Algorithmica (2012) 63:509–531
DOI 10.1007/s00453-011-9557-7

Average Case Analysis of Moore's State Minimization Algorithm

Frédérique Bassino · Julien David · Cyril Nicaud

Received: 27 July 2009 / Accepted: 27 July 2011 / Published online: 12 October 2011

Abstract We prove that the average complexity of Moore's state minimization algorithm is $\mathcal{O}(kn \log n)$, where n is the number of states of the input and k the size of the alphabet. This result holds for a whole family of probabilistic models on automata, including the uniform distribution over deterministic and accessible automata, as well as uniform distributions over classical subclasses, such as complete automata, acyclic automata, automata where each state is final with probability $\gamma \in (0, 1)$, and many other variations.

Keywords State minimization algorithms · Moore's algorithm · Average complexity · Finite automata

1 Introduction

Deterministic automata are a convenient way to represent regular languages and can be used to perform efficiently most usual computations involving regular languages. Therefore, finite-state automata appear in many fields of computer science, such as linguistics, data compression, bioinformatics, *etc.* One can associate a unique smallest deterministic automaton to any given regular language. This automaton is called

F. Bassino (✉) · J. David
LIPN UMR 7030, Université Paris 13—CNRS, 99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France
e-mail: Frederique.Bassino@lipn.univ-paris13.fr

J. David
e-mail: Julien.David@lipn.univ-paris13.fr

C. Nicaud
LIGM UMR 8049—CNRS, Université Paris Est, 5, bd Descartes, 77454 Marne-la-Vallé Cedex 2, France
e-mail: Cyril.Nicaud@univ-paris-est.fr

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 algorithms, which compute the minimal automaton of a regular language given by a deterministic automaton, are of great interest.

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 at most two parts, successive refinements lead to a partition whose elements are the subsets of indistinguishable states; these sets can be merged to form a smaller automaton that recognizes the same language. Since there are at most n such refinements, and since each of these refinements is done in linear 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. It selects carefully the parts that are split at each step. Using appropriate data structures, its worst-case complexity is $\mathcal{O}(kn \log n)$. It is the best known minimization algorithm and therefore it has been studied intensively: in [3, 4] different proofs of its correctness are given, in [5–7] the tightness of the complexity upper bound for various families of automata is proved, in [8, 9] a precise description of the data structures that are needed to reach the $\mathcal{O}(kn \log n)$ complexity is given. In [10, 11] two $\mathcal{O}(m \log n)$ solutions for incomplete automata are given, where m denotes the number of defined transitions, using improvements in Hopcroft's strategy together with advanced data structures. Note that Hopcroft and Ullman described another minimization algorithm which is much easier to implement in [12]: for every pair of states of the input automaton, it tests whether the two states are equivalent or not. Its complexity is $\Theta(kn^2)$.

Brzozowski's algorithm [13, 14] is another minimization algorithm. It works differently than the other ones. It has the advantage to be able to take non-deterministic automata as inputs. 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 often sub-exponential in practice. For further details on all these algorithms, the reader is invited to consult [15] where a taxonomy of minimization algorithms is presented.

Also note that for some specific families of automata, minimization can be done in linear time. See for instance [16] for acyclic automata, [17] for unary automata and [18] for local automata.

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

In the following we prove that Moore's algorithm performs only $\mathcal{O}(\log n)$ refinements on average, and therefore has average complexity $\mathcal{O}(kn \log n)$. Our result holds for any probabilistic model that satisfies the following conditions: the underlying graph of the automata and the set of final states are chosen independently from one another and every state is final with fixed probability $\gamma \in (0, 1)$. We call such a probabilistic model a *Bernoulli model*. For instance, our result holds for the uniform distribution on possibly incomplete automata, the uniform distribution on strongly connected (resp. acyclic, group, *etc.*) automata, since they are all Bernoulli models.

The $\mathcal{O}(kn \log n)$ average time complexity therefore holds for various probabilistic models, and the goal of this paper is to state a result that holds for as many distri-

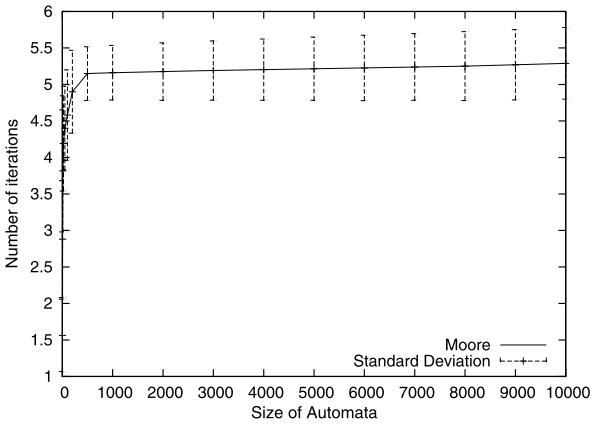

Fig. 1 The experimental results were obtained with the C++ library REGAL (available at: http://regal.univ-mlv.fr/) to randomly and uniformly generate deterministic, accessible and complete automata [19–21]. For each size the values are computed from 20 000 random automata over a 2-letter alphabet

butions as possible. Choosing one specific model may allow a more precise analysis; this is what the second author did in [22], proving that the average complexity is $\mathcal{O}(kn \log\log n)$ for the uniform distribution over deterministic and complete automata (not necessarily accessible), but such a result cannot be generalized directly to, for instance, acyclic automata.

A preliminary version of this work, where some proofs were omitted and the statements were less general, has been presented in [23].

