Sampling different kinds of acyclic automata using Markov chains

Vincent Carnino, Sven de Felice

Sampling Different Kinds of Acyclic Automata Using Markov Chains

Vincent Carnino[a], Sven De Felice[a,*]

[a] *LIGM, UMR 8049, Université Paris-Est et CNRS, 77454 Marne-la-Vallée, France.*

Abstract

We propose algorithms that use Markov chain techniques to generate acyclic automata uniformly at random. We first consider deterministic, accessible and acyclic automata, then focus on the class of minimal acyclic automata. In each case we explain how to define random local transformations that describe an ergodic and symmetric Markov chain: the distribution of the automaton obtained after T random steps in this Markov chain tends to the uniform distribution as T tends to infinity.

1. Introduction

In language theory, acyclic automata are exactly the automata that recognize finite languages. For this reason, they play an important role in some specific fields of applications, such as the treatment of natural languages. From an algorithmic point of view, they often enjoy more efficient solutions than general automata; a famous example is the linear minimization algorithm proposed by Revuz for deterministic acyclic automata [17]. They also appear as first steps in some algorithms, two examples of which are related to Glushkov construction [3, 4, 5] and an extension of Aho-Corasick automaton [16].

In the design and analysis of algorithms it is of great use to have access to exhaustive and random generators for the inputs of the algorithm one wants to study: the exhaustive generator is used to analyze the behavior of the algorithm for small inputs, but cannot be used for large inputs since there are too many of them; the number of size-n inputs typically grows at least exponentially in n. Those generators can be used either to test the correctness and the efficiency of an implementation, or to help the researcher while establishing theoretical results about the average case analysis of the algorithm.

An exhaustive generator for minimal deterministic acyclic automata has been given by Almeida, Moreira and Reis [1], and in this paper we propose an algorithm to generate at random deterministic, accessible and acyclic automata

*The second author was supported by ANR MAGNUM - project ANR-2010-BLAN-0204.

Email addresses: vcarnino@univ-mlv.fr (Vincent Carnino), defelic@univ-mlv.fr (Sven De Felice)

and minimal acyclic automata, with a distribution that is almost uniform, using Markov chain techniques. The idea is to start with a n-state acyclic automaton (or minimal acyclic automaton), then to perform a certain amount T of mutations of this automaton, a mutation being a small local transformation that preserves the required properties (deterministic, accessible and acyclic with the same number of states, or minimal with the same number of states). Since each mutation is performed in time $O(n)$, the complexity of our algorithm is $O(nT)$. The bigger T, faster the output distribution approaches the uniform distribution. For a given distance to uniformity, it is a generally difficult problem to give a good estimation of a corresponding value of T; this is directly related to the *mixing time* of the Markov chain, which is generally a difficult problem [11]. We do not address this problem in this article, but the simulations that we performed seem to indicate that a choice of T polynomial in n gives a correct random generator, at least for most applications.

Note that the other generic methods to generate combinatorial structures uniformly at random seem to fail here. For instance, recursive methods [8] or Boltzmann samplers [7], which have been used for deterministic automata [6, 2, 9], rely on a good recursive description of the input, which is not known for acyclic automata. To our knowledge, the only combinatorial result on acyclic automata is due to Liskovets [13], who gave a closed-form formula for the number of acyclic automata; unfortunately this formula cannot be directly translate into a tractable recursive description.

Related work: as mentioned above, our algorithm is a complement of the exhaustive generator of Almeida, Moreira and Reis [1] for testing conjectures and algorithms based on deterministic acyclic automata or minimal acyclic automata. The idea of using Markov chains for that kind of objects starts with works on acyclic graphs, which has been done for graph visualization purposes [14, 15]. Though using the same general idea, deterministic acyclic automata do not resemble acyclic graphs that much, mainly because they only have a linear number of edges (transitions). In particular, the diameter of the Markov chain, which is a lower bound for the mixing time, is quadratic for acyclic graphs but linear in our case. Moreover, automata considered in this article must be accessible, which is not a natural condition for graphs (there is no notion of distinguished initial vertex); Melançon and Philippe considered simply connected acyclic graphs in [15], but this is not the same notion as accessibility. For instance, they use a nice optimization based on reversing an edge, which preserves connectedness but not accessibility; hence it cannot be used here.

The paper is organized as follows. In Section 2, we recall basic notations about automata; and in Section 3 classical Markov chain concepts are detailed. The generator of acyclic automata is described in Section 4, and its correctness is given in Section 5. The algorithm on minimal acyclic automata is described in Section 6, and its correctness is given in Section 7. Finally, in Section 8, we explain how to adapt the second algorithm to handle the case of generating minimal acyclic automata with some constraints on the set of final states.

2. Definitions and Notations

For more information about classical automata theory, the reader is referred to the book of Hopcroft and Ullman [10].

2.1. Automata

Throughout this paper, a *deterministic finite automaton* is a tuple $\mathcal{A} = (Q, A, \delta, \{i_0\}, F)$, where Q is a finite set of *states*, A is a finite set of *letters* called the *alphabet*, $\delta : Q \times A \to Q$ is the (partial) *transition function*, $i_0 \in Q$ is the *initial state* and $F \subseteq Q$ is the set of *final states*.

If p is a state of $\mathcal{A}$ we denote by A_p the set of letters $a \in A$ such that $\delta(p, a)$ is defined. For any state $q \in Q$, the transition function $\delta(q, \cdot)$ is inductively extended to the set A^* of all finite words over A: $\delta(q, \varepsilon) = q$, where ε is the *empty word*, and for all $w \in A^*$ such that $w = w_1 w_2 \dots w_n$, then $\delta(q, w) := \delta(\delta(\dots \delta(\delta(q, w_1), w_2) \dots), w_n)$, when each of them is defined, and is undefined otherwise.

A state $q \in Q$ is *accessible* (resp. *co-accessible*) when there exists $w \in A^*$ such that $\delta(i_0, w) = q$ (resp. $\delta(q, w) \in F$). An automaton is *accessible* (resp. *co-accessible*) when all its states are accessible (resp. *co-accessible*). An automaton is *trim* when it is both accessible and co-accessible.

A state $q \in Q$ is *transient* if for all $w \in A^+$, $\delta(q, w) \neq q$. A state that is not transient is called *recurrent*. An automaton is *acyclic* when every state is transient. Another definition of acyclic automata is that the underlying directed graph is an acyclic graph. Note that it is impossible for a complete automaton to be acyclic.

For a given acyclic automaton, a *path* from a state q to a state p is a word w such as $\delta(q, w) = p$. The *length* of a path w is its size. If there exists a path from a state q to a state p and $p \neq q$, we call p an *ancestor* of q and q a *successor* of p. If there is a path of length 1 from p to q, we say that p is a *direct ancestor* of q and that q is a *direct successor* of p.

In the sequel, without loss of generality, the set of states Q of an n-state deterministic automaton will always be $\{1, \dots, n\}$ and 1 will always be the initial state. The *size* of an automaton is its number of states, and we furthermore assume from now on that $n \geq 2$. Moreover, since we always consider deterministic, accessible and acyclic automata in this article, we shall just denote them by "acyclic automata" for short. The set of all n-state acyclic automata is denoted by $\mathbb{A}_n$. We also assume from now on that $|A| \geq 2$, the case $|A| = 1$ being trivial for the questions considered in this article.

