Average Case Analysis of Moore's State Minimization Algorithm*

Frédérique Bassino
LIPN UMR 7030, Université Paris 13 - CNRS,
99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France.
Frederique.Bassino@lipn.univ-paris13.fr

Julien David
Institut Gaspard Monge, Université Paris Est,
77454 Marne-la-Vallée Cedex 2, France.
Julien.David@univ-paris-est.fr

Cyril Nicaud
Institut Gaspard Monge, Université Paris Est,
77454 Marne-la-Vallée Cedex 2, France.
Cyril.Nicaud@univ-paris-est.fr

July 29, 2009

Abstract. We prove that for the uniform distribution over accessible deterministic automata with n states, the average complexity of Moore's state minimization algorithm is in $\mathcal{O}(n \log n)$. The bound is shown to be tight in the case of unary automata. The average complexity of this algorithm for other related distributions, such as the uniform distribution over possibly incomplete or co-accessible automata, is also analyzed.
Key Words. state minimization algorithms, Moore's algorithm, average complexity, finite automata.

1 Introduction

Deterministic automata are a convenient way to represent regular languages that can be used to efficiently perform most of usual computations involving regular languages. Therefore finite state automata appear in many fields of computer science, such as linguistics, data compression, bioinformatics, *etc.* To a given regular language one can associate a unique smallest deterministic automaton, called its minimal automaton. This canonical representation of regular languages is compact and provides an easy way to check equality between regular languages. As a consequence, state minimization

*The authors were supported by the ANR (GAMMA - project BLAN07-2_195422)

algorithms that compute the minimal automaton of a regular language given by a deterministic automaton are fundamental.

Moore proposed a solution [1] that can be seen as a sequence of partition refinements. Starting from a partition of the set of states, of size n, into two parts, successive refinements lead to a partition whose elements are the subsets of indistinguishable sets, that can be merged to form a smaller automaton recognizing the same language. As there are at most $n-2$ such refinements, each of them requiring a linear running time, the worst-case complexity of Moore's state minimization algorithm is quadratic.

Hopcroft's state minimization algorithm [2] also uses partition refinements to compute the minimal automaton, selecting carefully the parts that are split at each step. Using suitable data structures, its worst-case complexity is in $\mathcal{O}(n \log n)$. It is the best known minimization algorithm and therefore it has been intensively studied: in [3, 4] the authors give different proofs of its correctness, in [5, 6, 7] they prove the tightness of the upper bound of the complexity for various family of automata, in [8, 9] they give a precise description of the data structures that are needed to reach the $\mathcal{O}(n \log n)$ complexity, and in [10, 11] they present some variations of the algorithm for incomplete automata. Note that Hopcroft and Ullman described another minimization algorithm in [12], which is much easier to implement: it tests, for every pair of states of the input automaton, whether the two states are equivalent or not. Its complexity is $\Theta(n^2)$.

In certain cases where the set of input automata is restricted by the specification of properties, there exist algorithms which compute the minimal automaton in linear time: see [13] for acyclic automata, [14] for unary automata and [15] for local automata.

Finally Brzozowski algorithm [16, 17] is different from the other ones. Its inputs may be non-deterministic automata. It is based on two successive determinization steps, and though its worst-case complexity is proved to be exponential, it has been noticed that it is is often sub-exponential in practice. The reader is invited to consult [18], which presents a taxonomy of minimization algorithms.

In this paper we study the average time complexity of Moore's algorithm. From an experimental point of view, the average number of partition refinements increases very slowly as the size of the input grows (Fig.1).

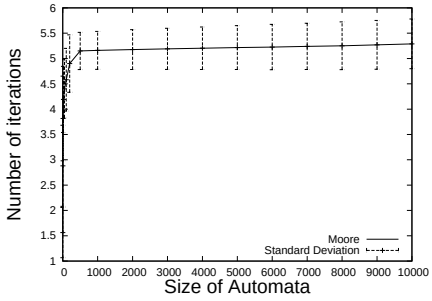

Figure 1: The experimental results were obtained with the C++ library REGAL (available at: `http://regal.univ-mlv.fr/`) to randomly generate accessible deterministic automata [19, 20, 21]. For each size the values are computed from 20 000 random automata over a 2-letter alphabet.

In the following we mainly prove that in average, for the uniform distribution

over accessible deterministic automata with n states, Moore's algorithm performs only $\mathcal{O}(\log n)$ refinements, thus its average complexity is $\mathcal{O}(n \log n)$.

After recalling the basics of minimization of automata in Section 2, we establish some results on Moore's algorithm applied to minimal automata in Section 3. We prove in Section 4 that the average complexity of Moore's algorithm is $\mathcal{O}(n \log n)$ (Theorem 2). In Section 5, we show that this bound is tight when the alphabet is unary, and study some generalizations of Theorem 2 to other distributions over natural classes of automata. Finally, we propose in Section 6 a conjecture about a tight upper bound for the average time complexity of Moore's algorithm when the alphabet in not unary.

A preliminary version of this work has been presented in [22].

2 Preliminaries

This section is devoted to basic notions related to the minimization of automata. We refer the reader to the literature for more details about this topic. Only a few definitions and results that will be useful for our purpose are recalled here.

2.1 Definitions and notations

A *finite deterministic automaton* $\mathcal{A} = (A, Q, \cdot, q_0, F)$ is a quintuple where Q is a finite set of *states*, $A = \{a_1, \ldots, a_k\}$ is a finite set of *letters* called *alphabet*, the *transition function* $\cdot$ is a mapping from $Q \times A$ to Q, $q_0 \in Q$ is the *initial state* and $F \subset Q$ is the set of final states. An automaton is *complete* when its transition function is total. The transition function can be extended by morphism to all words of A^*: $p \cdot \varepsilon = p$ for any $p \in Q$ and for any $u, v \in A^*$, $p \cdot (uv) = (p \cdot u) \cdot v$. A word $u \in A^*$ is recognized by an automaton when $p \cdot u \in F$. The set of all words recognized by $\mathcal{A}$ is denoted by $L(\mathcal{A})$. An automaton is *accessible* when for any state $p \in Q$, there exists a word $u \in A^*$ such that $q_0 \cdot u = p$.

A *transition structure* is an automaton where the set of final states is not specified. Given such a transition structure $\mathcal{T} = (A, Q, \cdot, q_0)$ and a subset F of Q, we denote by $(\mathcal{T}, F)$ the automaton $(A, Q, \cdot, q_0, F)$. For a given accessible deterministic transition structure with n states there are exactly 2^n distinct accessible deterministic automata that can be built from this transition structure. Each of them corresponds to a choice of set of final states.

In the following we only consider accessible complete deterministic automata and accessible complete deterministic transition structures, except in the presentation of the generalizations of the main theorem in Section 5. Consequently these objects will often just be called respectively *automata* or *transition structures*. The set Q of states of an n-state transition structure will be denoted by $\{1, \ldots, n\}$.

The cardinality of a finite set E is denoted by $|E|$. For a boolean condition $Cond$, the Iverson bracket $[\![Cond]\!]$ is equal to 1 if the condition $Cond$ is satisfied and 0 otherwise.

2.2 Myhill-Nerode Equivalence

Let $\mathcal{A} = (A, Q, \cdot, q_0, F)$ be an automaton. For any nonnegative integer i, two states $p, q \in Q$ are *i-equivalent*, denoted by $p \sim_i q$, when for all words u of length less than or equal to i, $[\![p \cdot u \in F]\!] = [\![q \cdot u \in F]\!]$. Two states are *equivalent* when for all $u \in A^*$, $[\![p \cdot u \in F]\!] = [\![q \cdot u \in F]\!]$. This equivalence relation is called *Myhill-Nerode equivalence* [23].