The paper is organized as follows. After recalling the basics of minimization algorithms in Sect. 2, we establish some results on Moore's algorithm when applied to automata that are already minimal in Sect. 3. Section 4 is devoted to the statement and the proof of our main result, starting with a description of the probabilistic models under which it holds. In Sect. 5, we present some extensions and other models that do not necessarily satisfy the hypothesis of our result, in order to illustrate the limits of what can be done with the techniques introduced in this paper. A conclusion is proposed in Sect. 6, with some perspectives and open problems.

2 Preliminaries

This section is devoted to basic notions related to the minimization of automata. We refer the reader to the literature [12] 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*, or *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 function from $Q \times A$ to Q, $q_0 \in Q$ is the *initial state* and $F \subseteq Q$ 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 the 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 $q_0 \cdot u \in F$. The set of all the

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 $q_0 \cdot u = p$. It is *co-accessible* when for any state $p \in Q$, there exists $u \in A^*$ such that $p \cdot u \in F$.

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

For any fixed alphabet A and any integer $n \geq 1$, we denote by $\mathcal{D}_n$ the set of all n-state deterministic and accessible automata on A and by $\mathcal{T}_n$ the set of all deterministic and accessible n-state transition structures on A. We also define $\mathcal{D} = \cup_{n\geq 1}\mathcal{D}_n$ and $\mathcal{T} = \cup_{n\geq 1}\mathcal{T}_n$.

In the following we only consider deterministic and accessible automata and deterministic and accessible transition structures. Consequently, these objects will often just be respectively called *automata* and *transition structures*. We also consider, without loss of generality, that the set of states of an n-state automaton or transition structure is always $Q = \{1, \ldots, n\}$.

When stating a result on the average time complexity of Moore's algorithm, n is implicitly always the number of states of the automaton, and k the size of the alphabet.

The cardinality of a finite set E is denoted by $|E|$. For a boolean condition (which we also call *property*) *Cond*, the Iverson bracket *Cond* is equal to 1 if the condition *Cond* is satisfied and 0 otherwise. If $P = \{E_1, \ldots, E_m\}$ is a partition of a set E, the E_i's are called the *parts* of E.

2.2 Completion of an Automaton

Moore's algorithm deals with deterministic accessible and complete automata. Though all automata considered in this article are deterministic and accessible, we will consider in the sequel several distributions on possibly incomplete automata. In that case, a simple preprocessing is needed before applying state minimization algorithm.

More precisely, if the automaton is not complete, just add a sink state, which becomes the target state of every undefined transition. This transformation, which will be called the *completion step* in the following, is done in time $\mathcal{O}(kn)$, in the worst case, where k is the size of the alphabet and n the number of states of the automaton.

2.3 The Myhill–Nerode Equivalence Relation for Complete Automata

Let $\mathcal{A} = (A, Q, \cdot, q_0, F)$ be a complete automaton. For any non-negative integer i, two states $p, q \in Q$ are *i-equivalent*, denoted by $p \sim_i q$, when for all words u of length smaller 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 the *Myhill–Nerode equivalence relation* [24], and is denoted by $p \sim q$.

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

$$\forall u \in A^*,\ \forall (p,q) \in Q^2, \quad p \approx q \quad \Rightarrow \quad p \cdot u \approx q \cdot u.$$

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

Proposition 1 *Let $\mathcal{A} = (A, Q, \cdot, q_0, F)$ be a complete n-state automaton. 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 $i \in \mathbb{N}$, the $(i+1)$-equivalence relation is equal to the i-equivalence relation then for every $j \geq i$, the j-equivalence relation is equal to the Myhill–Nerode equivalence relation $\sim$.*
4. *If $n \geq 2$, the $(n-2)$-equivalence relation is equal to the Myhill–Nerode equivalence relation. If $n = 0$ or $n = 1$ or if F is either empty or Q, then $\sim_0 = \sim$.*
5. *The Myhill–Nerode equivalence relation is right invariant.*

For any complete automaton $\mathcal{A} \in \mathcal{D}$, denote by $\textsc{Nerode}(\mathcal{A})$ the smallest integer m such that the m-equivalence relation $\sim_m$ is equal to the Myhill–Nerode equivalence relation.

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

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

where $Q/{\approx}$ is the set of equivalence 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 well-formedness of this definition follows from the right invariance of the equivalence relation $\approx$.

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

The quotient automaton $\mathcal{A}/{\sim}$ of Theorem 1 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 and complete automaton, and its uniqueness defines a bijection between regular languages and minimal automata.

Note that for our definition, a minimal automaton is always complete; the alternative definition, where the minimal automaton must be trim, is also used in the literature.

2.4 Moore's State Minimization Algorithm for Complete Automata

In this section we describe an algorithm due to Moore [1], which computes the minimal automaton of a regular language represented by a deterministic accessible and complete 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 the Myhill–Nerode equivalence relation. The algorithm relies mainly on properties *2* and *3* of Proposition 1: The partition π is initialized according to the 0-equivalence relation $\sim_0$, then in each iteration the partition corresponding to the $(i+1)$-equivalence relation $\sim_{i+1}$ is computed from the one corresponding to the i-equivalence relation $\sim_i$, using property *2*. The algorithm stops when no new partition refinement is obtained, and the result is the Myhill–Nerode equivalence relation according to property *3*. The minimal automaton can then be computed from the resulting partition since it is the quotient automaton of the input automaton by the Myhill–Nerode equivalence relation. The algorithm is detailed in Fig. 2.

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
```

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 are 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 these lines are executed.

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

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

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 yields a complexity in $\Theta(kn)$ for this part of the algorithm.

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[σ(p)] ≠ s[σ(p − 1)] then
8          i := i + 1
9      π'[σ(p)] := i
10 return π'
```

Fig. 2 Description of Moore's algorithm

Note that after i iterations in the main loop of the algorithm, the relation $\sim_i$ has been computed, and π' is the associated partition.