2.2. Hammock automata

The notion of Hammock automata will be specially useful in the second part of the paper when we focus on minimal acyclic automata.

Definition 1. An acyclic automaton $\mathcal{A}$ is called a *hammock* automaton if it has exactly one state with no outcoming transition, called the *target state*.

Proposition 1. *Let $\mathcal{A} = (Q, A, \delta, \{1\}, F)$ be a hammock automaton and let $s \in Q$ be a state of $\mathcal{A}$. Then there exists a path from s to the target state.*

Proof. Since $\mathcal{A}$ is acyclic any path of $\mathcal{A}$ is upper bounded by $n-1$. Let t denote the target state. Let w be a path such that $\delta(s, w)$ is defined and such the length of w is maximal amongst the paths starting at s. By contradiction suppose that $\delta(s, w) = r \neq t$. Then r has a transition labeled by a letter a. Thus $\delta(s, wa)$ is defined and $|wa| = |w| + 1 > |w|$. That is impossible since the length of w is maximal amongst the paths starting at s. Then $\delta(s, w) = t$. □

2.3. Minimal automata

The *Nerode equivalence* is the equivalence relation $\sim$ defined on the states of a given automaton $\mathcal{A}$ as follow:

$$p \sim q \iff \mathrm{Fut}_{\mathcal{A}}(p) = \mathrm{Fut}_{\mathcal{A}}(q),$$

where $\mathrm{Fut}_{\mathcal{A}}(p) = \{w \in A^* \mid \delta(p, w) \in F\}$ denotes the *future* of a state p.

If an automaton is deterministic this equivalence satisfies the following property.

$$p \sim q \iff \begin{cases} p \in F \iff q \in F, \\ A_p = A_q, \\ \forall a \in A_p,\ \delta(p, a) \sim \delta(p, b). \end{cases}$$

A deterministic automaton is *minimal* if it is trim[1] and every equivalence class is reduced to a singleton. Every regular language $\mathcal{L}$ is recognized by a unique minimal automaton (up to labelling), which is called the minimal automaton of $\mathcal{L}$, and which is the smallest deterministic automaton that recognizes $\mathcal{L}$. The classical way to define minimal automata is to use quotients, see [10] for more details.

The Nerode equivalence can be used to compute the minimal automaton of $\mathcal{L}$ starting from a deterministic automaton that recognizes $\mathcal{L}$: repeatedly merge two states that are equivalent until the equivalence relation is the equality. The automaton obtained is exactly the minimal automaton of the language, and most minimization algorithms proceed this way.

As we shall see in Section 7, minimal acyclic automata are always hammock automata. It is convenient, as we asked for the initial state to be labelled by 1, to choose a fixed label for it target state too: in the sequel we denote by $\mathbb{M}_n$ the set of all minimal acyclic automata with n states, whose initial state is 1 and whose target state is n.

[1] In the literature the minimal automaton is either defined as a complete automaton or as a trim automaton. We use the later definition here.

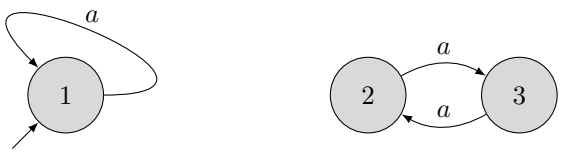

Figure 1: A deterministic automaton which is not accessible. Note that its group of automorphism is not trivial: if we permute the label 2 with the label 3 we obtain the same automaton.

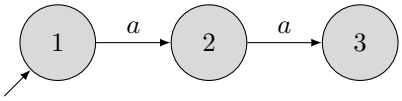

Figure 2: An accessible deterministic automaton. Unlike the previous automaton, any non trivial permutation of the label leads to a different automaton.

2.4. Automorphism of automata

Let $\mathcal{A} = (Q, A, \delta, \{1\}, F)$ be a deterministic automata. A one-to-one mapping ϕ from Q to Q is an *automorphism* when

- $\phi(1) = 1$,
- $\forall p \in Q$, $A_p = A_{\phi(p)}$,
- $\forall p \in Q$, $\forall a \in A_p$, $\delta(\phi(p), a) = \phi(\delta(p, a))$,
- $\forall p \in Q$, $p \in F \iff \phi(p) \in F$.

As remarked by Liskovets [12], the only automorphism of an accessible and deterministic automaton is the identity. This has the important combinatorial consequence that there are exactly $(n-1)!$ distinct ways of labelling with $\{1, \ldots, n\}$ the states of such a size-n automaton, with the convention that 1 is the initial state. In particular, for a family $\mathcal{F}_n$ of size-n accessible and deterministic automata, stable by relabelling, the process of generating uniformly at random a labelled element of $\mathcal{F}_n$, then removing the labels (formally: considering its equivalence class up to labelling) introduce no bias in uniformity.

For elements of $\mathbb{M}_n$, since we required that 1 is the initial state and n is the target state, the same result holds, but there are now exactly $(n-2)!$ elements in each equivalence class, up to labelling.

3. Markov Chains and Random Generation

In this section we describe the Markov chain used for generating almost uniformly at random elements of $\mathbb{A}_n$, for any fixed $n \geq 2$. In the process, we

recall the basic notion of Markov chain that we shall need in the sequel. More information on Markov chain for random generation can be found in [11].

The input of the algorithm consists of two positive integers: the number of states n, and the number of iterations T. The algorithm relies on a *Markov chain* process: it randomly moves in the set $\mathbb{A}_n$ and returns the automaton reached after T steps.

The Markov chain of the algorithm can be seen as a directed graph whose vertices are elements of $\mathbb{A}_n$. An edge from an automaton $\mathcal{A}$ to another automaton $\mathcal{B}$ is labelled by a real $r \in [0, 1]$, which represents the probability to move from automaton $\mathcal{A}$ to automaton $\mathcal{B}$ in one step. For two automata $\mathcal{A}, \mathcal{B} \in \mathbb{A}_n$ we denote by $\mathcal{P}_{\mathcal{A},\mathcal{B}}$ the label of the edge from $\mathcal{A}$ to $\mathcal{B}$, if it exists, otherwise we set $\mathcal{P}_{\mathcal{A},\mathcal{B}} = 0$. Since it is a probability, we have:

$$\forall \mathcal{A} \in \mathbb{A}_n, \sum_{\mathcal{B} \in \mathbb{A}_n} \mathcal{P}_{\mathcal{A},\mathcal{B}} = 1.$$

A distribution on $\mathbb{A}_n$ is a mapping p from $\mathbb{A}_n$ to $[0, 1]$ such that $\sum_{\mathcal{A} \in \mathbb{A}_n} p(\mathcal{A}) = 1$. A *stationary distribution* π *of a Markov chain* is a distribution that remains globally unchanged after each random move, that is,

$$\forall \mathcal{B} \in \mathbb{A}_n, \pi(\mathcal{B}) = \sum_{\mathcal{A} \in \mathbb{A}_n} \pi(\mathcal{A}) \times \mathcal{P}_{\mathcal{A},\mathcal{B}}.$$

A Markov Chain is called *irreducible* when its graph is strongly connected. For $i \in \mathbb{N}$, let $\mathcal{P}^{(i)}_{\mathcal{A},\mathcal{B}}$ be the probability to move from $\mathcal{A}$ to $\mathcal{B}$ in i steps of the algorithm. We define the *period* of a vertex $\mathcal{A}$ as the gcd of the lengths of all circuits on $\mathcal{A}$: $gcd(\{i \in \mathbb{N} \mid \mathcal{P}^{(i)}_{\mathcal{A},\mathcal{A}} > 0\})$. If there is a loop of length 1 on $\mathcal{A}$, the period of $\mathcal{A}$ is 1 by definition. A vertex is *aperiodic* if its period is 1. A Markov chain is *aperiodic* when all its states are aperiodic. A Markov chain is *ergodic* when it is both irreducible and aperiodic.

