Random Deterministic Automata

Cyril Nicaud

▶ **To cite this version:**

Cyril Nicaud. Random Deterministic Automata. MFCS 2014, Aug 2014, Budapest, Hungary. pp.5-23, 10.1007/978-3-662-44522-8_2 . hal-01772890

HAL Id: hal-01772890
https://hal.science/hal-01772890v1

Submitted on 20 Apr 2018

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.

Random Deterministic Automata*

Cyril Nicaud

LIGM, Université Paris-Est & CNRS, 77454 Marne-la-Vallée Cedex 2, France
cyril.nicaud@univ-mlv.fr

Abstract. In this article, we consider deterministic automata under the paradigm of average case analysis of algorithms. We present the main results obtained in the literature using this point of view, from the very beginning with Korshunov's theorem about the asymptotic number of accessible automata to the most recent advances, such as the average running time of Moore's state minimization algorithm or the estimation of the probability that an automaton is minimal. While focusing on results, we also try to give an idea of the main tools used in this field.

This article is dedicated to the memory of Philippe Flajolet.

1 Introduction

Automata theory [30] is a fundamental field of computer science, which has been especially useful in classifying formal languages, while providing efficient algorithms in several field of applications, such as text algorithms [17]. In this article, we consider deterministic automata under the paradigm of average case analysis of algorithms. We present the main results obtained in the literature with this approach, from the very beginning with Korshunov's theorem [35] to the most recent advances. We do not claim to be exhaustive, but hope to give a fair overview of the current state of this research area.

Following Knuth's seminal approach [33, Ch. 1.2.10], the area of *analysis of algorithms* aims at a more thorough analysis of algorithms by gaining insight on their running time in the best case, the worst case and the average case. Establishing that an algorithm behaves better in average than in the worst case is often done in two steps. First, one looks for properties of the inputs that makes the algorithm run faster. Then, using precise quantifications of various statistics, these properties are proved to hold with high probability[1] for random inputs.

To assist the researcher in spotting these properties, the community of analysis of algorithms also developed algorithms that randomly and uniformly generate a large variety of combinatorial structures. As we will see in the sequel, there several solutions available that are both efficient and quite generic. These random samplers are also very useful as substitutes for benchmarks for studying

* This work is supported by the French National Agency (ANR) through ANR-10-LABX-58 and through ANR-2010-BLAN-0204.

[1] i.e., with probability that tends to 1 as the size tends to infinity.

algorithms experimentally. A random generator is considered good if structures of size thousand can be generated in a few seconds. A very good generator can generated objects of size one billion within the same amount of time.

The two main mathematical tools for analyzing the average running time of algorithms are discrete probabilities [23] and *analytic combinatorics* [25]. The latter is a field of computer science that aims at studying large combinatorial structures (such as permutations, trees, ...) using their combinatorial properties. The analysis starts from an enumerative description of these objects, encoded into generating series, that is, the formal series $\sum_{n\geq 0} c_n z^n$, where c_n denote the number of objects of size n. When possible, these series are interpreted as analytic functions from $\mathbb{C}$ to $\mathbb{C}$. A precise analysis of these functions then provides information on the initial combinatorial structures, especially approximate estimations of various parameters, such as the expected number of cycles in a permutation, the typical height of a tree, and so on.

Under the impulsion of Flajolet, efforts were made to systematize the approach as much as possible. It was done in two main directions. First, by providing techniques to directly characterize the generating series from a combinatorial description of the structures of interest. Then, by stating theorems that yield useful and general results, while hiding the technical complex analysis methods within their proofs. Though its main domain of application is the average case analysis of algorithms and the design of efficient random generators, analytic combinatorics has also proved useful in various fields such as statistical physics or information theory.

As an example, consider the set of total maps from $[n]$ to itself.[2] If f is such a map, its *functional graph* is the directed graph with vertex set $[n]$ and with an edge from x to y whenever $f(x) = y$. Such a graph may consist of several components, each a cycle of trees (a forest whose roots are connected by a cycle). For n and k two positive integers, let $m_{n,k}$ denote the number of maps from $[n]$ to itself having exactly k cyclic points, where x is a cyclic point of f when there exists a positive i such that $f^i(x) = x$. Intuitively, cyclic points are the vertices of the functional graph belonging to a cycle. Assume we want to estimate the proportion of cyclic points in a random map from $[n]$ to $[n]$, for large values of n. The *symbolic method* [25, Ch. II] allows to directly translate the combinatorial specification "maps = set of cycles of trees" into the equality

$$M(z,u) := \sum_{n\geq 1}\sum_{k\geq 1} \frac{m_{n,k}}{n!} z^n u^k = \frac{1}{1-u\,T(z)} \quad \text{with} \quad T(z) = z\,e^{T(z)}\,.$$

At this point, $M(z,u)$ is viewed as an analytic function and the *singularity analysis* techniques [25, Ch. VI] yield that the average number of cyclic points is asymptotically equivalent to $\sqrt{\pi\,n/2}$.

Since a random deterministic automaton has a lot of useless states with high probability, we focus on the combinatorics of accessible automata in Section 2.

[2] For any positive integer n, $[n]$ denote the set $\{1,\ldots,n\}$.

We give some combinatorial bijections between automata and other combinatorial structures, as well as asymptotic results on the number of automata. In Section 3, we present on a toy example a useful technique, which is widely used in the analysis of random automata. With this technique, one can export a probabilistic property of random automata to random accessible automata at low cost, avoiding the difficulty of dealing with accessible automata directly. We investigate the problem of generating accessible automata uniformly at random in Section 4. In Section 5, we briefly explained an important result, which states that a non negligible ratio of accessible automata are minimal. We then present the average case analysis of an algorithm, namely Moore's state minimization algorithm, in Section 6. In Section 7, we state some recent results about the random synchronization of automata and present some open problems that seem to be relevant for further studies.

Though not always explicitly mentioned, analytic combinatorics and discrete probabilities are used to establish most of the results presented in this article.[3]

2 Combinatorics of Automata

We start our presentation by studying the combinatorics of deterministic automata. As a guideline, we will answer the following question:[4]

> **Question 1:** *What is the asymptotic number of accessible deterministic and complete automata with n states?*

Let us formalize the question first. Let A be a fixed alphabet with $k \geq 2$ letters.[5] For any $n \geq 1$, an *n-state transition structure* $\mathcal{T}$ on A is a pair (q_0, δ), where $q_0 \in [n]$ is the *initial state* and δ is a total map from $[n] \times A$ to $[n]$ called the *transition function* of $\mathcal{T}$. Since δ is a total map, a transition structure is a classical deterministic and complete automaton with set of states $[n]$, with q_0 as initial state, but with no final states. An n-state deterministic and complete automaton on A is a tuple (q_0, δ, F), where (q_0, δ) is an n-state transition structure and $F \subseteq [n]$ is the *set of final states*. Let $\mathfrak{T}_n$ and $\mathfrak{A}_n$ denote the set of n-state transition structures and n-state automata respectively. Obviously, we have $|\mathfrak{T}_n| = n \cdot n^{kn}$ and $|\mathfrak{A}_n| = n \cdot n^{kn} \cdot 2^n$. Since we will only consider deterministic and complete automata, we simply call them *automata* in the sequel.

