On the Average Complexity of Moore's State Minimization Algorithm

Frédérique Bassino, Julien David, Cyril Nicaud

Symposium on Theoretical Aspects of Computer Science 2009 (Freiburg), pp. 123–134
www.stacs-conf.org

ON THE AVERAGE COMPLEXITY OF MOORE'S STATE MINIMIZATION ALGORITHM

FRÉDÉRIQUE BASSINO [1] AND JULIEN DAVID [2] AND CYRIL NICAUD [2]

[1] LIPN UMR 7030, Université Paris 13 - CNRS, 99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France.
E-mail address: Frederique.Bassino@lipn.univ-paris13.fr

[2] Institut Gaspard Monge, Université Paris Est, 77454 Marne-la-Vallée Cedex 2, France
E-mail address: Julien.David@univ-paris-est.fr,Cyril.Nicaud@univ-paris-est.fr

Abstract. We prove that, for any arbitrary finite alphabet and for the uniform distribution over deterministic and accessible automata with n states, the average complexity of Moore's state minimization algorithm is in $\mathcal{O}(n \log n)$. Moreover this bound is tight in the case of unary 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. As a consequence, state minimization algorithms that compute the minimal automaton of a regular language, given by a deterministic automaton, are fundamental.

Moore proposed a solution [15] 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 [11] 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, see [1, 5, 9, 12] for instance. Finally Brzozowski's algorithm [6, 7] is different from the other ones. Its inputs may be non-deterministic automata.

1998 ACM Subject Classification: F.2 Analysis of algorithms and problem complexity.
Key words and phrases: finite automata, state minimization, Moore's algorithm, average complexity.

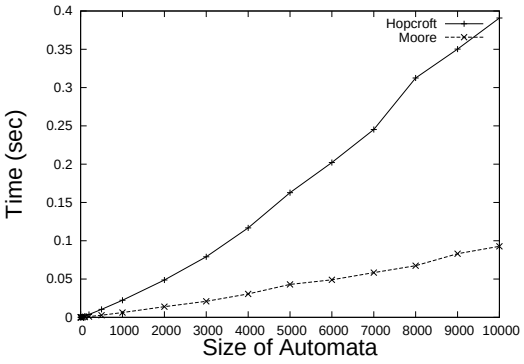

Figure 1: Time complexity of Moore's and Hopcroft's algorithms

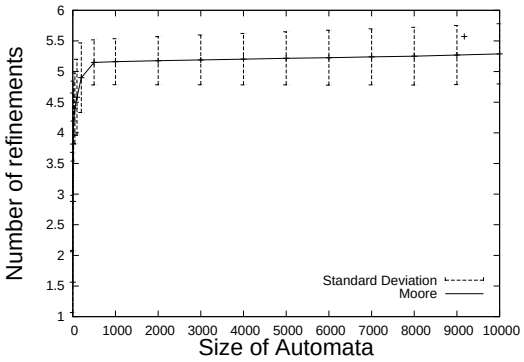

Figure 2: Number of partition refinements in Moore's algorithm

The results of Fig.1 and Fig.2 were obtained with the C++ library REGAL (available at: `http://regal.univ-mlv.fr/`) to randomly generate deterministic accessible automata [2, 3, 4]. Each value is computed from 20 000 automata over a 2-letter alphabet.

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 [17], which presents a taxonomy of minimization algorithms, for a more exhaustive list.

In this paper we study the average time complexity of Moore's algorithm. From an experimental point of view, the average complexity of Moore's algorithm seems to be smaller than the complexity of Hopcroft's algorithm (Fig.1) and the number of partition refinements increases very slowly as the size of the input grows (Fig.2). In the following we mainly prove that in average, for the uniform distribution, Moore's algorithm performs only $\mathcal{O}(\log n)$ refinements, thus its average complexity is in $\mathcal{O}(n \log n)$.