A famous property of ergodic Markov chains with a finite number of vertices is that they have a unique stationary distribution and that starting at any vertex the distribution obtained after T steps tends to this stationary distribution as T tends to infinity [11]. This gives a general framework to build a random generator on a non-empty finite set E: design an ergodic Markov chain whose set of vertices is E and such that the stationary distribution is the uniform distribution. Start from any vertex, then move randomly for a long enough time to obtain a random element of E almost uniformly.

This is exactly what we do in this article. A small part of the Markov chain that is behind our algorithm is depicted in Figure 3. Each step consists either in doing nothing or in making a transition. The complete description of the algorithm is done in Section 4. Our main result, which is proved in Section 5 is the following:

Theorem 1. *For any $n \geq 2$, the Markov chain for $\mathbb{A}_n$ is ergodic and its stationary distribution is the uniform distribution.*

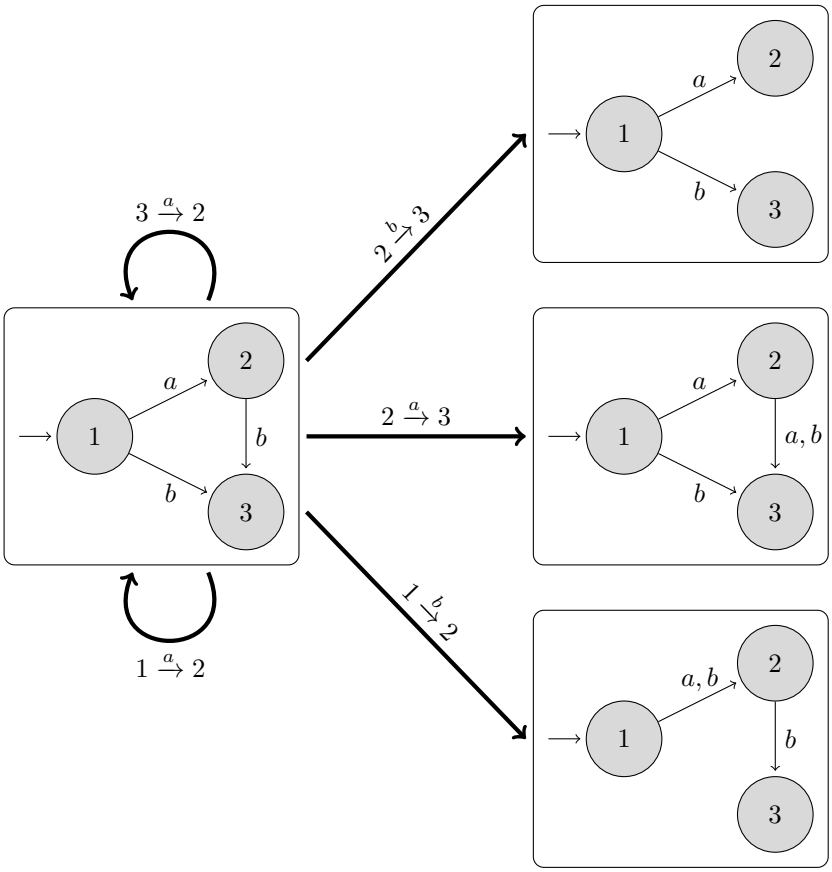

Figure 3: Part of the Markov chain: at each iteration an element $p \xrightarrow{a} q$ of $Q \times A \times Q$ is chosen randomly. If it corresponds to a transition of the automaton, as $2 \xrightarrow{b} 3$, then it is removed. If there is no transition labelled by a and starting at p, it is added; this is the case for $2 \xrightarrow{a} 3$. When there already is a transition labelled by a and starting at p, it is redirected to q; this is the case for $1 \xrightarrow{b} 2$. The mutation is not done if the automaton is not acyclic anymore ($3 \xrightarrow{a} 2$) or if it is not accessible anymore ($1 \xrightarrow{a} 2$).

Since the isomorphism classes of n-state automata have the same cardinality, our uniform random generator on $\mathbb{A}_n$ yields a generator on isomorphic classes of automata which is also uniform (see Section 2.4).

Note that the number of iterations T must be large enough in order to approach closely the uniform distribution. The choice of T is a difficult problem [11] and it is not cover in this paper. The diameter of the Markov chain's graph is a lower bound for T, and we will show in Section 5 that this diameter is linear in our case.

4. Algorithm for acyclic automata

```
AcyclicAutomatonGeneration(n, T)
1  A ← any deterministic, accessible and acyclic automaton with n states
2  i ← 0
3  while i < T do
4      p ← Uniform(Q), a ← Uniform(A), q ← Uniform(Q \ {p})
5      if δ(p, a) is undefined then
6          if IsAcyclic(A ⊕ p -a-> q) then A = A ⊕ p -a-> q
7      else if δ(p, a) = q then
8          if IsAccessible(A ⊖ p -a-> q) then A = A ⊖ p -a-> q
9      else
10         r ← δ(p, a)
11         if IsAccessible(A ⊖ p -a-> r) then
12             A = A ⊖ p -a-> r
13             if IsAcyclic(A ⊕ p -a-> q) then
14                 A = A ⊕ p -a-> q
15             else
16                 A = A ⊕ p -a-> r
17     i ← i + 1
18 Randomly choose the set of final states of A
19 return A
```

We represent a transition $\delta(p, a) = q$, with $(p, q) \in Q^2$ and $a \in A$, by $p \xrightarrow{a} q$. The notation $\mathcal{A} \oplus p \xrightarrow{a} q$ represents the automaton $\mathcal{A}$ with the additional transition $p \xrightarrow{a} q$. Similarly, the notation $\mathcal{A} \ominus p \xrightarrow{a} q$ represents the automaton $\mathcal{A}$ where the transition $p \xrightarrow{a} q$ has been removed, if it exists.

The algorithm has two arguments: the number n of states and a value T which indicates the desired number of iterations (it is quite difficult to know when the uniform distribution is reached so it is convenient to specify it). After choosing any acyclic automaton $\mathcal{A} \in \mathbb{A}_n$ to start with, the algorithm repeats the following steps T times: choose uniformly a labelled edge $p \xrightarrow{a} q$ with $p \neq q$ ($p = q$ is not interesting since we are considering acyclic automata). Then there are three possible cases:

- There is no transition starting from p and labelled with a. In such a case, we try to add $p \xrightarrow{a} q$ to $\mathcal{A}$ and test if it is still acyclic. The transition is added only if it is.
- There already is a transition $p \xrightarrow{a} q$ in $\mathcal{A}$. In that case, we test if $\mathcal{A}$ is still accessible if we remove it. If it is, the transition is removed, else $\mathcal{A}$ remains unchanged.
- There is a transition starting from p, labelled with a and reaching a state r, with $r \neq q$. In this last case, we first test whether $\mathcal{A}$ is still accessible if we redirect $\delta(p, a)$ to q. If it is, we do the redirection, otherwise $\mathcal{A}$ remains unchanged.

