Improving the Florentine algorithms: recovering algorithms for Motzkin and Schröder paths

Axel Bacher

February 19, 2018

Abstract

We present random sampling procedures for Motzkin and Schröder paths, following previous work on Dyck paths. Our algorithms follow the anticipated rejection method of the Florentine algorithms (Barcucci et al. 1994+), but introduce a recovery idea to greatly reduce the probability of rejection. They use an optimal amount of randomness and achieve a better time complexity than the Florentine algorithms.

1 Introduction

This paper discusses random sampling procedures for two classical families of lattice paths: Motzkin and Schröder paths, shown in Figure 1. We are interested in *positive paths* (paths staying above the x-axis) and *excursions* (positive paths with final height zero). Together with Dyck paths, Motzkin and Schröder paths are widely studied in combinatorics. Their counting sequences are the Catalan, Motzkin and Schröder numbers; as can be seen in their OEIS entries [13] (A001405, A000108, A005773, A001006, A026003, A006318 and related ones), they are in bijection with hundreds of combinatorial objects—most notably, binary and unary-binary trees—making interesting and useful the problem of their efficient random sampling.

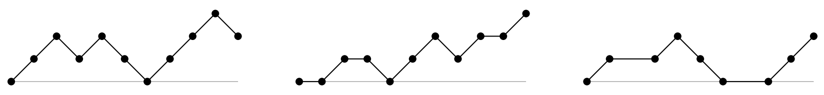
Figure 1: Dyck, Motzkin and Schröder positive paths of length 10.

The efficiency of an algorithm is measured, of course, by its time complexity (since all the algorithms discussed here only need negligible storage in addition to the output, space is not an issue). Also of interest for randomized algorithms is entropy complexity, which is a measure of the randomness consumed by the algorithm. Our model of entropy complexity closely follows that of [8], which takes its roots in Shannon's information theory and where the unit of complexity is the random bit. This framework aims at capturing the cost of random primitives in a realistic way and avoids unreasonable assumptions, like having access to random real numbers in constant time.

Many algorithms exist for sampling lattice paths or plane trees: Boltzmann samplers [7], Devroye's algorithm based on the cycle lemma [6], Rémy's algorithm [12], etc. However, none of these algorithms have a linear complexity for exact-size sampling: Boltzmann samplers provide approximate size, needing costly rejection to get exact size, while both others use an entropy of $n \log n$.

The Florentine algorithms [4, 5, 11] are, for their part, linear. They use an extremely simple method called *anticipated rejection*. To sample a positive path of length n, the path is drawn step by step at random, until either the length n is reached—at which point the path is output—or the path goes below the x-axis—at which point the path is deleted and the procedure started over. This is surprisingly efficient: sampling a positive path of length n requires, on average, to draw $2n$ random steps (n for the successful run, n for all the failed runs). This is due to the fact that rejection occurs, on average, on comparatively small paths. A detailed analysis is found in [10, 3]. Anticipated rejection was also used in random sampling algorithms for classes of trees, with similar complexities [2].

In the case of Dyck paths, an improved algorithm is given in [1] (actually, it works in the slightly more general case of m-Dyck paths). The idea of this algorithm, used before for binary trees in [2], is a *recovery* method: it follows the Florentine algorithm, but if the path goes below the x-axis, a "recovering" procedure is used to turn it into a random positive path, from which the algorithm is resumed. By avoiding rejection altogether, this algorithm only consumes asymptotically n random bits—which is optimal—and reads and writes $5n/4$ steps, better than the Florentine algorithm.

In this paper, we extend this recovery idea to Motzkin and Schröder paths. We retain a small probability of rejection, but this does not affect the complexity: the algorithms are still optimal in terms of entropy and have better time complexity than their Florentine counterparts.

The paper is organized as follows: Section 2 states general remarks useful in all models; in Sections 3 and 4, we give the algorithms for Motzkin and Schröder paths, respectively; finally, the complexity analysis is done in Section 5.

2 Preliminaries

We start by giving basic definitions and notations. We denote by u, f and d the up, flat and down steps, with height 1, 0 and -1 respectively. A *path* is a word on $\{\mathsf{u}, \mathsf{f}, \mathsf{d}\}$; the *height* of a path ω, denoted by $h(\omega)$, is the sum of the heights of its steps. A path is *positive* if all its prefixes have height ≥ 0; an *excursion* is a positive path with height 0; a path is *Łukasiewicz* if all its prefixes are positive except the path itself, which has height < 0. We denote by ε the empty path.