Recall that an equivalence relation $\equiv$ defined on the set of states Q of a deterministic automaton is said to be *right invariant* when

$$\text{for all } u \in A^* \text{ and for all } p, q \in Q, \quad p \equiv q \Rightarrow p \cdot u \equiv q \cdot u.$$

The following proposition summarizes the properties of Myhill-Nerode equivalence that will be used in the next sections.

Proposition 1. *Let $\mathcal{A} = (A, Q, \cdot, q_0, F)$ be a deterministic automaton with n states. The following properties hold:*

1. *For all $i \in \mathbb{N}$, $\sim_{i+1}$ is a partition refinement of $\sim_i$, that is, for all $p, q \in Q$, if $p \sim_{i+1} q$ then $p \sim_i q$.*
2. *For all $i \in \mathbb{N}$ and for all $p, q \in Q$, $p \sim_{i+1} q$ if and only if $p \sim_i q$ and for all $a \in A$, $p \cdot a \sim_i q \cdot a$.*
3. *If, for some $i \in \mathbb{N}$, $(i+1)$-equivalence is equal to i-equivalence then for every $j \geq i$, j-equivalence is equal to Myhill-Nerode equivalence.*
4. *$(n-2)$-equivalence is equal to Myhill-Nerode equivalence.*
5. *Myhill-Nerode equivalence is right invariant.*

Let $\mathcal{A} = (A, Q, \cdot, q_0, F)$ be an automaton and $\equiv$ be a right invariant equivalence relation on Q. The *quotient automaton* of $\mathcal{A}$ by $\equiv$ is the automaton

$$(\mathcal{A}/\!\equiv) = (A, Q/\!\equiv, *, [q_0], \{[f], f \in F\}),$$

where $Q/\!\equiv$ is the set of equivalent classes, $[q]$ is the class of $q \in Q$, and $*$ is defined for any $a \in A$ and any $q \in Q$ by $[q] * a = [q \cdot a]$. The correctness of this definition relies on the right invariance of the equivalence relation $\equiv$.

Theorem 1. *For any accessible complete deterministic automaton $\mathcal{A}$, the automaton $\mathcal{A}/\!\sim$ is the unique smallest complete deterministic automaton (in terms of the number of states) that recognizes the same language as the automaton $\mathcal{A}$. It is called the* minimal automaton *of $L(\mathcal{A})$.*

The uniqueness of the minimal automaton is up to labelling of the states. Theorem 1 shows that the minimal automaton is a fundamental notion in language theory: It is the most space efficient representation of a regular language by a deterministic automaton, and its uniqueness defines a bijection between regular language and minimal automata.

2.3 Moore's State Minimization Algorithm

In this section we describe an algorithm due to Moore [1] which computes the minimal automaton of a regular language represented by a deterministic automaton. The analysis of the average complexity of this algorithm is the main purpose of this article.

Recall that Moore's algorithm builds the partition of the set of states of the input automaton corresponding to Myhill-Nerode equivalence. It mainly relies on Properties *2* and *3* of Proposition 1: The partition π is initialized according to 0-equivalence $\sim_0$, then at each iteration the partition corresponding to $(i+1)$-equivalence $\sim_{i+1}$ is computed from the one corresponding to i-equivalence $\sim_i$ using Property *2*. The algorithm halts when no new partition refinement is obtained, and the result is Myhill-Nerode equivalence according to Property *3*. The minimal automaton can then be

Algorithm 1: Moore's algorithm

```
1  if F = ∅ then
2      return (A, {1}, ∗, 1, ∅)
3  if F = {1, ..., n} then
4      return (A, {1}, ∗, 1, {1})
5  forall p ∈ {1, ..., n} do
6      π'[p] = [[p ∈ F]]
7  π = undefined
8  while π ≠ π' do
9      π = π'
10     compute π' from π
11 return the quotient of A by π
```

The computation of the new partition is done using the following property on associated equivalence relations:

$$p \sim_{i+1} q \Leftrightarrow \begin{cases} p \sim_i q, \\ \forall a \in A \quad p \cdot a \sim_i q \cdot a. \end{cases}$$

To each state p is associated a signature $s[p]$ such that $p \sim_{i+1} q$ if and only if $s[p] = s[q]$. The states are then sorted according to their signature, in order to compute the new partition. The use of a lexicographic sort provides a complexity in $\Theta(kn)$ for this part of the algorithm.

In this description of Moore's algorithm, $*$ denotes the function such that $1 * a = 1$ for all $a \in A$. Lines 1-4 correspond to the special cases where $F = \emptyset$ or $F = Q$. In the process π' is the new partition and π the former one. Lines 5-6 is the initialization of π' to the partition of $\sim_0$, π is initially undefined. Lines 8-10 form the main loop of the algorithm where the new partition is computed, using the second algorithm below. The *number of iterations* of Moore's algorithm is the number of times those lines are executed.

Algorithm 2: Computing π' from π

```
1  forall p ∈ {1, ..., n} do
2      s[p] = (π[p], π[p · a_1], ..., π[p · a_k])
3  compute the permutation σ that
   sorts the states according to s[]
4  i = 0
5  π'[σ(1)] = i
6  forall p ∈ {2, ..., n} do
7      if s[p] ≠ s[p − 1] then i = i + 1
8      π'[σ(p)] = i
9  return π'
```

Figure 2: Description of Moore's algorithm

computed from the resulting partition since it is the quotient automaton of the input automaton by Myhill-Nerode equivalence. The algorithm is detailed in Figure 2.

The worst-case time complexity of Moore's algorithm is in $\Theta(n^2)$. The following result is a more precise statement about the worst-case complexity of this algorithm that will be used in the proof of the main theorem (Theorem 2). For sake of completeness we also give a justification of this statement.

For any integer $n \geq 1$ and any $m \in \{0, \ldots, n-2\}$, we denote by $\mathcal{A}_n^{(m)}$ the set of automata with n states for which m is the smallest integer such that m-equivalence $\sim_m$ is equal to Myhill-Nerode equivalence. We also denote by $\textsc{Moore}(\mathcal{A})$ the number of iterations of the main loop when Moore's algorithm is applied to the automaton $\mathcal{A}$.

Lemma 1. *For any automaton $\mathcal{A}$ of $\mathcal{A}_n^{(m)}$, the following properties hold:*

- *The number of iterations* $\textsc{Moore}(\mathcal{A})$ *of the main loop in Moore's algorithm is equal* 0 *if* $L(\mathcal{A}) = \emptyset$ *or* $L(\mathcal{A}) = A^*$ *and is equal to* $m + 1$ *otherwise.*
- $\textsc{Moore}(\mathcal{A})$ *is always less than or equal to* $n - 1$.

- *The worst-case time complexity $\mathcal{W}(\mathcal{A})$ of Moore's algorithm is uniformly in $\Theta((m+1)n)$ for $m \in \{0, \ldots, n-2\}$, or equivalently there exist two positive real numbers C_1 and C_2 independent of n and m such that $C_1(m+1)n \leq \mathcal{W}(\mathcal{A}) \leq C_2(m+1)n$.*

Proof. The loop is iterated exactly $m+1$ times when the set F of final states is neither empty nor equal to $\{1, \ldots, n\}$. Moreover from Property 4 of Proposition 1 the integer m is less than or equal to $n-2$. If F is empty or equal to $\{1, \ldots, n\}$, then necessarily $m = 0$, and the time complexity of the determination of the size of F is $\Theta(n)$.

