



Exact-size Sampling for Motzkin Trees in Linear Time via Boltzmann Samplers and Holonomic Specification

Axel Bacher, Olivier Bodini, Alice Jacquot

► To cite this version:

Axel Bacher, Olivier Bodini, Alice Jacquot. Exact-size Sampling for Motzkin Trees in Linear Time via Boltzmann Samplers and Holonomic Specification. ANALCO 2013, Jan 2013, New Orleans, United States. pp.Pages 52–61. hal-00989647

HAL Id: hal-00989647

<https://hal.science/hal-00989647>

Submitted on 12 May 2014

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

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

Exact-size Sampling for Motzkin Trees in Linear Time via Boltzmann Samplers and Holonomic Specification

A. Bacher ^{*} O. Bodini [†] A. Jacquot [†]

Abstract

Boltzmann samplers are a kind of random samplers; in 2004, Duchon, Flajolet, Louchard and Schaeffer showed that given a combinatorial class and a combinatorial specification for that class, one can automatically build a Boltzmann sampler. In this paper, we introduce a Boltzmann sampler for Motzkin trees built from a holonomic specification, that is, a specification that uses the pointing operator. This sampler is inspired by Rémy's algorithm on binary trees. We show that our algorithm gives an exact size sampler with a linear time and space complexity in average.

Introduction

Trees are certainly among the most classical objects of computer science and also a central object of study in analytic combinatorics. In this paper, we are interested in the difficult question of the efficient random generation of trees.

The class of binary planar trees, or Catalan trees, is perhaps the simplest class of trees. The number of Catalan trees with $2n + 1$ nodes is the n -th Catalan number, that we denote by C_n . Rémy's algorithm [11, 10] gives an efficient way to uniformly sample a binary planar tree (Catalan tree) of a given size. It is based on the following recurrence on the Catalan numbers:

$$(n + 1)C_n = 2(2n - 1)C_{n-1},$$

with initial condition $C_0 = 1$. Let $C(z) = \sum_{n \geq 0} C_n z^{2n+1}$ be the generating function of Catalan trees counted according to their total number of nodes. The above recurrence translates to the holonomic equation:

$$(0.1) \quad C(z) + C^\bullet(z) = 2z + 4z^2 C^\bullet(z),$$

where $C^\bullet(z) = zC'(z)$ is the generating function of Catalan trees pointed on a node. While Rémy's algorithm is quite simple to implement and of linear complexity for drawing a tree of size n , this method seems ad-hoc to Catalan trees and cannot be directly adapted to other classes of trees. Indeed, numerous authors tried to extend this algorithm without true success.

Our main observation in this paper is to interpret Rémy's algorithm in the context of modern Boltzmann sampling. Boltzmann samplers were introduced in [7]. They are a universal sampling theory, in the sense where they can (automatically) built from any specification of combinatorial classes. The classical Boltzmann sampler for Catalan trees, which is akin to a Galton-Watson process, exploits the algebraic equation for the generating function $C(z)$ defined above:

$$(0.2) \quad C(z) = z + zC(z)^2.$$

Contrary to Rémy's algorithm, Boltzmann samplers output an object with a random size (but objects of the same size are still drawn with the same probability).

^{*}LIX - École Polytechnique, Palaiseau, France. E-mail: bacher@lix.polytechnique.fr.

[†]Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS(UMR 7030), F-93430, Villetaneuse, France. E-mail: bodini@lipn.fr, jacquot@lipn.fr.

In this paper, we present a Boltzmann sampler for Catalan trees that uses a specification based on the holonomic equation (0.1), rather than the algebraic equation (0.2). This sampler turns out to be very similar to Rémy’s algorithm. We then expand this idea to the class of rooted unary-binary planar trees, or Motzkin trees, for which we obtain a linear-time exact-size sampler based on a holonomic specification. Several difficulties emerge: first, we need to find a combinatorial specification corresponding to the holonomic equation of the generating function of Motzkin trees; second, the Boltzmann sampler is much more intricate than the one for Catalan trees; third, the sampler, which outputs a tree of random size, must be efficiently turned into an exact-size sampler.

The originality of this work is to propose a method based on a holonomic specification. Other works allow to sample Motzkin trees in linear time, using bijections with other objects [1, 5]; our algorithm samples a Motzkin tree in average linear time using a rejection scheme (the probability of rejection is asymptotically $1/3$).

