Boltzmann samplers and beyond

KEYWORDS: analytic combinatorics, trees, dynamic programming, recursive method, Boltzmann sampler, uniform distribution

1 Warm-up: binary trees and Catalan numbers

Let us gently start by recalling

- What is a rooted tree and a binary rooted tree?
- What is a generating function?
- How many binary trees does there exist with given number of nodes? (only on the level of recurrences, not the exact number)

From graph theory, we are familiar with the concept of tree, which is a graph without cycles. The tree is called **rooted** if one of the vertices is distinguished. This distinguished vertex is called a **root**.
The height of a node is the distance from this node to the root. If two nodes are adjacent then the node with greater height is called a **child**, and the other one is called a **parent**. Binary trees are those trees whose nodes have either zero or two children.
Then, binary trees admit a kind of recursive definition which defines binary trees in terms of themselves:

Definition 1. Every binary rooted tree is either a **leaf** or a **root** with two **binary trees** attached.

This recursive-type definition can be given as well to ordinary rooted trees, with arbitrary number of children:

Definition 2. A rooted tree is a **root** and a sequence (of possibly zero length) of **rooted trees**.

Binary trees can be also represented by an unambiguous context-free grammar (the same as given for correct bracket expressions):

$$B = \mathcal{Z} \mid (B)\mathcal{Z}(B)$$

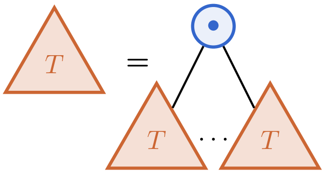


Figure 1: Combinatorial definition of a rooted tree

The symbol $\mathcal{Z}$ is a terminal symbol of this grammar (which corresponds to a leaf vertex), and the brackets «(», «)» are auxilliary symbols. Thus, this grammar describes the set of words

$$\{\mathcal{Z}, (\mathcal{Z})\mathcal{Z}(\mathcal{Z}), ((\mathcal{Z})\mathcal{Z}(\mathcal{Z}))\mathcal{Z}(\mathcal{Z}), (\mathcal{Z})\mathcal{Z}((\mathcal{Z})\mathcal{Z}(\mathcal{Z})), \dots\}$$

We shall say that a tree has **size** n if it has n vertices. In the sequel, we will be interested in the problem of random generation of combinatorial objects like binary trees (maybe a bit more complicated than just binary trees). If we fix the size of a tree n, the problem is to pick uniformly at random among the trees of given size.

Problem 1. Give an algorithm which generates uniformly at random binary trees with n nodes.

In the long run, the problem of random generation will become a bit more general than this, but we need to start with a simple example. Essentially, we consider two algorithms which solve this problem.

1.1 The recursive method

The recursive method is a kind of algorithm which uses dynamic programming, first described in [NW78]. In order to generate a tree of size n, we choose a number $k \in \{0, \dots, n\}$ at random, and generate left subtree with k nodes and right subtree with $n-k-1$ nodes. But what should be the distribution of random variable k in order to obtain the resulting tree uniformly at random?
In order to give the answer, let's turn to combinatorics. Suppose that the number of binary trees with k nodes is equal to T_k and has been precomputed for all $k \leq n$. How to compute the number of trees of size n if all the numbers T_k are known for $k < n$? After summing over all possible k, we

obtain:

$$T_n = \sum_{k=1}^{n} T_k T_{n-k-1} \quad .$$

The probability of having k vertices in the left subtree should be proportional to k-th summand in the previous sum:

$$\mathbb{P}(k \text{ nodes in the left subtree}) = \frac{T_k T_{n-k-1}}{T_n}$$

Once all the values $T_1, \ldots, T_{n-1}$ are precomputed, it is possible to recursively generate each required probability distribution on the number of nodes in the left subtree. The tree is generated from the root until each «subprocess» becomes a leaf node.