The initialization and the construction of the quotient are both done in $\Theta(n)$. The complexity of each iteration of the main loop is in $\Theta(n)$: this can be achieved using a lexicographic sort algorithm. Moreover in this case the constants C_1 and C_2 do not depend on m, proving the uniformity of both the upper and lower bounds. □

Note that Lemma 1 gives a proof that the worst-case complexity of Moore's algorithm is in $\mathcal{O}(n^2)$, as there are no more than $n-1$ iterations in the process of the algorithm. To be more precise, the worst-case complexity of the algorithm is in $\Theta(n^2)$, as it is shown by the automaton of Figure 3, for instance.

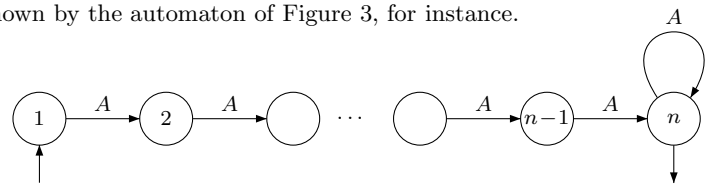

Figure 3: The minimal automaton of the language $A^{n-1}A^*$. The states 1 or 2 are $(n-3)$-equivalent, but not $(n-2)$-equivalent. Moore's algorithm performs $n-1$ iterations before halting.

3 Moore's Algorithm on Minimal Automata

Before analyzing the average behavior of Moore's algorithm, which is the main purpose of this paper, we establish two results on what happens when it is used on minimal automata. First, we prove that the number of iterations of Moore's algorithm only depends on the recognized language, and therefore it is the same for all deterministic automata recognizing the same language. Next we establish a lower bound of the number of iterations of Moore's algorithm when it is applied to minimal automata of size, that will be useful in the forthcoming discussions.

Lemma 2. *The number of iterations of Moore's algorithm applied to any accessible deterministic automaton is equal to the number of iterations of this algorithm when it is applied to the associated minimal automaton.*

Proof. Let $\mathcal{A} = (A, Q, \cdot, q_0, F)$ be the automaton and $\mathcal{A}_{min} = (A, Q/\sim, *, [q_0], F')$ be the associated minimal automaton. If the language recognized by $\mathcal{A}$ and $\mathcal{A}_{min}$ is either A^* or $\emptyset$, from Lemma 1 $\textsc{Moore}(\mathcal{A}) = \textsc{Moore}(\mathcal{A}_{min}) = 0$.

Otherwise let m be the smallest integer such that $\sim_m = \sim$ for the automaton $\mathcal{A}$. Then from Lemma 1 $\textsc{Moore}(\mathcal{A}) = m+1$. Moreover by definition of $\sim_m$ one has

$$\textsc{Moore}(\mathcal{A}) = m+1 = \min\{i \in \mathbb{N} \mid \forall (p,q) \in Q^2 \text{ such that } p \nsim q, \\ \exists u \in A^{\leq i}, [\![p \cdot u]\!] \neq [\![q \cdot u]\!]\} + 1.$$

As by definition $p \nsim q$ if and only if $[p] \neq [q]$ and since for all $p \in Q$ and all $u \in A^*$, $[\![p \cdot u]\!] = [\![[p] * u]\!]$, one can write

$$\textsc{Moore}(\mathcal{A}) = \min\{i \in \mathbb{N} \mid \forall (p,q) \in Q^2 \text{ such that } [p] \neq [q], \\ \exists u \in A^{\leq i}, [\![[p] * u]\!] \neq [\![[q] * u]\!]\} + 1.$$

Thus $\textsc{Moore}(\mathcal{A}) = \textsc{Moore}(\mathcal{A}_{min})$, concluding the proof. □

According to the previous lemma, it is interesting to study the behavior of Moore's algorithm when applied on minimal automata. The worst-case complexity of the algorithm is still $\Theta(n^2)$, since the automaton depicted in Figure 3 is minimal. As for a minimal automaton it is guaranteed that all states are alone in their equivalence class, a minimum number of iterations in the algorithm is required to distinguish all of them, as it is proved in the following proposition.

Proposition 2. *Moore's algorithm applied on a minimal automaton of size n has a complexity in $\Omega(\frac{k}{\log k} n \log\log n)$ for an alphabet of size $k \geq 2$ and in $\Omega(n \log n)$ for a one-letter alphabet.*

Proof. The algorithm ends when each state of the minimal automaton $\mathcal{A}$ is isolated in a subset of the partition. The number of subsets is equal to the number of states in $\mathcal{A}$. For any integer i and any state p, consider the mapping $\phi_p^{(i)} : A^{\leq i} \to \{0,1\}$ defined by $\phi_p^{(i)}(u) = [\![p \cdot u]\!]$. As there are k^j words of length j, the number of distinct words of length at most i, for a fixed integer i, is

$$\sum_{j=0}^{i} k^j = \begin{cases} \frac{k^{i+1}-1}{k-1} & \text{when } k \geq 2, \\ i+1 & \text{when } k = 1. \end{cases}$$

Therefore, there are at most $2^{\frac{k^{i+1}-1}{k-1}}$ (resp. 2^{i+1}) distinct $\phi_p^{(i)}$, for finite alphabets of size at least two (resp. equal to one). As $p \sim_i q$ if and only if $\phi_p^{(i)} = \phi_q^{(i)}$, there are at most $2^{\frac{k^{i+1}-1}{k-1}}$ (resp. 2^{i+1}) distinct subsets in the partition at the i-th iteration of Moore's algorithm. Since the algorithm halts when the n parts are computed, one has

$$\begin{cases} n \leq 2^{\frac{k^{\textsc{Moore}(\mathcal{A})+1}-1}{k-1}} & \text{when } k \geq 2, \\ n \leq 2^{\textsc{Moore}(\mathcal{A})+1} & \text{when } k = 1, \end{cases}$$

concluding the proof since the cost of an iteration is $\Theta(kn)$. □

4 Average Case Analysis for Accessible Deterministic Complete Automata

4.1 Probabilistic Model and Main Result

The choice of the distribution is crucial for average case analysis of algorithms. Here we are considering an algorithm that builds the minimal automaton of the language recognized by a given accessible complete deterministic one. We focus our study on the average complexity of this algorithm for the uniform distribution over accessible complete deterministic automata with n states and as n tends toward infinity. Some extensions of the main result to other distributions are given in Section 5.

Note that for the uniform distribution over automata with n states, the probability for a given set to be the set of final states is equal to 2^n. Therefore, even if it is possible, the probability that all states are final (or non-final) is exponentially unlikely.

The average case analysis of algorithms handling automata is often difficult. The general framework of this domain [24] is based on a good understanding of the enumeration properties of studied objects, most often given by generating functions. For accessible deterministic automata, this first step is already complex. Although the asymptotic number of such automata is known, it can not be easily handled: a result due to Korshunov [25], rewritten in terms of Stirling numbers of the second kind in [19] and generalized to possibly incomplete automata in [21], is that the number of accessible deterministic automata with n states is asymptotically equal to $\alpha\beta^n n^{(|A|-1)n}$ where α and β are constants depending on the cardinality $|A|$ of the alphabet, and α depends on whether we are considering complete automata or possibly incomplete automata.