Recall that a word u is *recognized* by the automaton $\mathcal{A}$ when it labels a path from the initial state to a final state. To define it formally, we extend δ to words in A^* by setting inductively that for every $p \in [n]$, $u \in A^*$ and $a \in A$, $\delta(p, \varepsilon) = p$ and $\delta(p, ua) = \delta(\delta(p, u), a)$. A word u is *recognized* by $\mathcal{A}$ when $\delta(q_0, u) \in F$. Let $\mathcal{L}(\mathcal{A})$ denote the *language recognized by* $\mathcal{A}$, i.e., the set of words it recognizes.

[3] For instance, the probabilistic study of the number of cyclic points in a random map, presented above, is one of the cornerstone of [10, 19, 20, 42].

[4] For the exact number of automata, see Remark 7.

[5] Automata on alphabets with only one letter are very specific. See [40] for information on their typical behavior when taken uniformly at random.

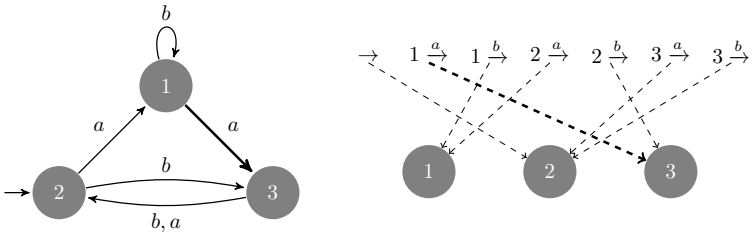

Fig. 1. An accessible automaton and its associated surjection ϕ: a transition $p \xrightarrow{a} q$ indicates that $\phi(p \xrightarrow{a}) = q$ and $\phi(\rightarrow)$ designates the initial state.

A state p of an automaton or a transition structure is *accessible* when there exists $u \in A^*$ such that $\delta(q_0, u) = p$. An automaton or a transition structure is *accessible* when all its states are accessible. Similarly, a state p of an automaton is *co-accessible* when there exists $u \in A^*$ such that $\delta(p, u) \in F$, and an automaton is *co-accessible* when all its states are co-accessible.

States that are not accessible or not co-accessible are useless, since they can be removed without changing the recognized language. We will see that from a probabilistic point of view, an accessible automaton is co-accessible with high probability (see Remark 9). The number of automata with no useless states is therefore asymptotically equivalent to the number of accessible automata. This justifies the choice of Question 1, and we now turn our attention to the set $\mathfrak{C}_n$ of n-state transition structures that are accessible.

In a transition structure, the action of each letter $a \in A$, i.e, the map $p \mapsto \delta(p, a)$, is a total map from $[n]$ to $[n]$. Using for instance techniques of analytic combinatorics [25], it is not difficult to establish that the expected number of elements with no preimage by a random map from $[n]$ to $[n]$ is $e^{-1}n$ (see [24]). Hence, roughly speaking, in a random element of $\mathfrak{T}_n$ on a two-letter alphabet, there are around $e^{-1}n$ states with no incoming transition labelled by a and $e^{-1}n$ states with no incoming transition labelled by b. "Therefore", there are around $e^{-2}n$ states with no incoming transition. This informal argument can be turned into a proof. It establishes that with high probability an element of $\mathfrak{T}_n$ is not accessible, as only the initial state can have no incoming transition in an accessible structure. Hence, $|\mathfrak{C}_n|$ is asymptotically much smaller than $|\mathfrak{T}_n|$.

The idea of Korshunov [35] is to consider elements $\mathfrak{T}_n$ whose states have at least one incoming transition, except possibly the initial state. Let $\mathfrak{T}'_n$ denote the set of such transition structures. Of course, an element of $\mathfrak{T}'_n$ is not always accessible: two strongly connected components that are totally disjoint form a non-accessible element of $\mathfrak{T}'_n$. But we will see in the sequel that it is a reasonable approximation of $\mathfrak{C}_n$. Let $E_n = [n] \times A \cup \{\rightarrow\}$ and let $\mathcal{S}_n$ denote the set of surjections from E_n onto $[n]$. To each element $\mathcal{T} = (q_0, \delta)$ of $\mathfrak{T}'_n$ one can associate bijectively an element f of $\mathcal{S}_n$, by setting $f(\rightarrow) = q_0$ and $f((p, a)) = \delta(p, a)$ (an example is depicted in Fig. 1). Hence $|\mathfrak{T}'_n|$ is equal to the number $S(kn+1, n)$ of

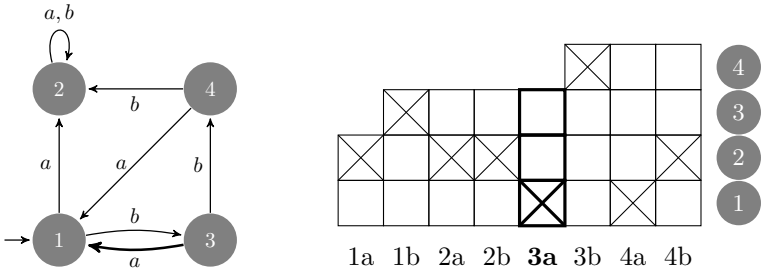

Fig. 2. An accessible automaton and its associated diagram, as introduced in [7]. The states are labelled following a breadth-first traversal of the automaton, taking a-transitions before b-transitions. Every column of the diagram corresponds to a transition $p \xrightarrow{a} q$. The height of a column is the number of states discovered so far in the traversal, and there is a cross in the row corresponding to the target q of the transition. For example, 3 states were discovered when considering the transition $3 \xrightarrow{a} 1$. Therefore, the associated column has height 3 and there is a cross in the first row, as 1 is the target state of this transition.

surjections from a set with $kn+1$ elements onto a set with n elements. Good [27] used the *saddle point method* (see [25, Ch. VIII]) to obtain an asymptotic equivalent of $S(n, m)$ when $m = \Theta(n)$. Using his result we get that the number of surjections from $[kn + 1]$ onto $[n]$ satisfies

$$S(kn + 1, n) \sim \alpha_k \, \beta_k^n \, n^{kn+1} \, ,$$

where $\alpha_k > 0$ and $\beta_k \in (0, 1)$ are two computable constants.

Remark 1. The quantity $\alpha_k \beta_k^n$ is exactly the probability that a mapping from $[kn + 1]$ to $[n]$ is a surjection. By Good's result, this probability is hence exponentially small.