After briefly recalling the basics of minimization of automata in Section 2, we prove in Section 3 that the average time complexity of Moore's algorithm is $\mathcal{O}(n \log n)$ and show in Section 4 that this bound is tight when the alphabet is unary. The paper closes with a short discussion about generalizations of our main theorem to Bernoulli distributions and to incomplete automata in Section 5, and the presentation of a conjecture based on the slow growth of the number of refinements (Fig.2 when the alphabet is not unary in Section 6.

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 minimization of automata [10, 14, 18]. We only record a few definitions and results that will be useful for our purpose.

2.1. Finite automata

A *finite deterministic automaton* $\mathcal{A}$ is a quintuple $\mathcal{A} = (A, Q, \cdot, q_0, F)$ where Q is a finite set of *states*, A 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 deterministic and accessible transition structure with n states there are exactly 2^n distinct deterministic and accessible 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 complete accessible deterministic automata and complete accessible 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 $\{1, \cdots, n\}$.

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, $[\![p \cdot u \in F]\!] = [\![q \cdot u \in F]\!]$, where the Iverson bracket $[\![Cond]\!]$ is equal to 1 if the condition $Cond$ is satisfied and 0 otherwise. 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*. An equivalence relation $\equiv$ defined on the set of states Q of a deterministic automaton is said to be *right invariant* when

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

The following proposition [10, 14, 18] summarizes the properties of Myhill-Nerode equivalence that will be used in the next sections.

Proposition 2.1. *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 2.2. *For any complete, accessible and deterministic automaton $\mathcal{A}$, the automaton $\mathcal{A}/\sim$ is the unique smallest 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 2.2 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 the uniqueness defines a bijection between regular language and minimal automata. For instance, to check whether two regular languages are equal, one can compare their minimal automata. It is one of the motivations for the algorithmic study of the computation of the minimal automaton of a language.

2.3. Moore's state minimization algorithm

In this section we describe the algorithm due to Moore [15] which computes the minimal automaton of a regular language represented by a deterministic automaton. Recall that Moore's algorithm builds the partition of the set of states corresponding to Myhill-Nerode equivalence. It mainly relies on properties (2) and (3) of Proposition 2.1: the partition π is initialized according to the 0-equivalence $\sim_0$, then at each iteration the partition corresponding to the $(i+1)$-equivalence $\sim_{i+1}$ is computed from the one corresponding to the 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 computed from the resulting partition since it is the quotient automaton by Myhill-Nerode equivalence.

Algorithm 1: Moore

```
1  if F = ∅ then
2  |  return (A, {1}, ∗, 1, ∅)
3  end
4  if F = {1, ··· , n} then
5  |  return (A, {1}, ∗, 1, {1})
6  end
7  forall p ∈ {1, ··· , n} do
8  |  π'[p] = [[p ∈ F]]
9  end
10 π = undefined
11 while π ≠ π' do
12 |  π = π'
13 |  compute the partition π' s.t.
   |  π'[p] = π'[q] iff π[p] = π[q]
14 |  and ∀a ∈ A π[p · a] = π[q · a]
15 end
16 return the quotient of A by π
```

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

In the description of Moore's algorithm, $*$ denotes the function such that $1 * a = 1$ for all $a \in A$. Lines 1-6 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 7-9 consists of the initialization of π' to the partition of $\sim_0$, π is initially undefined. Lines 11-14 are the main loop of the algorithm

where π is set to π' and the new π' is computed. Line 13 is described more precisely in the algorithm on the right: with each state p is associated a $k+1$-uple $s[p]$ such that two states should be in the same part in π' when they have the same $k+1$-uple. The matches are found by sorting the states according to their associated string.

The worst-case time complexity of Moore's algorithm is in $\mathcal{O}(n^2)$. 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 3.1). For sake of completeness we also give a justification of this statement.