The worst-case time complexity of Moore's algorithm is $\Theta(kn^2)$. Lemma 1 below is a more precise statement that will be used in the proof of the main theorem (Theorem 2).

If $\mathcal{A} \in \mathcal{D}$, then the number of iterations of the main loop when Moore's algorithm is applied to $\mathcal{A}$, denoted by $\textsc{Moore}(\mathcal{A})$, is directly related to $\textsc{Nerode}(\mathcal{A})$, as stated in the following lemma.

Lemma 1 *For any automaton $\mathcal{A}$ of $\mathcal{D}_n$, the following properties hold*:

- *The number of iterations* $\textsc{Moore}(\mathcal{A})$ *of the main loop in Moore's algorithm is equal to* 0 *if* $L(\mathcal{A}) = \emptyset$ *or* $L(\mathcal{A}) = A^*$ *and is equal to* $\textsc{Nerode}(\mathcal{A}) + 1$ *otherwise.*
- $\textsc{Moore}(\mathcal{A})$ *is always less than or equal to* $n - 1$.
- *If* $L(\mathcal{A}) \neq \emptyset$ *and* $L(\mathcal{A}) \neq A^*$, *then the time complexity* $\mathcal{W}(\mathcal{A})$ *of Moore's algorithm applied to* $\mathcal{A}$ *is* $\Theta(\textsc{Moore}(\mathcal{A})kn)$.

Proof If F is empty or equal to $\{1, \ldots, n\}$, then $\textsc{Nerode}(\mathcal{A}) = 0$, and the time complexity to compute the size of F is $\Theta(n)$.

The loop is iterated exactly $\textsc{Nerode}(\mathcal{A}) + 1$ times when the set F of final states is neither empty nor equal to $\{1, \ldots, n\}$, because the algorithm needs one more iteration to obtain that $\pi = \pi'$. Moreover, by property *4* of Proposition 1, $\textsc{Nerode}(\mathcal{A})$ is less than or equal to $n - 2$. The initialization and the construction of the quotient automaton are both done in $\Theta(kn)$. The complexity of each iteration of the main loop is in $\Theta(kn)$: this can be achieved using a lexicographic sort algorithm, and yields the announced result for $\mathcal{W}(\mathcal{A})$. □

Lemma 1 gives proof that the worst-case complexity of Moore's algorithm is $\mathcal{O}(kn^2)$, as there are no more than $n - 1$ iterations in the process of the algorithm, for n large enough. More precisely, the worst-case complexity of the algorithm is $\Theta(kn^2)$; this can be shown using the automata depicted in Fig. 3.

When the automaton that must be minimized is not complete, the completion step described in Sect. 2.2 is applied before running Moore's algorithm. The time complexity of the completion step followed by Moore's algorithm is of the same order of magnitude as the one of the state minimization alone.

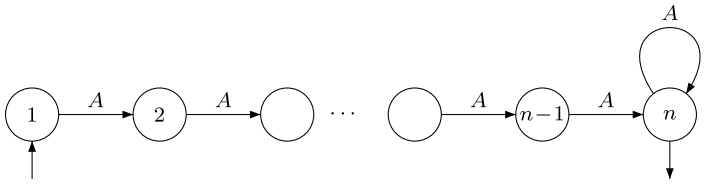

Fig. 3 The minimal automaton of the language $A^{n-1}A^*$, for $n \geq 3$. The states 1 and 2 are $(n-3)$-equivalent, but not $(n-2)$-equivalent: Moore's algorithm performs $n-1$ iterations before halting

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 depends only on the recognized language, and therefore it is the same for all deterministic accessible and complete 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, which will be useful in the forthcoming discussions.

Lemma 2 *The number of iterations of Moore's algorithm applied to any deterministic accessible and complete 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 $\emptyset$, by Lemma 1 we have $\textsc{Moore}(\mathcal{A}) = \textsc{Moore}(\mathcal{A}_{min}) = 0$.

Otherwise, by Lemma 1, $\textsc{Moore}(\mathcal{A}) = \textsc{Nerode}(\mathcal{A}) + 1$. Moreover, by definition one has

$$\textsc{Nerode}(\mathcal{A}) = \min\Big\{ i \geq 0 \mid \forall (p,q) \in Q^2 : p \nsim q \Rightarrow \Big(\exists u \in A^{\leq i} : [\![p \cdot u]\!] \neq [\![q \cdot u]\!]\Big)\Big\}.$$

Since 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

$$\textsc{Nerode}(\mathcal{A}) = \min\Big\{ i \geq 0 \mid \forall (p,q) \in Q^2 : [p] \neq [q] \Rightarrow \Big(\exists u \in A^{\leq i} : [\![[p] * u]\!] \neq [\![[q] * u]\!]\Big)\Big\}.$$

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 to minimal automata. The worst-case complexity of the algorithm is still $\Theta(kn^2)$, since the automaton depicted in Fig. 3 is minimal. Since all the states of a minimal automaton are alone in their equivalence class, a minimum number of iterations in the algorithm is required to distinguish all of them, as proved in the following proposition.

Proposition 2 *Moore's algorithm applied to a minimal n-state automaton 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 part of the partition, and needs one more iteration to find out that the partition

has not changed. The number of parts 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 $\phi_p^{(i)}(u) = [\![p \cdot u]\!]$. Since 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). Since $p \sim_i q$ 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 parts in the partition at the i-th iteration of Moore's algorithm. When $n \geq 2$, the set of final states of the minimal automaton $\mathcal{A}$ is non trivial and the algorithm halts one iteration after the n parts have been computed. Hence

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

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

4 Average Case Analysis

4.1 Probabilistic Model

Our main theorem is established for a family of distributions that we call *Bernoulli model* on automata. We define them formally in this section.