In this process, we need to check regularly the accessibility and the acyclicity of $\mathcal{A}$.

The accessibility test is implemented the following way. We keep up-to-date, for each state q, a counter that indicates the total number of transitions ending in q. Each time we add or remove such a transition, this counter is increased or decreased. Thus, to test the accessibility, we just have to check, after the transition has been removed, whether the counter on the state that ends the transition reaches 0 or not; this is a consequence of Lemma 1 (see Section 5). It clearly has a $O(1)$ time complexity.

The acyclicity is tested by the classical algorithm, using a depth-first-search algorithm which runs in time $O(n)$, since the number of transitions is linear in a deterministic automaton.

We therefore get the following result.

Proposition 2. *Each iteration of the algorithm is performed in time $O(n)$. The worst case time complexity of the algorithm is $O(Tn)$ and its space complexity is $O(n).$*

5. Proofs

In this section, we prove the main facts that are used for the first algorithm to correctly generate an acyclic automaton with almost uniform distribution, and with the announced complexity.

An operation which consists of removing, adding or changing a transition is called an *elementary operation.*

Lemma 1. *Let $\mathcal{A}$ be an acyclic automaton of size n and $\mathcal{B} = \mathcal{A} \ominus p \xrightarrow{a} q$, where $q \neq 1$ and $p \xrightarrow{a} q$ is any transition of $\mathcal{A}$. If there is at least one transition that ends in q in $\mathcal{B}$ then $\mathcal{B}$ is accessible.*

Proof. $\mathcal{B}$ is clearly acyclic. $\mathcal{B}$ has a transition $r \xrightarrow{b} q$, for some state r and some letter b. The state r is accessible in $\mathcal{A}$, and $r \neq q$. Since $\mathcal{A}$ and $\mathcal{B}$ only differ by a transition that ends in q, r is also accessible in $\mathcal{B}$. Therefore, q is accessible in $\mathcal{B}$ because one can follow a path from 1 to r, then use the transition

$r \xrightarrow{b} q$. Since all other states remain accessible for the same reason as r, $\mathcal{B}$ is accessible. □

Note that the result of Lemma 1 does not hold for automata that are not acyclic.

Lemma 2. *The Markov chain of the algorithm is symmetric, that is, for all* $\mathcal{A}, \mathcal{B} \in \mathbb{A}_n$, $\mathcal{P}_{\mathcal{A},\mathcal{B}} = \mathcal{P}_{\mathcal{B},\mathcal{A}}$.

Proof. Recall that the probability to draw a given triplet (p, a, q) with $p \in Q$, $q \in Q\backslash\{p\}$, and $a \in A$ is $\frac{1}{n(n-1)|A|}$. Let $\mathcal{A}$, $\mathcal{B}$ be in $\mathbb{A}_n$ such that $\mathcal{P}_{\mathcal{A},\mathcal{B}} > 0$. Then there exists an elementary operation that transforms $\mathcal{A}$ into $\mathcal{B}$. Suppose $\mathcal{B} = \mathcal{A} \oplus p \xrightarrow{a} q$. The probability to draw the triplet (p, a, q) is $\frac{1}{n(n-1)|A|}$. Now from $\mathcal{B}$ the only possible elementary operation to reach $\mathcal{A}$ is to remove the transition $p \xrightarrow{a} q$. Thus, we need to draw the triplet (p, a, q) and the probability of this event is $\frac{1}{n(n-1)|A|}$ too. If $\mathcal{B} = \mathcal{A} \ominus p \xrightarrow{a} q$ then $\mathcal{A} = \mathcal{B} \oplus p \xrightarrow{a} q$ thus we are in the same case as above and $\mathcal{P}_{\mathcal{A},\mathcal{B}} = \mathcal{P}_{\mathcal{B},\mathcal{A}}$.

Suppose the elementary operation that transforms $\mathcal{A}$ to $\mathcal{B}$ is to redirect the transition $p \xrightarrow{a} q$ of $\mathcal{A}$ to obtain $p \xrightarrow{a} s$ in $\mathcal{B}$. To get this, we need to draw the triplet (p, a, s) and the probability of this event is $\frac{1}{n(n-1)|A|} = \mathcal{P}_{\mathcal{A},\mathcal{B}}$. The only possible elementary operation to reach $\mathcal{A}$ from $\mathcal{B}$ is to redirect the new transition $p \xrightarrow{a} s$ to $p \xrightarrow{a} q$ which has the same probability, for the same reasons. Hence $\mathcal{P}_{\mathcal{A},\mathcal{B}} = \mathcal{P}_{\mathcal{B},\mathcal{A}}$ in this case too. □

Lemma 3. *The Markov chain of the algorithm is ergodic.*

Proof. We need to prove that it is both irreducible and aperiodic.

To prove the irreducibility, we show that, in the Markov chain, there is a path from any acyclic automaton $A \in \mathbb{A}_n$ to an automaton $S_n \in \mathbb{A}_n$, where S_n is the acyclic automaton whose only transitions are $i \xrightarrow{a} i+1$, for $i \in \{1, \ldots, n-1\}$ and for all a:

Let $\mathcal{A}$ be any acyclic automaton and let a be a letter in A. Let E be the set of states that are accessible from the initial state by reading only the letter a. E is not empty since it contains at least the initial state 1. Repeatedly remove every transition $p \xrightarrow{\alpha} q$ where $q \in E$ and $p \notin E$. Then repeatedly remove every remaining transition $p \xrightarrow{\alpha} q$ where $p, q \in E$ and $\alpha \neq a$. These actions are valid moves in the Markov chain by Lemma 1 since we always keep the transitions $p \xrightarrow{a} q$ with $p, q \in E$. Let ℓ be the only state in E with no outgoing transition labelled with a.

If $|E| < n$, choose a state s of $\mathcal{A}$ that is not in E and add a transition $\ell \xrightarrow{a} s$. Since there is no path between s and a state of E, this operation cannot create a cycle. Repeatedly remove all transitions directed toward s except $\ell \xrightarrow{a} s$. Add

s to E, the set E is one state bigger. The size of E being finite, this operations can be repeated until E contains all states of $\mathcal{A}$.

Hence, at some point $|E| = n$ and $\mathcal{A}$ is isomorphic to S_n, since every state but the initial one has exactly one incoming transition, which is labelled by a. The only difference with S_n is that the states are not necessarily in the correct order. We now explain how they can be re-ordered.

Let $b \in A$, $b \neq a$, for each transition $p \xrightarrow{a} q$ of $\mathcal{A}$, we add to $\mathcal{A}$ the transition $p \xrightarrow{b} q$ by elementary operations, which do not create any cycle. Now we remove all transitions labelled by a, $\mathcal{A}$ remains accessible because of the transitions labelled by b. We are in the case $|E| < n$ above, where the set E only contains the state 1. To reach the automaton S_n, it is sufficient to choose the new states added to E in the order of their label. After removing all transitions labelled by b, we finally obtain the automaton S_n.