For any integer $n \geq 1$ and any $m \in \{0, \cdots, n-2\}$, we denote by $\mathcal{A}_n^{(m)}$ the set of automata with n states for which m is the smallest integer such that the 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 2.3. *For any automaton $\mathcal{A}$ of $\mathcal{A}_n^{(m)}$,*

- *the number of iterations $\textsc{Moore}(\mathcal{A})$ of the main loop in Moore's algorithm is at most equal to $m+1$ and always less than or equal to $n-1$.*
- *the worst-case time complexity $\mathcal{W}(\mathcal{A})$ of Moore's algorithm is in $\Theta((m+1)n)$, where the Θ is uniform for $m \in \{0, \cdots, n-2\}$, 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 result holds since the loop is iterated exactly $m+1$ times when the set F of final states is neither empty nor equal to $\{1, \cdots, n\}$. Moreover from property (4) of Proposition 2.1 the integer m is less than or equal to $n-2$. If F is empty or equal to $\{1, \cdots, n\}$, 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 classically 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 2.3 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.

2.4. Probabilistic model

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 deterministic and complete one. We focus our study on the average complexity of this algorithm for the uniform distribution over accessible deterministic and complete automata with n states, and as n tends toward infinity. 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 $1/2^n$. Therefore the probability that all states are final (or non-final) is exponentially unlikely. Some extensions of the main result to other distributions are given in Section 5.

The general framework of the average case analysis of algorithms [8] is based on the enumeration properties of studied objects, most often given by generating functions. For accessible and deterministic automata, this first step is already not easy. Although the

asymptotic of the number of such automata is known, it can not be easily handled: a result from Korshunov [13], rewritten in terms of Stirling numbers of the second kind in [2] and generalized to possibly incomplete automata in [4], is that the number of accessible and 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.

Here 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.

3. Main result

This section is devoted to the statement and the proof of the main theorem.

Theorem 3.1. *For any fixed integer $k \geq 1$ and for the uniform distribution over the accessible deterministic and complete automata of size n over a k-letter alphabet, the average complexity of Moore's state minimization algorithm is $\mathcal{O}(n \log n)$.*

Note that this bound is independent of k, the size of the alphabet considered. Moreover, as we shall see in Section 4, it is tight in the case of a unary alphabet.

Before proving Theorem 3.1 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 $p' \neq q'$. We 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:

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

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

From the previous 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 $\{1, \cdots, n\}$, 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 on the cardinality of $\mathcal{F}_\ell(p, q, p', q')$ in Lemma 3.3.

Lemma 3.2. *For any integer $\ell \in \{1, \cdots, n-1\}$ and any states $p, q, p', q' \in \{1, \cdots, n\}$ with $p \neq q$, $p' \neq q'$ such that $\mathcal{F}_\ell(p, q, p', q')$ is not empty, there exists an acyclic subgraph of $\mathcal{G}_\ell(p, q, p', q')$ with ℓ edges.*

Proof. If $\mathcal{F}_\ell(p,q,p',q')$ is not empty, let $u = u_1 \cdots u_\ell$ 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 construction.

For every $i \in \{0, \cdots, \ell-1\}$, 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 $p \sim_{\ell-1} q$. Moreover, the graphs $(G_{\ell,i})_{0 \le i \le \ell-1}$ have the following properties:

(1) For each $i \in \{0, \cdots, \ell-2\}$, $G_{\ell,i}$ is a strict subgraph of $G_{\ell,i+1}$. 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 $G_{\ell,i}$, for 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. Hence this edge is a new one.
(2) For each $i \in \{0, \cdots, \ell-1\}$, $G_{\ell,i}$ contains $i+1$ edges. It is a consequence of property (1), since $G_{\ell,0}$ has only one edge between p and q.
(3) For each $i \in \{0, \cdots, \ell-1\}$, $G_{\ell,i}$ contains no loop. Indeed $p \cdot v \neq q \cdot v$ for any prefix of u since $p \not\sim q$ for any automaton $(\mathcal{T}, F)$ with $F \in \mathcal{F}_\ell(p,q,p',q')$, which is not empty.
(4) For each $i \in \{0, \cdots, \ell-1\}$, if there exists a path in $G_{\ell,i}$ from s to t, then $s \sim_{\ell-1-i} t$ in every automata $(\mathcal{T}, F)$ with $F \in \mathcal{F}_\ell(p,q,p',q')$. This property can be proved by induction.