The main result of Korshunov in [35] is that, asymptotically, the number of accessible transition structures differs from $|\mathfrak{T}'_n|$ by a multiplicative constant: if $|A| \geq 2$ then $|\mathfrak{C}_n| \sim \gamma_k \, |\mathfrak{T}'_n|$, where $\gamma_k > 0$ is an explicit constant. The proof is too complicated to be sketched here. It relies on a precise combinatorial study of the shape of a random element of $\mathfrak{T}'_n$. Using Good's estimation for the number of surjections, we therefore get the answer to Question 1:

$$|\mathfrak{C}_n| \sim \gamma_k \, \alpha_k \, \beta_k^n \, n^{kn+1} \, 2^n \, .$$

Remark 2. If ρ_k is the smallest positive solution of the equation $x = k(1 - e^{-x})$, then $\beta_k = \frac{k^k (e^{\rho_k} - 1)}{e^k \rho_k^k}$. Numerically, we have $\beta_2 \approx 0.836$ and $\beta_3 \approx 0.946$.

Remark 3. Korshunov gave a complicated formula for γ_k, using limits of converging series. Lebensztayn greatly simplified it with the theory of Lagrange inversion [36].

Fig. 3. On the left, an automaton with inner symmetries: there are only 3 different ways to label this shape. The automaton on the right is accessible; hence it has no inner symmetry and there are 3! different ways to label it.

Remark 4. Korshunov also proved that the number of strongly connected transition structures has the same order of growth: it is asymptotically equivalent to $\delta_k\, \beta_k^n\, n^{kn+1}$, for some positive δ_k.

Remark 5. In [7], Bassino and the author used another encoding for elements of $\mathfrak{T}'_n$. Instead of surjections, these transition structures are represented by diagrams obtained during a breadth-first traversal, as shown on Fig. 2. They have the advantage that there is a simple characterization of diagrams that correspond to accessible transition structures. These diagrams were used by Bassino, David and Sportiello to count the asymptotic number of minimal automata [6]. This result is presented in Section 5.

The answer we gave to Question 1 may seem to be unsatisfactory, since we consider two automata that only differ by their state labels as different. An *isomorphism of transition structures* is a bijection ϕ from the states of a transition structure $\mathcal{T} = (q_0, \delta)$ to those of $\mathcal{T}' = (q'_0, \delta')$ such that $\phi(q_0) = q'_0$ and for every state p and every letter a, $\delta'(\phi(p), a) = \phi(\delta(p, a))$. For the definition of *isomorphism of automata* we also require that $\phi(p)$ is final if and only if p is. It can seem more relevant to count the number of isomorphic classes rather than the number of transition structures or automata.

There is not the same number of automata in every isomorphic class, as depicted in Fig. 3. Such situations can make the counting of the number of classes quite difficult. Fortunately, the situation is easier when considering only the number of accessible structures: each state p of an n-state accessible automaton (or transition structure) is completely characterized by the smallest word u for the *radix order*,[6] also called the *length-lexicographic order*, such that $\delta(q_0, u) = p$. Hence, every bijection from the set of those words to $[n]$ define a different labelling for the automaton. Each isomorphic class of an accessible automaton or transition structure therefore contains exactly $n!$ elements. Thus, using Stirling formula, we can give the final answer to Question 1:

> **Answer 1:** *The number of isomorphic classes of accessible automata with n states is asymptotically equivalent to $\gamma'_k\, \beta'^n_k\, n^{(k-1)n+1/2}\, 2^n$, where $\gamma'_k = \frac{\gamma_k \alpha_k}{\sqrt{2\pi}}$ and $\beta'_k = e \cdot \beta_k$ are two computable positive constants.*

[6] Compare the length first, and use the lexicographic order if they have same length.

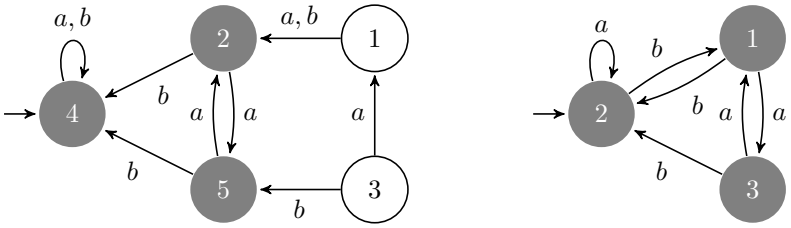

Fig. 4. On the left, a transition structure $\mathcal{T}$ with 5 states. Its states 1 and 3 are not accessible. The accessible part of $\mathcal{T}$ is depicted on the right. It is obtained by removing 1 and 3, and by normalizing the remaining state labels while preserving their relative order: $2 \mapsto 1$, $4 \mapsto 2$ and $5 \mapsto 3$.

3 From Automata to Accessible Automata

Our next goal is to obtain information on the typical properties of random accessible automata. However, we saw in the previous section that working with accessible structures can be quite complicated. In particular, we do not know any useful formula for their generating series. We therefore cannot directly use well-established techniques, such as analytic combinatorics, in order to obtain statistics on accessible automata. In this section we will explain how, in some situations, we can establish a property on random accessible automata by first proving it on random automata. We illustrate this technique on Question 2 below, which is a toy question we use to emphasize the method. More important applications will be presented in the following sections. Recall that a state p is a *sink state* if for every $a \in A$, $\delta(p, a) = p$.

Question 2: *Does a random accessible automaton often has a sink state?*

The question is easy if we remove the accessibility condition. A random automaton with n states for the uniform distribution can be seen as choosing the initial state uniformly in $[n]$, then, independently, choosing $\delta(p, a)$ uniformly in $[n]$ for every $p \in [n]$ and every $a \in A$. Assume for this section that $A = \{a, b\}$. The probability that a given state p is a sink state is therefore $\frac{1}{n^2}$. Hence, by the union bound, the probability that there is a sink state is[7] at most $\frac{1}{n}$: a uniform random automaton has no sink state with high probability. We will now show how to use this simple result to establish a similar statement for random accessible automata.

For any $\mathcal{T} \in \mathfrak{T}_n$, the *accessible part* of $\mathcal{T}$ is the accessible transition structure obtained by removing the states that are not accessible. If the accessible part has m states, we relabel them with $[m]$ while preserving their relative order, as depicted in Fig. 4.

Let m be an integer with $1 \leq m \leq n$ and let $\mathcal{T}$ be an element of $\mathfrak{C}_m$. We want to compute the number of elements of $\mathfrak{T}_n$ whose accessible part is $\mathcal{T}$. To

[7] In fact, it is exactly $1 - (1 - \frac{1}{n^2})^n$.

build such a transition structure, we first choose one of the $\binom{n}{m}$ possible size-m subsets of $[n]$ for labelling the states of $\mathcal{T}$. Then observe that the transitions outgoing from an accessible state are already defined by the choice of $\mathcal{T}$ and that the other transitions can end anywhere without changing the accessible part. Therefore, there are exactly $\binom{n}{m} n^{k(n-m)}$ elements of $\mathfrak{T}_n$ whose accessible part is $\mathcal{T}$. A crucial observation here is that this quantity only depends on n and m: two elements of $\mathfrak{C}_m$ are the accessible part of a random element of $\mathfrak{T}_n$ with the same probability. Using the vocabulary of probability: *conditioned by its size m, the accessible part of a uniform random element of $\mathfrak{T}_n$ is a uniform random element of $\mathfrak{C}_m$.*