Since we are dealing with finite sets only, every universe on which probabilities are defined is also finite; a probability law on such a finite universe U can be defined by a *probability mass function* $p : U \to [0,1]$, that is, a function such that $\sum_{x \in U} p(x) = 1$. The associated probability law, that we also denote by p, is defined for all $X \subseteq U$ by

$$p(X) = \sum_{x \in X} p(x).$$

Recall that every automaton $\mathcal{A} \in \mathcal{D}_n$ can be seen as a pair $(T_\mathcal{A}, F_\mathcal{A})$, where $T_\mathcal{A} \in \mathcal{T}_n$ is a transition structure and $F_\mathcal{A} \subseteq \{1, \ldots, n\}$ is a set of final states.

If for a given non-empty finite set E, we choose a subset of E by selecting every element of E with probability γ, where $\gamma \in (0,1)$, we get a probability on subsets of E, which is formally described in the following definition.

Definition 1 Let E be a non-empty finite set. For any $\gamma \in (0,1)$, the probability r defined on subsets X of E by $r(X) = \gamma^{|X|}(1-\gamma)^{|E|-|X|}$ is called the *Bernoulli distribution of parameter γ on elements of E*.

Definition 2 For any $n \geq 1$, a probability p defined on $\mathcal{D}_n$ is a *Bernoulli model* when there exists a probability q defined on $\mathcal{T}_n$ and a real number $\gamma \in (0, 1)$ such that, for any $\mathcal{A} \in \mathcal{D}_n$,

$$p(\mathcal{A}) = q(T_{\mathcal{A}})\, r(F_{\mathcal{A}}), \quad \text{with } \mathcal{A} = (T_{\mathcal{A}}, F_{\mathcal{A}}),$$

where r is the Bernoulli distribution of parameter γ on states.

This means that the transition structures follow any probability law q on $\mathcal{T}_n$, and that each state is final according to a Bernoulli law of parameter γ, independently from the other states and from the transition structure. If we want to specify the parameter, we shall say that p is a *Bernoulli model of parameter* γ.

Since we are interested in the asymptotic complexity of an algorithm, we are working on a sequence of probabilities, one for each $\mathcal{D}_n$, with $n \geq 1$. We can avoid too much formalism using the following definition.

Definition 3 Let $\gamma \in (0, 1)$. Let $p : \mathcal{D} \to [0, 1]$ be a mapping such that for any $n \geq 1$, the restriction of p to $\mathcal{D}_n$ is a Bernoulli model of parameter γ. Then p is a *Bernoulli model on* $\mathcal{D}$ (of parameter γ).

Note that it is important in the above definition that γ is fixed and therefore does not depend on n.

In analysis of algorithms, one is often interested in uniform distributions on inputs or on subfamilies of inputs. A *uniform distribution* on a non-empty finite set E is a probability p such that, for any $e \in E$, $p(e) = 1/|E|$. Because of the required independency between transition structures and sets of final states in Definition 2, the uniform distribution on a subset E of $\mathcal{D}_n$ is not always a Bernoulli model. Nonetheless, as stated in the following lemma, Bernoulli models can be obtained by restricting the allowed transition structures.

Lemma 3 *Let P be a property defined on $\mathcal{T}$. For any $n \geq 1$, let $\mathcal{E}_n \subseteq \mathcal{D}_n$ be the subset of n-state automata whose transition structures satisfy P. For any $n \geq 1$ such that $\mathcal{E}_n \neq \emptyset$, the uniform distribution on $\mathcal{E}_n$ is a Bernoulli model.*

Proof Let p be the probability defined for any $\mathcal{A} \in \mathcal{D}_n$, with $\mathcal{A} = (T_{\mathcal{A}}, F_{\mathcal{A}})$, by

$$p(\mathcal{A}) = \frac{[\![P(T_{\mathcal{A}})]\!]}{t_n} \frac{1}{2^n},$$

where t_n is the number of transition structures in $\mathcal{T}_n$ that satisfy P. Then p is the uniform distribution on $\mathcal{E}_n$ and it is a Bernoulli model with $q(T) = [\![P(T)]\!]/t_n$ and $\gamma = \frac{1}{2}$. □

Acyclic automata, strongly connected automata, group automata, *etc.* are examples of families of automata that are defined by a property on the transition structures only. As a consequence, the uniform distribution on such a class of automata is a Bernoulli model.

4.2 Main Result

The main result is the following:

Theorem 2 *For any Bernoulli model on $\mathcal{D}$, the average time complexity of Moore's state minimization algorithm, possibly preceded by a completion step, is $\mathcal{O}(kn \log n)$.*

As a consequence of Theorem 2 and Lemma 3, we have the following corollary:

Corollary 1 *The average complexity of Moore's state minimization algorithm, possibly preceded by a completion step, is $\mathcal{O}(kn \log n)$ for the following distributions*:

- *The uniform distribution on deterministic and accessible automata.*
- *The uniform distribution on deterministic, accessible and complete automata.*
- *The uniform distribution on deterministic, accessible and acyclic automata.*
- *The uniform distribution on deterministic and accessible inverse automata (the action of each letter is a partial injection on the set of states).*
- *Any of the above, using a Bernoulli model of fixed parameter $\gamma \in (0, 1)$ for the sets of final states.*

A typical example that does not meet the condition of Theorem 2 is the uniform distribution on complete and co-accessible automata, since the co-accessibility condition depends on the set of final states (see Sect. 5.4 for more details).