We claim that every $G_{\ell,i}$ is acyclic. Assume that it is not true, and let $j \ge 1$ be the smallest integer such that $G_{\ell,j}$ contains a cycle. By property (1), $G_{\ell,j}$ is obtained from $G_{\ell,j-1}$ by adding an 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 $\mathcal{G}_\ell(p,q,p',q')$ with ℓ edges according to property (2), which concludes the proof. ∎

Lemma 3.3. *Given a transition structure $\mathcal{T}$ of size $n \ge 1$ and an integer ℓ with $1 \le \ell < n$, for all states p,q,p',q' of $\mathcal{T}$ with $p \neq q$ and $p' \neq q'$ the following result holds:*

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

Proof. If $\mathcal{F}_\ell(p,q,p',q')$ is empty, the result holds. Otherwise, from Lemma 3.2, there exists an acyclic subgraph G of $\mathcal{G}_\ell(p,q,p',q')$ with ℓ edges. Let m be the number of connected components of G that are not reduced to a single vertex. The states in such a component are either all final or all non-final. Therefore there are at most m choices to make to determine whether the states in those components are final or non-final. As the graph G is acyclic, there are exactly $m + \ell$ vertices that are not isolated in G. Hence there are at

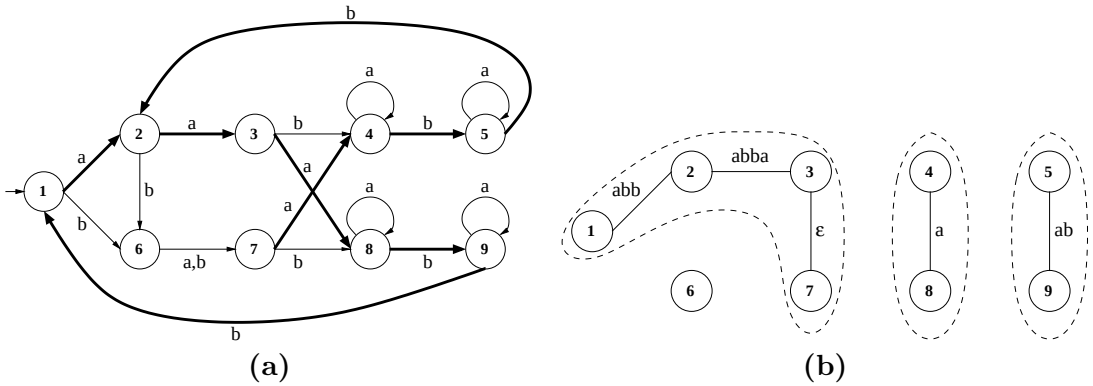

Figure 3: Illustration of the proof of Lemma 3.2 for $n = 9$, $\ell = 5$, $p = 3$, $q = 7$, $p' = 3$ and $q' = 8$ on a given transition structure. **(a)** $u = abbaa$ is the smallest word of length 5, for the lexicographic order, such that $3 \cdot u = 3$ and $7 \cdot u = 8$. The set $\mathcal{F}_5(3, 7, 3, 8)$ 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.

most $2^m 2^{n-(m+\ell)} = 2^{n-\ell}$ elements in $\mathcal{F}_\ell(p, q, p', q')$: 2^m corresponds to the possible choices for the connected components and $2^{n-(m+\ell)}$ to the choices for the isolated vertices. ∎

Proposition 3.4. *Let $k \geq 1$. There exists a positive real constant C such that for any positive integer n and any deterministic and complete transition structure $\mathcal{T}$ of size n over a k-letter 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 upper bounded by $C \log n$.*

Proof. Let $\mathcal{T}$ be a deterministic and complete transition structure of size n over a k-letter 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 exists 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