2.1 Unfolding and folding

In all three models, an essential ingredient of our algorithms is a classical bijection (see, e.g., [9, Chapter 9]), which we call *unfolding*. This bijection is illustrated in Figure 2. Consider a Łukasiewicz path factorized as $\sigma\tau$, with $\tau \neq \varepsilon$. If $h(\sigma) = k$, the path τ is of the form:

$$\tau = \tau_k \mathsf{d} \cdots \tau_0 \mathsf{d},$$

where the τ_i's are excursions (Figure 2, left). Define:

$$\tilde{\tau} = \mathsf{u}\tau_k \cdots \mathsf{u}\tau_0. \qquad (1)$$

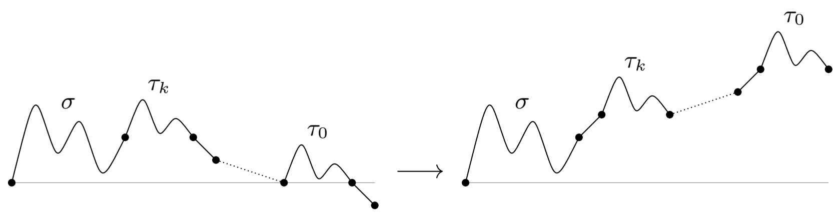

Figure 2: The unfolding operation, turning the Łukasiewicz path $\sigma\tau$ (left) into the positive path of odd height $\sigma\tilde{\tau}$ (right).

Proposition 1. *Every positive path of odd height can be written in a unique way as $\sigma\tilde{\tau}$, where $\sigma\tau$ is a Łukasiewicz path.*

Proof. Let ω be a positive path of height $2k+1$. According to the definition (1), it can be written $\sigma\tilde{\tau}$ only if $\tilde{\tau}$ is the suffix going up to the last visit at height $h(\sigma)$. Moreover, the path $\sigma\tau$ is Łukasiewicz if and only if $h(\sigma) = k$. This shows the proposition. □

In the following, we call *mid-height factorization* of ω the factorization $\sigma\tilde{\tau}$.

Remark. For the purpose of this paper, it is also acceptable, and perhaps simpler, to define $\tilde{\tau}$ as the *mirror* of τ (read τ backwards and change every u to a d and vice versa). We prefer the definition (1) because it seems more robust theoretically (the mirror does not work in the case of m-Dyck paths discussed in [1]) and because it only involves reading τ once in the forward direction, which is better in practice.

2.2 Structure of a recovering algorithm

Like in the Florentine algorithms, recovering algorithms build a path by adding random steps drawn according to some basic distribution. They also use a function, which we denote by recover, operating from Łukasiewicz paths to positive paths. The general structure is as follows.

Algorithm 1: Recovering algorithm for a random positive path of length n

```
1 $\omega \leftarrow \varepsilon$
2 while $|\omega| < n$ do
3     add a random step to $\omega$
4     if $h(\omega) < 0$ then $\omega \leftarrow \text{recover}(\omega)$
5 return $\omega$
```

If $n \geq 0$, consider the random path when it reaches a length at least n for the first time. Let $\mathcal{P}_n$ be the distribution of that path conditioned to be positive and $\mathcal{L}_n$ be the distribution of that path conditioned to be Łukasiewicz.

Theorem 2. *Assume that the* recover *function, when its input is distributed like $\mathcal{L}_n$, outputs a path distributed like $\mathcal{P}_n$. Then Algorithm 1 outputs a path distributed like $\mathcal{P}_n$.*

Proof. By induction, assume that the path ω is distributed like $\mathcal{P}_{n-1}$ when it first reaches a length $\geq n-1$. When it first reaches a length $\geq n$, it is either positive, in which case it is distributed like $\mathcal{P}_n$, or Łukasiewicz, in which case it is distributed like $\mathcal{L}_n$. After recovering, it is therefore distributed like $\mathcal{P}_n$. □

In the case of Dyck paths [1], the recover function works by taking a random factorization $\omega = \sigma\tau$ and unfolding; the result is a uniformly distributed positive path by Proposition 1.

In the cases of Motzkin and Schröder paths presented in this paper, the recover function does not work so perfectly: we retain some measure of anticipated rejection. To represent this, we define it as a *partial* function, meaning that it may be undefined with some probability. By convention, whenever an algorithm computes an undefined result, it immediately rejects the sample and terminates. The algorithm is then restarted until it produces an output. For the recovering algorithm to work, the output of the recover function only needs to follow the distribution $\mathcal{P}_n$ when it is defined.

If ω is a path, denote by $\langle\omega\rangle_k$ the path ω if its height is at least k and undefined otherwise.

3 Motzkin paths

In the Motzkin case, we build the path ω by adding $\mathtt{u}$, $\mathtt{d}$ and $\mathtt{f}$ steps with probability 1/3, 1/3 and 1/3. The distributions $\mathcal{P}_n$ and $\mathcal{L}_n$ are the uniform distributions on positive and Łukasiewicz paths of length n, respectively.

