rejection process be made reasonably efficient. The major conclusion from this and the next section is as follows: in many cases, including all the examples seen so far, *approximate-size sampling is achievable in linear time under the real-arithmetic model* of Theorem 1. The constants appear to be not too large if a "reasonable" tolerance on size is allowed.

The outcome of a basic Boltzmann sampler has a random size N whose distribution is exactly described by Proposition 1. First, for the rejection sampler tuned at the "natural" value $x = x_n$ such that $\mathbb{E}_{x_n}(N) = n$, a direct application of Chebyshev's inequalities gives:

Theorem 2. *Let $\mathcal{C}$ be a specifiable class and ε a* fixed nonzero *tolerance on size. Assume the following Mean Value and Variance Conditions,*

$$\lim_{x\to\rho^-} \mathbb{E}_x(N) = +\infty, \qquad \lim_{x\to\rho^-} \frac{\sqrt{\mathbb{E}_x(N^2) - \mathbb{E}_x(N)^2}}{\mathbb{E}_x(N)} = 0. \tag{2}$$

Then, the rejection sampler equipped with the value $x = x_n$ defined by the inversion relation $x_n C'(x_n)/C(x_n) = n$ succeeds in one trial with probability tending to 1 as $n \to \infty$. Its total cost is $O(n)$ on average.

The mean and variance conditions *are* satisfied by the class $\mathcal{S}$ of set partitions (Example 4) and the class $\mathcal{F}$ of assemblies of filaments (Example 7).

It is possible to discuss at a fair level of generality cases where rejection sampling is efficient, even though the strong moment conditions of Theorem 2 may not hold. The discussion is fundamentally based on the types of singularities that the generating functions exhibit. This is an otherwise well-researched topic as it is central to asymptotic enumeration [7, 13].

Theorem 3. *Let $\mathcal{C}$ be a combinatorial class that is specifiable. Assume that the generating function $C(z)$ (for $z \in \mathbb{C}$) has an isolated singularity at ρ, which is the unique dominant singularity. Assume also that the singular expansion of $C(z)$ at ρ is of the form (with P a polynomial)*

$$C(z) \underset{z\to\rho}{\sim} P(z) + c_0(1 - z/\rho)^{-\alpha} + o((1 - z/\rho)^{-\alpha}). \tag{3}$$

When the exponent $-\alpha$ is negative, for any fixed nonzero *tolerance ε, the rejection sampler corresponding to $x = x_n$ succeeds in an expected number of trials asymptotic to the constant*

$$\frac{1}{\xi_\alpha(\varepsilon)}, \quad \textit{where} \quad \xi_\alpha(\varepsilon) = \frac{\alpha^\alpha}{\Gamma(\alpha)} \int_{-\varepsilon}^{\varepsilon} (1+s)^{\alpha-1} e^{\alpha(1+s)}\, ds.$$

Moreover the total cost of this rejection sampler is $O(n)$ on average.

Words without long runs, surjections, and permutations (Examples 1, 5, and 6) have generating functions with a polar singularity, corresponding to the singular exponent -1, and hence satisfy the conditions above.

We note here that a condition $-\alpha < 0$ can often be ensured by successive differentiations of generating functions. Combinatorially, this corresponds to a *"pointing"* construction. Boltzmann sampling combined with pointing and rejection is developed in the full article [5] as a viable optimization technique.

7 Singular Boltzmann samplers.

We now discuss two infinite categories of models, where it is of advantage to place oneself right at the singularity $x = \rho_C$ in order to develop a rejection sampler from a Boltzmann model for $\mathcal{C}$. One category covers several of the sequence constructions, the other one corresponds to a wide set of recursive specifications.

Singular samplers for sequences. Define a sequence construction $\mathcal{C} = \mathfrak{S}(\mathcal{A})$ to be supercritical if $\rho_A > \rho_C$. The generating function of $\mathcal{C}$ and $\mathcal{A}$ satisfy $C(x) = (1 - A(x))^{-1}$, so that the supercriticality condition corresponds to $A(\rho_C) = 1$, with the (dominant) singularity ρ_C of $C(x)$ being necessarily a pole.

Theorem 4. *Consider a sequence construction $\mathcal{C} = \mathfrak{S}(\mathcal{A})$ that is supercritical. Generate objects from $\mathcal{A}$ sequentially according to $\Gamma A(\rho_C)$ until the total size becomes at least n. With probability tending to 1 as $n \to \infty$, this produces a random $\mathcal{C}$ object of size $n + O(1)$ in* one *trial. Exact-size random generation is achievable from this generator by rejection in expected time $O(n)$.*

This theorem applies to "cores" of words without long runs (from Example 1) and surjections (Example 5), for which exact-size generation become possible in linear time. It also provides a global setting for a variety of *ad hoc* algorithms developed by Louchard in the context of efficient generation of certain types (directed, convex) of random planar diagrams known as "animals" and "polyominos".