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

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

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

and using Lemma 3.3 and estimating the number of choices of the four points p, q, p', q', we have

$$|\mathcal{F}^{\geq \ell}| \leq n(n-1)\, n(n-1) 2^{n-\ell} \leq n^4 2^{n-\ell}. \tag{3.1}$$

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

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

where $\mathcal{F}^{<\ell}$ is the complement of $\mathcal{F}^{\geq \ell}$ in the set of all subsets of states. Moreover by Lemma 2.3, for any $F \in \mathcal{F}^{<\ell}$, $\textsc{Moore}(\mathcal{T}, F) \leq \ell$. Therefore, since $|\mathcal{F}^{<\ell}| \leq 2^n$

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

Using Lemma 2.3 again to give an upper bound for $\textsc{Moore}(\mathcal{T}, F)$ when $F \in \mathcal{F}^{\geq \ell}$ and the estimate of $|\mathcal{F}^{\geq \ell}|$ given by Equation 3.1 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 3.1:

Proof of the main theorem: Let $\mathcal{T}_n$ denote the set of deterministic, accessible and complete 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 deterministic, accessible and complete automata with n states is in bijection with the pairs $(\mathcal{T}, F)$ consisting of a deterministic, accessible and complete transition structure $\mathcal{T} \in \mathcal{T}_n$ with n states and a subset $F \subset \{1, \cdots, n\}$ 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

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

Using Proposition 3.4 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 by $C \log n$, and by Lemma 2.3, the average complexity of Moore's algorithm is upper bounded by $C_1 C n \log n$, concluding the proof. □

4. Tight bound 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.

We shall use the following result on words, whose proof is given in detail in [8, p. 285]. For a word u on the binary alphabet $\{0, 1\}$, the *longest run of* 1 is the length of the longest consecutive block of 1's in u.

Proposition 4.1. *For any real number h and for the uniform distribution on binary words of length n, the probability that the longest run of* 1 *is smaller than $\lfloor \log_2 n + h \rfloor$ is equal to*

$$e^{-\alpha(n)2^{-h-1}} + \mathcal{O}\left(\frac{\log n}{\sqrt{n}}\right),$$

where the $\mathcal{O}$ is uniform on h, and $\alpha(n) = 2^{\log n - \lfloor \log n \rfloor}$.

Corollary 4.2. *For the uniform distribution on binary words of length n, the probability that the longest run of* 1 *is smaller than $\lfloor \frac{1}{2} \log_2 n \rfloor$ is smaller than $e^{-\sqrt{n}/2}$.*

Proof. Set $h = -\frac{1}{2} \log_2 n$ in Proposition 4.1 and use that for any integer n, $\alpha(n) \geq 1$. ∎

The shape of an accessible deterministic and complete automaton with n states on a one-letter alphabet $A = \{a\}$ is very specific. If we label the states using the depth-first order, then for all $q \in \{1, \cdots, n-1\}$ $q \cdot a = q + 1$. The state $n \cdot a$ entirely determines the transition structure of the automaton. Hence there are $n2^n$ distinct unary automata with n states. We shall also use the following result from [16]:

Proposition 4.3. *For the uniform distribution on unary automata with n states, the probability that an automaton is minimal is asymptotically equal to $\frac{1}{2}$.*

We can now prove the optimality of the $\mathcal{O}(n \log n)$ bound for unary automata:

Theorem 4.4. *For the uniform distribution on unary automata with n states, the average time complexity of Moore's state minimization algorithm is $\Theta(n \log n)$.*

Proof. From Theorem 3.1 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.