As we shall see in the analysis presented in the following, some good properties of Myhill-Nerode equivalence allow us to work independently and uniformly on each transition structure. In this way the enumeration problem mentioned above can be avoided. Nevertheless it should be necessary to enumerate some subsets of this set of automata in order to obtain a more precise result. One refers the readers to the discussion of Section 6 for more details.

The main result of this article is the following theorem. The remainder of this section is devoted to its proof.

Theorem 2. *For any fixed integer $k \geq 1$ and for the uniform distribution over the accessible complete deterministic automata of size n over a k-letters alphabet, the average complexity of Moore's state minimization algorithm is $\mathcal{O}(n \log n)$.*

Note that this bound is valid for any size $k \geq 1$ of the alphabet considered. Moreover, as we shall see in Section 5, it is tight for a unary alphabet.

4.2 Dependency Graph

Before proving Theorem 2. we introduce some definitions and preliminary results. Let $\mathcal{T}$ be a fixed transition structure with n states and ℓ be an integer such that $1 \leq \ell < n$. Let p, q, p', q' be four states of $\mathcal{T}$ such that $p \neq q$ and $p' \neq q'$.

Define $\mathcal{F}_\ell(p, q, p', q')$ as the set of sets of final states F for which in the automaton $(\mathcal{T}, F)$ the states p and q are $(\ell-1)$-equivalent, but not ℓ-equivalent, because of a word of length ℓ mapping p to p' and q to q' where p' and q' are not 0-equivalent. In other words $\mathcal{F}_\ell(p, q, p', q')$ is the following set:

$$\mathcal{F}_\ell(p, q, p', q') = \{F \subset \{1, \ldots, n\} \mid \text{for } (\mathcal{T}, F),\ p \sim_{\ell-1} q, [\![p' \in F]\!] \neq [\![q' \in F]\!], \\ \exists u \in A^\ell,\ p \cdot u = p' \text{ and } q \cdot u = q'\}$$

Note that when ℓ grows, the definition of $\mathcal{F}_\ell$ is more constrained and consequently there are fewer non-empty sets $\mathcal{F}_\ell$.

With the set $\mathcal{F}_\ell(p, q, p', q')$ one can define the undirected graph $\mathcal{G}_\ell(p, q, p', q')$, called the *dependency graph*, as follows:

- its set of vertices is $\{1, \ldots, n\}$, the set of states of $\mathcal{T}$;
- there is an edge (s, t) between two vertices s and t if and only if for all $F \in \mathcal{F}_\ell(p, q, p', q')$, $[\![s \in F]\!] = [\![t \in F]\!]$.

The dependency graph contains some information that is a basic ingredient of the proof: it is a convenient representation of necessary conditions for a set of final states

to be in $\mathcal{F}_\ell(p,q,p',q')$, that is, for Moore's algorithm to require more than ℓ iterations because of p, q, p' and q'. These necessary conditions will be used to give an upper bound of the cardinality of $\mathcal{F}_\ell(p,q,p',q')$ in Lemma 6.

A first bound on the cardinality of $\mathcal{F}_\ell(p,q,p',q')$ is given by the following simple result:

Lemma 3. *If $\mathcal{G}_\ell(p,q,p',q')$ has m connected components, then $|\mathcal{F}_\ell(p,q,p',q')| \leq 2^m$.*

Proof. It directly follows from the definition of $\mathcal{G}_\ell(p,q,p',q')$: two states in the same connected components must be both final or both not final for the set of final states to be in $\mathcal{F}_\ell(p,q,p',q')$. □

4.3 Bounding Above the Number of Connected Components

For this part, we assume that $\mathcal{F}_\ell(p,q,p',q')$ is not empty.

In the following, we extract a big enough acyclic subgraph (*i.e.*, a forest) from $\mathcal{G}_\ell(p,q,p',q')$, that gives an upper bound of the number of connected components.

For any integer $\ell \in \{1,\dots,n-1\}$ and any states $p,q,p',q' \in \{1,\dots,n\}$ with $p \neq q$, $p' \neq q'$, let $u = u_1 \dots u_\ell$ with $u_i \in A$ be the smallest (for the lexicographic order) word of length ℓ such that $p \cdot u = p'$ and $q \cdot u = q'$. Note that every word u of length ℓ such that $p \cdot u = p'$ and $q \cdot u = q'$ can be used. But a non-ambiguous choice of this word u guarantees a complete description of the following construction.

For every $i \in \{0,\dots,\ell-1\}$, let $G_{\ell,i}$ be the subgraph of $\mathcal{G}_\ell(p,q,p',q')$ whose edges are defined as follows. An edge (s,t) is in $\mathcal{G}_{\ell,i}$ if and only if there exists a prefix v of u of length less than or equal to i such that $s = p \cdot v$ and $t = q \cdot v$. In other words the edges of $\mathcal{G}_{\ell,i}$ are exactly the edges $(p \cdot v, q \cdot v)$ between the states $p \cdot v$ and $q \cdot v$ where v ranges over the prefixes of u of length less than or equal to i. Such edges belong to $\mathcal{G}_\ell(p,q,p',q')$ since $p \sim_{\ell-1} q$. An illustration of the arguments used in the proof is presented in Figure 4.

Lemma 4. *The following properties hold for the subgraphs $G_{\ell,i}$:*

1. *For each $i \in \{0,\dots,\ell-2\}$, $G_{\ell,i}$ is a strict subgraph of $G_{\ell,i+1}$.*
2. *For each $i \in \{0,\dots,\ell-1\}$, $G_{\ell,i}$ contains $i+1$ edges.*
3. *For each $i \in \{0,\dots,\ell-1\}$, $G_{\ell,i}$ contains no loop.*
4. *For each $i \in \{0,\dots,\ell-1\}$, if there exists a path in $G_{\ell,i}$ from s to t, then $s \sim_{\ell-1-i} t$ in every automaton $(\mathcal{T},F)$ with $F \in \mathcal{F}_\ell(p,q,p',q')$.*

Proof. We prove each of the properties of Lemma 4.

1. The graph $G_{\ell,i+1}$ is obtained from $G_{\ell,i}$ by adding an edge from $p \cdot w$ to $q \cdot w$, where w is the prefix of u of length $i+1$. This edge does not belong to $G_{\ell,i}$, otherwise there would exist a strict prefix z of w such that either $p \cdot z = p \cdot w$ and $q \cdot z = q \cdot w$ or $p \cdot z = q \cdot w$ and $q \cdot z = p \cdot w$. In this case, let w' be the word such that $u = ww'$, then either $p \cdot zw' = p'$ and $q \cdot zw' = q'$, or $p \cdot zw' = q'$ and $q \cdot zw' = p'$. Therefore there would exist a word of length less than ℓ, zw', such that, for $F \in \mathcal{F}_\ell(p,q,p',q')$, $[\![p \cdot zw' \in F]\!] \neq [\![q' \cdot zw' \in F]\!]$ which is not possible since $p \sim_{\ell-1} q$ and $\mathcal{F}_\ell(p,q,p',q')$ is not empty.
2. It is a consequence of Property *1* that can be established by induction on i since $G_{\ell,0}$ has only one edge between p and q.
3. For any automaton $(\mathcal{T},F)$ with $F \in \mathcal{F}_\ell(p,q,p',q')$, which is not empty, $p \nsim q$. Hence for any prefix v of u, $p \cdot v \neq q \cdot v$.