Algorithm 1: Recursive algorithm for binary trees

```
Data: Integer number $n$
Result: Random tree of size $n$
begin
    Precompute array $(T_k)_{k=1}^{n}$ using the reccurence
        $$T_n = \sum_{k=1}^{n} T_k T_{n-k-1} \quad , \quad T_1 = 1$$
    For each $n$, precompute the probability distribution $\mathcal{P}_n$:
        $$p_k^{(n)} = \frac{T_k T_{n-k-1}}{T_n}$$
    Function Generate($n$):
        if $n = 1$ then
            return $\mathcal{Z}$ ;
        Sample random $k$ from the probability distribution $\mathcal{P}_n$ ;
        $L :=$ Generate($k$) ;
        $R :=$ Generate($n - k - 1$) ;
        return $(L)\mathcal{Z}(R)$;
```

Here are several trees of size 9 generated by **Algorithm 1**:

```
(((Z)Z(Z))Z(Z))Z((Z)Z(Z))
(Z)Z((((Z)Z(Z))Z(Z))Z(Z))
(Z)Z(((Z)Z(Z))Z((Z)Z(Z)))
(((Z)Z((Z)Z(Z)))Z(Z))Z(Z)
(Z)Z(((Z)Z((Z)Z(Z)))Z(Z))
```

Remark 1. We use quadratic algorithm to precompute the values T_k. In fact, this can be done in linear time using a recursive formula (exercise)

$$T_{2n+1} = \frac{2(2n-1)}{n+1} T_{2n-1}$$

Moreover, for any combinatorial system corresponding to **unambiguous context-free grammar** this can be done in linear arithmetic time (using so-called **holonomic specifications**). Keywords: "every algebraic function is holonomic".

1.2 Boltzmann sampler

Arnold's principle. Every principle doesn't carry the name of its inventor. Arnold's principle is applicable to itself.
According to Arnold's principle, Boltzmann sampler was not invented by Boltzmann. It was introduced in the paper of Duchon, Flajolet, Louchard and Schaeffer.
In contrast to the previous sampler, Boltzmann sampler doesn't return objects of fixed size, the size is a random variable. Hovewer, the sampler has an additional parameter as an input, and this parameter can be changed to give different expected values of size.
At this point we need to define **generating function** of binary trees. Generating function is a formal power series that contains information about all the coefficients T_k in the following way:

$$T(z) := T_0 + T_1 z + T_2 z^2 + \dots$$

This function can be even computed at some points z, for example (exercise!):

$$T(\tfrac{1}{2}) = 1.$$

Pure magic and beauty.
There is a famous formula for Catalan numbers.

$$T(z) = \frac{1 - \sqrt{1 - 4z^2}}{2z}$$

which can be proved by solving quadratic equation with respect to T

$$T(z) = z + zT^2(z)$$

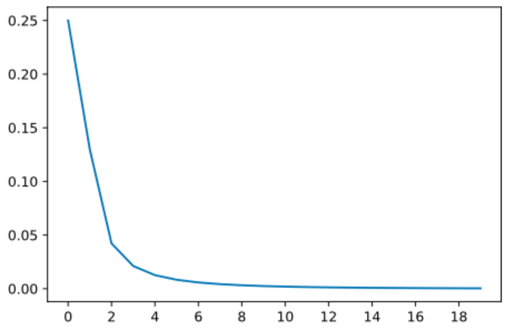
which, in its turn, follows from the reccurence relation on its coefficients

$$T_n = \sum_{k=1}^{n-1} T_k T_{n-k-1}.$$

Algorithm 2: Boltzmann sampler algorithm for binary trees

```
Data: Integer number $n$, real value $z$
Result: Random tree of variable size, target expected size $n$
begin
    Function Generate($n$):
        Carefully look at the equation
            $T(z) = z + zT^2(z)$
        Generate Bernoulli random variable $X$
            $\mathbb{P}(X = 0) = \frac{z}{z + zT^2(z)}$
            $\mathbb{P}(X = 1) = \frac{zT^2(z)}{z + zT^2(z)}$
        if $X = 0$ then
            return $\mathcal{Z}$;
        if $X = 1$ then
            $L$ := Generate($z$);
            $R$ := Generate($z$);
            return $(L)\mathcal{Z}(R)$;
```