3.1 The recover operation

The difficulty in constructing a recovering procedure for Motzkin paths is the fact that, for any given n, there are Motzkin positive paths of length n of both odd and even height. Since unfolding only produces paths of odd height, we need a way to turn a path of odd height into one of even height. This is the purpose of the following involution, defined for paths which are not all d steps:

$$\mathrm{flip}\colon \sigma\mathsf{f}\mathsf{d}^k \longleftrightarrow \sigma\mathsf{u}\mathsf{d}^k. \tag{2}$$

This operation is illustrated in Figure 3.

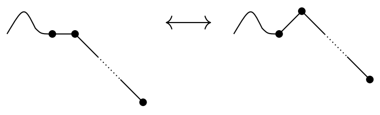

Figure 3: The flip operation, which changes the parity of the final height.

Let $q_n = 1/(2n+1)$. We define the recover operation, which takes a Łukasiewicz path of length n and outputs a random positive path as follows:

$$\text{recover}\colon \begin{cases} \sigma\tau \mapsto \sigma\tilde{\tau} & \text{with proba. } q_n, & \text{(3a)} \\ \sigma\tau \mapsto \mathrm{flip}(\sigma\tilde{\tau}) & \text{with proba. } q_n, & \text{(3b)} \\ \omega \mapsto \langle \mathrm{flip}(\omega)\rangle_0 & \text{with proba. } q_n. & \text{(3c)} \end{cases}$$

Since there are n possible factorizations $\omega = \sigma\tau$ with $\tau \neq \varepsilon$, the cases (3a) and (3b) are taken with probability $n/(2n+1)$ each for any given ω. Thus, if $\langle \mathrm{flip}(\omega)\rangle_0$ is undefined (either $\omega = \mathsf{d}$ or ω ends with $\mathsf{u}\mathsf{d}^k$), then rejection occurs with probability q_n.

Lemma 3. *Assume that ω is equal to every Łukasiewicz path of length n with probability p. Then, for every positive path ω' of length n, we have:*

$$\mathbf{P}\big[\mathrm{recover}(\omega) = \omega'\big] = pq_n.$$

Proof. We distinguish three cases: if ω' has odd height, it can only be built with (3a); if ω' has even height and $\mathrm{flip}(\omega')$ is positive, it can only be built by (3b); if ω' has height zero and $\mathrm{flip}(\omega)$ is Łukasiewicz, it can only be built by (3c). In all three cases, by Proposition 1, there is only one way to build ω'; thus, it is output with probability pq_n. □

3.2 Main algorithms

We are now ready to write the algorithms sampling random Motzkin positive paths and excursions. We only give here the proofs of their correction; complexity is discussed in Section 5.

The uniformity of the output of Algorithm 2 is an immediate consequence of Theorem 2 and Lemma 3. To show that the excursion output by Algorithm 3 is uniform, let p be the probability of any positive path of length $n+1$ to be drawn at line 1. After line 2, the path ω is equal to every positive path of odd height with probability $2p$ and, after line 3, to every Łukasiewicz path with probability $2(n+1)p$.

Algorithm 2: Random Motzkin positive path of length n

```
1 $\omega \leftarrow \varepsilon$
2 for $i = 1, \dots, n$ do
3     $\omega \leftarrow \omega\mathsf{u}$, $\omega\mathsf{f}$ or $\omega\mathsf{d}$ with probabilities 1/3, 1/3 and 1/3
4     if $h(\omega) = -1$ then $\omega \leftarrow \mathrm{recover}(\omega)$
5 return $\omega$
```

Algorithm 3: Random Motzkin excursion of length n

```
1 $\omega \leftarrow$ random Motzkin positive path of length $n+1$
2 if $h(\omega)$ is even then $\omega \leftarrow \langle \mathrm{flip}(\omega) \rangle_1$
3 $\sigma\tilde{\tau} \leftarrow$ mid-height factorization of $\omega$
4 return $\sigma\tau$ minus the last $\mathsf{d}$ step
```

3.3 Colored Motzkin paths

In this section, we consider Motzkin paths where the flat step carries a given positive real weight (this may be the case if there are several kinds of flat steps, hence the name colored Motzkin paths). We call *weight* of a path ω and denote by $\mathrm{wt}(\omega)$ the product of the weights of its steps. Florentine algorithms for these paths are discussed in [5].

It is also possible to define colored Motzkin paths with a weight for the up step, but this is not as interesting: if that weight is > 1 (positive drift), paths naturally go away from the x-axis and the Florentine algorithm is already asymptotically optimal; if it is < 1 (negative drift), paths naturally go below the x-axis and the Florentine algorithm is exponential. If we are interested in excursions, we do not lose any generality by assuming that the weight of u is 1. In any case, we need this condition so that the unfold function does not change the weight of the path.