The paper is organized as follows. Section 1 deals with Catalan trees, recalling Rémy’s algorithm and presenting our simple Boltzmann sampler. Section 2 presents a holonomic specification for Motzkin trees. Finally, Section 3 describes a Boltzmann sampler based on this specification and shows how to optimize it to reach a linear exact-size sampler.

1 Specification and random sampling of binary trees

1.1 Holonomic functions and specification The goal of this section is to present a Boltzmann sampler for Catalan trees based on a holonomic specification. A holonomic specification of a class is a specification corresponding to a holonomic equation (or linear differential equation with polynomial coefficients) on the generating function of that class; such a specification uses the pointing operator [8, Section I.6.2].

As every algebraic power series is holonomic (or D-finite), it is always possible, given an algebraic class, to find a holonomic equation satisfied by its generating function. However, finding a combinatorial specification based on this equation is not automatic. In the case of Catalan trees, we show that the problem is relatively straightforward, and that the Boltzmann sampler obtained is essentially similar to Rémy’s algorithm. Thus, this section is a preamble to the much harder method we will develop on Motzkin trees.

As said in the introduction, the generating function of the class \mathcal{C} of Catalan trees satisfies the equation 0.1. In this section, we use the following specification for the class \mathcal{C} :

$$\mathcal{C}^{\bullet(\text{leaf})} = \mathcal{Z} + 2\mathcal{Z}^2\mathcal{C}^{\bullet},$$

where $\mathcal{C}^{\bullet(\text{leaf})}$ is the class of binary trees pointed on a leaf and \mathcal{C}^{\bullet} is the class of binary trees pointed on any node. Note that this point is independant to the root. This correspond to line 4 and 5 of figure 2.

Remark: As there are $2n - 1$ nodes in a tree with n leaves, this is equivalent, via a canonical repointing $2\mathcal{C}^{\bullet(\text{leaf})} \simeq \mathcal{C}^{\bullet} + \mathcal{C}$, to:

$$\mathcal{C}^{\bullet} + \mathcal{C} = 2\mathcal{Z} + 4\mathcal{Z}^2\mathcal{C}^{\bullet},$$

which directly gives the equation (0.1).

1.2 A differential Boltzmann sampler for Catalan trees versus Rémy algorithm. First, let us recall Rémy algorithm. This algorithm draws in a unique way every Catalan tree with n leaves pointed on a leaf.

Algorithm 1: Rémy’s algorithm

Input: A size n

Output: a Catalan tree of size n .

Start from a single leaf labelled 1

For i from 2 to n do:

Choose a node at random, add a new binary node between it and its father, and equiprobably a right or left sibling leaf. This new leaf is labelled by i .

Rémy proves in [11] that this algorithm draws a binary tree of size n uniformly at random. An example of a run can be find in Figure 1.

Now, let us recall briefly what Boltzmann samplers are. These samplers were first introduced in [7]. They provide a theoretical and efficient framework for the generation of combinatorial objects uniformly at random.

This approach is based on the *Boltzmann distribution* of parameter¹ x : $\mathbb{P}_x(\alpha) = x^{|\alpha|}/A(x)$ for every α in a combinatorial class \mathcal{A} with generating function $A(z)$. A *Boltzmann sampler* is a random generator such that the probability of drawing a given object follows a Boltzmann distribution. Boltzmann samplers can be built directly from the specification thanks to a dictionary describing the way to combine classical operators of the symbolic method (+, \times , Seq, ...).

Now, we succinctly describe a Boltzmann sampler for Catalan trees, based on the holonomic specification.

Algorithm 2: $\Gamma_x \mathcal{C}^\bullet + \mathcal{C}$

Input: the parameter x

Output: an object in \mathcal{C} .

If Bernoulli $\left(\frac{2x}{\mathcal{C}^\bullet + \mathcal{C}(x)}\right) = 1$
 return a leaf

Else

return draw $\gamma = \Gamma_x \mathcal{C}^\bullet$. From the marked vertex of γ add a new binary node between it and its father, and equiprobably a right or left sibling leaf.