Hence for every $\mathcal{A} \in \mathbb{A}_n$, there exists a path from $\mathcal{A}$ to S_n in the Markov chain. By Lemma 2 there also exists a path from S_n to $\mathcal{A}$: the Markov chain is therefore irreducible. For every automaton $\mathcal{A} \in \mathbb{A}_n$ and any state $p \neq 1$ and any letter $a \in A$, if the edge chosen by the algorithm is $(p, a, 1)$ then $\mathcal{A}$ remains the same: adding the transitions would make $\mathcal{A}$ cyclic. Hence every vertex has a loop of length 1 in the Markov chain, it is therefore aperiodic. □

Lemma 4. *The diameter of the Markov chain is in* $\Theta(n)$.

Proof. Using the construction proposed in the proof of Lemma 3, every $\mathcal{A} \in \mathbb{A}_n$ is at distance at most $(|A|+5)n$ of S_n. The diameter of the Markov chain is thus at most $2(|A|+5)n$, which is $O(n)$. The lower bound in $\Omega(n)$ is obtained by considering the distance from S_n to an acyclic automaton whose edges are all labelled by a letter $b \neq a$. □

Theorem 1 is a consequence of the lemmas above: by Lemma 3 the Markov chain of the algorithm is ergodic and by Lemma 2 it is symmetric. According to a classical result in Markov chain theory [11], its stationary distribution is the uniform distribution on $\mathbb{A}_n$.

6. Algorithm for Minimal Acyclic Automata

```
MinimalAcyclicAutomatonGeneration(n, T)
1  A ← any minimal and acyclic automaton of M_n
2  i ← 0
3  while i < T do
4      x ← Uniform({0,1})
5      if x = 0 then
6          p ← Uniform(Q), a ← Uniform(A), q ← Uniform(Q \ {p})
7          if δ(p,a) is undefined then
8              if IsAcyclic(A ⊕ p -a-> q) and IsMinimal(A ⊕ p -a-> q) then
                   A = A ⊕ p -a-> q
9          else if δ(p,a) = q then
10             if IsTrim(A ⊖ p -a-> q) and IsMinimal(A ⊖ p -a-> q) then
                   A = A ⊖ p -a-> q
11         else
12             r ← δ(p,a)
13             if IsTrim(A ⊖ p -a-> r) then
14                 A = A ⊖ p -a-> r
15                 if IsAcyclic(A ⊕ p -a-> q) and IsMinimal(A ⊕ p -a-> q) then
16                     A = A ⊕ p -a-> q
17                 else
18                     A = A ⊖ p -a-> r
19     else
20         p ← Uniform(Q)
21         if IsMinimal(A ⊕ p̄) then
22             A = A ⊕ p̄
23     i ← i + 1
24 return A
```

This second algorithm is quite similar to the previous one. It has two arguments, n the number of states and T the desired number of iterations. The algorithm start from a minimal acyclic automaton of size n and randomly move T times in a Markov chain on $\mathbb{M}_n$, the set of n-size minimal acyclic automata. Then it returns the last automaton reached.

A move in the Markov chain consists in performing one of the two following cases, each with probability $\frac{1}{2}$:

- Uniformly draw a state $p \in Q$ and change its final status (this operation is represented by $\overline{p}$). This means that if p is a final state, we make it non final, and if it is not we make it final.
- Uniformly draw two states $p, q \in Q$ and a letter $a \in A$. Then we have

the same three cases as in the previous algorithm except that here we also need to check if the resulting automaton is still minimal. Furthermore, we do not only check for accessibility but also for co-accessibility, which means that the automaton must be trim.

Note that in this algorithm, we use three tests to decide whether or not the automaton is acyclic, trim and minimal.

In order to know if the automaton is acyclic, we use the same depth-first-search algorithm as before running in time $O(n)$. We still need to check accessibility but we also need to check co-accessibility because we need the resulting automaton to be trim. So we use the same principle as for accessibility and apply it to co-accessibility by counting output transitions of each state. The demonstration of the correctness of this test is done in the Section 7. It is still clear that this test runs in constant time. For the minimality test, we can use the Revuz' algorithm [17] which specifically deals with acyclic automata and whose running time is linear in the number of states.

For each move the algorithm performs at most four tests, the accessibility test, the co-accessibility test, the acyclicity test and the test of minimality. The time complexity of each test is in $O(n)$ (except the accessibility and co-accessibility tests which both run in constant time) and because we do T moves in the algorithm, we obtain the following result.

Proposition 3. *The worst time complexity of the algorithm that generates minimal acyclic automata is in $O(nT)$ and its space complexity is $O(n)$.*

The following property proves the effectiveness of the algorithm.

Theorem 2. *The Markov chain of the algorithm that generates minimal acyclic automata is ergodic and its stationary distribution is the uniform distribution on $\mathbb{M}_n$.*

Our proof of Theorem 2 also gives informations on the diameter of the underlying Markov chain.

Proposition 4. *The diameter of the Markov chain that generates minimal acyclic automata is in $\Theta(n)$.*

7. Proofs

This section is devoted to the proof of the Theorem 2 and to the estimation of the diameter of the underlying Markov chain.

Since a hammock automaton is acyclic, the length of a path from any state to the state n is upper bounded by $n-1$. We use this remark in order to define the rank of a state in a hammock automaton.

Definition 2. Let $\mathcal{A}$ be a hammock automaton. For each state p of $\mathcal{A}$, the *rank* of p, denoted by $\eta_{\mathcal{A}}(p)$ (or $\eta(p)$ if there is no ambiguity), is the length of the path of maximal length among all paths from p to the target state n.

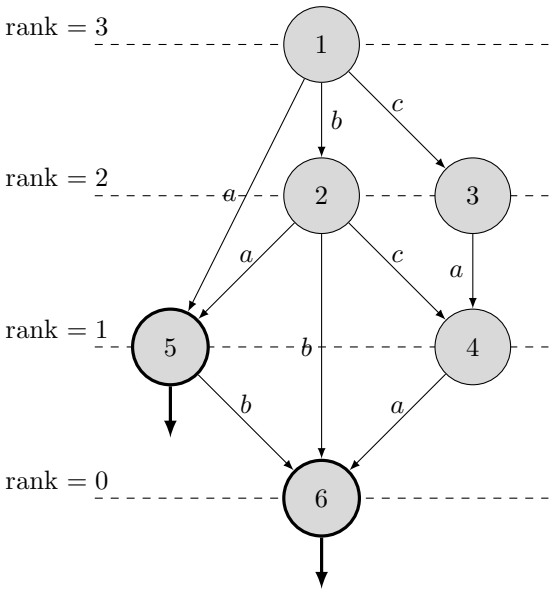

Figure 4: A hammock automaton with 6 states. The states have been organized to emphasize the value of their rank.

Since two equivalent states have the same future, they have the same rank. Note also that if p is an ancestor of q then $\eta(p) > \eta(q)$. Moreover if p is not the target state, then it has a direct successor q such $\eta(p) = \eta(q) + 1$.

Definition 3. Let $\mathcal{A} = (Q, A, \delta, \{1\}, F)$ be an acyclic automaton. Two states p and q are *directly equivalent* if they satisfy the following properties.

- $p \in F \iff q \in F$,
- $A_p = A_q$,
- $\forall a \in A_q$, $\delta(p, a) = \delta(q, a)$.

Observe that two states directly equivalent are Nerode equivalent.