Let us take a look at the distribution of the size of generated tree



Here are several trees generated by **Algorithm 2** for different values of z. For $z = 0.45$:

```
Z
Z
(Z)Z(((Z)Z(Z))Z(Z))
(Z)Z(Z)
Z
Z
Z
Z
(Z)Z(Z)
((Z)Z(Z))Z(Z)
```

For $z = 0.49$:

```
Z
((Z)Z(Z))Z((((Z)Z((Z)Z(((Z)Z(Z))Z(Z))))Z(Z))Z(Z))
Z
Z
Z
(Z)Z((Z)Z(Z))
Z
```

For $z = 0.5$:

```
Z
Z
(Z)Z(Z)
Z
```

```
(Z)Z(Z)
(((((Z)Z(Z))Z((Z)Z(Z)))Z(((Z)Z(Z))Z(((Z)Z((Z)Z(((Z)Z((((Z)Z(((((Z)
Z(Z))Z(Z))Z((Z)Z(((((Z)Z(Z))Z(Z))Z((Z)Z(Z)))Z(Z))))Z(Z)))Z(Z))Z((Z
)Z(((Z)Z((Z)Z((Z)Z((((((((Z)Z((((Z)Z(Z))Z(Z))Z(Z)))Z((((((Z)Z(Z))Z
(Z))Z((Z)Z(Z)))Z(Z))Z(Z)))Z(Z))Z(Z))Z(((((Z)Z(Z))Z(Z))Z((Z)Z(Z)))Z
(Z)))Z(((Z)Z(Z))Z(Z)))Z(Z)))))Z((Z)Z(((Z)Z(((Z)Z(Z))Z((Z)Z(Z))))Z(
(Z)Z(Z))))))))Z((((Z)Z((Z)Z(Z)))Z(Z))Z(Z)))))Z((Z)Z(Z)))))Z((Z)Z((
(Z)Z((Z)Z((((((Z)Z(Z))Z(Z))Z((Z)Z(((((Z)Z(((Z)Z((Z)Z((Z)Z((((((Z)Z
(Z))Z((((Z)Z((Z)Z(Z)))Z(Z)))Z(((Z)Z((Z)Z(Z)))Z((((((Z)Z(Z))Z(Z))Z((
Z)Z(Z)))Z(Z))Z((Z)Z(Z)))))Z((Z)Z(Z)))Z(((((Z)Z(Z))Z(Z))Z(Z))Z(Z)))
)))Z(Z)))Z((((Z)Z(Z))Z(Z))Z(((Z)Z(Z))Z(Z))))Z(Z))Z((((Z)Z(Z))Z(Z))
Z((Z)Z(((((Z)Z(Z))Z((Z)Z(Z)))Z(Z))Z(Z)))))))Z((((Z)Z(Z))Z(Z))Z(Z))
)Z(((Z)Z(Z))Z((Z)Z(((Z)Z(Z))Z(Z)))))))Z((Z)Z(Z)))))Z(((((((Z)Z(Z))
Z((Z)Z(Z)))Z(Z))Z(Z))Z(((Z)Z(Z))Z(Z)))Z((((Z)Z(Z))Z((Z)Z(((Z)Z((Z)
Z(((((((Z)Z((Z)Z((Z)Z(Z))))Z(((Z)Z(((Z)Z(((Z)Z(Z))Z((((Z)Z((Z)Z(Z)
))Z(Z))Z(Z))))Z(Z)))Z(Z)))Z(Z))Z(Z))Z(Z))Z((((((((Z)Z((((Z)Z(Z))Z(
Z))Z(((((Z)Z((((((Z)Z((Z)Z((((Z)Z((((Z)Z(Z))Z(Z))Z(Z)))Z(Z))Z(((Z)
Z((Z)Z(Z)))Z(((Z)Z(Z))Z(Z))))))Z(((((Z)Z(Z))Z(((Z)Z(Z))Z(Z)))Z((Z)
Z((((Z)Z(Z))Z(Z))Z((Z)Z((Z)Z(((Z)Z(Z))Z(Z)))))))Z(Z)))Z((((Z)Z(Z))
Z((((Z)Z(Z))Z(Z)))Z(Z)))Z(Z))Z(Z)))Z(Z))Z(Z))Z(Z))))Z(Z))Z(Z))Z(Z))
Z(Z))Z(Z))Z((Z)Z((((Z)Z(Z))Z(((Z)Z((Z)Z(Z)))Z((((Z)Z(Z))Z(Z))Z(Z))
))Z((Z)Z(Z))))))))Z((((((Z)Z((((Z)Z((Z)Z((Z)Z(Z))))Z(Z))Z((Z)Z((((
Z)Z(((Z)Z((Z)Z(Z)))Z((Z)Z(Z))))Z((Z)Z(Z)))Z(Z)))))Z((Z)Z(Z)))Z((((
((Z)Z(Z))Z((Z)Z(((Z)Z(Z))Z((Z)Z((((Z)Z(Z))Z(Z))Z(Z)))))))Z(Z))Z(Z))
Z((Z)Z(Z))))Z(Z))Z(Z)))))Z(((Z)Z(Z))Z((Z)Z(Z)))))
```