Of course, this algorithm is incomplete: we need to explain how to obtain the Boltzmann sampler $\Gamma_x \mathcal{C}^\bullet$ from a recursive call to $\Gamma_x \Gamma_x \mathcal{C}^\bullet + \mathcal{C}$. For that, we would like to bias the Boltzmann distribution, as already developped in [4, 3, 2]. Here, this tool does not apply directly and we need a sampler for $(\mathcal{C}^\bullet + \mathcal{C})^\bullet$. By classical derivations, we have:

$$(\mathcal{C}^\bullet + \mathcal{C})^\bullet = 2\mathcal{Z} + 4\mathcal{Z}^2 \mathcal{C}^\bullet + 4\mathcal{Z}^2 (\mathcal{C}^\bullet + \mathcal{C})^\bullet.$$

This automatically translate to a Boltzmann sampler, from wich we can obtain a Boltzmann sampler for \mathcal{C}^\bullet (see Algorithm 5).

From these samplers, we derive an infinite process by choosing $x = 1/4$ (wich makes the density drop to 0 when $u \neq 1$). We can reverse the process in order to get a imperative form:

Algorithm 3: $\Gamma_{1/4} \mathcal{C}^\bullet + \mathcal{C}$

Input: none.

Output: does not terminate.

$T :=$ a single leaf.

Repeat: draw a node of T . From the marked vertex of T add a new binary node between it and its father, and equiprobably a right or left sibling leaf

This process can be seen as the limit of the Boltzmann sampler when the parameter x tends to $1/4$. Indeed, in this case, the probability to stop tends to 0. At each step of this process, the binary tree T is uniformly distributed; therefore, by artificially stopping the process when a targeted size is reached, we get an exact size sampler. In fact, this algorithm is identical to Rémy's algorithm.

We are now using the same method to find an algorithm “à la Rémy” for Motzkin trees: first we find a holonomic specification, then transform it into Boltzmann sampler and finally tune this sampler into an exact size sampler.

2 Holonomic specification for Motzkin trees

A unary-binary tree, or a *Motzkin tree* is a planar tree where all internal nodes have one or two children. An algebraic specification for the class \mathcal{M} of Motzkin trees counted according to the total number of nodes is

$$\mathcal{M} = \mathcal{Z} + \mathcal{Z}\mathcal{M} + \mathcal{Z}\mathcal{M}^2.$$

This specification uses a product which implies ramifications in the Boltzmann samplers (branching process). We want to avoid this type of construction in favor of an extension “à la Rémy” by splitting and grafting.

In the same vein that the holonomic specification of binary trees uses pointing on the leaves as a internal step, we need here two kinds of pointing on leaves. We can say that we have a blue pointing (first kind) and a green

¹Note that x needs to be a non-negative real number in the disk of convergence of $A(z)$.



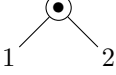
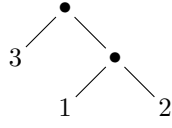
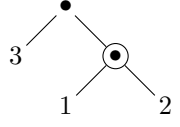
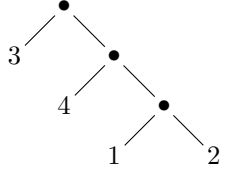
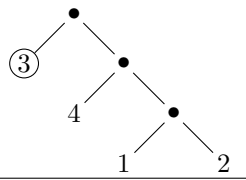
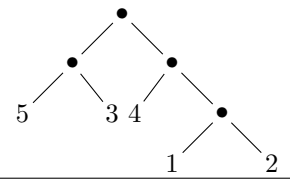
i	Pointed tree	Side of sibling	New tree
2		right	
3		left	
4		left	
5		left	

Figure 1: A example of a run of Remy algorithm for $n = 5$

pointing (second kind) and one red on unary nodes. We respectively denote these classes by $\mathcal{M}^{\bullet_1(\text{leaf})}$, $\mathcal{M}^{\bullet_2(\text{leaf})}$ and $\mathcal{M}^{\bullet_3(\text{unary})}$.

Computations, for example using the procedure `algeqtodiffeq` of the maple-package Gfun [12], gives the following linear differential equation for the generating function $M(z)$ of the class \mathcal{M} :

$$M(z) + zM'(z) = 2z + 3z^3M'(z) + z^2M'(z) + z(M(z) + zM'(z))$$

We use this information to find a combinatorial holonomic specification for the class of Motzkin trees. Let us begin by this easy lemma:

LEMMA 2.1. *The following two classes are isomorphic:*

$$\mathcal{M}^{\bullet_1(\text{leaf})} + \mathcal{M}^{\bullet_2(\text{leaf})} + \mathcal{M}^{\bullet_3(\text{unary})} \simeq \mathcal{M} + \mathcal{M}^{\bullet}$$

Proof. The total number of nodes in a Motzkin tree is twice the number of leaves plus the number of unary nodes minus 1. The relation ensues immediately.