Lemma 5. *Let $\mathcal{A} = (A, Q, \delta, \{1\}, F)$ be an acyclic automaton such that state n has no transition. $\mathcal{A}$ is a minimal automaton if and only if $n \in F$, $\mathcal{A}$ is a hammock automaton and such that two distinct states of $\mathcal{A}$ are never directly equivalent.*

Proof. Assume first that $\mathcal{A}$ is a minimal acyclic automaton. Let t and r be two states of $\mathcal{A}$ with no outgoing transition. The states t and r must be final states, otherwise $\mathcal{A}$ is not trim. Thus $t \in F$ and $r \in F$ and $A_r = A_t = \emptyset$. Therefore $t \sim s$ and since $\mathcal{A}$ is minimal, then $t = s$. This proves that n is the only state with no outcoming transition and that $n \in F$. Hence $\mathcal{A}$ is a

hammock automaton. Let p and q be two states which are directly equivalent and therefore equivalent. Since $\mathcal{A}$ is minimal $p = q$.

Assume now that A is a hammock automaton of target state n, which is final, and such that two different states are never directly equivalent. There is a path from any state to state n, which is in F. This means that $\mathcal{A}$ is co-accessible, and therefore $\mathcal{A}$ is trim. Since $\mathcal{A}$ is a hammock automaton we can use the notion of rank (see Definition 2) to prove that $\mathcal{A}$ is minimal. By contradiction, suppose that $\mathcal{A}$ has two different states that are equivalent, we are going to prove that they are directly equivalent. The set $E = \{s \in Q \mid \exists s', s \neq s',\ s \sim s'\}$ is not empty. Let $s \in E$ be an element of minimal rank $\eta(s)$. Since $s \in E$, there exists $s' \neq s$ such that $s \sim s'$, and therefore $\eta(s) = \eta(s')$. For every $a \in A_s$, $\delta(s, a) \sim \delta(s', a)$, thus $\eta(\delta(s, a)) = \eta(\delta(s', a)) < \eta(s)$. But $\eta(s)$ is minimal in E, so that $\delta(s, a)$ is not in E and therefore $\delta(s, a) = \delta(s', a)$. This means that s and s' are directly equivalent, which is a contradiction: $\mathcal{A}$ is minimal. □

Lemma 6. *The Markov chain of the algorithm that generates minimal acyclic automata is irreducible.*

Proof. In order to prove that the Markov chain is irreducible, we need to show that for any two minimal acyclic automata there exists a path from one to the other in the Markov chain. Since the Markov chain is symmetric it is sufficient to prove that from any automaton $\mathcal{A}$, there exists a path to a specific automaton. We start from $\mathcal{A} = (Q, A, \delta_{\mathcal{A}}, \{1\}, F)$ in $\mathbb{M}_n$ and prove that we can reach the line automaton $\mathcal{D} = (Q, A, \delta_{\mathcal{D}}, \{1\}, F_{\mathcal{D}})$, where $F_{\mathcal{D}} = \{n\}$, $\forall p \in Q \setminus \{n\}$, $A_p = \{a\}$ and $\delta(p, a) = p+1$. Since we consider hammock automata only (see Proposition 5), we can use the notion of rank for the states.

First, we assume that $\eta_{\mathcal{A}}(1) = n - 1$ and show that we can reach the line automaton from $\mathcal{A}$ by making moves in the Markov chain of $\mathbb{M}_n$. Since $\eta_{\mathcal{A}}(1) = n - 1$, all states have different ranks, which range from 0 to $n - 1$. Any state s has at least one transition directed towards the state of rank $\eta(s) - 1$. For each state $s \neq n$, we remove all transitions starting from this state except one directed towards the state of rank $\eta(s) - 1$ (these are valid moves in the Markov chain). We then remove all states from the set F of final state, except state n. This results in a line automaton $\mathcal{B}$ where the states are not necessarily in order and where the transitions are not necessarily labeled by the letter a. Note that after each modification the states keep their rank, therefore the automaton remains minimal and acyclic.

Since $|A| \geq 2$, then for each state s in $\mathcal{B}$ there is a letter b such as $\delta_{\mathcal{B}}(s, b)$ is not defined. Now we will show that for two states p and q such as p is an ancestor of q, $q \neq n$, we can reorder the state using Markov chain moves, in a way that p becomes a direct ancestor of q without changing the order of p's ancestors. When all states will be in order we can replace each transition by a transition labelled by a, reaching the automaton $\mathcal{D}$. The process, depicted in Figure 7 is the following. Add a transition from p to q, using a remaining letter. Make q final. Note that until now the rank of the states have not changed, and the automaton is still a minimal acyclic automaton. Let s be the direct ancestor

of q and r be the direct successor of q. Redirect the transition between s and q towards r. Notice that the automaton remains a hammock automaton, and that except state r and state 1, all states have only one incoming transition. Moreover q and s are not directly equivalent because q is a final state and s is not. Hence the automaton is still a minimal acyclic automaton. Let t be the direct successor of p. Since q is only accessible from p, t is not an ancestor of q. Redirect the transition of q towards t. p is an ancestor of q then p and q cannot be directly equivalent. Now remove the transition between p and t and remove q from the set of final state. We have changed the direct successor of p. The order of all ancestor of p have not change.

Using the process repeatedly, we can reorder the states to obtain the automaton $\mathcal{D}$.

Assume now that $\eta_{\mathcal{A}}(1) < n-1$. Using moves in the Markov chain, we will change $\mathcal{A}$ into an automaton $\mathcal{B}$, where the rank of 1 in $\mathcal{B}$ is greater than the rank of 1 in $\mathcal{A}$. After less than $n-1$ moves in the Markov chain we will reach an automaton where the rank of 1 is $n-1$, so that we can apply the construction just above.

For all state $s \in Q$ we have $0 \leq \eta_{\mathcal{A}}(s) \leq \eta_{\mathcal{A}}(1) \leq n-2$. Since $|Q| = n$ there exist two states p, p' such that $\eta_{\mathcal{A}}(p) = \eta_{\mathcal{A}}(p')$.

Let p be a state such that $\eta_{\mathcal{A}}(p)$ is minimal and there exists another state p' such that $\eta_{\mathcal{A}}(p) = \eta_{\mathcal{A}}(p')$. Let $q \neq p$ be another state, such that $\eta_{\mathcal{A}}(q)$ is maximal in the set of states that are not an ancestor of p (this set is not empty since it contains at least p'). State p has at least one transition directed towards a state r of rank $\eta_{\mathcal{A}}(r) = \eta_{\mathcal{A}}(p) - 1$. Let $\mathcal{B}$ be the automaton obtained from $\mathcal{A}$ by redirecting this transition towards q.

We now prove that $\mathcal{B}$ is minimal and acyclic. It is clear that $\mathcal{B}$ is deterministic, and, since the redirection is performed toward q, which is not an ancestor of p in $\mathcal{A}$, $\mathcal{B}$ is also acyclic. The state p' has a transition directed towards a state r' such as $\eta_{\mathcal{A}}(p') - 1 = \eta_{\mathcal{A}}(r')$. Therefore $\eta_{\mathcal{A}}(r') = \eta_{\mathcal{A}}(r) < \eta_{\mathcal{A}}(p)$. Therefore $r = r'$, since there cannot be two distinct states of same rank strictly smaller than $\eta_{\mathcal{A}}(p)$, by definition of p. As a consequence there is a transition from p' to r in $\mathcal{B}$, and $\mathcal{B}$ is accessible. Since the only state of $\mathcal{A}$ with no outgoing transition is n, n is also the only state with no outgoing transition in $\mathcal{B}$. Thus $\mathcal{B}$ is a hammock automaton. We denote by $\eta_{\mathcal{B}}$ the rank of the states of $\mathcal{B}$ and by $\delta_{\mathcal{B}}$ its transition function. Note that $\eta_{\mathcal{A}}(p) < \eta_{\mathcal{B}}(p)$ since $\eta_{\mathcal{A}}(q) \geq \eta_{\mathcal{A}}(p') = \eta_{\mathcal{A}}(p)$, and since there is a transition form p to q in $\mathcal{B}$.