The strange thing happening is that the size distribution has a heavy tail, so sometimes we obtain objects of very large size. The size of the generated object is not well concentrated.

Exercise 1. Show that conditioned on size, the distribution of the objects inside a class is uniform.

Exercise 2. Show that the expected size of the generated object from a Boltzmann sampler is equal to $z\dfrac{T'(z)}{T(z)}$. Show that this function is non-decreasing in argument z.

Hint. Show that under ordinary Boltzmann model, the probability generating function of the size of generated objects is

$$\sum_n \mathbb{P}_x(N = n) z^n = \frac{F(xz)}{F(x)}.$$

Show the monotonicity by proving the identity

$$x\frac{d}{dx}\mathbb{E}_x(N) = \mathbb{V}_x(N).$$

Exercise 3. Show that the variance of the size of the generated binary tree is infinite.

1.3 Generating rooted trees

After having generated binary trees, we proceed to the trees without fixed number of children, i.e. rooted plane trees. Let us recall the combinatorial definition of a rooted plane tree:

Definition 3. A rooted plane tree is a **root** and a **sequence** (possibly empty) of sub-trees, each is also a rooted plane tree.

Suppose that $T(z) = T_0 + T_1 z + T_2 z^2 + \ldots$ is a generating function for trees, i.e. T_n is equal to the number of trees of size n. Then, this generating function satisfies functional equation

$$T(z) = z \cdot (1 + T(z) + T(z)^2 + \ldots) = z \cdot \frac{1}{1 - T(z)}$$

There are two different ways to sample random trees using Boltzmann sampler. These two ways differ in computational complexity (more precisely, in the number of random bits required), but result in the same distribution. One approach is to treat the infinite sum analogous to that of binary trees, and sample a random variable X with distribution

$$\mathbb{P}(X = k) = \frac{T(z)^k}{1 + T(z) + T^2(z) + \ldots}$$

This random variable is responsible for the number of children of the root node. The generation proceeds recursively into each of these k children then. This can be viewed in **Algorithm 3**.
A different option would be to define the sequence of trees as a separate combinatorial class recursively:

$$S := \frac{1}{1 - T(z)}, \quad S(z) = 1 + S(z) \cdot T(z)$$

and define a Boltzmann generator separately for S and for T recursively in terms of each other.

Exercise 4. Extend the Boltzmann sampling framework onto labelled structures, i.e. those having exponential generating functions as describing them. Show that the SET and CYC operators can be constructed as Poisson and another specific discrete distribution with p.g.f. proportional to $\frac{C^k}{k}$, respectively.