Let $\#\mathtt{acc}(\mathcal{A}_n)$ be the random variable associated with the number of states of the accessible part of a random element $\mathcal{A}_n$ of $\mathfrak{T}_n$. By summing the contribution of all $\mathcal{T} \in \mathfrak{C}_m$, and since $|\mathfrak{T}_n| = n^{kn+1}$, we get that

$$\mathbb{P}\big(\#\mathtt{acc}(\mathcal{A}_n) = m\big) = \frac{1}{n}\binom{n}{m} |\mathfrak{C}_m|\, n^{-km}\,. \tag{1}$$

This is the starting point of the study of $\#\mathtt{acc}(\mathcal{A}_n)$ done by Carayol and the author in [13]. Thanks to Korshunov's result presented in Section 2, a fine analysis of the distribution of $\#\mathtt{acc}(\mathcal{A}_n)$ is possible. In particular, we proved that there exists a computable constant $v_k \in (0,1)$ such that $\mathbb{E}[\#\mathtt{acc}(\mathcal{A}_n)] \sim v_k\, n$. Numerically, $v_2 \approx 0.7968$, meaning that about 80% of the states are accessible in a typical automaton over a two-letter alphabet. The distribution is also concentrated around its mean: for any $\epsilon > 0$ the accessible part has size between $(1-\epsilon)v_k\, n$ and $(1+\epsilon)v_k\, n$ with high probability. Moreover, there exists a positive constant ω_k such that

$$\mathbb{P}\Big(\#\mathtt{acc}(\mathcal{A}_n) = \lfloor v_k\, n \rfloor\Big) \sim \frac{\omega_k}{\sqrt{n}}\,. \tag{2}$$

Remark 6. The main contribution of [13] is that the sequence of random variables $\#\mathtt{acc}(\mathcal{A}_n)$ is asymptotically Gaussian, of expectation and standard deviation asymptotically equivalent to $v_k\, n$ and $\sigma_k \sqrt{n}$ respectively, where v_k and σ_k are two computable positive constants. Hence once properly normalized, it converges in distribution to the normal law.

We can now give an answer to Question 2. Let $\mathcal{A}_n$ denote a random element of $\mathfrak{T}_n$, let $\mathtt{acc}(\mathcal{A}_n)$ denote its accessible part and let $\#\mathtt{acc}(\mathcal{A}_n)$ denote the number of states in $\mathtt{acc}(\mathcal{A}_n)$. We first use the fact that conditioned by its size, the accessible part of an automaton is a uniform accessible automaton: the probability that an element of $\mathfrak{C}_m$ has a sink state is equal to the probability that an element of $\mathfrak{T}_n$ has a sink state in its accessible part given the accessible part has size m. Therefore, we aim at studying the quantity $\mathbb{P}(\mathtt{acc}(\mathcal{A}_n) \text{ has a sink state } \mid \#\mathtt{acc}(\mathcal{A}_n) = m)$. But if there is a sink state in the accessible part, there is a sink state in the automaton. Hence,

$$\begin{aligned}\mathbb{P}\big(\mathtt{acc}(\mathcal{A}_n) \text{ has a sink state } &\mid \#\mathtt{acc}(\mathcal{A}_n) = m\big)\\ &\leq \mathbb{P}\big(\mathcal{A}_n \text{ has a sink state } \mid \#\mathtt{acc}(\mathcal{A}_n) = m\big)\,.\end{aligned}$$

Using the definition of conditional probability, we get

$$\begin{aligned}\mathbb{P}\big(\mathcal{A}_n \text{ has a sink state} \mid \#\mathtt{acc}(\mathcal{A}_n) = m\big) \\ &= \frac{\mathbb{P}\big(\mathcal{A}_n \text{ has a sink state, } \#\mathtt{acc}(\mathcal{A}_n) = m\big)}{\mathbb{P}\big(\#\mathtt{acc}(\mathcal{A}_n) = m\big)} \\ &\leq \frac{\mathbb{P}(\mathcal{A}_n \text{ has a sink state})}{\mathbb{P}\big(\#\mathtt{acc}(\mathcal{A}_n) = m\big)}\,.\end{aligned}$$

We already proved that the numerator is at most $\frac{1}{n}$ at the beginning of the section. Moreover, by Equation (2), if we choose n such that $m = \lfloor v_k\, n \rfloor$, then $n = \Theta(m)$ and the denominator is in $\Theta(\frac{1}{\sqrt{m}})$. Therefore, the probability that an element of $\mathfrak{C}_m$ has a sink state is in $\mathcal{O}(\frac{1}{\sqrt{m}})$. This gives the answer to Question 2:

> **Answer 2:** *With high probability, a random accessible automaton has no sink state.*

Remark 7. Equation (1) rewrites in $n^{kn+1} = \sum_{m=1}^{n} \binom{n}{m} |\mathfrak{C}_m|\, n^{k(n-m)}$. This is a way to calculate the values of $|\mathfrak{C}_m|$ using a computer. The first values of the number $\frac{1}{m!}|\mathfrak{C}_m|$ of accessible transition structures up to isomorphism are,[8] for $k = 2$,

$$1,\ 12,\ 216,\ 5248,\ 160675,\ 5931540,\ 256182290,\ \ldots$$

This formula for $|\mathfrak{C}_m|$ was given by Liskovets [37]. See also Harrison [28] for the first formulas that enumerate several kind of automata.

Remark 8. The answer to Question 2 we gave follows the article [13], but the idea we used is already in Korshunov's paper [35]. To apply the method and prove that a property P does not hold with high probability for $\mathfrak{C}_n$, it is sufficient that (i) the probability P holds for $\mathfrak{T}_n$ is in $o(\frac{1}{\sqrt{n}})$ and that (ii) if $\mathcal{T} \in \mathfrak{C}_n$ satisfies P, then any transition structure whose accessible part is $\mathcal{T}$ also satisfies P.

Remark 9. By moving randomly the initial state and by using our result on the size of the accessible part, we can observe that a random automaton should have a unique stable[9] strongly connected component with high probability, which has size around $v_k\, n$. This implies that for any state p, there exists a path from p to this stable connected component. Moreover this component has a final state with high probability. Using the same technique as for sink states, this "proves" that an accessible automaton is co-accessible with high probability. By controlling the error terms, this informal argument can be turned into a full proof [20].

Remark 10. In the classical Erdős-Rényi model, a random graph with n vertices has an edge between any two vertices with probability p_n, independently. The phase transition result [11] states that there are three phases for the connectedness of such a graph: if $p_n \ll \frac{1}{n}$ then a typical graph is completely disconnected,

[8] This is the sequence **A006689** of the Online Encyclopedia of Integer Sequences.
[9] A set S is stable when there is no transition $p \to q$ for $p \in S$ and $q \notin S$.

with very small connected components; if $\frac{1}{n} \ll p_n \ll \frac{\log n}{n}$, there is a giant connected component of linear size and the other connected components are much smaller; if $\frac{\log n}{n} \ll p_n$, the graph is connected with high probability. This result have been extended by Karp to directed graphs [31]. One could think that by taking $p_n = \frac{k}{n}$, one would obtain a good approximation of the underlying structure of an automaton over a k-letter alphabet. In particular, the expected number of edges is kn. However, this is not really the case since random automata have a unique component with high probability according to the previous remark, whereas random graphs with $p_n = \frac{k}{n}$ do not.