By contradiction, suppose that $\mathcal{B}$ is not minimal; then there exist two states o and o', $o' \neq o$, that are directly equivalent in $\mathcal{B}$ (see Lemma 5). Notice that $o = p$ or $o' = p$, otherwise the transitions of o and o' would be the same in both $\mathcal{A}$ and $\mathcal{B}$, thus o and o' would be directly equivalent in $\mathcal{A}$, which is not possible. Without loss of generality we suppose that $p = o'$. Since o is directly equivalent to p in $\mathcal{B}$, then o is not an ancestor of p in $\mathcal{B}$ and o is an ancestor of q in $\mathcal{B}$. The transitions of o are the same in both $\mathcal{A}$ and $\mathcal{B}$. As a consequence o is not an ancestor of p in $\mathcal{A}$ and is an ancestor of q in $\mathcal{A}$. Thus $\eta_{\mathcal{A}}(q) < \eta_{\mathcal{A}}(o)$, which is a contradiction since $\eta_{\mathcal{B}}(q)$ is maximal amongst states that are not ancestor

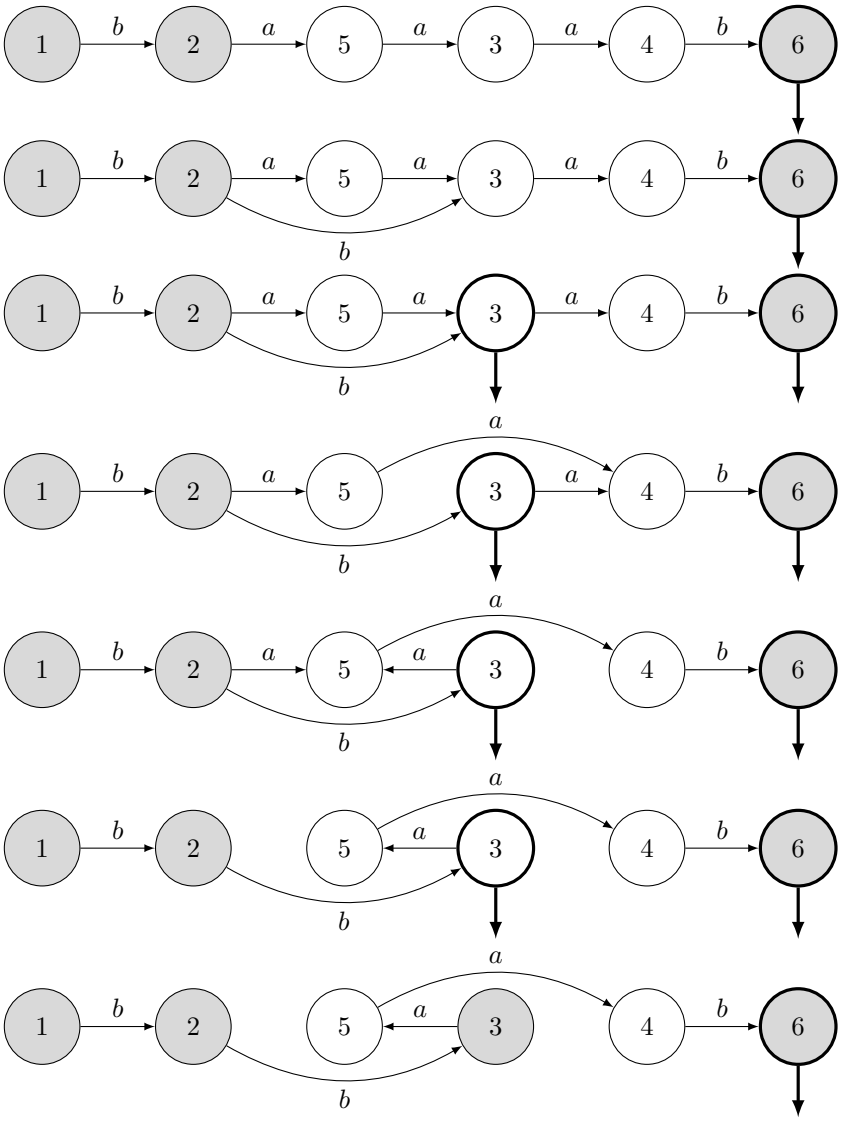

Figure 5: Different moves in a Markov chain to position the state 3 after the state 2. At the beginning only states 2,6 and 1 are correctly ordered. Then after five moves the state 1,2,6 and 3 are in order

of p in $\mathcal{A}$. Therefore $\mathcal{B}$ is minimal.

We now prove that the rank of every ancestor s of p has increased: $\eta_{\mathcal{A}}(s) < \eta_{\mathcal{B}}(s)$. By contradiction, suppose that the set $J = \{j \in Q \mid j \text{ is an ancestor of } p \text{ in } \mathcal{B} \text{ and } \eta_{\mathcal{B}}(j) \leq \eta_{\mathcal{A}}(j)\}$ is not empty. Let s be a state of J such that $\eta_{\mathcal{A}}(s)$ is minimal. Let t be a direct successor of s such that $\eta_{\mathcal{A}}(t) = \eta_{\mathcal{A}}(s) - 1$ in $\mathcal{A}$. Three cases occur.

• there is no path from t to p in $\mathcal{A}$: since s is an ancestor of p in both $\mathcal{B}$ and $\mathcal{A}$ then $\eta_{\mathcal{B}}(p) < \eta_{\mathcal{B}}(s)$. Moreover $s \in J$, therefore $\eta_{\mathcal{B}}(s) \leq \eta_{\mathcal{A}}(s)$, so that $\eta_{\mathcal{B}}(p) \leq \eta_{\mathcal{B}}(s) - 1 \leq \eta_{\mathcal{A}}(s) - 1 = \eta_{\mathcal{A}}(t)$. Since p is a direct ancestor of q in $\mathcal{B}$, by definition of the transformation leading to automaton $\mathcal{B}$, then $\eta_{\mathcal{B}}(q) < \eta_{\mathcal{B}}(p)$, thus $\eta_{\mathcal{B}}(q) < \eta_{\mathcal{A}}(t)$. Since by construction there is no path from q to p in $\mathcal{A}$, the rank of q is not changed during the transformation: $\eta_{\mathcal{B}}(q) = \eta_{\mathcal{A}}(q)$. Hence $\eta_{\mathcal{A}}(q) < \eta_{\mathcal{A}}(t)$. This is not possible since q was chosen as a state of maximal rank in $\mathcal{A}$ amongst those that are not an ancestor of p, and t is such a state.