4.3 Outline of the Proof

The proof of Theorem 2 can be summarized informally as follows.

We first consider a fixed n-state complete transition structure T. We establish necessary conditions for a set of final states F to be such that Moore's algorithm requires at least ℓ iterations when applied to the automaton (T, F). Under a Bernoulli model, the probability that an automaton satisfies these conditions is proved to be exponentially small in ℓ.

Therefore, for a good choice of $\ell = \Theta(\log n)$, we can prove that the probability for a set of final states F to induce more than ℓ iterations in Moore's algorithm is $\mathcal{O}(\frac{1}{n})$: the contribution to the average value of such sets is negligible. Hence most of the time, less than ℓ iterations are performed before the algorithm halts. This is stated in Proposition 3 and Proposition 4.

Moreover, the upper bound obtained for the average number of iterations does not depend on the transition structure T. Therefore, since the distribution of transition structures and sets of final states are independent under a Bernoulli model, a simple computation using the uniform bound completes the proof of Theorem 2.

4.4 Compatible Partitions to Obtain at Least ℓ Iterations

In this section, we introduce some definitions and preliminary results that will be used in the main proof.

The notion of compatible partition will be of great importance in the following.

Definition 4 Let E be a non-empty finite set and $P = \{E_1, \ldots, E_m\}$ be a partition of E. A subset X of E is *compatible* with P when it is a union of parts of P: there exists $I \subseteq \{1, \ldots, m\}$ such that $X = \bigcup_{i \in I} E_i$.

Let T be a fixed complete transition structure of $\mathcal{T}_n$ and ℓ be an integer such that $1 \leq \ell < n$. Let also $p, q, p', q' \in \{1, \ldots, n\}$ be four states of T.

Define $\mathcal{F}_\ell(p, q, p', q')$ as the set of sets of final states F for which in the automaton (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:

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

We assume for the discussion below that $\mathcal{F}_\ell(p, q, p', q')$ is not empty; the cases where it is empty are trivial and are handled easily.

We build a partition $P_\ell(p, q, p', q')$, denoted P_ℓ for short, that represents necessary conditions for a set of states to belong to $\mathcal{F}_\ell(p, q, p', q')$. 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 $q \cdot u = q'$.

For every $i \in \{0, \ldots, \ell\}$, we build a partition P_i of the set of states $\{1, \ldots, n\}$ in the following way:

- $P_0 = \{\{1\}, \ldots, \{n\}\}$ is the partition into n parts, where each state is alone in its part.
- For any $i \in \{0, \ldots, \ell - 1\}$, the partition P_{i+1} is obtained from P_i by merging the parts of the states $p \cdot v$ and $q \cdot v$, where v is the prefix of length i of u: if E is the part of $p \cdot v$ and E' the part of $q \cdot v$, P_{i+1} is obtained from P_i by removing the parts E and E' and by adding the part $E \cup E'$. Note that if $p \cdot v$ and $q \cdot v$ are in the same part, then $P_{i+1} = P_i$, but we will show that it cannot happen.

Lemma 4 *If $\mathcal{F}_\ell(p, q, p', q') \neq \emptyset$, then the partition $P_\ell(p, q, p', q')$ has exactly $n - \ell$ parts, and every element of $\mathcal{F}_\ell(p, q, p', q')$ is compatible with $P_\ell(p, q, p', q')$.*

Proof Let $(p_i)_{0 \leq i \leq \ell}$ and $(q_i)_{0 \leq i \leq \ell}$ denote the sequences of states such that for every $i \in \{0, \ldots, \ell\}$, $p_i = p \cdot v$ and $q_i = q \cdot v$, where v is the prefix of length i of u. In particular, $p_0 = p$, $q_0 = q$, $p_\ell = p'$ and $q_\ell = q'$.

Note that for every $F \in \mathcal{F}_\ell(p, q, p', q')$ and every $i \in \{0, \ldots, \ell - 1\}$, $p_i \sim_{\ell-i-1} q_i$ but $p_i \not\sim_{\ell-i} q_i$, since by definition $p \sim_{\ell-1} q$ and $p \not\sim_\ell q$. Therefore we can prove by induction on $i \in \{0, \ldots, \ell - 1\}$ that if x and y are in the same part of P_i then $x \sim_{\ell-i} y$, in any automaton (T, F) such that $F \in \mathcal{F}_\ell(p, q, p', q')$: Since P_0 contains only trivial parts, it is true for $i = 0$. For the induction step, we prove that if it is true at rank $i \in \{0, \ldots, \ell - 2\}$, then it is also true at rank $i + 1$. Indeed, let x and y be in the same part of P_{i+1}. If x and y are also in the same part of P_i, the induction hypothesis applies, $x \sim_{\ell-i} y$ and therefore $x \sim_{\ell-(i+1)} y$. Otherwise, x and y are in different parts of P_i, which means that in P_i, x is in the same part as p_i and y is in the same