4 Random Generation of Accessible Automata

As explained in the introduction, random generation is an important subfield of analysis of algorithms. Given a combinatorial class $\mathcal{C}$, the goal is to build an algorithm that efficiently generates elements of $\mathcal{C}$ of size n with the required distribution (usually the uniform distribution, in which all elements of size n have equal probability). Most basic structures, such as permutations or binary trees, have their own ad-hoc random generators. But researchers of this field also developed generic methods that translate combinatorial specifications into random samplers. They can directly be applied for objects like set partitions, partial injections, and so on. For more intricate structures, such as accessible automata, some additional work is usually required. But these general techniques form a guideline to design advanced generators, as we will see by addressing the following question:

> **Question 3:** *Is there an efficient algorithm to generate accessible automata with n states uniformly at random?*

A first idea could be to use a rejection algorithm: repeatedly generate a random automaton until it is accessible. Unfortunately, the probability p_n that an automaton is accessible is exponentially small. Since the number of iterations of this process follows a geometric law of parameter p_n, the average running time of this generator is exponential in n.

A second idea is to use the encoding into surjections of $[kn+1]$ to $[n]$ presented in Section 2. If we can generate efficiently such a surjection, then we can successfully use the idea of a rejection algorithm. Indeed, by Korshunov's result, the probability that a random surjection corresponds to an accessible automaton tends to a positive constant. The average number of rejections is therefore in $\mathcal{O}(1)$. Moreover, using appropriate data structures, building the automaton from the surjection and testing whether it is accessible can be done in linear time. Hence, the limiting factor of this approach is the efficient generation of a random surjection from $[kn+1]$ onto $[n]$.

Such a generator can be built using the *recursive method*, which has been introduced by Nijenhuis and Wilf [43] and developed by Flajolet, Zimmermann and Van Cutsem [26]. Let us illustrate this method on our example. Recall that $S(m,n)$ denote the number of surjections from $[m]$ onto $[n]$. In such a

surjection f we distinguish two cases, depending on whether $f(m)$ has one or more preimage by f (m is of course one of these preimages). If $f(m)$ has only one preimage, then the restriction of f to $[m-1]$ is a surjection onto $[n] \setminus \{f(m)\}$; since there are n choices for $f(m)$, there are $n\,S(m-1,n-1)$ such surjections. Similarly, there are $n\,S(m-1,n)$ surjections such that $f(m)$ has more than one preimage. Hence, adding the correct initial conditions, we have the recursive formula $S(m,n) = n\,S(m-1,n-1) + n\,S(m-1,n)$. The recursive method consists of two steps. First, all the values $S(i,j)$ are computed, for $i \in [kn+1]$ and $j \in [n]$. Then, we use the formula to build the surjection inductively: we randomly generate the image of m by f, then decide whether it has one or more preimage. It has one preimage with probability $\frac{nS(m-1,n-1)}{S(m,n)}$, in which case we remove $f(m)$ from the possible images. We then switch to $m-1$, and so on. The running time of the preprocess is $\mathcal{O}(n^2)$ and then each surjection is generated in linear time. But this analysis holds for a RAM model, where each arithmetic operation and each random choice is done in constant time. This is not realistic, since we saw that $S(kn+1,n)$ grows extremely fast. In practice, it is hard to generate accessible automata with more than a few thousand states using this method. We have to find a better solution.

Remark 11. Following [21], it is possible to use floating point approximations to avoid the computation and the storage of $\mathcal{O}(n^2)$ big integers. Assume that we have an approximate value $p_\approx$ of a probability p, with $|p - p_\approx| \leq \epsilon$. To draw a Bernoulli law of parameter p, we generate an element x of $[0,1]$. If $x < p_\approx - \epsilon$ we return 1, if $x > p_\approx + \epsilon$ we return 0, and if x is in the unknown zone, i.e., $|x - p_\approx| \leq \epsilon$, we compute a more precise estimation of p. This idea is classical in random generation but requires a careful implementation.

Remark 12. Instead of generating the surjections and rejecting those that are not *valid* (not associated to an accessible automaton), we can directly work on valid surjections. Indeed, valid surjections satisfies the same recurrence formula as surjections, but with different border conditions. This is the technique developed in [16, 41], using valid diagrams of Remark 5 instead of valid surjections.

Remark 13. The algorithm designed by Almeida, Moreira and Reis [2] is another example of using the recursive method for generating accessible automata. The encoding is different, as they use string representations for accessible automata.

Boltzmann samplers were introduced in [22] and quickly became very popular in the field. This is an elegant technique that proved very efficient in many situations. Boltzmann samplers are parameterized by a positive real number x. They do not generate objects of fixed size, and the value of x has to be tuned so that the expected size of a generated object is near n. However, two objects of the same size always have the same probability to be generated. If the distribution of sizes is concentrated enough, the algorithm produces objects of size around n with high probability. The main idea of this method is thus to allow variations on the sizes in order to obtain more efficient samplers.

In our setting, the Boltzmann samplers for surjections onto $[n]$ consists of the following steps. First generates the sizes $s_1, \ldots, s_n$ of the preimage of each element of $[n]$. Each preimage size is generated independently using a non-zero Poisson law[10] of parameter x. Then fill every preimage with elements of $[m]$ where $m = \sum_{i=1}^{n} s_i$. This can be done by generating a random permutation of $[m]$ once and taking the elements in that order. The law for the s_i's guarantees that we sample surjections onto $[n]$ following a Boltzmann distribution of parameter x. For x correctly chosen, the value of m is concentrated around $kn+1$. But we need m to be exactly $kn+1$ for our transformation into automata to doable. This can be achieved by doing an extra rejection step: if $m \neq kn+1$ we start the process again from the beginning. Every construction can be done in linear time, and it can be shown that the average number of rejections is in $\Theta(\sqrt{n})$. All in all, it results in an efficient random sampler for accessible automata of size n with an average running time in $\Theta(n^{3/2})$.

Remark 14. The correct value for the Boltzmann parameter x is the ρ_k of Remark 2. It also satisfies $\rho_k = k + W_0(-k\,e^{-k})$, where W_0 is the Lambert-W function. The series expansion of W_0 is $W_0(z) = \sum_{n\geq 1} \frac{(-n)^n}{n!} z^n$, which can be used to compute a good approximation of ρ_k, as $-k\,e^{-k}$ is small. This approximation is necessary for the algorithm, in order to generate the s_i's.

The third approach to random generation consists in using the result on the accessible part of a random automaton [13], which has been presented in Section 3. Recall that if $m \leq n$, then conditioned by its size m, the accessible part of a random automaton is a uniform accessible automaton with m states. Since the size of the accessible part is concentrated around $v_k\, n$, one can simply build a random sampler for size-m accessible automata by generating an automaton of size $\frac{n}{v_k}$ and extracting its accessible part. This is particularly efficient if we allow approximate sampling: if we use rejections until the resulting accessible automaton has size in $[(1-\epsilon)m, (1+\epsilon)m]$, the average running time is linear. To generate an accessible automaton of size exactly m, we use a rejection algorithm once again, and the average running time of the process is $\Theta(m^{3/2})$. It is therefore competitive with the previous method and much simpler to implement.