THEOREM 2.1. *The following specification is a constructive combinatorial specification for the class of unary binary trees \mathcal{M} :*

$$\mathcal{M}^{\bullet_1(\text{leaf})} + \mathcal{M}^{\bullet_2(\text{leaf})} + \mathcal{M}^{\bullet_3(\text{unary})} = 2Z + 3Z^2\mathcal{M}^{\bullet} + Z\mathcal{M}^{\bullet} + Z(\mathcal{M}^{\bullet_1(\text{leaf})} + \mathcal{M}^{\bullet_2(\text{leaf})} + \mathcal{M}^{\bullet_3(\text{unary})}).$$

Proof. In Appendix 5.1

The combinatorial description of the specification is summarized in Figure 2.

3 Random sampling of Motzkin trees

This section is dedicated to the samplers for Motzkin trees. In a first part, we build a free Boltzmann sampler according to the constructible holonomic specification that we elaborated. By free Boltzmann sampler, we mean a generator without rejection on the size. So, the distribution of the output is a Boltzmann distribution. In a second part, we optimize this sampler. For that, we use the properties of non-ramifying Boltzmann sampling (i.e. without independent calls), to show that we can sample, in exact size, Motzkin trees, in linear expected time. The main idea is called the *Hand of God*. It corresponds to controlling the process of sampling arbitrarily and deciding to stop it at a given moment before its natural termination.

Term	Original tree	Final tree	contributes to	Description
\mathcal{Z}			$\mathcal{M}^{\bullet_1(\text{leaf})}$	A tree reduced at a 1-pointed leaf
\mathcal{Z}			$\mathcal{M}^{\bullet_2(\text{leaf})}$	A tree reduced at a 2-pointed leaf
$\mathcal{Z}\mathcal{M}^{\bullet}$			$\mathcal{M}^{\bullet_3(\text{unary})}$	A tree 3-pointed on a unary node
$\mathcal{Z}^2\mathcal{M}^{\bullet}$			$\mathcal{M}^{\bullet_1(\text{leaf})}$	a tree with a 1-pointed leaf right child of a binary node
$\mathcal{Z}^2\mathcal{M}^{\bullet}$			$\mathcal{M}^{\bullet_1(\text{leaf})}$	a tree with a 1-pointed leaf left child of a binary node
$\mathcal{Z}\mathcal{M}^{\bullet_1(\text{leaf})}$			$\mathcal{M}^{\bullet_1(\text{leaf})}$	a tree with a 1-pointed leaf child of a unary node
$\mathcal{Z}^2\mathcal{M}^{\bullet}$			$\mathcal{M}^{\bullet_2(\text{leaf})}$	a tree 2-pointed on a leaf right child of a binary node
$\mathcal{Z}\mathcal{M}^{\bullet_2(\text{leaf})}$			$\mathcal{M}^{\bullet_2(\text{leaf})}$	a tree 2-pointed on a leaf child of a unary node
$\mathcal{Z}\mathcal{M}^{\bullet_3(\text{unary})}$			$\mathcal{M}^{\bullet_2(\text{leaf})}$	a tree 2-pointed on a leaf left child of a binary node

Figure 2: A constructive interpretation of the holonomic specification for Motzkin trees.

3.1 Free Boltzmann sampler Let us begin by a Boltzmann sampler for $\mathcal{M} + \mathcal{M}^{\bullet}$. This algorithm follows rigorously the precepts of Boltzmann sampling. This gives an automatic proof of its validity. Note that we can use this sampler to draw a tree then forget the point: every tree of the same size will still be drawn with the same probability. So, let

$$F(z) = \frac{1 + 3z - \sqrt{1 - 2z - 3z^2}}{2\sqrt{1 - 2z - 3z^2}}$$

be the generating function of $\mathcal{M} + \mathcal{M}^{\bullet}$, and

$$G(z) = \frac{1 - z - \sqrt{1 - 2z - 3z^2}}{2\sqrt{1 - 2z - 3z^2}z}$$

be the generating function of \mathcal{M}^{\bullet} . Algorithm 4 is the Boltzmann sampler for $\mathcal{M} + \mathcal{M}^{\bullet}$ based on the specification of Section 2.