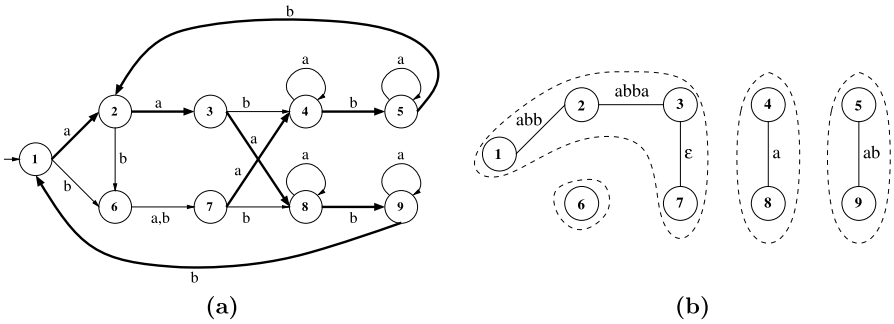

Fig. 4 Illustration of Lemma 4 for $n = 9$, $\ell = 5$, $p = 3$, $q = 7$, $p' = 3$ and $q' = 8$. (**a**) $u = abbaa$ is the smallest word of length 5, for the lexicographic order, such that $3 \cdot u = 3$ and $7 \cdot u = 8$. The set $\mathcal{F}_5(3, 7, 3, 8)$ is not empty, since it contains at least $\{4, 8\}$. The bold transitions are the ones followed when reading u from p and from q. (**b**) For every prefix v of u, with $v \neq u$, we put an edge between $p \cdot v$ and $q \cdot v$. The partition P_ℓ corresponds to the connected components of the resulting graph: two states in the same connected component must be either both final or both not final, for the automaton to be in $\mathcal{F}_\ell(p, q, p', q')$

part as q_i (or we swap x and y by symmetry); by induction hypothesis, $x \sim_{\ell-i} p_i$ and $y \sim_{\ell-i} q_i$. Hence $x \sim_{\ell-(i+1)} p_i$ and $y \sim_{\ell-(i+1)} q_i$, and since $p_i \sim_{\ell-i-1} q_i$, we obtain that $x \sim_{\ell-(i+1)} y$, concluding the proof by induction.

The property shown above proves that p_i and q_i are not in the same part of P_i, since they are not $(\ell - i)$-equivalent, for any choice of set of final states $F \in \mathcal{F}_\ell(p, q, p', q')$. Therefore, for every $i \in \{0, \ldots, \ell - 1\}$, $P_{i+1} \neq P_i$. Hence, a direct induction shows that the number of parts in P_i is exactly $n - i$, for any $i \in \{0, \ldots, \ell\}$.

Moreover, if x and y are in the same part of P_ℓ, we have proved that $x \sim_0 y$ for any $F \in \mathcal{F}_\ell(p, q, p', q')$. This means that both x and y are in F or neither of them are in F. Therefore F is a union of parts of P_ℓ and F is compatible with P_ℓ. □

Figure 4 illustrates the proof of Lemma 4 with the construction of a graph. This is the way the main proof was done in [23].

4.5 Main Proof

In this section, we present the proof of our main theorem. We consider separately the case of complete and incomplete transition structures.

4.5.1 Complete Transition Structures

Lemma 5 *Let r be a Bernoulli distribution of parameter $\gamma \in (0, 1)$ on elements of a non-empty finite set E. Let P be a partition of E in $m \geq 1$ parts. The probability, under r, that a subset of E is compatible with P is at most β^{n-m}, where $\beta = \max(\gamma, 1 - \gamma)$.*

Proof Let $P = \{E_1, \ldots, E_m\}$. Since every X compatible with P is a union of parts of the partition, such an X is determined by choosing the subset I of $\{1, \ldots, m\}$ satisfying $X = \cup_{i \in I} E_i$. Furthermore, using properties of the Bernoulli model, we can work on each E_i independently: for every $i \in \{1, \ldots, m\}$, either $E_i \cap X = \emptyset$ or $E_i \subseteq X$;

the probability that the elements of E_i are all in X is $\gamma^{|E_i|}$, and the probability that they all are not in X is $(1-\gamma)^{|E_i|}$. With $\beta = \max(\gamma, 1-\gamma)$, the probability of having either event is therefore $\gamma^{|E_i|} + (1-\gamma)^{|E_i|} \leq \gamma\beta^{|E_i|-1} + (1-\gamma)\beta^{|E_i|-1} = \beta^{|E_i|-1}$. Therefore, by independency, the probability that X is compatible with P is bounded from above by $\prod_{i=1}^{m} \beta^{|E_i|-1} = \beta^{n-m}$. □

Proposition 3 (Complete transition structures) *Let $k \geq 1$, $n \geq 1$ and let $T \in \mathcal{T}_n$ be a complete transition structure over a k-letter alphabet. For the Bernoulli distribution of parameter γ on final states, the average number of iterations of the main loop of Moore's algorithm applied to (T, F) is bounded from above by $5\log_{\frac{1}{\beta}} n + 2$, where $\beta = \max(\gamma, 1-\gamma)$.*

Proof Denote by $\mathcal{F}^{\geq \ell}$ the set of sets of final states such that the execution of Moore's algorithm on (T, F) requires more than ℓ iterations. Equivalently, $\mathcal{A} \in \mathcal{F}^{\geq \ell}$ if and only if $\textsc{Nerode}(\mathcal{A}) \geq \ell$, by Lemma 1.

A necessary and sufficient condition for F to be in $\mathcal{F}^{\geq \ell}$ is that there exists two states p and q 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

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

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 for the probability of belonging to $\mathcal{F}^{\geq \ell}$. By Lemma 4, for every $p, q, p', q' \in \{1, \ldots, n\}$, every $F \in \mathcal{F}_\ell(p, q, p', q')$ is compatible with a partition $P_\ell(p, q, p', q')$ into $n - \ell$ parts. Let r be the Bernoulli distribution of parameter γ on $\{1, \ldots, n\}$. By Lemma 5 we have

$$r(\mathcal{F}^{\geq \ell}) \leq \sum_{p,q,p',q' \in \{1,\ldots,n\}} r(\mathcal{F}_\ell(p, q, p', q')) \leq n^4 \beta^\ell. \tag{1}$$