Example 8. Coin fountains ($\mathcal{O}$). These were enumerated by Odlyzko and Wilf. They correspond to Dyck paths taken according to area (disregarding length). The OGF is the continued fraction $O(z) = 1 \big/ (1 - z \big/ (1 - z^2 \big/ (1 - z^3 \big/ (\cdots))))$. At top level, the singular Boltzmann sampler of Theorem 4 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 high precision as $\rho = 0.5761487691\cdots$. The objects of $\mathcal{Q}$ needed are with high probability of size at most $O(\log n)$, so that they can be generated by whichever subexponential method is convenient. 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.

Singular samplers for recursive structures. What we call a recursive class $\mathcal{C}$ is the component $\mathcal{C} = \mathcal{F}_1$ of a system of mutually dependent equations:

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

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 (everybody depends on everybody else). In such a case, the singular type of the generating functions is a square-root, as follows from a mild generalization of a famous theorem by Drmota, Lalley, and Woods; see [7, Ch. 8] and references therein. A consequence is that coefficients of generating functions are of the universal form $\rho^{-n}n^{-3/2}$. In particular objects of a small size are likely to be produced by the singular generator $\Gamma C(\rho_C)$ whereas the expectation of size $\mathbb{E}_{\rho_C}(N)$ is infinite. (In other words, a very high dispersion of sizes is observed.) The singular sampler considered here simply uses the singular value $\rho = \rho_C$ together with an "early-abort" strategy: it aborts its execution as soon as the size of the partially generated object exceeds the tolerance upper bound. The process is repeated till an object within the tolerance bounds is obtained.

Theorem 5. *Let $\mathcal{C}$ be a combinatorial class given by a recursive specification that is irreducible and aperiodic. For any* fixed nonzero *tolerance ε, the "early-abort" rejection sampler succeeds in a number of trials that is $O(n^{1/2})$ on average. Furthermore, the total cost K_n of this sampler satisfies*

$$\mathbb{E}(K_n) \sim \frac{n}{\varepsilon}\left((1-\varepsilon)^{1/2} + (1+\varepsilon)^{1/2}\right). \tag{4}$$

For exact-size generation, the "early-abort" rejection sampler has complexity $O(n^2)$.

The early-abort sampler thus gives linear-time approximate-size random generation for all the simple varieties of trees of Example 2 (including binary trees, unary-binary trees, 2–3 trees, and so on) and for secondary structures (Example 3). For all these cases, exact-size is also achievable in quadratic time. The method is roughly comparable to drawing from a suitably dimensioned critical branching process in combination with abortion and rejection.

The rejection algorithm above is akin to the "Florentine algorithm" invented by Barcucci–Pinzani–Sprugnoli [1] to generate prefixes of Motzkin words and certain directed plane animals. The cost analysis is related to Louchard's work [11].

8 Conclusions

As shown here, combinatorial decompositions allow for random generation in low polynomial time. In particular, approximate-size random generation is often of a linear time complexity. Given the large number of combinatorial decompositions that have been gathered over the past two decades (see, e.g., [2, 7, 9], we estimate to perhaps a hundred the number of classical combinatorial structures that are amenable to efficient Boltzmann sampling. In contrast with the recursive method [3, 8, 12], memory requirements are kept to a minimum since only a table of constants of size $O(1)$ is required.

In forthcoming works starting with [5], we propose to demonstrate the versatility of Boltzmann sampling including: the generation of unlabelled multisets and powersets, the encapsulation of constructions like substitution and pointing, and the realization of Boltzmann samplers at bit-level. (Linear boolean complexity seems to be achievable in many cases of practical interest.)

Acknowledgements: The authors are grateful to Brigitte Vallée for several architectural comments on an early version of this manuscript. Thanks also to Bernard Ycart, Jim Fill, Marni Mishna, and Paul Zimmermann for encouragements and constructive observations. This work was supported in part by the ALCOM-FT Project IST-1999-14186 of the European Union.

References

1. Barcucci, E., Pinzani, R., and Sprugnoli, R. The random generation of directed animals. *Theoretical Computer Science 127*, 2 (1994), 333–350.
2. Bergeron, F., Labelle, G., and Leroux, P. *Combinatorial species and tree-like structures.* Cambridge University Press, Cambridge, 1998. Translated from the 1994 French original by Margaret Readdy, With a foreword by Gian-Carlo Rota.
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. Duchon, P., Flajolet, P., Louchard, G., and Schaeffer, G. Boltzmann samplers for random combinatorial generation. In preparation, 2002.
6. Flajolet, P., Salvy, B., and Zimmermann, P. Automatic average–case analysis of algorithms. *Theoretical Computer Science 79*, 1 (Feb. 1991), 37–109.
7. 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.
8. Flajolet, P., Zimmermann, P., and Van Cutsem, B. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science 132*, 1-2 (1994), 1–35.
9. Goulden, I. P., and Jackson, D. M. *Combinatorial Enumeration.* John Wiley, New York, 1983.
10. 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.
11. Louchard, G. Asymptotic properties of some underdiagonal walks generation algorithms. *Theoretical Computer Science 218*, 2 (1999), 249–262.
12. Nijenhuis, A., and Wilf, H. S. *Combinatorial Algorithms*, second ed. Academic Press, 1978.
13. 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.
14. Sloane, N. J. A. *The On-Line Encyclopedia of Integer Sequences.* 2000. Published electronically at `http://www.research.att.com/~njas/sequences/`.