We further impose that the weight of the f step is greater than 1. In this case, we may assume that there are four kinds of steps: u, f and d, with weight 1 each, and a fourth kind, $\mathsf{f_c}$, with a weight $c > 0$. If ω is a colored Motzkin path, define the *flippable step* (FS) of ω to be the last u or f step, if it exists. Let $\mathrm{flip}(\omega)$ be the path obtained by changing the FS from u to f or vice versa.

Let $q_n = 1/[2n + \max(1, c)]$. Define the new recover function as follows:

$$\mathrm{recover}\colon \begin{cases} \sigma\tau \mapsto \sigma\tilde{\tau} & \text{with proba. } q_n, & & \text{(4a)} \\ \sigma\tau \mapsto \mathrm{flip}(\sigma\tilde{\tau}) & \text{with proba. } q_n, & & \text{(4b)} \\ \omega \mapsto \mathrm{flip}(\omega) & \text{with proba. } q_n & \text{if FS} = \mathsf{f}, & \text{(4c)} \\ \omega\mathsf{d} \mapsto \omega\mathsf{f_c} & \text{with proba. } q_n c & \text{otherwise.} & \text{(4d)} \end{cases}$$

By construction, the probabilities sum to at most 1. Rejection occurs for some paths when $c \neq 1$. When $c = 1$, there is no rejection; this is the case of bicolored Motzkin paths, which are famously counted by the Catalan numbers. The last two cases are illustrated in Figure 4.

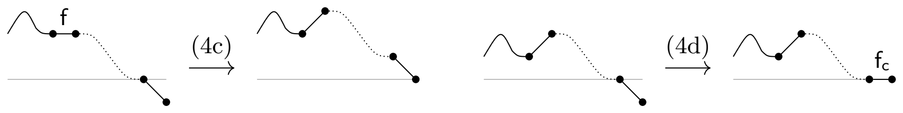

Figure 4: Left: the case (4c), producing excursions with a u FS ending with d. Right: the case (4d), producing excursions with a u FS ending with $\mathsf{f_c}$ (as well as the excursion with all $\mathsf{f_c}$ steps). The dotted part consists of d and $\mathsf{f_c}$ steps.

Lemma 4. *Assume that ω is equal to every Łukasiewicz path of length n with probability $p\,\mathrm{wt}(\omega)$. Then, for every positive path ω' of length n, we have:*

$$\mathbf{P}\big[\mathrm{recover}(\omega) = \omega'\big] = pq_n\,\mathrm{wt}(\omega').$$

Proof. A positive path can be built by the recover function in four different ways: paths with odd height are built with (4a); paths with even non-zero height and excursions with an f FS are built with (4b); excursions with a u FS and ending with d are built with (4c); excursions with a u FS and ending with $\mathsf{f_c}$, plus the excursion $\mathsf{f_c}^n$, are built with (4d).

Every path can be built exactly once in this way. Moreover, the number of steps $\mathsf{f_c}$ is not changed by recover, except in the case (4d), which adds one $\mathsf{f_c}$ step and happens with a probability multiplied by c. □

To sample a positive path or excursion, we use Algorithm 2 or 3 with the new recover function. The proofs are identical.

4 Schröder paths

In a Schröder path, flat steps have length 2 instead of 1. To avoid ambiguities, we denote by $|\omega|$ the number of steps of ω, by $|\omega|_{\mathsf{f}}$ its number of flat steps, and by $\ell(\omega) = |\omega| + |\omega|_{\mathsf{f}}$ its length. Following [11], we build Schröder paths by adding u, f and d steps with probabilities r, r^2 and r, where $r = \sqrt{2} - 1$, satisfying $2r + r^2 = 1$, is the radius of convergence of the generating function of Schröder paths.

Like a Dyck path, the length and height of a Schröder path have the same parity. When n is odd, $\mathcal{L}_n$ is the uniform distribution on Łukasiewicz paths of length n. However, a positive path may either reach length n exactly or overstep it with a flat step. Therefore, a path distributed like $\mathcal{P}_n$ is equal to every positive path of length n with probability proportional to 1 and to every positive path of length $n+1$ ending with f with probability proportional to r.

4.1 The recover operation

The recover operation should take a Łukasiewicz path of length n and output a path of length either n or $n+1$. Our first tool is the following random function, which extends a path of length n into a path of length $n+1$ in a recursive manner:

$$\text{extend}\colon \begin{cases} \omega \mapsto \omega\mathsf{u} & \text{with proba. } r, & (5a)\\ \omega \mapsto \omega\mathsf{d} & \text{with proba. } r, & (5b)\\ \omega\mathsf{u} \mapsto \omega\mathsf{f} & \text{with proba. } r^2, & (5c)\\ \omega\mathsf{d} \mapsto \omega\mathsf{f} & \text{with proba. } r^2, & (5d)\\ \omega\mathsf{f} \mapsto \text{extend}(\omega)\mathsf{f} & \text{with proba. } r^2. & (5e) \end{cases}$$

Rejection occurs with probability r^2 if $\omega = \varepsilon$ and, because of the recursion, with probability r^{2k+2} if $\omega = \mathsf{f}^k$. This function is illustrated in Figure 5.