Moreover, the average number of iterations of the main loop of Moore's algorithm is by definition

$$\sum_{F \subseteq \{1,\ldots,n\}} r(F) \cdot \textsc{Moore}(T, F) = \sum_{F \in \mathcal{F}^{<\ell}} r(F) \cdot \textsc{Moore}(T, F) + \sum_{F \in \mathcal{F}^{\geq \ell}} r(F) \cdot \textsc{Moore}(T, F),$$

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

$$\sum_{F \in \mathcal{F}^{<\ell}} r(F) \cdot \textsc{Moore}(T, F) \leq \ell \sum_{F \in \mathcal{F}^{<\ell}} r(F) \leq \ell.$$

Bounding $\textsc{Moore}(T, F)$ from above by n when $F \in \mathcal{F}^{\geq \ell}$ (see Lemma 1) and estimating $r(\mathcal{F}^{\geq \ell})$ with (1), we get that

$$\sum_{F\in\mathcal{F}^{\geq\ell}} r(F)\cdot \textsc{Moore}(T,F) \leq n^5\beta^\ell.$$

Finally, choosing $\ell = \lceil 5\log_{\frac{1}{\beta}} n\rceil$, we obtain that

$$\sum_{F\subseteq\{1,\ldots,n\}} r(F)\cdot \textsc{Moore}(T,F) \leq \lceil 5\log_{\frac{1}{\beta}} n\rceil + n^5\beta^{\lceil 5\log_{\frac{1}{\beta}} n\rceil} \leq 5\log_{\frac{1}{\beta}} n + 2.$$

This concludes the proof. □

4.5.2 *Incomplete Transition Structures*

Recall that before applying Moore's algorithm to an incomplete automaton, a sink state which is not final is added, as described in Sect. 2.2. Starting from such an n-state automaton $\mathcal{A} = (T, F)$, the result is an $(n+1)$-state complete automaton $\mathcal{A}' = (T', F)$, where T' is complete.

The proof for incomplete transition structures is roughly the same as before, but the distribution on set of final states has changed, since the new sink state is always non-final: though very similar, this is not a Bernoulli distribution anymore. Let r be the Bernoulli distribution of parameter γ on elements of $\{1,\ldots,n\}$. We denote by $\tilde{r}$ the probability it induces on $\{1,\ldots,n+1\}$, when sets containing $n+1$ have probability zero: for any $X\subseteq\{1,\ldots,n+1\}$, $\tilde{r}(X) = r(X)$ if $n+1\notin X$ and $\tilde{r}(X) = 0$ if $n+1\in X$.

Hence if we fix $T\in\mathcal{T}_n$ and consider the automaton $\mathcal{A} = (T, F)$, where the final states of F are distributed following r, the final states of the automaton $\mathcal{A}'$ are distributed following $\tilde{r}$.

We first establish a result similar to Lemma 5.

Lemma 6 *Let r be a Bernoulli distribution of parameter $\gamma\in(0,1)$ on elements of $\{1,\ldots,n\}$, for $n\geq 1$. Let P be a partition of $\{1,\ldots,n+1\}$ into $m\geq 1$ parts. The probability, under $\tilde{r}$, that a subset of $\{1,\ldots,n+1\}$ is compatible with P is at most β^{n+1-m}, where $\beta = \max(\gamma, 1-\gamma)$.*

Proof Let $P = \{E_1,\ldots,E_m\}$ and assume without loss of generality that $n+1\in E_1$. Since for $\tilde{r}$, $n+1$ cannot be in a subset having positive probability, it is necessary that $X\cap E_1 = \emptyset$ for a random X that is compatible with P. This happens with probability $(1-\gamma)^{|E_1|-1}$. For the remaining elements, observe that $\tilde{r}$ induces an independent Bernoulli distribution on $\{1,\ldots,n\}\setminus E_1$, that must be compatible with the partition $\{E_2,\ldots,E_m\}$. By Lemma 5, the probability of this event is bounded from above by $\beta^{|E_2|+\cdots+|E_m|-(m-1)}$. By independency, X is compatible with P with probability bounded from above by $(1-\gamma)^{|E_1|-1}\beta^{|E_2|+\cdots+|E_m|-(m-1)} \leq \beta^{n+1-m}$. □

Proposition 4 (Incomplete transition structures) *Let $k\geq 1$, $n\geq 1$ and let $T\in\mathcal{T}_n$ be an incomplete transition structure over a k-letter alphabet. For the Bernoulli distribution of parameter γ on final states, the average number of iterations of the main loop of Moore's algorithm applied to (T', F), where T' is obtained by completing T, is bounded from above by $5\log_{\frac{1}{\beta}}(n+1)+2$.*

Proof The proof mimics the one of Proposition 3, but with $n+1$ instead of n, since we have the additional sink state (the bound obtained in Lemma 6 is compatible with the $n+1$ shift). The final upper bound is therefore $5\log_{\frac{1}{\beta}}(n+1)+2$, concluding the proof. □

4.5.3 Proof of Theorem 2

We have all the ingredients to prove our main theorem.

Proof Let C_n and $\overline{C}_n$ respectively denote the sets of complete and incomplete transition structures of $\mathcal{T}_n$.

If p is a Bernoulli model on $\mathcal{D}$ of parameter γ, then by definition, for every $(T,F)\in\mathcal{D}_n$, $p((T,F)) = q(T)\,r(F)$ for a probability q over $\mathcal{T}_n$ and where r is a Bernoulli model of parameter γ on elements of $\{1,\dots,n\}$. The average number of iterations in Moore's algorithm, for n-state automata is therefore

$$\sum_{\substack{(T,F)\in\mathcal{D}_n\\ T\in\mathcal{C}_n}} p(\mathcal{A})\cdot \textsc{Moore}(T,F) + \sum_{\substack{(T,F)\in\mathcal{D}_n\\ T\in\overline{\mathcal{C}}_n}} p(\mathcal{A})\cdot \textsc{Moore}(T',F),$$

where T' is the complete automaton associated with T.