For any binary word u of size n, we denote by $F(u)$ the subset of $\{1, \cdots, n\}$ such that $i \in F(u)$ if and only if the i-th letter of u is 1. The map F is clearly a bijection between the binary words of length n and the subsets of $\{1, \cdots, n\}$. Therefore a unary automaton with n states is completely defined by a word u of length n, encoding the set of final states, and an integer $m \in \{1, \cdots, n\}$ corresponding to $n \cdot a$; we denote such an automaton by the pair $(u, m) \in \{0, 1\}^n \times \{1, \cdots, n\}$. Let ℓ be the integer defined by $\ell = \lfloor \frac{1}{2} \log_2 n \rfloor$. Let M_n be the set of minimal unary automata with n states, and S_n be the subset of M_n defined by

$$S_n = \{(u, m) \in M_n \mid \text{the longest run of 1 in } u \text{ is smaller than } \ell\}$$

As the number of element in S_n is smaller than the number of automata (u, m) whose longest run of 1 in u is smaller than ℓ, from Corollary 4.2, we have $|S_n| = o(n2^n)$. Let (u, m) be a minimal automaton in $M_n \setminus S_n$. The word u has a longest run of 1 greater or equal to ℓ. Let $p \in \{1, \cdots, n\}$ be the index of the beginning of such a longest run in u. The states p and $p + 1$ requires ℓ iterations in Moore's algorithm to be separated, as $p \cdot a^i$ and $(p + 1) \cdot a^i$ are both final for every $i \in \{0, \cdots, \ell - 2\}$. They must be separated

by the algorithm at some point since (u, m) is minimal. Hence $\textsc{Moore}((u, m)) \geq \ell$ for any $(u, m) \in M_n \setminus S_n$. Therefore

$$\begin{aligned}\frac{1}{n2^n} \sum_{(u,m)\in\{0,1\}^n\times\{1,\cdots,n\}} \textsc{Moore}((u, m)) &\geq \frac{1}{n2^n} \sum_{(u,m)\in M_n\setminus S_n} \textsc{Moore}((u, m)) \\ &\geq \frac{1}{n2^n}|M_n \setminus S_n|\ell \geq \frac{1}{n2^n}|M_n|\ell - \frac{1}{n2^n}|S_n|\ell \\ &\geq \frac{1}{2}\ell - o(\ell)\end{aligned}$$

The last inequality is obtained using Proposition 4.3, concluding the proof since by hypothesis $\ell = \lfloor \frac{1}{2} \log_2 n \rfloor$. ∎

5. Extensions

In this section we briefly present two extensions of Proposition 3.4 and Theorem 3.1.

5.1. Bernoulli distributions for the sets of final states

Let p be a fixed real number with $0 < p < 1$. Let $\mathcal{T}$ be a transition structure with n states. Consider the distribution on the sets of final states for $\mathcal{T}$ defined such that each state as a probability p of being final. The probability for a given subset F of $\{1, \cdots, n\}$ to be the set of final states is $\mathbb{P}(F) = p^{|F|}(1 - p)^{n-|F|}$.

A statement analogous to Proposition 3.4 still holds in this case. The proof is similar although a bit more technical, as Proposition 3.4 corresponds to the special case where $p = \frac{1}{2}$. Hence, for this distribution of sets of final states, the average complexity of Moore's algorithm is also $\mathcal{O}(n \log n)$.

5.2. Possibly incomplete automata

Now consider the uniform distribution on possibly incomplete deterministic automata with n states and assume that the first step of Moore's algorithm applied to an incomplete automaton consists in the completion of the automaton making use of a sink state. In this case Proposition 3.4 still holds. Indeed, Lemma 3.3 is still correct, even if the sets of final states F are the sets that do not contain the sink state. As a consequence, if a transition structure $\mathcal{T}$ is incomplete, the average complexity of Moore's algorithm for the uniform choice of set of final states of the completed transition structure, such that the sink state is not final, is in $\mathcal{O}((n + 1) \log(n + 1)) = \mathcal{O}(n \log n)$.

6. Open problem

We conjecture that for the uniform distribution on complete, accessible and deterministic automata with n states over a k-letter alphabet, with $k \geq 2$, the average time complexity of Moore's algorithm is in $\mathcal{O}(n \log \log n)$.