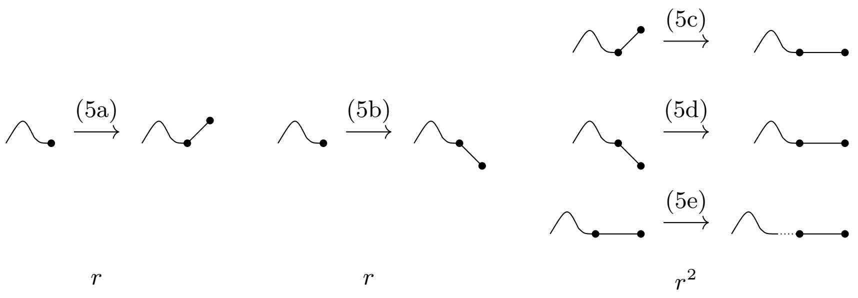

Figure 5: The three ways of extending a Schröder path: with probability r, adding an up step (5a); with probability r, adding a down step (5b); with probability r^2, turning the last up (5c) or down (5d) step into a flat step, or recursively extend the part before the last flat step (5e).

Lemma 5. *Assume that ω is equal to every positive path of length n with some probability p. For every positive path ω' of length $n+1$ and height > 0, we have:*

$$\mathbf{P}\big[\text{extend}(\omega) = \omega'\big] = pr.$$

Proof. We establish the result by induction on n. If ω' ends with a u (resp. d), then it can only be built with (5a) (resp. (5b)), yielding a probability pr. If it ends with an f, then it can be built with (5c), (5d) or (5e), yielding a probability $pr^2 + pr^2 + pr^3 = pr$ by induction hypothesis. Note that the condition $h(\omega') > 0$ is necessary in order to build ω' using (5d). ☐

Let $q_n = 1/(n+r)$. We define the recover function in the following way:

$$\text{recover}\colon \begin{cases} \sigma\tau \mapsto \sigma\tilde{\tau} & \text{with proba. } q_n, & (6a)\\ \sigma\mathsf{f}\tau \mapsto \langle\text{extend}(\sigma\tilde{\tau})\rangle_2\,\mathsf{f} & \text{with proba. } q_n, & (6b)\\ \omega\mathsf{d} \mapsto \omega\mathsf{f} & \text{with proba. } q_n r. & (6c) \end{cases}$$

To legitimize this definition, we note that there are $|\omega|$ factorizations of ω of type $\sigma\tau$ and $|\omega|_{\mathsf{f}}$ factorizations of type $\sigma\mathsf{f}\tau$; those two numbers sum to n. Rejection occurs in the case (6b) when extend$(\sigma\tilde{\tau})$ has height 0.

Lemma 6. *Let n be odd and let ω be equal to every Lukasiewicz path of length n with probability p. Then, for every positive path ω' of length n, we have:*

$$\mathbf{P}\left[\mathrm{recover}(\omega) = \omega'\right] = pq_n$$

and for every positive path ω' of length $n+1$ ending with f, we have:

$$\mathbf{P}\left[\mathrm{recover}(\omega) = \omega'\right] = pq_n r.$$

Proof. If $\ell(\omega') = n$, then ω' can only be produced with (6a), with a probability pq_n. If $\ell(\omega') = n+1$, then if $h(\omega') > 0$, it can only be produced by (6b), and Lemma 5 shows that it occurs with probability $pq_n r$; if $h(\omega') = 0$, it can only be produced by (6c), with probability $pq_n r$. □

4.2 Main algorithms

We are finally ready to present our algorithms for random Schröder positive paths and excursions, which are a bit more involved than in the Motzkin case. Again, analysis is postponed to Section 5.

Algorithm 4: Random Schröder positive path of length n or $n-1$

```
1 $\omega \leftarrow \varepsilon$
2 while $\ell(\omega) < n$ do
3     $\omega \leftarrow \omega\mathsf{u}$, $\omega\mathsf{f}$ or $\omega\mathsf{d}$ with probabilities $r$, $r^2$ and $r$
4     if $h(\omega) = -1$ then $\omega \leftarrow \mathrm{recover}(\omega)$
5 if $\omega = \sigma\mathsf{f}$, $\ell(\sigma) = n-1$ then $\omega \leftarrow \sigma$
6 return $\omega$
```

As a consequence of Theorem 2 and Lemma 6, the output of Algorithm 4 is either a path of length n with some probability p or a path of length $n-1$ with probability pr.