Now, we note that our algorithm makes call to a Boltzmann sampler of \mathcal{M}^{\bullet} . One might hope to bias the Boltzmann distribution $\Gamma_x(\mathcal{M} + \mathcal{M}^{\bullet})$ in order to simulate $\Gamma_x\mathcal{M}^{\bullet}$ and obtain a recursive algorithm. More precisely, following the classical idea already developed in [4, 3, 2], we can try to find a density of probability δ such that, by choosing a real u following δ , the sampler $\Gamma_{ux}(\mathcal{M} + \mathcal{M}^{\bullet})$ simulates exactly $\Gamma_x\mathcal{M}^{\bullet}$. But the conditions required for

Algorithm 4: $\Gamma_x(\mathcal{M} + \mathcal{M}^\bullet)$

Input: A parameter x .

Output: A 1, 2 or 3-pointed Motzkin tree

With Probability $\frac{x}{F(x)}$

return a tree reduced to a 1-pointed leaf.

With Probability $\frac{x}{F(x)}$

return a tree reduced to a 2-pointed leaf

With Probability $\frac{xG(x)}{F(x)}$

$T := \Gamma_x \mathcal{M}^\bullet$ a pointed tree.

 Add a 3-pointed unary node before the pointed node.

 Forget the original point.

return the obtained 3-pointed tree.

With Probability $\frac{x^2G(x)}{F(x)}$

$T := \Gamma_x \mathcal{M}^\bullet$ a pointed tree.

 Add a binary node before the pointed node, the pointed node is the right child, the left child is a 1-pointed leaf.

 Forget the original point.

return the obtained 1-pointed tree.

With Probability $\frac{x^2G(x)}{F(x)}$

$T := \Gamma_x \mathcal{M}^\bullet$ a pointed tree.

 Add a binary node before the pointed node, the pointed node is the left child, the right child is a 1-pointed leaf.

 Forget the original point.

return the obtained 1-pointed tree.

With Probability $\frac{x^2G(x)}{F(x)}$

$T := \Gamma_x \mathcal{M}^\bullet$ a pointed tree.

 Add a binary node before the pointed node, the pointed node is the right child, the left child is a 2-pointed leaf.

 Forget the original point.

return the obtained 2-pointed tree.

With Probability $\frac{xG(x)}{F(x)}$

$T := \Gamma_x(\mathcal{M} + \mathcal{M}^\bullet)$ a 1, 2 or 3-pointed tree.

If T is a 1 or 2-pointed tree **then**

 Add a unary node before the pointed node.

return the obtained 1 or 2-pointed tree.

Else

 Transform the 3-pointed unary node into a binary node by adding a 2-pointed leaf as its left child. Its original sub-tree is the right child.

 Forget the original point.

return the obtained 3-pointed tree.

the existence of δ correspond to a moment problem that is not feasible (Indeed, Stieljes conditions are violated). To circumvent this difficulty², we point the holonomic specification for Motzkin trees. We thus obtain the new specification:

$$\begin{aligned} & (\mathcal{M} + \mathcal{M}^\bullet)^\bullet \\ &= 2\mathcal{Z} + 6\mathcal{Z}^2\mathcal{M}^\bullet + 3\mathcal{Z}^2\mathcal{M}^{\bullet\bullet} + \mathcal{Z}\mathcal{M}^\bullet + \mathcal{Z}\mathcal{M}^{\bullet\bullet} + \mathcal{Z}(\mathcal{M} + \mathcal{M}^\bullet) + \mathcal{Z}(\mathcal{M} + \mathcal{M}^\bullet)^\bullet \\ &= 2\mathcal{Z} + 3\mathcal{Z}^2\mathcal{M}^\bullet + 3\mathcal{Z}^2(\mathcal{M} + \mathcal{M}^\bullet)^\bullet + \mathcal{Z}(\mathcal{M} + \mathcal{M}^\bullet) + 2\mathcal{Z}(\mathcal{M} + \mathcal{M}^\bullet)^\bullet. \end{aligned}$$

We easily construct a Boltzmann sampler $\Gamma_x(\mathcal{M} + \mathcal{M}^\bullet)^\bullet$ similar to the sampler $\Gamma_x(\mathcal{M} + \mathcal{M}^\bullet)$. This sampler makes recursive calls to itself, to $\Gamma_x\mathcal{M}^\bullet$ and to $\Gamma_x(\mathcal{M} + \mathcal{M}^\bullet)$. But now, $\Gamma_x\mathcal{M}^\bullet$ can be simulated using a biased version of $\Gamma_x(\mathcal{M} + \mathcal{M}^\bullet)^\bullet$, as shown below (Algorithm 5). [4]