• $t = p$: then we have $\eta_{\mathcal{A}}(t) < \eta_{\mathcal{B}}(t)$, since the rank of p has increased. Then $\eta_{\mathcal{A}}(t) + 1 = \eta_{\mathcal{A}}(s) \leq \eta_{\mathcal{B}}(t)$. But since s is in J, $\eta_{\mathcal{B}}(s) \leq \eta_{\mathcal{A}}(s)$, and therefore $\eta_{\mathcal{B}}(s) \leq \eta_{\mathcal{B}}(t)$. This is also a contradiction since s is an ancestor of t in $\mathcal{B}$.

• t is an ancestor of p: since s is an ancestor of t in $\mathcal{A}$ then $\eta_{\mathcal{A}}(t) < \eta_{\mathcal{A}}(s)$. By hypothesis, the rank of s in $\mathcal{A}$ is minimal amongst the states of J, therefore $t \notin J$. Since $t \notin J$ and t is an ancestor of p, we have $\eta_{\mathcal{A}}(t) < \eta_{\mathcal{B}}(t)$. Then $\eta_{\mathcal{A}}(t) + 1 = \eta_{\mathcal{A}}(s) \leq \eta_{\mathcal{B}}(t)$. Since s is in J, $\eta_{\mathcal{B}}(s) \leq \eta_{\mathcal{A}}(s)$. Therefore $\eta_{\mathcal{B}}(s) \leq \eta_{\mathcal{B}}(t)$. This is another contradiction, since s is an ancestor of t in $\mathcal{B}$.

Hence, J is empty and for any ancestor s of p, we have proved that $\eta_{\mathcal{A}}(s) < \eta_{\mathcal{B}}(s)$. This is in particular true for the initial state 1, whose rank has increased during the transformation from $\mathcal{A}$ to $\mathcal{B}$, $\mathcal{B}$ still being a minimal acyclic automaton. Repeating this construction leads to a minimal automaton whose initial state has rank $n - 1$, which has been analyzed in the first part of the proof: starting from any minimal acyclic automaton $\mathcal{A}$, there is a path in the Markov chain that reaches $\mathcal{D}$. □

We have all the elements to prove Theorem 2.

Proof (of Theorem 2). To show the ergodicity of the Markov chain we have to prove that it is irreducible and aperiodic. The irreducibility of the Markov chain is proved in Lemma 6. Observe that in the algorithm, from any minimal automaton we have a nonzero probability to choose a triplet (p, a, q) which lead to stay at the same place in the chain. All vertices of the Markov chain are therefore aperiodic and the Markov chain is aperiodic too. Then the Markov chain is ergodic, and it has only one stationary distribution, with convergence to it. And since the Markov chain is symmetric by construction, the stationary distribution is the uniform distribution on $\mathbb{M}_n$. □

An estimation of the diameter of the Markov chain can be deduced from the proof of Lemma 6.

Proof (of Proposition 4). Let $\mathcal{A}$ be any minimal acyclic automaton of $\mathbb{M}_n$. In the proof of Lemma 6 we used at most $n - 1$ moves in the Markov to change

$\mathcal{A}$ into a minimal automaton $\mathcal{B}$ such that $\eta_{\mathcal{B}}(1) = n-1$. Then for each state we remove at most $|A|$ transitions to obtain a line automaton, hence at most $n|A|$ globally. We then reordered the states labels, using at most 6 moves for each state, as depicted in Figure 7, which corresponds to at most $6n$ moves globally. We replace each transition by another transition labeled by the letter a, taking at most $2n$ moves in the Markov chain. Finally, using at most n moves, we can remove all final states but the last one, leading to automaton $\mathcal{D}$ with a path of at most $O(n)$ moves. The lower bound of $\Omega(n)$ is easily obtained, starting with a line automaton having no a-transition.

Hence the farthest automaton from $\mathcal{D}$ is at distance $\Theta(n)$ and the diameter of the chain is therefore also $\Theta(n)$. □

8. Conclusion

In this article we have seen on two instances how to build Markov chains for the purpose of generating deterministic and accessible acyclic automata, either in general, or restricted to the class of minimal acyclic automata. These techniques can easily be adapted to many other families of deterministic acyclic automata: simple constraints, for instance, can easily be handled by slightly changing the algorithm. Such a simple constraint could be "the initial state has no outgoing transition labelled by a" or "the word ab is not recognized", for example.

More interestingly, the Markov chain for generating minimal acyclic automata can be tuned to consider minimal acyclic automata having a fixed number of final states. The possibility of changing one state from final to non-final or from non-final to final must be removed, replaced by the possibility of exchanging the final status of two random states (to keep the number of final states constant). The proof of ergodicity can readily be adapted and the diameter is roughly the same as in the article. This may be useful to avoid producing minimal automata with a large number of final states with high probabilities, as it is the case for the uniform distribution on the set of minimal acyclic automata.

The main remaining question, which seems to be quite challenging, is to obtain interesting upper bounds on the mixing time of the Markov chains described in this article.

Acknowledgement: we would like to thanks Cyril Nicaud for his precious help in most stages of this work.

[1] Marco Almeida, Nelma Moreira, and Rogério Reis. Exact generation of minimal acyclic deterministic finite automata. *Int. J. Found. Comput. Sci.*, 19(4):751–765, 2008.

[2] Frédérique Bassino and Cyril Nicaud. Enumeration and random generation of accessible automata. *Theor. Comput. Sci.*, 381(1-3):86–104, 2007.

[3] Pascal Caron, Jean-Marc Champarnaud, and Ludovic Mignot. Small extended expressions for acyclic automata. In Sebastian Maneth, editor,

CIAA, volume 5642 of *Lecture Notes in Computer Science*, pages 198–207. Springer, 2009.

[4] Pascal Caron, Jean-Marc Champarnaud, and Ludovic Mignot. Acyclic automata and small expressions using multi-tilde-bar operators. *Theor. Comput. Sci.*, 411(38-39):3423–3435, 2010.

[5] Pascal Caron and Djelloul Ziadi. Characterization of glushkov automata. *Theor. Comput. Sci.*, 233(1-2):75–90, 2000.

[6] Jean-Marc Champarnaud and Thomas Paranthoën. Random generation of DFAs. *Theor. Comput. Sci.*, 330(2):221–235, 2005.

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

[8] Philippe Flajolet, Paul Zimmermann, and Bernard Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theor. Comput. Sci.*, 132(2):1–35, 1994.

[9] Pierre-Cyrille Héam, Cyril Nicaud, and Sylvain Schmitz. Parametric random generation of deterministic tree automata. *Theor. Comput. Sci.*, 411(38-39):3469–3480, 2010.

[10] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[11] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times.* AMS, 2008.

[12] Valery A. Liskovets. The number of initially connected automata. *Cybernetics*, 4:259–262, 1969. English translation of Kibernetika (3) 1969, 16-19.

[13] Valery A. Liskovets. Exact enumeration of acyclic deterministic automata. *Discrete Applied Mathematics*, 154(3):537–551, 2006.

[14] Guy Melançon, Isabelle Dutour, and Mireille Bousquet-Mélou. Random generation of directed acyclic graphs. *Electronic Notes in Discrete Mathematics*, 10:202–207, 2001.

[15] Guy Melançon and Fabrice Philippe. Generating connected acyclic digraphs uniformly at random. *Inf. Process. Lett.*, 90(4):209–213, 2004.

[16] Mehryar Mohri. String-matching with automata. *Nord. J. Comput.*, 4(2):217–231, 1997.

[17] Dominique Revuz. Minimisation of acyclic deterministic automata in linear time. *Theor. Comput. Sci.*, 92(1):181–189, 1992.