This conjecture comes from the following observations. First, Figure 2 seems to show a sub-logarithmic asymptotic number of iterations in Moore's algorithm. Second, if the automaton with n states is minimal, at least $\Omega(\log \log n)$ iterations are required to isolate every state: $\log n$ words are needed, and this can be achieved in the best case using all the words of length less than or equal to $\log \log n$. Moreover, in [2] we conjectured that a constant part of deterministic automata are minimal; if it is true, this would suggest that $\Omega(\log \log n)$ is a lower bound for the average complexity of Moore's algorithm. The conjecture above is that this lower bound is tight.

References

[1] M. Baclet, C. Pagetti, Around Hopcroft's Algorithm In *CIAA'2006* volume 4094 in Lect. Notes Comput. Sci., p. 114-125, Springer-Verlag, 2006.
[2] F. Bassino, C. Nicaud, Enumeration and random generation of accessible automata, *Theoret. Comput. Sci.*, 381, p. 86-104, 2007.
[3] F. Bassino, J. David, C. Nicaud, REGAL: a Library to randomly and exhaustively generate automata, in Jan Holub, Jan Zdarek, editors, *12th International Conference on Implementation and Application of Automata (CIAA'07)*, Vol. 4783 in Lect. Notes Comput. Sci., p. 303-305, Springer, 2007.
[4] F. Bassino, J. David, C. Nicaud, Random generation of possibly incomplete deterministic automata, in *Génération Aléatoire de Structures COMbinatoires (Gascom'08)*, p. 31- 40.
[5] J. Berstel, O. Carton, On the complexity of Hopcroft's state minimization algorithm, In *CIAA'2004*, Vol. 3317 of LNCS, p. 35-44, Springer-Verlag, 2004.
[6] J.A. Brzozowski, Canonical regular expressions and minimal state graphs for definite events, Proc. Symp. on. the Mathematical Theory of Automata, Vol. 12 of MRI Symposia Series, p. 529-561, Polytechnic Press, Polytechnic Institute of Brooklyn, New York, 1962.
[7] J.-M. Champarnaud, A. Khorsi, T. Paranthoen. *Split and Join for Minimizing: Brzozowski's algorithm* PSC'02 Proceedings, Prague Stringology Conference, Research report DC-2002-03, p. 96-104, 2002.
[8] P. Flajolet, R. Sedgewick, *Analytic combinatorics*, Cambridge University Press, 2009.
[9] D. Gries, Describing an Algorithm by Hopcroft, *Acta Inf.*, 2, p. 97-109, 1973.
[10] J.E. Hopcroft, J.D. Ullman *Introduction to Automata Theory, Languages and Computation*. Addison-Weisley Publishing Company, 1979.
[11] J.E. Hopcroft. *An $n \log n$ algorithm for minimizing the states in a finite automaton*, in The Theory of Machines and Computations, p 189-196, Academic Press, New York, 1971.
[12] T. Knuutila. Re-describing an algorithm by Hopcroft, *Theoret. Comput. Sci.*, 250, p. 333-363, 2001
[13] D. Korshunov. Enumeration of finite automata, *Problemy Kibernetiki*, 34, p. 5–82, 1978, In Russian.
[14] M. Lothaire. *Applied combinatorics on words*, Vol 105 of Encyclopedia of mathematics and its application. Cambridge University Press, 2005.
[15] E.F. Moore. *Gedanken experiments on sequential machines*, in C.E Shannon and J. McCarthy, Automata Studies, Princeton Univ. Press, p. 129-153, 1956.
[16] C. Nicaud. Average State Complexity of Operations on Unary Automata, In *MFCS 1999* volume 1672 in Lect. Notes Comput. Sci., p. 231 - 240, Springer-Verlag, 1999.
[17] B.W. Watson. A taxonomy of algorithms for constructing minimal acyclic deterministic finite automata *South African Computer Journal*, Vol. 27, p. 12-17, 2001.
[18] D. Wood *Theory of Computation*. John Wiley & Sons, 1987.