Remark 15. Computing v_k is not difficult, as $v_k = \frac{\rho_k}{k}$, where ρ_k can be approximated as explained in Remark 14.

The story is not over yet. In a recent preprint [8], Bassino and Sportiello presented a new method to achieve the random generation of surjections. It is based on the recursive method, mixed with the idea presented in Remark 11: probabilities are estimated more precisely only when needed. Using their method, the random generation of a surjection from $[kn+1]$ onto $[n]$ can be done in linear expected time, yielding a linear expected time algorithm for generating accessible automata. Remark that implementing completely this technique seems

[10] That is, $\mathbb{P}(s = i) = \frac{x^i}{i!(e^x-1)}$, for any $i \geq 1$.

to be quite challenging. We therefore choose to state the answer to Question 3 as follows.

> **Answer 3:** *Using simple algorithms, one can randomly generate accessible automata with n states in expected time $\Theta(n^{3/2})$, and accessible automata having between $(1-\epsilon)n$ and $(1+\epsilon)n$ states in linear expected time. Advanced techniques under development will soon allow the generation of accessible automata with n states in linear expected time.*

Remark 16. Some implementations of these algorithms are available, such as Regal [3] for the method using Boltzmann samplers and Fado [1] that uses the recursive method on string representations (see Remark 13). The algorithm that consists in extracting the accessible part of a random automaton can be easily implemented: the random generation of an automaton is elementary, the accessible part is computed using a depth-first traversal and a good evaluation of $v_k = \frac{\rho_k}{k}$ is obtained by truncating the series of Remark 14.

5 Proportion of Minimal Automata

Let $\mathcal{A} = (q_0, \delta, F)$ be an n-state automaton. For every state $p \in [n]$, let $\mathcal{L}_p(\mathcal{A}) = \{u \in A^* : \delta(p, u) \in F\}$, i.e., the words recognized when the initial state is moved to p. Two states p and q of $\mathcal{A}$ are *equivalent* when $\mathcal{L}_p(\mathcal{A}) = \mathcal{L}_q(\mathcal{A})$. We write $p \sim q$ when p and q are equivalent. An automaton is *minimal* when its states are pairwise non-equivalent. Minimal automata are important in automata theory. In particular, up to isomorphism, there is a unique minimal automata that recognizes a given regular language $\mathcal{L}$. Moreover, it is the deterministic automaton recognizing $\mathcal{L}$ that minimizes the number of states.

Experimentations done at the end of the nineties [41] suggested that the proportion of minimal automata amongst accessible automata is not negligible. This motivate the question of this section:

> **Question 4:** *What is the probability that a random accessible automaton is minimal?*

Bassino, David and Sportiello gave the answer to this question [6]. Their proof is complicated, we will just give the main ideas here. Two states p and q are *strongly equivalent* when both are in F or both are not in F, and for all $a \in A$, $\delta(p, a) = \delta(q, a)$. Clearly, if p and q are strongly equivalent then they are equivalent, and the automaton is not minimal.

The first step of their proof is to establish that if a random accessible automaton is not minimal, then with high probability it contains two states that are strongly equivalent. To do so, they used the technique we presented in Section 3 and proved it for random automata first. It is easier but still quite involved.

They then estimated precisely the probability of having no pair of strongly equivalent states. The critical case is for two-letter alphabets, for which this probability tends to a positive constant. Intuitively, in a random automaton, the probability that 4 given states p, p', q_a and q_b are such that $\delta(p, a) = \delta(p', a) = q_a$ and $\delta(p, b) = \delta(p', b) = q_b$ is $\frac{1}{n^4}$. Since there are $\Theta(n^4)$ choices for these 4

states, this indicates that there should be a positive probability that a random automaton is not minimal. Turning this intuition into a formal proof is difficult, especially since we have to deal with accessible automata. To do so, they use the diagram encoding depicted in Fig. 2. The main theorem of [6] is a very beautiful result, which answers Question 3:

> **Answer 4:** *If $|A| = 2$, then the probability that an accessible automaton is minimal tends to a computable constant $c_2 \in (0, 1)$. If $|A| \geq 3$, the probability that a random accessible automaton is minimal tends to* 1 *as the number of states tends to infinity.*

Remark 17. There is a related work of Berend and Kontorovich where they study the size of the minimal automaton of the language recognized by a random automaton [9]. They mostly rely on discrete probabilities to establish that when minimizing a random automaton, one obtains an automaton with $v_k n + \mathcal{O}(\sqrt{n} \log n)$ states with high probability.

Remark 18. Thanks to the answer to Question 4, the algorithms of Section 4 can be directly used to generate minimal automata with a given number of states. We just need to add a rejection step where we test whether the accessible automaton is minimal, and start again from the beginning if it is not. Testing minimality can be done in time $\mathcal{O}(n \log n)$ and the average number of rejections is bounded. The average running time is therefore $\mathcal{O}(n^{3/2})$ or $\mathcal{O}(n \log n)$ depending on the algorithm used for generating accessible automata.

6 Average Case Analysis of Minimization Algorithms

A minimization algorithm computes the minimal automaton of a regular language, which is usually given by an automaton. Minimal automata are important in automata theory, and there is a rich literature on the topic, with many algorithms, experimental studies, worst-case running time analysis, and so on. The best known solution to the minimization problem is Hopcroft's algorithm [29], whose worst-case running time is $\mathcal{O}(n \log n)$. This algorithm can be viewed as a tight optimization of Moore's algorithm [39], whose worst-case running time is $\mathcal{O}(n^2)$. Amongst the many other solutions, the most famous one is probably Brzozowski's algorithm [12], which is based on a different idea[11] and which also works if the input is a non-deterministic automaton. However, the running time of this elegant algorithm is exponential in the worst case, even for deterministic automata.[12]

Despite its quadratic worst-case running time, authors of libraries that implements classical algorithms for automata[13] noticed that Moore's minimization

[11] This is not entirely true, there are works that explain how it is related to the other minimization algorithms [15].

[12] For non-deterministic automata, the exponential blow up cannot be prevented in the worst case.

[13] Such as Vaucanson [38].

algorithm behaves well in practice. This motivates the question of this section, which is the following.