Algorithm 3: Boltzmann sampler for rooted plane trees

```
Data: Integer number $n$, real value $z$
Result: Random tree of variable size, target expected size $n$
begin
    Function Generate($n$):
        Carefully look at the equation
            $$T(z) = z \cdot (1 + T(z) + T(z)^2 + \ldots)$$
        Generate random variable $X$ from geometric distribution
          with parameter $T(z)$:
            $$\mathbb{P}(X = k) = T(z)^k (1 - T(z))$$
        for $i = 1$ to $k$ do
            $T_i :=$ Generate($z$) ;
        return $\mathcal{Z}(T_1 T_2 \ldots T_k)$ ;
```

2 Different combinatorial classes

According to grammar types, so-defined combinatorial classes that we consider (binary trees) can be replaced by more sophisticated objects, according to the nature of its generating function. It may satisfy a system of equations, which themselves can be of different complexity, like rational, algebraic and differential systems.
Also, in the previous part we have tuned the expected size of the object, but we can have control over some additional parameters like number of leaves of the generated tree.

2.1 Tree with given number of leaves

Exercise 5. Show that the expected number of leaves in a random tree is $n/8$.

Exercise 6. Show that the expected number of leaves is equal to

$$\mathbb{E}_{z,u} \#_{\text{of}} \text{ leaves} = u \frac{\partial_u F(z,u)}{F(z,u)}$$

Functional equation	**Exact sampling**	**In expectation**
Rational	$O(n)$ [BG12]	$O(n)$
Algebraic	$O(n^2)$ [DZ99]	$O(n)$
Unlabeled tree-like	$O(n^2)$	$O(n)$
First-order differential		$O(n)$
Second-order differential		$O(n)$
Dirichlet samplers		$O(n)$
Non-analytic ODE		doesn't exist
FE with substitutions		doesn't exist
Multiparametric algebraic	$O^*(n^2)$	$O^*(n)$

Table 1: Possible situations for Boltzmann sampling

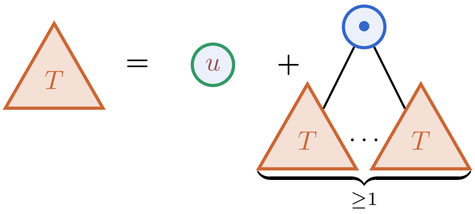


Figure 2: Trees with labeled leaves.

2.2 Multiparametric sampling

Exercise 7. Exact multiparametric sampling from algebraic grammars is NP-complete.

Hint. Consider an NP-complete problem 1-in-3 SAT which is a version of 3-SAT with a requirement that in each clause, exactly one of the variables is satisfied. Show that given an instance of 1-in-3 SAT problem, it is possible to construct a corresponding algebraic grammar, such that it is possible to sample at least one object from this grammar if and only if the formula is satisfiable.

Exercise 8. Exact multiparametric sampling from algebraic grammars is #P-complete.

However, if the goal is to have the *expected* values of given parameters, it is possible to have a polynomial algorithm. In our recent paper [BBD18], we construct a polynomial-time oracle for a problem of multidimensional tuning. The previous algorithm [BP10] only guaranteed convergence if the starting vector was close enough to the target solution.

2.3 Rational and algebraic specifications, Pólya structures

The basic case of combinatorial families whose generating functions satisfy the functional equation

$$\boldsymbol{F} = \boldsymbol{\Phi}(\boldsymbol{F}, z)$$

is covered in the original paper [DFLS04] and the corresponding oracle is constructed in [PSS12] (in a uniparametric setting). More on Pólya structures (also known as unlabelled structures) can be found in [FFP07, BFKV11].

2.4 First-order and second-order differential specifications

The cases when the differential function is analytic and satisfies a first-order differential equation (oracle provided) is settled in [BRS12]. This work was continued to obtain second-order differential samplers [BDF$^+$16], with some applications to concurrent processes.