Algorithm 5: $\Gamma_x\mathcal{M}^\bullet$ from $\Gamma_x(\mathcal{M} + \mathcal{M}^\bullet)^\bullet$

Input: the parameters x

Output: an object in \mathcal{M}^\bullet .

draw a real number v according to the density law $\frac{H(xv)}{vG(x)}$, where $H(z)$ is the generating function of $(\mathcal{M} + \mathcal{M}^\bullet)^\bullet$.

return $\gamma = \Gamma_{vx}(\mathcal{M} + \mathcal{M}^\bullet)^\bullet$ and forget the extra point.

Finally, one may object that we just have a Boltzmann sampler for $\mathcal{M} + \mathcal{M}^\bullet$. But this sampler can again be biased by a probabilistic modification of its parameter to have a sampler for \mathcal{M} . Thus, we are able to sample Motzkin trees according to the Boltzmann distribution $\mathbb{P}_x(T) = x^{|T|}/M(x)$.

3.2 Exact-size sampler Now, we optimize our free Boltzmann sampler to obtain an exact-size sampler in average linear time. We propose three different optimizations.

The first one consists in tuning the free Boltzmann sampler in the dominant singularity $z = 1/3$. Indeed, we are interested in generating the largest trees possible (in order to limit the sampling of small trees). A classical way to do that is to set the parameter of the sampler to the dominant singularity of the generating function. This can always be done with simple tree structures according to the Drmota-Lalley-Wood theorem ([6, 9, 13] and unified in [8, pp. 446–451]) which ensures that the dominant singularity is of type $(1 - z/\rho)^{1/2}$.

Here, we have a potential problem: we deal with some derivative of trees whose singularity is of type $(1 - z/\rho)^{-1/2}$. Nevertheless, due to the argument of the Hand of God, which consists in breaking the process of sampling when the size of the current (partial) tree reaches the fixed value n and to show that this premature tree is drawn uniformly along the Motzkin trees of the same size, we avoid the difficulty of the “natural” termination and we prove that we can still generate in the singularity $z = 1/3$.

The second optimization is to turn the recursive algorithm into an iterative algorithm. Thus, we avoid the stack of recursion. Note that when popping the recursive calls of each step of the recursive Boltzmann algorithm, we add 1 or 2 nodes to the tree returned by the recursive call, depending of a Bernoulli choice.

The third optimization concerns the line 13 and 14 of the exact-size algorithm 6. Without optimization, these two lines would be:

- 13** Transform the pointed T in $\mathcal{M}^{\bullet_1(\text{leaf})} + \mathcal{M}^{\bullet_2(\text{leaf})} + \mathcal{M}^{\bullet_3(\text{unary})}$ into a pointing \bar{T} in $\mathcal{M} + \mathcal{M}^\bullet$
14 If \bar{T} in \mathcal{M} then restart the sampler.

But the bijection between $\mathcal{M}^{\bullet_1(\text{leaf})} + \mathcal{M}^{\bullet_2(\text{leaf})} + \mathcal{M}^{\bullet_3(\text{unary})}$ and $\mathcal{M} + \mathcal{M}^\bullet$ cannot be easily done in constant time and this approach adds an extra rejection when \bar{T} is in \mathcal{M} . But, at this stage, we are interesting in a random sampling of a element in \mathcal{M}^\bullet . As the tree T is uniformly distributed among the Motzkin trees of the same size, we have just to select uniformly a node of T to obtain what we want.

This optimization gives Algorithm 6.

²If we accept rejection, we can also simulate $\Gamma_x\mathcal{M}^\bullet$, just by drawing elements in $\Gamma_x(\mathcal{M} + \mathcal{M}^\bullet)$ until we reach an element in \mathcal{M}^\bullet . The rejection is asymptotically negligible. So, it changes only by a constant the complexity of the sampler

Algorithm 6: $\hat{\Gamma}\mathcal{M}$

Input: a targeted size n .

Output: A 1, 2 or 3-pointed Motzkin tree of size n .