Algorithm 5: Random Schröder positive path of length n (odd)

```
1 $\omega \leftarrow$ random positive path of length $n$ or $n-1$
2 if $\ell(\omega) = n-1$ then $\omega \leftarrow \langle\mathrm{extend}(\omega)\rangle_1$
3 return $\omega$
```

In Algorithm 5, by Lemma 5, every path of length n is output with probability $p(1+r^2)$.

In Algorithm 6, if the **if** branch is taken, extending produces every positive path of length $n+1$ with probability pr. After folding, we get every Łukasiewicz path of length $n+1$ with a probability pr for each factorization of type $\sigma\tau$ and each factorization of type $\sigma\mathsf{f}\tau$, which adds up to $pr(n+1)$ for each path.

In Algorithm 7, finally, the branch in lines 1–3 produces every positive path with probability $p(n+1)/(n+1+r)$ and every non-excursion positive path with

Algorithm 6: Random Schröder excursion of length n (even)

```
1 $\omega \leftarrow$ random positive path of length $n$ or $n-1$
2 if $\ell(\omega) = n$ then
3     $\omega \leftarrow \langle \text{extend}(\omega) \rangle_1$
4     $\sigma\tilde{\tau} \leftarrow$ mid-height factorization of $\omega$
5     return $\sigma\tau$ minus the last $\mathsf{d}$ step
6 else
7     $\sigma\tilde{\tau} \leftarrow$ mid-height factorization of $\omega$
8     return $\sigma\mathsf{f}\tau$ minus the last $\mathsf{d}$ step
```

Algorithm 7: Random Schröder positive path of length n (even)

```
1 with probability $(n+1)/(n+1+r)$ do
2     $\omega \leftarrow$ random positive path of length $n$ or $n-1$
3     if $\ell(\omega) = n-1$ then $\omega \leftarrow \langle \text{extend}(\omega) \rangle_2$
4 with probability $r/(n+1+r)$ do
5     $\omega \leftarrow$ random excursion of length $n$
6 return $\omega$
```

probability $pr^2(n+1)/(n+1+r)$. The second branch produces every excursion with probability $pr(n+1)r/(n+1+r)$. In total, we get every positive path with probability $p(1+r^2)(n+1)/(n+1+r)$.

4.3 Little Schröder paths

Our last algorithms concern *little Schröder paths*, which are Schröder paths without flat steps at height 0 (entries A247623 and A001003 of the OEIS). Non-little and little paths are closely linked, as evidenced by the following classical bijection: by decomposing at the first flat step at height 0, any non-little Schröder path can be written in a unique way as $\sigma\mathsf{f}\tau$, where σ is a little excursion. Define:

$$\text{lift}\colon \sigma\mathsf{f}\tau \mapsto \sigma\mathsf{u}\tau.$$

Obviously, lift is a bijection between non-little paths of length n and little paths of length $n-1$ and height ≥ 1. Moreover, $\omega \mapsto \text{lift}(\omega)\mathsf{d}$ is a bijection between non-little and little excursions. This shows the well-known fact that the number of Schröder excursions of any positive length is exactly twice the number of little Schröder excursions and immediately gives Algorithm 8.

Sampling little positive paths is a bit more complicated. We need the following lemma, which establishes how the extend operation behaves with respect to little Schröder paths.

Lemma 7. *Assume that ω is equal to every little positive path of length n with probability p. For every little positive path ω' of length $n+1$, unless ω' has*

Algorithm 8: Random little Schröder excursion of length n (even)

```
1 $\omega \leftarrow$ random excursion of length $n$
2 if $\omega$ is not little then  $\omega \leftarrow \mathrm{lift}(\omega)\mathtt{d}$
3 return $\omega$
```

height 1 *and ends with* `f`*, we have:*

$$\mathbf{P}\big[\mathrm{extend}(\omega) = \omega'\big] = pr.$$

Proof. We proceed similarly to Lemma 5, by induction on n. If ω' ends with `u` or `d`, it has probability pr. If it ends with `f`, then it has height at least 2 (since it is a little path not ending with `f` at height 1). Write $\omega' = \sigma'\mathtt{f}$. Then ω can be $\sigma'\mathtt{u}$, $\sigma'\mathtt{d}$ or $\sigma\mathtt{f}$ when $\mathrm{extend}(\sigma) = \sigma'$. In the latter case, σ has height at least 1, which shows that $\sigma\mathtt{f}$ is a little path. By induction hypothesis, we get a probability of $pr^2 + pr^2 + pr^3 = pr$. ☐