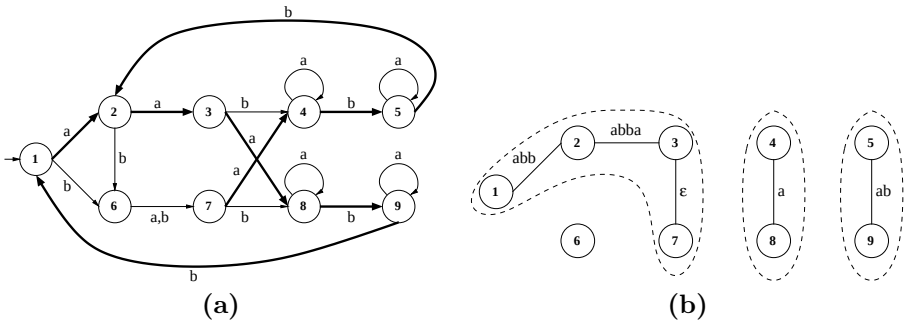

Figure 4: Illustration of the construction for $n = 9$, $\ell = 5$, $p = 3$, $q = 7$, $p' = 3$ and $q' = 8$ on a given transition structure. **(a)** $u = abbaa$ is the smallest word of length 5, for the lexicographic order, such that $3 \cdot u = 3$ and $7 \cdot u = 8$. The set $\mathcal{F}_5(3, 7, 3, 8)$ is not empty, as it contains $\{4, 8\}$. The bold transitions are the ones followed when reading u from p and from q. **(b)** The construction of an acyclic subgraph of $\mathcal{G}_5(3, 7, 3, 8)$ with 5 edges. To each strict prefix v of $u = abbaa$ is associated an edge between $3 \cdot v$ and $7 \cdot v$. It encodes some necessary conditions for a set of final states F to be in $\mathcal{F}_5(3, 7, 3, 8)$, as two states in the same connected component must be either both final or both not final.

4. The property is proved by induction on $i \in \{0, \ldots, \ell-1\}$. For $i = 0$, $G_{\ell,0}$ contains only one edge, between p and q, and $p \sim_{\ell-1} q$. Assume that the property holds for $i \in \{0, \ldots, \ell-2\}$. Let x and y be two vertices such that there exists a simple path from x to y in $G_{\ell,i+1}$. If this path is in $G_{\ell,i}$, *i.e.* it does not use the added edge, then $x \sim_{\ell-1-i} y$ by induction hypothesis, hence $x \sim_{\ell-2-i} y$. Otherwise the path use the added edge between $p \cdot v$ and $q \cdot v$, where v is the prefix of length $i+1$ of u. Assume by symmetry that the path reach $p \cdot v$ first. Then the part of the path between x and $p \cdot v$ belongs to $G_{\ell,i}$ and therefore $x \sim_{\ell-1-i} p \cdot v$ by hypothesis. Similarly, $y \sim_{\ell-1-i} q \cdot v$. But $p \cdot v \sim_{\ell-2-i} q \cdot v$ since $p \sim_{\ell-1} q$. Hence $x \sim_{\ell-2-i} y$, concluding the proof by induction.

So the four properties are established. □

Lemma 5. *The subgraph $G_{\ell,\ell-1}$ is an acyclic subgraph of $\mathcal{G}_\ell(p, q, p', q')$ with ℓ edges.*

Proof. We first prove that $G_{\ell,\ell-1}$ is acyclic. Assume that it is not true, and let $j \geq 1$ be the smallest integer such that $G_{\ell,j}$ contains a cycle. By Property *1* of Lemma 4, $G_{\ell,j}$ is obtained from $G_{\ell,j-1}$ by adding a new edge between $p \cdot w$ and $q \cdot w$ where w is the prefix of length j of u. As $G_{\ell,j-1}$ is acyclic, this edge forms a cycle in $G_{\ell,j}$. Hence in $G_{\ell,j-1}$ there already exists a path between $p \cdot w$ and $q \cdot w$. Therefore by Property *4*, $p \cdot w \sim_{\ell-j} q \cdot w$ in any automaton $(\mathcal{T}, F)$ with $F \in \mathcal{F}_\ell(p, q, p', q')$. Let w' be the word such that $u = ww'$. The length of w' is $\ell - j$, hence $p \cdot u$ and $q \cdot u$ are both in F or both not in F, which is not possible since $F \in \mathcal{F}_\ell(p, q, p', q')$.

Thus $G_{\ell,\ell-1}$ is an acyclic subgraph of $\mathcal{G}_\ell(p, q, p', q')$. It contains exactly ℓ edges according to Property *2*, which concludes the proof. □

Lemma 6. *Given a transition structure $\mathcal{T}$ of size $n \geq 1$ and an integer ℓ with $1 \leq \ell < n$, for all states p, q, p', q' of $\mathcal{T}$ with $p \neq q$ and $p' \neq q'$ the following result holds:*

$$|\mathcal{F}_\ell(p, q, p', q')| \leq 2^{n-\ell}.$$

Proof. If $\mathcal{F}_\ell(p, q, p', q')$ is empty, the result holds. Otherwise, from Lemma 5, there exists an acyclic subgraph G of $\mathcal{G}_\ell(p, q, p', q')$ with ℓ edges. Hence, G contains a forest with ℓ edge, and therefore has at most $n - \ell$ connected components. We conclude using Lemma 3. □

4.4 Proof of Theorem 2

Proposition 3. *Let $k \geq 1$. There exists a positive real constant C such that for any positive integer n and any deterministic complete transition structure $\mathcal{T}$ of size n over a k-letters alphabet, for the uniform distribution over the sets F of final states, the average number of iterations of the main loop of Moore's algorithm applied to $(\mathcal{T}, F)$ is bounded above by $C \log n$.*

Proof. Let $\mathcal{T}$ be a complete deterministic transition structure of size n over a k-letters alphabet. Denote by $\mathcal{F}^{\geq \ell}$ the set of sets F of final states such that the execution of Moore's algorithm on $(\mathcal{T}, F)$ requires more than ℓ iterations or equivalently such that $(\mathcal{T}, F) \in \mathcal{A}_n^{(m)}$ with $m \geq \ell$ (see Section 2.3 for notation).

A necessary condition for F to be in $\mathcal{F}^{\geq \ell}$ is that there exist two states p and q with $p \neq q$ and such that $p \sim_{\ell-1} q$ and $p \not\sim_\ell q$. Therefore there is a word u of length ℓ such that $[\![p \cdot u]\!] \neq [\![q \cdot u]\!]$. Hence $F \in \mathcal{F}_\ell(p, q, p \cdot u, q \cdot u)$ and

$$\mathcal{F}^{\geq \ell} = \bigcup_{\substack{p,q,p',q' \in \{1,\ldots,n\} \\ p \neq q,\ p' \neq q'}} \mathcal{F}_\ell(p, q, p', q').$$

In this union the sets $\mathcal{F}_\ell(p, q, p', q')$ are not disjoint, but this characterization of $\mathcal{F}^{\geq \ell}$ is precise enough to obtain a useful upper bound of the cardinality of $\mathcal{F}^{\geq \ell}$. From the description of $\mathcal{F}^{\geq \ell}$ we get

$$|\mathcal{F}^{\geq \ell}| \leq \sum_{\substack{p,q,p',q' \in \{1,\ldots,n\} \\ p \neq q,\ p' \neq q'}} |\mathcal{F}_\ell(p, q, p', q')|.$$