$S := 1$ a global parameter for the size of the current pre-tree

$T :=$ a 1 or 2-pointed leaf. (T is the current pre-tree)

while $S < n$ **do**

if $\text{Bernoulli}(1/3) = 1$ **then**

if T is a 1 or 2-pointed tree **then**

 Add a unary node before the pointed node of T

$S = S + 1$.

else

 Transform the 3-pointed unary node into a binary node by adding a 2-pointed leaf as its left child. Its original sub-tree is the right child.

$S = S + 1$.

 Forget the original point.

end

else

 Forget the pointing of T .

 Point at random a node in T in order to get a tree \bar{T} of \mathcal{M}^\bullet

if $\text{Bernoulli}(1/2) = 1$ **then**

 Add a 3-pointed unary node before the pointed node of \bar{T} .

 Forget the original point.

$S = S + 1$.

else

Equiprobably choose one of the tree following cases:

 - Add a binary node before the pointed node of \bar{T} , the pointed node is the right child, the left child is a 1-pointed leaf.

 Forget the original point.

$S = S + 2$.

 - Add a binary node before the pointed node of \bar{T} , the pointed node is the left child, the right child is a 1-pointed leaf.

 Forget the original point.

$S = S + 2$.

 - Add a binary node before the pointed node of \bar{T} , the pointed node is the right child, the left child is a 2-pointed leaf.

 Forget the original point.

$S = S + 2$.

end

end

end

if $S = n$ **then**

return T

else

return $\hat{\Gamma}\mathcal{M}$

end

THEOREM 3.1. *Algorithm 6 draws a Motzkin tree of size n uniformly among all Motzkin trees of size n in linear time.*

Proof. [Sketch of the proof] We proceed by transformation of the algorithm 4. Firstly, let us explain the Hand of God. Consider in all generality that we have a free Boltzmann sampler without ramification for a class \mathcal{C} (i.e. that the tree of recursive calls is reduced to a path.) If we can force during the process of sampling the Bernoulli choice of termination to succeed, the generated object is drawn uniformly among the objects of \mathcal{C} of the same size. The proof is quite obvious, but unconventional: if the Bernoulli choice naturally succeeds, by Boltzmann sampler properties we have the uniformity. Now assume that we are God, we can decide that the Alea is the termination. Thus, as the probability to stop naturally doesn't depend of the current size, we have the uniformity. So, with this property, we can add an “external” stop to Algorithm 4 if the size of the object exceeds n .

Now, thanks to this external stop, we can set the parameter x to $1/3$. Note that the probability of the natural termination tends to 0 and the probability $xG(x)/F(x)$, $x^2G(x)/F(x)$ and x tends respectively to $1/3$, $1/9$ and $1/3$.

Finally, we just rewrite in iterative version by recursion elimination the sampler and we optimize the lines concerning the repointing.

For the complexity, the only point to check is that the rejection line 33 which occurs when the sampling process jumped over the size n (from a tree of size $n - 1$ to a tree of size $n + 1$) just modified by a multiplicative constant the complexity. But, with the specification the number of trees of size $n + 1$ not built from trees of size n is $[z^{n+1}]3z^3M'(z)$ among $[z^{n+1}]M(z)$. Therefore, asymptotically, $1/3$ of the trees are rejected, which proves that the algorithm is linear in average.

4 Conclusion

This paper presents a new linear random sampler for the class \mathcal{M} of Motzkin trees, based on a holonomic specification of Motzkin trees in the same vein as Rémy's algorithm for binary trees. The first step to construct such an algorithm is to find a holonomic combinatorial specification for Motzkin trees. From this specification, we then build a Boltzmann sampler; this is almost automatic, but finding a Boltzmann sampler for \mathcal{M} from one for $\mathcal{M} + \mathcal{M}^\bullet$ required attention. Finally, we transform this sampler into a iterative exact-size sampler. Two main tools are necessary to do so: the possibility to choose the singularity as the Boltzmann parameter and the *Hand of God* property.

This work could be extended to other classes, for instance other classes of trees, by going through the same steps. The main obstacle is to find a holonomic combinatorial specification for the desired class; we expect that the other steps can be carried off in a manner similar to Motzkin trees.

This work was supported by the ANR Project MAGNUM ANR 2010 BLAN 0204 grant of the French Agence Nationale de la Recherche.