Algorithm 9: Random little Schröder positive path of length n (even)

```
1 $\omega \leftarrow$ random positive path of length $n$
2 if $\omega$ is not little then
3     $\omega \leftarrow \mathrm{extend}(\mathrm{lift}(\omega))$
4     if $\omega$ is not little then reject
5 return $\omega$
```

Algorithm 10: Random little Schröder positive path of length n (odd)

```
1 $\omega \leftarrow$ random little positive path of length $n - 1$
2 $\omega \leftarrow \mathrm{extend}(\omega)$
3 if $\omega = \sigma\mathtt{f}$ and $h(\omega) = 1$ then reject
4 if $\omega = \sigma\mathtt{dd}$ and $h(\omega) = -1$ then $\omega \leftarrow \sigma\mathtt{f}$
5 return $\omega$
```

To show the uniformity of the output of Algorithm 9, note that since n is even, every positive path of length $n - 1$ has height > 0. Therefore, $\mathrm{lift}\,\omega$ is a uniformly distributed little positive path. Moreover, no path of length n has height 1, so by Lemma 7, $\mathrm{extend}(\mathrm{lift}\,\omega)$ is also uniform when it is a little path.

Let p be the probability of any path to be output by Algorithm 9. In Algorithm 10, since n is odd, the result of $\mathrm{extend}(\omega)$ is either a little positive path or a little Łukasiewicz path. In the latter case, it necessarily ends with `dd` (since ω is a little path); every such path appears with probability pr. Lemma 7 therefore shows the uniformity of the output.

Note. For both excursions and positive paths, the lift operation can be made to work in constant time: throughout all operations in the algorithms for Schröder paths—adding a step, extending, unfolding—it is possible to keep track of the first f step at height 0, if any, without any significant extra cost.

5 Complexity analysis

We analyse the algorithms presented in this paper in time and entropy complexity, going as far as limit law analysis. For entropy complexity, we use a slightly modified version of the model of [8]: we assume that the algorithms have access to a primitive giving a random step (a fair coin toss in the Dyck case, a fair 3-sided die in the Motzkin case, a 3-sided die with probabilities r, r and r^2 in the Schröder case), which carries a cost equal to the entropy of its distribution. Since the algorithms do not make any expensive computations (outside of drawing random steps, which we already accounted for), we choose for our measure of time complexity the number of steps in memory read or written.

Define the *time factor* of an algorithm to be its time complexity divided by the number of steps of the output and, similarly, the *entropy factor* to be the entropy complexity divided by the entropy of the output. In each case—positive paths, excursions, exact-size or not—the entropy of the output is asymptotic to the entropy of the random path of length n, which is equal to the entropy of the random steps needed to generate it.

Let R be an inhomogeneous Poisson point process on $(0, 1]$ with density function $\lambda(x) = 1/(2x)$. Let L be the sum, for all $x \in R$, of independent variables distributed like $\mathrm{Unif}[0, x]$ (this law is the same as in [1], and more information—moments, density, tail distribution—can be found in that paper; note that L is well-defined because, almost surely, the set R is infinite but summable). Finally, let U be a random variable distributed like $\mathrm{Unif}[0, 1]$ independent from L.

Theorem 8. *Recovering algorithms for positive paths (Algorithms 2, 4, 5, 7, 9 and 10) have an entropy factor tending to* 1 *and a time factor tending in distribution to* $1 + L$ *(expected value* $5/4$*). Algorithms for excursions (Algorithms 3, 6 and 8) have an entropy factor tending to* 1 *and a time factor tending in distribution to* $1 + L + U$ *(expected value* $7/4$*).*

For comparison, the Florentine algorithms for positive paths have, on average, entropy and time factors of 2. Their limit law is discussed in [3].

Proof. Let us start with Algorithms 2 and 4 (the basic recovering algorithms). We also, for the moment, ignore the possibility of rejection and consider only the final, successful run. There are two costs to account for: the cost of drawing steps and writing them to memory (a fixed time and entropy factor of 1) and the cost of recovery.

Let R_n be the set of lengths where recovery occurs. By Lemmas 3 and 6, the probability that i is in R_n is $q_i/(1 + q_i)$, where q_i is asymptotic to $1/(2i)$ in the Motzkin case and to $1/i$ when i is odd in the Schröder case. Moreover, since the distribution of the path is the same whether or not we recover, the events $i \in R_n$ are independent. This shows that R_n/n tends, as a point process, to R. Since recovering costs $\mathcal{O}(\log i)$ entropy and time equal to the size of a uniformly distributed right factor of the path (the part τ of the path $\sigma\tau$), its total cost in entropy is $\mathcal{O}(\log^2 n)$ and its total contribution to the time factor tends to L.