Using Lemma 6 and estimating the number of choices of the four points p, q, p', q', we have

$$|\mathcal{F}^{\geq \ell}| \leq n(n-1)\, n(n-1) 2^{n-\ell} \leq n^4 2^{n-\ell}. \tag{1}$$

Moreover for a fixed integer ℓ and for the uniform distribution over the sets F of final states, the average number of iterations of the main loop of Moore's algorithm is by definition

$$\frac{1}{2^n} \sum_{F \subset \{1,\ldots,n\}} \textsc{Moore}(\mathcal{T}, F) = \frac{1}{2^n} \sum_{F \in \mathcal{F}^{<\ell}} \textsc{Moore}(\mathcal{T}, F) + \frac{1}{2^n} \sum_{F \in \mathcal{F}^{\geq \ell}} \textsc{Moore}(\mathcal{T}, F),$$

where $\mathcal{F}^{<\ell}$ is the complement of $\mathcal{F}^{\geq \ell}$ in the set of all subsets of states. From Lemma 1, for any $F \in \mathcal{F}^{<\ell}$, $\textsc{Moore}(\mathcal{T}, F) \leq \ell$. Therefore, since $|\mathcal{F}^{<\ell}| \leq 2^n$

$$\frac{1}{2^n} \sum_{F \in \mathcal{F}^{\leq \ell}} \textsc{Moore}(\mathcal{T}, F) \leq \ell.$$

Bounding above $\textsc{Moore}(\mathcal{T}, F)$ when $F \in \mathcal{F}^{\geq \ell}$ with the help of Lemma 1 and estimating $|\mathcal{F}^{\geq \ell}|$ with Equation (1) we have

$$\frac{1}{2^n} \sum_{F \subset \mathcal{F}^{\geq \ell}} \textsc{Moore}(\mathcal{T}, F) \leq n^5 2^{-\ell}.$$

Finally, choosing $\ell = \lceil 5 \log_2 n \rceil$, we obtain that there exists positive real C such that

$$\frac{1}{2^n} \sum_{F \subset \{1,\ldots,n\}} \textsc{Moore}(\mathcal{T}, F) \leq \lceil 5 \log_2 n \rceil + n^5 2^{-\lceil 5 \log_2 n \rceil} \leq C \log n,$$

concluding the proof. □

Now we prove Theorem 2: Let $\mathcal{T}_n$ denote the set of accessible complete deterministic transition structures with n states. For a transition structure $\mathcal{T} \in \mathcal{T}_n$, there are exactly 2^n distinct automata $(\mathcal{T}, F)$.

Recall that the set $\mathcal{A}_n$ of accessible complete deterministic automata with n states is in bijection with the pairs $(\mathcal{T}, F)$ consisting of an accessible complete deterministic transition structure $\mathcal{T} \in \mathcal{T}_n$ with n states and a subset $F \subset \{1, \ldots, n\}$ of final states. Therefore, for the uniform distribution over the set $\mathcal{A}_n$, the average number of iterations of the main loop when Moore's algorithm is applied to an element of $\mathcal{A}_n$ is

$$\frac{1}{|\mathcal{A}_n|} \sum_{\mathcal{A} \in \mathcal{A}_n} \textsc{Moore}(\mathcal{A}) = \frac{1}{2^n |\mathcal{T}_n|} \sum_{\mathcal{T} \in \mathcal{T}_n} \sum_{F \subset \{1,\ldots,n\}} \textsc{Moore}(\mathcal{T}, F)$$

Using Proposition 3 we get

$$\frac{1}{|\mathcal{A}_n|} \sum_{\mathcal{A} \in \mathcal{A}_n} \textsc{Moore}(\mathcal{A}) \leq \frac{1}{|\mathcal{T}_n|} \sum_{\mathcal{T} \in \mathcal{T}_n} C \log n \leq C \log n.$$

Hence the average number of iterations is bounded above by $C \log n$, and from Lemma 1 the average complexity of Moore's algorithm is bounded above by $C_1 C n \log n$, concluding the proof.

5 Related Results

5.1 Tightness for Unary Automata

In this section we prove that the bound $\mathcal{O}(n \log n)$ is optimal for the uniform distribution on unary automata with n states, that is, automata on a one-letter alphabet.

Proposition 4. *For the uniform distribution on unary automata with n states, the average time complexity of Moore's state minimization algorithm is $\Theta(n \log n)$.*

To prove Proposition 4 we use the following result from [14]:

Proposition 5 ([14])**.** *For the uniform distribution on unary automata with n states, the probability for an automaton to be minimal is asymptotically equivalent to $\frac{1}{2}$.*

Proof. (of Proposition 4) From Theorem 2 this time complexity is $\mathcal{O}(n \log n)$. It remains to study the lower bound of the average time complexity of Moore's algorithm.

From Proposition 2, there exists a positive constant $C > 0$ such that, for any $n \geq 1$, the complexity of Moore's algorithm applied to a unary automaton with n states is at least $Cn \log n$. Let m_n denote the number of minimal unary automata

with n states and a_n the number of accessible complete deterministic unary automata with n states. Taking into account only the contribution of minimal automata, the average complexity of Moore's algorithm is bounded below by

$$\frac{m_n}{a_n} Cn \log n \sim \frac{1}{2} Cn \log n,$$

using Proposition 5 to compute the equivalent. Hence the average complexity is in $\Omega(n \log n)$, concluding the proof. □

5.2 Binomial Distribution for the Sets of Final States

A natural extension of Theorem 2 consists in using a Bernoulli distribution for the probability of each state to be final. Then the number of final states has a binomial distribution.

Theorem 3. *Let $\mathcal{T}$ be an accessible complete deterministic transition structure with n state over a k-letters alphabet. For the distribution over the sets of final states where each state as a probability $x \in]0,1[$ to be final, the average complexity of Moore's state minimization algorithm is in $\mathcal{O}(n \log n)$.*

Proof. Let x be a fixed real number with $0 < x < 1$. Let $\mathcal{T}$ be a transition structure with n states. Consider the distribution over the sets of final states for $\mathcal{T}$ defined such that each state as a probability x to be final. The probability associated to a given subset F of $\{1, \ldots, n\}$ is then $P_F = x^{|F|}(1-x)^{n-|F|}$. For fixed values p, q, p', q' and ℓ, let $P_\ell(p, q, p', q')$ be the probability for a set of final states to be in $\mathcal{F}_\ell(p, q, p', q')$. Since $\mathcal{G}_\ell(p, q, p', q')$ only models a subset of constraints on the sets of final states of $\mathcal{F}_\ell(p, q, p', q')$, $P_\ell(p, q, p', q')$ is less than the probability for a set of final states to verify the constraints implied by the graph.