References

- [1] L. Alonso, J.L. Rémy, and R. Schott. A linear time algorithm for the generation of trees, 1996.
- [2] O. Bodini. How to generate an object under an ordinary Boltzmann distribution via an exponential boltzmann sampler. *CoRR*, abs/1006.2902, 2010.
- [3] O. Bodini and Jacquot A. Boltzmann samplers for v -balanced cycles. *TCS Theoretical Computer Science*, 2012.
- [4] O. Bodini, O. Roussel, and M. Soria. Boltzmann samplers for first order combinatorial differential equations. *Discrete Applied Mathematics*, pages 1–17 to appear, 2012.
- [5] L. Devroye. Simulating size-constrained galton-watson trees. *SIAM J. Comput.*, 41(1):1–11, 2012.
- [6] M. Drmota. Systems of functional equations. *Random Structures & Algorithms*, 10:103–124, 1999.
- [7] P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, Probability and Computing*, 13:577–625, 2004.
- [8] P. Flajolet and R. Sedgewick. *Analytic combinatorics*. Cambridge University Press, Cambridge, 2009.
- [9] S. P. Lalley. Finite range random walk on free groups and homogeneous trees. *The Annals of Probability*, 21(4):2087–2130, 1993.
- [10] E. Mäkinen. Generating random binary trees - a survey. *Inf. Sci.*, 115(1-4):123–136, April 1999.
- [11] J.L. Remy. Un procédé itératif de dénombrement d’arbres binaires et son application a leur génération aléatoire. *ITA*, 19(2):179–195, 1985.
- [12] B. Salvy and P. Zimmermann. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software*, 20(2):163–177, 1994.
- [13] A. R. Woods. Coloring rules for finite trees, and probabilities of monadic second order sentences. *Random Struct. Algorithms*, 10(4):453–485, July 1997.

5 Appendix

5.1 Proof of Theorem 2.1 To prove the validity of this specification, we describe the isomorphism:

$$\mathcal{M}^{\bullet_1(\text{leaf})} + \mathcal{M}^{\bullet_2(\text{leaf})} + \mathcal{M}^{\bullet_3(\text{unary})} \xrightarrow{f} 2\mathcal{Z} + 3\mathcal{Z}^2\mathcal{M}^{\bullet} + \mathcal{Z}\mathcal{M}^{\bullet} + \mathcal{Z}(\mathcal{M}^{\bullet_1(\text{leaf})} + \mathcal{M}^{\bullet_2(\text{leaf})} + \mathcal{M}^{\bullet_3(\text{unary})}).$$

We distinguish three cases.

- Let T be in $\mathcal{M}^{\bullet_3(\text{unary})}$; let u be the pointed unary node of T and let x be the child of u . To build the tree $f(T)$, we remove the node u and point its child x . We get a tree in $\mathcal{Z}\mathcal{M}^{\bullet}$.
- Let T be in $\mathcal{M}^{\bullet_1(\text{leaf})}$ and let ℓ be the pointed leaf of T . We further distinguish four cases, according to the father of ℓ .
 - The tree T is reduced to a leaf, which is counted by the term \mathcal{Z} .
 - The father of ℓ is a unary node u . The tree $f(T)$ is obtained by deleting the node u ; it is in the class $\mathcal{Z}\mathcal{M}^{\bullet_1(\text{leaf})}$.
 - The leaf ℓ is the left child of a binary node b . Let x be the right child of b . To build the tree $f(T)$, we remove the nodes b and ℓ and point the node x . This tree is in the class $\mathcal{Z}^2\mathcal{M}^{\bullet}$.
 - The leaf ℓ is the right child of a binary node b . This case is handled symmetrically from the previous one. The tree $f(T)$ is again in the class $\mathcal{Z}^2\mathcal{M}^{\bullet}$.
- Let T be in $\mathcal{M}^{\bullet_2(\text{leaf})}$ and let ℓ be the pointed leaf of T . Again, we distinguish several cases according to the father of ℓ .
 - The tree T is reduced to a leaf, ℓ is the child of a unary node, or the left child of a binary node. In this case, we proceed as in the previous case. The tree $f(T)$ is in the class $\mathcal{Z} + \mathcal{Z}\mathcal{M}^{\bullet_2(\text{leaf})} + \mathcal{Z}^2\mathcal{M}^{\bullet}$.
 - The pointed leaf ℓ is the right child of a binary node b . To build the tree $f(T)$, we delete the leaf ℓ and turn the node b into a unary node; we then point this new unary node. This tree is in the class $\mathcal{Z}\mathcal{M}^{\bullet_3(\text{unary})}$.

It is straightforward to prove that all these operations can be reversed. This shows that f is an isomorphism.