Let us now estimate the cost of the unsuccessful runs. The probability of rejection at length i is $\mathcal{O}(1/i)$ if we have to recover, or $\mathcal{O}(1/i^2)$ total. The probability of reaching length at least n without rejection is therefore:

$$\prod_{i=1}^{n} 1 - \mathcal{O}(1/i^2) = \Omega(1).$$

Thus, we only reject $\mathcal{O}(1)$ times on average. Moreover, if we do reject before length n, the average length we reach is:

$$\sum_{i=1}^{n} i \cdot \mathcal{O}(1/i^2) = \mathcal{O}(\log n).$$

Therefore, the total cost of rejection is $\mathcal{O}(\log n)$, which is negligible.

Finally, except in cases of probability $\mathcal{O}(1/n)$, all algorithms for positive paths work by calling the basic algorithm and then performing operations taking constant time, so their complexity is the same. The algorithms for excursions, again except in cases with probability $\mathcal{O}(1/n)$, work by sampling a positive path and folding; the latter costs no entropy and entails accessing a uniformly distributed right factor of the path, hence the result. □

Note. We can compute explicitly the probability of never having to reject. Let p_n be the probability of any one positive path of length n to be output. In the Motzkin case, by Lemma 3, we have $p_n = p_{n-1}/3\,[1 + 1/(2n+1)]$, which entails:

$$p_n = 3^{-n} \prod_{i=1}^{n} \frac{2i+2}{2i+1} = 3^{-n} \frac{\Gamma(n+2)\Gamma(3/2)}{\Gamma(n+1/2)} \sim 3^n \frac{\sqrt{\pi n}}{2}.$$

Let M_n be the number of positive paths of length n. Classically, we have $M_n \sim 3^{n+1/2}/\sqrt{\pi n}$. The probability of reaching length n is therefore:

$$p_n M_n \to \frac{\sqrt{3}}{2}.$$

This means that we have a more than 86% chance of succeeding in the first try.

In the Schröder case, we have, by Lemma 6, $p_n = p_{n-1} r[1 + 1/(n+r)]$ if n is odd and $p_n = p_{n-1} r$ if n is even. Therefore, we have:

$$p_n = r^n \prod_{i=1}^{\lceil n/2 \rceil} \frac{2i+r}{2i-1+r} \sim r^n \sqrt{n/2} \frac{\Gamma\left(\frac{1+r}{2}\right)}{\Gamma\left(1 + \frac{r}{2}\right)}.$$

Let S_n be the number of positive paths of length n. Using the estimate $S_n \sim 2^{-3/4}/(r^{n+1}\sqrt{\pi n})$, the probability of reaching at least length n is:

$$p_n S_n + p_n r S_{n-1} \to \frac{2^{1/4}}{\sqrt{\pi}} \frac{\Gamma\left(\frac{\sqrt{2}}{2}\right)}{\Gamma\left(\frac{1+\sqrt{2}}{2}\right)}.$$

Thus, we have a more than 94% chance of succeeding in the first try.

References

[1] A. Bacher. A new bijection on m-Dyck paths with application to random sampling. In *GASCom 2016*, 2016. `http://arxiv.org/abs/1603.06290`.

[2] A. Bacher, O. Bodini, and A. Jacquot. Efficient random sampling of binary and unary-binary trees via holonomic equations. *Theoretical Computer Science*, 695(Supplement C):42 – 53, 2017.

[3] A. Bacher and A. Sportiello. Complexity of anticipated rejection algorithms and the Darling–Mandelbrot distribution. *Algorithmica*, 74(1):1–20, 2015.

[4] E. Barcucci, R. Pinzani, and R. Sprugnoli. The random generation of directed animals. *Theoret. Comput. Sci.*, 127(2):333–350, 1994.

[5] E. Barcucci, R. Pinzani, and R. Sprugnoli. The random generation of underdiagonal walks. *Discrete Math.*, 139(1-3):3–18, 1995. Formal power series and algebraic combinatorics (Montreal, PQ, 1992).

[6] L. Devroye. Simulating size-constrained Galton-Watson trees. *SIAM J. Comput.*, 41(1):1–11, 2012.

[7] P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combin. Probab. Comput.*, 13(4-5):577–625, 2004.

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

[9] M. Lothaire. *Applied combinatorics on words*, volume 105 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2005.

[10] G. Louchard. Asymptotic properties of some underdiagonal walks generation algorithms. *Theoret. Comput. Sci.*, 218(2):249–262, 1999. Caen '97.

[11] J.-G. Penaud, E. Pergola, R. Pinzani, and O. Roques. Chemins de Schröder et hiérarchies aléatoires. *Theoret. Comput. Sci.*, 255(1-2):345–361, 2001.

[12] J. Rémy. Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire. *ITA*, 19(2):179–195, 1985.

[13] N. J. A. Sloane. The on-line encyclopedia of integer sequences. *Notices Amer. Math. Soc.*, 50(8):912–915, 2003. `http://oeis.org`.