Let m be the number of connected components in $G_{\ell,\ell-1}$ containing at least two vertices and let $c_1, \ldots, c_m$ be the sizes of these components. Since $G_{\ell,\ell-1}$ is acyclic and contains exactly ℓ edges, the following equality holds:

$$\sum_{i=1}^{m} c_i = m + \ell$$

The probability for a random set to satisfy the conditions implied by the i-th connected component is equal to $x^{c_i} + (1-x)^{c_i}$ (either all states of the subset are final, either none). Let r be the real number defined by $r = \max\{x, 1-x\} \in [1/2, 1[$. For all $n \geq 1$, $x^n + (1-x)^n \leq r^{n-1}(x + 1 - x) = r^{n-1}$. Hence

$$P_\ell(p, p', q, q') \leq \Pi_{i=1}^{m} r^{c_i - 1} = r^{\sum_{i=1}^{m}(c_i - 1)} = r^\ell$$

For a fixed transition structure $\mathcal{T}$, the probability for a set of final states F to be such that $\textsc{Moore}(\mathcal{T}, F) \geq \ell$ is at most

$$\sum_{\substack{p,q,p',q' \in \{1,\ldots,n\} \\ p \neq q,\ p' \neq q'}} P_\ell(p, p', q, q') \leq n^4 r^\ell$$

Taking $\ell = 5 \log_r n$ concludes the proof. □

5.3 Possibly Incomplete Automata

We assume that in order to apply Moore's algorithm on an incomplete automaton, the first step consists in adding a sink state. From an average point of view, the distribution has been altered a bit, but not enough to change the result.

Proposition 6. *For the uniform distribution over the accessible possibly incomplete deterministic automata with n states on a k-letters alphabet, the average complexity of Moore's state minimization algorithm is in $\mathcal{O}(n \log n)$.*

Proof. Let $\mathcal{T}$ be a possibly incomplete transition structure, and $\mathcal{T}'$ be the complete transition structure associated with $\mathcal{T}$ (which can be equal to $\mathcal{T}$ if it was already complete). From Lemma 6 and since there are n or $n+1$ states in $\mathcal{T}'$, there are at most $2^{n+1-\ell}$ elements in $\mathcal{F}_\ell(p, p', q, q')$, for any $\ell \geq 1$ and any states p, p', q, q' such that $p \neq p'$ and $q \neq q'$. Note that this upper bound is a bit less tight than in the case of complete automata, since the sink state cannot be final. Nonetheless the bound is enough precise to prove the negligible contribution of the set of final states for which Moore's algorithm performs at least ℓ iterations, when $\ell = \lceil 5 \log n \rceil$, as in the proof of Theorem 2. □

5.4 Co-accessible Automata

When considering the uniform distribution on co-accessible automata with n states, the upper bound of the number of iterations of Moore's algorithm still holds. The main reason is that the proportion of co-accessible automata amongst accessible ones is big enough as it can be proved using the following result by Korshunov [26]:

Theorem 4 ([26])**.** *There exists a real constant $0 < c < 1$, depending on the size of the alphabet, such that for the uniform distribution over the accessible complete deterministic automata with n states, the probability for an automaton to be strongly connected tends toward c as n tends toward infinity.*

Proposition 7. *For the uniform distribution over the accessible and co-accessible complete deterministic automata with n on a k-letters alphabet, the average complexity of Moore's state minimization algorithm is in $\mathcal{O}(n \log n)$.*

Proof. Let $\mathcal{C}_n$ denote the set of accessible and co-accessible automata with n states. Let $\mathcal{C}_n^{<\ell}$ and $\mathcal{C}_n^{\geq \ell}$ denote the subsets of $\mathcal{C}_n$ consisting of the automata $\mathcal{A}$ such that respectively $\textsc{Moore}(\mathcal{A}) < \ell$ and $\textsc{Moore}(\mathcal{A}) \geq \ell$.

The number of n-state automata $\mathcal{A}$ such that $\textsc{Moore}(\mathcal{A}) \geq \ell$ is in $\mathcal{O}(\frac{1}{n}|\mathcal{A}_n|)$ when $\ell = \lceil 5 \log n \rceil$, using the same arguments as in the proof of Proposition 3. Hence $|\mathcal{C}_n^{\geq \ell}| \leq C \frac{1}{n} |\mathcal{A}_n|$ for some constant $C > 0$.

Therefore the average number of iterations is, when $\ell = \lceil 5 \log n \rceil$,

$$\begin{aligned}\frac{1}{|\mathcal{C}_n|} \sum_{\mathcal{A} \in \mathcal{C}_n} \textsc{Moore}(\mathcal{A}) &= \frac{1}{|\mathcal{C}_n|} \sum_{\mathcal{A} \in \mathcal{C}_n^{<\ell}} \textsc{Moore}(\mathcal{A}) + \frac{1}{|\mathcal{C}_n|} \sum_{\mathcal{A} \in \mathcal{C}_n^{\geq \ell}} \textsc{Moore}(\mathcal{A}) \\ &\leq \frac{|\mathcal{C}_n^{<\ell}|}{|\mathcal{C}_n|} \ell + n \frac{|\mathcal{C}_n^{\geq \ell}|}{|\mathcal{C}_n|} \leq \ell + \frac{n}{|\mathcal{C}_n|} \mathcal{O}\left(\frac{1}{n} |\mathcal{A}_n|\right)\end{aligned}$$

Moreover as a consequence of Theorem 4, $|\mathcal{A}_n|/|\mathcal{C}_n| = \mathcal{O}(1)$, and thus

$$\frac{1}{|\mathcal{C}_n|} \sum_{\mathcal{A} \in \mathcal{C}_n} \textsc{Moore}(\mathcal{A}) \leq \ell + \mathcal{O}(1) = \mathcal{O}(\log n),$$

concluding the proof. □

6 Discussion

6.1 Fixed Number of Final States

All the distributions studied in this article produce automata having a large number of final states with high probability. It is quite natural, for some applications, to consider distributions of automata with only a few final states. In that case the method we used is unlikely to work, as illustrated by the following result.

Proposition 8. *For the uniform distribution over unary automata with exactly one final state, the average time complexity of Moore's state minimization algorithm is in* $\Theta(n^2)$.

Proof. As it is described in [14] the transition structure of a unary automaton $\mathcal{U}$ over the alphabet $\{a\}$ is characterized by the number of states n and the arrival state l of the transition $n \cdot a$. If $l = 1$, meaning that the transition leaving the last state of the automaton ends in the first state, the automaton is a loop. Otherwise a unary automaton contains both a loop and a queue, where the loop is the set of states $L = \{l, .., n\}$ and the queue is the set $Q \setminus L$.

Let $\mathcal{U}$ be a unary automaton with n states and one final state f. For any state p, let i_p be the greatest integer such that for all $j \in \mathbb{N}$ with $j < i_p$, $p \cdot a^j$ is not final, if it exists, and 0 otherwise. Let q be the state such that i_q is maximal. If $i_q \geq 1$ then, from the shape of the automaton, $i_{q \cdot a} = i_q - 1$. In particular $\textsc{Moore}(\mathcal{U}) = i_q + 1$, as q and $q \cdot a$ are $i_q - 1$ equivalent, but not i_q equivalent. This is also true if $i_q = 0$, when the initial state is also the final state and $\mathcal{U}$ is not a loop: $\textsc{Moore}(\mathcal{U}) = 1$.

The states that maximize i_q can be either the initial state 1 or the state $f \cdot a$ (or both). If $f \notin L$, *i.e.* $f < l$, then the initial state maximizes i_q and $\textsc{Moore}(\mathcal{U}) = i_1 + 1 = (f - 1) + 1 = f$. If f is in the loop, one has $i_1 = f - 1$ and $i_{f \cdot a} = n - l$, since the length of the loop is $n - l + 1$. Hence when $f \in L$ $\textsc{Moore}(\mathcal{U}) = \max\{f, n - l + 1\}$.