> **Question 5:** *What is the average running time of Moore's minimization algorithm?*

Recall that two states p and q of an automaton are equivalent when the languages $\mathcal{L}_p$ and $\mathcal{L}_q$ are equal, where $\mathcal{L}_p$ is the language recognized if the initial state is moved to p. If p is a state and $\ell \geq 0$ is an integer, let $\mathcal{L}_p^{\leq \ell} = \mathcal{L}_p \cap A^{\leq \ell}$ denote the set of words of length at most ℓ that are in $\mathcal{L}_p$. Two states p and q are *ℓ-equivalent*, $p \sim_\ell q$, when $\mathcal{L}_p^{\leq \ell} = \mathcal{L}_q^{\leq \ell}$. It can be shown that if $\sim_\ell = \sim_{\ell+1}$, then $\sim_j = \sim$ for every $j \geq \ell$. Moreover, in an n-state automaton we always have $\sim_{n-2} = \sim$. Based on this facts, Moore's algorithm iteratively computes $\sim_0$, $\sim_1$, ... until $\sim_\ell = \sim_{\ell-1}$. Using appropriate data structures, each iteration can be done in linear time. Hence, the running time of the algorithm is $\mathcal{O}(n\ell)$, where ℓ is the number of iterations, that is, the smallest ℓ such that $\sim_\ell = \sim_{\ell-1}$. The minimal automaton is then built by merging states that are in the same equivalence class.

Let $\mathcal{A}$ be an automaton such that $\sim_\ell \neq \sim_{\ell-1}$ and $\sim_{\ell+1} = \sim_\ell$ for some given $\ell \geq 1$. Then there exists two states p and q that are distinguished after ℓ iterations, but not before: $p \sim_{\ell-1} q$ and there exists a word u of length ℓ such that $u \in \mathcal{L}_p$ and $u \notin \mathcal{L}_q$ (or $u \notin \mathcal{L}_p$ and $u \in \mathcal{L}_q$). Thus for every prefix v of u that is not equal to u, $\delta(p, v)$ and $\delta(q, v)$ are both final or both not final. Let G be the undirected graph whose vertices are the states of $\mathcal{A}$ and with an edge between $\delta(p, v)$ and $\delta(q, v)$ for any prefix v of u that is not equal to u. In the conference[14] paper [4], Bassino, David and the author proved that this graph is acyclic and has exactly ℓ edges. Observe that in a connect component of this graph, either all the states are final or none of them is final. But for fixed p, q and u, the graph only depends on the transition structure of $\mathcal{A}$: if we randomly choose the set of final states, the probability that the connected components satisfy the property is $2^{-\ell}$. For a good choice of $\ell \in \Theta(\log n)$, this proves that the average running time of Moore's algorithm is in $\mathcal{O}(n \log n)$.

Importantly, the proof we just sketched does not depend on the shape of the automaton: if we consider a probabilistic model where a transition structure with n states is chosen following *any* distribution and then a set of final states is added uniformly at random, then the result still holds. In particular it holds for subclasses of automata such as acyclic automata, group automata, and so on. David [18] proved that for the uniform distribution of automata, the average running time of Moore's algorithm is in $\Theta(n \log \log n)$. The proof is too involved to be presented here, but this gives the answer to Question 5:

> **Answer 5:** *For the uniform distribution, the average running time of Moore's algorithm is in $\mathcal{O}(n \log \log n)$. For any distribution on transition structures, if final states are added uniformly at random then the average running time of Moore's algorithm is in $\mathcal{O}(n \log n)$.*

[14] In the journal version [5] there is no reference to this graph, the proof is done on partitions directly.

Remark 19. The proofs of [5, 18] also work if each state is final with fixed probability $p \in (0, 1)$ independently. David's result, though stated for random automata, is still valid for the uniform distribution on accessible automata, using the technique presented in Section 3 (see [6, 13]).

Remark 20. The proofs do not work for distributions with few final states, such as the uniform distribution on accessible automata with exactly one final state. See Section 7 for a discussion on such models.

Remark 21. Hopcroft's algorithm maintains a set of tasks to be performed. This set can be implemented in many ways (queue, stack, ...) without affecting its worst-case running time. David [18] proposed a structure for this set of tasks that guarantees that the algorithm performs at least as well as Moore's algorithm. Hence, using this implementation for the set of tasks, the average running time of Hopcroft's algorithm satisfies the same bounds as those stated in Answer 5 for Moore's algorithm.

Remark 22. De Felice and the author proved that not only Brzozowski's algorithm is inefficient in the worst case, but also that its running time is super-polynomial[15] with high probability [19]. It is thus also super-polynomial on average. This result uses a famous theorem of Erdős and Túran, which states that the order of a random permutation of $[n]$ is super-polynomial with high probability.[16]

7 Recent Results, Ongoing Works and Open Questions

An automaton is *synchronizing* when there exists a word that maps every state to the same state. Such a word is called a *synchronizing word*. In 1964, Černý [14] gave a family of synchronizing n-state automata whose smallest synchronizing word has length $(n-1)^2$ and asked whether this bound is tight: does every synchronizing n-state automaton admit a synchronizing word of length at most $(n-1)^2$? The question, now known as the Černý conjecture, is still open and is one of the most famous conjecture in automata theory.

The probabilistic version of this conjecture is to ask whether random automata are often synchronizing, and whether the Černý conjecture holds with high probability. In a recent preprint [10], Berlinkov proved that a random automaton is synchronizing with probability $1 - \Theta(\frac{1}{n})$ on a two-letter alphabet. This is a deep and difficult result, which was expected for quite some time, since simulations clearly shows that automata are synchronizing with high probability. Berlinkov's proof is based on classical techniques developed around the Černý conjecture and uses advanced properties of random maps from $[n]$ to $[n]$, following the approach of [34]. In another preprint [42], the author uses a different

[15] i.e., grows faster that any polynomial in n.

[16] Their result is much more precise, giving a limit law for the random variable $\log O_n$, where O_n is the order of a random permutation of $[n]$.

method to establish that with high probability[17] there exists a synchronizing word of length $\mathcal{O}(n^{1+\epsilon})$, for any positive ϵ. Hence, the Černý conjecture holds with high probability. There is still room for sharper results in this direction, as experimentations [32] seem to indicate that the expected length of the smallest synchronizing word grows in $\sqrt{n}$.

Automata taken uniformly at random tends to have too many final states, as in practice automata with few final states are not uncommon. Unfortunately, as stated in Remark 20, most results presented in this article cannot be adapted to automata with, say, one final state. The only exception is the recent analysis of Brzozowski's algorithm for that kind of distributions [20]. One of the interesting open questions here is to confirm experimental studies that indicates that a random automaton with one final state should be minimal with non-negligible probability.

An other series of questions that is widely open is the study of the average state complexity of the classical operations on regular languages. The *state complexity* of a regular language is the number of states of its minimal automaton. A typical question in this area is "What is the average state complexity of the intersection of two languages of state complexities n?". The only known results are for unary alphabets [40] and for the reverse operation [19].

In this article, we presented different results about random deterministic automata. It is natural to try to answer the same kind of questions for non-deterministic automata too. Unfortunately, it is quite challenging to define distributions on non-deterministic automata that are both meaningful and mathematically tractable. For instance, a non-deterministic automaton taken uniformly at random recognizes all the words with high probability. The uniform model is therefore not relevant. Proving formally the experimental results such as those presented in [44], where non-deterministic automata are drawn under the Erdős-Rényi model for random graphs [11], would be an important first step in the analysis of the typical properties of random non-deterministic automata.

Acknowledgments. The author would like to thank Arnaud Carayol for his precious help during the preparation of this article.