2.5 Differential equations with zero radius of convergence and functional equations with substitutions

Some differential equations have solutions which are not analytic at zero. The simplest possible example is the functional equation

$$z^2 f'(z) + (z-1)f(z) = 0$$

which gives a solution

$$f(z) = \sum_{n \geq 0} n! z^n.$$

For such examples it is not yet known how to construct Boltzmann-style samplers. Also certain equations have much more complicated structure, for example the generating function for lambda terms in unary notation [BGGJ13]:

$$L(z) = zM(z) + zL(z)^2 + zL\left(\frac{z}{1-2zM(z)}\right)$$

where $M(z)$ is the generating function for Motzkin numbers.

It is remarkable that for the case of Motzkin trees it turned out to be possible to interpret combinatorially the corresponding holonomic specification to create a linear-time sampling algorithm [BBJ13]. The corresponding generating function is analytic at zero.

2.6 Dirichlet samplers

Boltzmann samplers for Dirichlet generating functions are discussed in [Bod10].

References

[BBD18] Maciej Bendkowski, Olivier Bodini, and Sergey Dovgal. Polynomial tuning of multiparametric combinatorial samplers. In *2018 Proceedings of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 92–106. SIAM, 2018.

[BBJ13] Axel Bacher, Olivier Bodini, and Alice Jacquot. Exact-size sampling for motzkin trees in linear time via boltzmann samplers and holonomic specification. In *Proceedings of the Meeting on Analytic Algorithmics and Combinatorics*, pages 52–61, 2013.

[BDF$^+$16] Olivier Bodini, Matthieu Dien, Xavier Fontaine, Antoine Genitrini, and Hsien-Kuei Hwang. Increasing diamonds. In *Latin American Symposium on Theoretical Informatics*, pages 207–219, 2016.

[BFKV11] Manuel Bodirsky, Éric Fusy, Mihyun Kang, and Stefan Vigerske. Boltzmann samplers, pólya theory, and cycle pointing. *SIAM J. Comp.*, 40(3):721–769, 2011.

[BG12] Olivier Bernardi and Omer Giménez. A linear algorithm for the random sampling from regular languages. *Algorithmica*, 62(1):130–145, 2012.

[BGGJ13] Olivier Bodini, Danièle Gardy, Bernhard Gittenberger, and Alice Jacquot. Enumeration of generalized *bci* lambda-terms. *arXiv preprint arXiv:1305.0640*, 2013.

[Bod10] Olivier Bodini. *Autour de la génération aléatoire sous modèle de Boltzmann [On random generation under Boltzmann models]*. habilitation, Université Pierre et Marie Curie, Paris, 2010.

[BP10] Olivier Bodini and Yann Ponty. Multi-dimensional boltzmann sampling of context-free languages. In *21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'10)*, volume AM, 2010.

[BRS12] Olivier Bodini, Olivier Roussel, and Michèle Soria. Boltzmann samplers for first-order differential specifications. *Disc. App. Math.*, 160(18):2563–2572, 2012.

[DFLS04] Philippe Duchon, Philippe Flajolet, Guy Louchard, and Gilles Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, Probability & Computing*, 13(4-5):577–625, 2004.

[DZ99] Alain Denise and Paul Zimmermann. Uniform random generation of decomposable structures using floating-point arithmetic. *Theoretical Computer Science*, 218(2):233–248, 1999.

[FFP07] Philippe Flajolet, Éric Fusy, and Carine Pivoteau. Boltzmann sampling of unlabelled structures. In *Proceedings of the Meeting on Analytic Algorithmics and Combinatorics*, pages 201–211, 2007.

[NW78] Albert Nijenhuis and Herbert S. Wilf. *Combinatorial Algorithms.* Academic Press, 2 edition, 1978.

[PSS12] Carine Pivoteau, Bruno Salvy, and Michèle Soria. Algorithms for combinatorial structures: well-founded systems and newton iterations. *J. Comb. Th.*, 119(8):1711–1773, 2012.