The number of unary automata with a unique final state is n^2 since all choices of $l \in \{1, .., n\}$ and $f \in \{1, .., n\}$ correspond to distinct automata. The average number of iterations is therefore equal to

$$\frac{1}{n^2} \sum_{l=1}^{n} \left(\sum_{f=1}^{l-1} f + \sum_{f=l}^{n-l} (n - l + 1) + \sum_{f=n-l+1}^{n} f \right) = \frac{1}{6n^2}(n^3 + 3n^2 - n) = \Theta(n),$$

concluding the proof. □

Nonetheless Moore's algorithm over automata with a unique final state seems to perform less than $\Theta(n^2)$ operations when the alphabet is of size at least two, as illustrated in Figure 5. A proof of such a result cannot be based on an independent treatment of transition structures and sets of final states, as we did in this article, since the restrictions to some transition structures provide a quadratic complexity (Proposition 8).

6.2 Hopcroft's Algorithm

Hopcroft's algorithm performs the minimization in time $\mathcal{O}(n \log n)$ in the worst case. As stated in [9] Hopcroft's algorithm is closely related to Moore's algorithm, refining the partition part by part, avoiding useless computations. It is likely that Hopcroft's

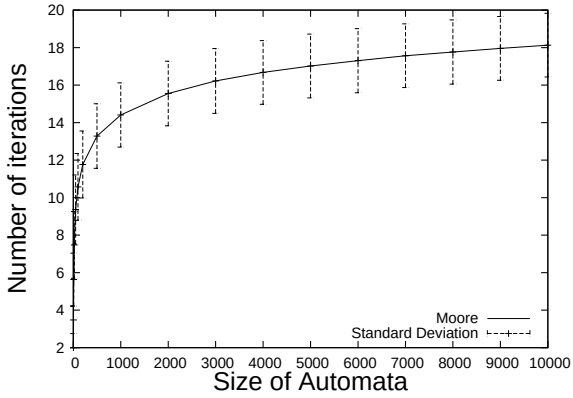

Figure 5: Experimental study of the average number of iterations in Moore's algorithm for the uniform distribution over automata with only one final states. For each size the values are computed from 20 000 random automata over a 2-letter alphabet.

algorithm halts quickly when Moore's algorithm does, at least for some choices of implementations (there is a set of tasks to perform in Hopcroft's algorithm, that can be implemented in different ways: queue, stack, and so on).

The drawback of Hopcroft's algorithm is that it uses complicated data structures, that require lots of elementary instructions to be updated at each step.

So, if one of the distributions analyzed in this paper models correctly a given practical case, we believe that Moore's algorithm is worth considering, as it can be faster than Hopcroft's algorithm.

6.3 A Conjecture

To conclude this paper we conjecture that when the alphabet is not unary the lower bound given in Proposition 2 for minimal automata is tight for the general case or, in other words, that the average complexity of Moore's algorithm for the uniform distribution over automata with n states is in $\Theta(n \log\log n)$. A first reason is that the proportion of minimal automata amongst accessible deterministic ones is conjectured positive (see [19]). It it is true, the lower bound in $\Omega(n \log\log n)$ would hold for the distribution over all automata. Secondly, the tests of Figure 1 show a very slow growth of the average number of iterations as n tends toward infinity.

References

[1] Moore, E.F.: Gedanken experiments on sequential machines. In: Automata Studies. Princeton U. (1956) 129–153.

[2] Hopcroft, J.E.: An n log n algorithm for minimizing states in a finite automaton. Technical report, Stanford, CA, USA (1971)

[3] Gries, D.: Describing an algorithm by Hopcroft. Acta Inf. **2** (1973) 97–109.

[4] Knuutila, T.: Re-describing an algorithm by Hopcroft. Theor. Comput. Sci. **250**(1-2) (2001) 333–363.

[5] Berstel, J., Carton, O.: On the complexity of Hopcroft's state minimization algorithm. In Domaratzki, M., Okhotin, A., Salomaa, K., Yu, S., eds.: CIAA. Volume 3317 of Lecture Notes in Computer Science., Springer (2004) 35–44.

[6] Castiglione, G., Restivo, A., Sciortino, M.: Hopcroft's algorithm and cyclic automata. In Martín-Vide, C., Otto, F., Fernau, H., eds.: LATA. Volume 5196 of Lecture Notes in Computer Science., Springer (2008) 172–183.

[7] Castiglione, G., Restivo, A., Sciortino, M.: On extremal cases of Hopcroft's algorithm. In Maneth, S., ed.: CIAA. Volume 5642 of Lecture Notes in Computer Science., Springer (2009) 14–23.

[8] Blum, N.: An O(n log n) implementation of the standard method for minimizing n-state finite automata. Inf. Process. Lett. **57**(2) (1996) 65–59.

[9] Lothaire, M.: Applied Combinatorics on Words. Volume 105 of Encyclopedia of mathematics and its application. Cambridge University Press. (2005)

[10] Valmari, A., Lehtinen, P.: Efficient minimization of DFAs with partial transition. In Albers, S., Weil, P., eds.: STACS. Volume 08001 of Dagstuhl Seminar Proceedings., Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany (2008) 645–656.

[11] Beal, M.P., Crochemore, M.: Minimizing incomplete automata. In: Finite-State Methods and Natural Language Processing (FSMNLP'08). (2008) 9–16.

[12] Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)

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

[14] Nicaud, C.: Average state complexity of operations on unary automata. In Kutylowski, M., Pacholski, L., Wierzbicki, T., eds.: MFCS. Volume 1672 of Lecture Notes in Computer Science., Springer (1999) 231–240.

[15] Beal, M.P., Crochemore, M.: Minimizing local automata. In G. Caire, M.F., ed.: IEEE International Symposium on Information Theory (ISIT'07). (2007) 1376–1380.

[16] Brzozowski, J.A.: Canonical regular expressions and minimal state graphs for definite events. In: Symposium on the Mathematical Theory of Automata. Volume 12., Polytechnic Institute of Brooklyn, New York, Polytechnic Press (1962) 529–561.

[17] Champarnaud, J.M., Khorsi, A., Paranthoen, T.: Split and join for minimizing: Brzozowski's algorithm. In: PSC'02 Proceedings. (2002) 96–104.

[18] Watson, B.W.: A taxonomy of finite automata minimization algorithms. Technical Report of Faculty of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands (1994)

[19] Bassino, F., Nicaud, C.: Enumeration and random generation of accessible automata. Theor. Comput. Sci. **381** (2007) 86–104.

[20] Bassino, F., David, J., Nicaud, C.: REGAL: a library to randomly and exhaustively generate automata. In: Implementation and Application of Automata, 12th International Conference, CIAA 2007. Volume Lecture Notes in Computer Science 4783. (2007) 303–305.

[21] Bassino, F., David, J., Nicaud, C.: Enumeration and random generation of possibly incomplete deterministic automata. Pure Mathematics and Applications (to appear)

[22] Bassino, F., David, J., Nicaud, C.: On the average complexity of Moore's state minimization algorithm. In Albers, S., Marion, J.Y., eds.: STACS 2009. Volume 09001 of Dagstuhl Seminar Proceedings., Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany (2009) 123–134

[23] Nerode, A.: Linear automaton transformation. In: Proc. American Mathematical Society. (1958) 541–544.

[24] Flajolet, P., Sedgewick, R.: Analytic combinatorics. Cambridge University Press (2009)

[25] Korshunov, D.: Enumeration of finite automata. Problemy Kibernetiki **34** (1978) 5–82.

[26] Korshunov, A.D.: On the number of non-isomorphic strongly connected finite automata. Elektronische Informationsverarbeitung und Kybernetik **22**(9) (1986) 459–462.