References

1. A. Almeida, M. Almeida, J. Alves, N. Moreira, and R. Reis. Fado and guitar. In S. Maneth, editor, *CIAA'09*, volume 5642 of *LNCS*, pages 65–74. Springer, 2009.
2. M. Almeida, N. Moreira, and R. Reis. Enumeration and generation with a string automata representation. *Theor. Comput. Sci.*, 387(2):93–102, 2007.
3. F. Bassino, J. David, and C. Nicaud. REGAL: A library to randomly and exhaustively generate automata. In J. Holub and J. Zdárek, editors, *CIAA'07*, volume 4783 of *LNCS*, pages 303–305. Springer, 2007.

[17] Contrarily to Berlinkov's result, there is no tight bound on the error term with this technique.

4. F. Bassino, J. David, and C. Nicaud. On the average complexity of Moore's state minimization algorithm. In S. Albers and J.-Y. Marion, editors, *STACS'09*, volume 3 of *LIPIcs*, pages 123–134. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.
5. F. Bassino, J. David, and C. Nicaud. Average case analysis of Moore's state minimization algorithm. *Algorithmica*, 63(1-2):509–531, 2012.
6. F. Bassino, J. David, and A. Sportiello. Asymptotic enumeration of minimal automata. In C. Dürr and T. Wilke, editors, *STACS'12*, volume 14 of *LIPIcs*, pages 88–99. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.
7. F. Bassino and C. Nicaud. Enumeration and random generation of accessible automata. *Theor. Comput. Sci.*, 381(1-3):86–104, 2007.
8. F. Bassino and A. Sportiello. Linear-time generation of specifiable combinatorial structures: general theory and first examples. *arXiv*, abs/1307.1728, 2013.
9. D. Berend and A. Kontorovich. The state complexity of random DFAs. *arXiv*, abs/1307.0720, 2013.
10. M. V. Berlinkov. On the probability to be synchronizable. *arXiv*, abs/1304.5774, 2013.
11. B. Bollobás. *Random Graphs*. Cambridge University Press, 2001.
12. J. A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. In *Mathematical theory of Automata*, pages 529–561. Polytechnic Press, Polyt. Instit. of Brooklyn, N.Y., 1962. Volume 12 of MRI Symposia Series.
13. A. Carayol and C. Nicaud. Distribution of the number of accessible states in a random deterministic automaton. In C. Dürr and T. Wilke, editors, *STACS'12*, volume 14 of *LIPIcs*, pages 194–205. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.
14. J. Černý. Poznámka k. homogénnym experimentom s konecnymi automatmi. *Matematicko-fyzikalny Časopis Slovensk*, 14, 1964.
15. J.-M. Champarnaud, A. Khorsi, and T. Paranthoën. Split and join for minimizing: Brzozowski's algorithm. In M. Balík and M. Simánek, editors, *Stringology'02*, pages 96–104, 2002.
16. J.-M. Champarnaud and T. Paranthoën. Random generation of DFAs. *Theor. Comput. Sci.*, 330(2):221–235, 2005.
17. M. Crochemore and W. Rytter. *Text Algorithms*. Oxford University Press, 1994.
18. J. David. Average complexity of Moore's and Hopcroft's algorithms. *Theor. Comput. Sci.*, 417:50–65, 2012.
19. S. De Felice and C. Nicaud. Brzozowski algorithm is generically super-polynomial for deterministic automata. In M.-P. Béal and O. Carton, editors, *DLT'13*, volume 7907 of *LNCS*, pages 179–190. Springer, 2013.
20. S. De Felice and C. Nicaud. On the average complexity of Brzozowski's algorithm for deterministic automata with a small number of final states. In *DLT'14*, 2014. to appear in LNCS.
21. A. Denise and P. Zimmermann. Uniform random generation of decomposable structures using floating-point arithmetic. *Theor. Comput. Sci.*, 218(2):233–248, 1999.
22. P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Comb. Prob. Comp.*, 13(4-5):577–625, 2004.
23. W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 1968.

24. P. Flajolet and A. M. Odlyzko. Random mapping statistics. In J.-J. Quisquater and J. Vandewalle, editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 329–354. Springer, 1989.
25. P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.
26. P. Flajolet, P. Zimmermann, and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theor. Comput. Sci.*, 132(2):1–35, 1994.
27. I. J. Good. An asymptotic formula for the differences of the powers at zero. *Ann. Math. Statist.*, 32:249–256, 1961.
28. M. Harrison. A census of finite automata. *Canadian J. Math.*, 17:100–113, 1965.
29. J. E. Hopcroft. An $n \log n$ algorithm for minimizing the states in a finite automaton. In Z. Kohavi, editor, *The Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.
30. J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Co., Reading, Mass., 1979. Addison-Wesley Series in Computer Science.
31. R. M. Karp. The transitive closure of a random digraph. *Random Struct. Algorithms*, 1(1):73–94, 1990.
32. A. Kisielewicz, J. Kowalski, and M. Szykula. A fast algorithm finding the shortest reset words. In D.-Z. Du and G. Zhang, editors, *COCOON'13*, volume 7936 of *LNCS*, pages 182–196. Springer, 2013.
33. D. E. Knuth. *The Art of Computer Programming, Volume I: Fundamental Algorithms, 3rd Edition*. Addison-Wesley, 1997.
34. V. Kolčin. *Random Mappings: Translation Series in Mathematics and Engineering*. Translations series in mathematics and engineering. Springer London, Limited, 1986.
35. A. Korshunov. Enumeration of finite automata. *Problemy Kibernetiki*, 34:5–82, 1978. In russian.
36. E. Lebensztayn. On the asymptotic enumeration of accessible automata. *DMTCS*, 12(3), 2010.
37. V. A. Liskovets. The number of initially connected automata. *Cybernetics*, 4:259–262, 1969. English translation of Kibernetika (3) 1969, 16-19.
38. S. Lombardy, Y. Régis-Gianas, and J. Sakarovitch. Introducing VAUCANSON. *Theor. Comput. Sci.*, 328(1-2):77–96, 2004.
39. E. F. Moore. Gedanken experiments on sequential machines. In *Automata Studies*, pages 129–153. Princeton U., 1956.
40. C. Nicaud. Average state complexity of operations on unary automata. In M. Kutylowski, L. Pacholski, and T. Wierzbicki, editors, *MFCS'99*, volume 1672 of *LNCS*, pages 231–240. Springer, 1999.
41. C. Nicaud. Comportement en moyenne des automates finis et des langages rationnels. *PhD Thesis, Université Paris VII*, 2000. In french.
42. C. Nicaud. Fast synchronization of random automata. *arXiv*, abs/1404.6962, 2014.
43. A. Nijenhuis and H. S. Wilf. *Combinatorial Algorithms*. Academic Press, 1978.
44. D. Tabakov and M. Y. Vardi. Experimental evaluation of classical automata constructions. In G. Sutcliffe and A. Voronkov, editors, *LPAR'05*, volume 3835 of *LNCS*, pages 396–411. Springer, 2005.