For complete automata, we have

$$\sum_{\substack{(T,F)\in\mathcal{D}_n\\ T\in\mathcal{C}_n}} p(\mathcal{A})\cdot \textsc{Moore}(T,F) = \sum_{T\in\mathcal{C}_n} q(T) \sum_{F\subseteq\{1,\dots,n\}} r(F)\cdot \textsc{Moore}(T,F).$$

By Proposition 3, the latter sum is bounded from above by $5\log_{\frac{1}{\beta}} n+2$, hence by $5\log_{\frac{1}{\beta}}(n+1)+2$. Moreover, $\sum_{T\in\mathcal{C}_n} q(T) = q(\mathcal{C}_n)$, therefore

$$\sum_{\substack{(T,F)\in\mathcal{D}_n\\ T\in\mathcal{C}_n}} p(\mathcal{A})\cdot \textsc{Moore}(T,F) \le q(\mathcal{C}_n)\left(5\log_{\frac{1}{\beta}}(n+1)+2\right).$$

For incomplete automata, recall that the definition of $\tilde{r}$ is given in Sect. 4.5.2, and we have

$$\sum_{\substack{(T,F)\in\mathcal{D}_n\\ T\in\overline{\mathcal{C}}_n}} p(\mathcal{A})\cdot \textsc{Moore}(T',F) = \sum_{T\in\overline{\mathcal{C}}_n} q(T) \sum_{F\subseteq\{1,\dots,n+1\}} \tilde{r}(F)\cdot \textsc{Moore}(T',F).$$

By Proposition 4, we obtain

$$\sum_{\substack{(T,F)\in\mathcal{D}_n\\ T\in\overline{\mathcal{C}}_n}} p(\mathcal{A})\cdot \textsc{Moore}(T',F) \le q(\overline{\mathcal{C}}_n)\left(5\log_{\frac{1}{\beta}}(n+1)+2\right).$$

Therefore, since $q(\mathcal{C}_n)+q(\overline{\mathcal{C}}_n)=1$, the average number of iteration is $\mathcal{O}(\log n)$, and by Lemma 1, the average complexity of Moore's algorithm is bounded from above by $\mathcal{O}(kn\log n)$, concluding the proof. □

5 Additional Results

In this section we analyze several specific distributions on automata, to emphasize the limits of the method presented in this paper.

5.1 Variations on Bernoulli Models

In fact, Theorem 2 can be established for more general distributions of automata. But this requires a heavier formalism, describing new families of distributions that resemble Bernoulli models a lot. The main point is that the proofs of Proposition 3 and Proposition 4 rely on properties formalized in Lemma 5 and Lemma 6: for the considered distributions of final states, the probability that a size-n set of final states is compatible with a given partition P with m parts is exponentially small in m.

Hence, the proof of Theorem 2 can readily be adapted whenever the probabilistic model on $\mathcal{D}$ satisfies:

i. The transition structures and the set of final states are chosen independently.
ii. The transition structures follow any distribution on $\mathcal{T}$.
iii. The probability r on sets of final states satisfies the following property: there exist $\alpha > 0$ and $\beta \in (0, 1)$ such that, for any partition $P = \{E_1, \ldots, E_m\}$ of $\{1, \ldots, n\}$, the probability that $X \subseteq \{1, \ldots, n\}$ is compatible with P is at most $\alpha\beta^{n-m}$.

The bounds given in Proposition 3 and Proposition 4 change a bit, by some additive constant terms depending on α and β.

As an example, consider the uniform distribution r on non-trivial subsets of $\{1, \ldots, n\}$: For $n = 1$, let $r(\{1\}) = r(\emptyset) = \frac{1}{2}$; for $n \geq 2$, define for any $X \in \{1, \ldots, n\}$,

$$r(X) = \begin{cases} 0 & \text{if } X = \emptyset \text{ or } X = \{1, \ldots, n\}, \\ \frac{1}{2^n - 2} & \text{otherwise.} \end{cases}$$

Let us establish property iii for this distribution. For any partition P in m parts of $\{1, \ldots, n\}$, denote by C_P the set of subsets of $\{1, \ldots, n\}$ that are compatible with P. Note that $\emptyset \in C_P$ and $\{1, \ldots, n\} \in C_P$. Hence, for $n \geq 2$,

$$\sum_{X \in C_P} r(X) = \frac{|C_P| - 2}{2^n - 2} \leq \frac{|C_P|}{2^n - 2}.$$

By Lemma 5 applied with $\gamma = \frac{1}{2}$,

$$\sum_{X \in C_P} \frac{1}{2^n} = \frac{|C_P|}{2^n} \leq \left(\frac{1}{2}\right)^{n-m}.$$

Therefore, for any $n \geq 2$,

$$\sum_{X \in C_P} r(X) \leq \frac{2^n}{2^n - 2} \left(\frac{1}{2}\right)^{n-m} \leq 2 \left(\frac{1}{2}\right)^{n-m}.$$

Hence, the uniform distribution on automata with non-trivial sets of final states satisfies the condition i, ii and iii above, with $\alpha = 2$ and $\beta = \frac{1}{2}$. The average complexity of Moore's algorithm is $\mathcal{O}(kn \log n)$ for this distribution too.

5.2 Unary Automata

一元自动机有利于构造反例

In the remainder, we will use the uniform probabilistic model on complete unary automata. It is quite simple and well-known [17], and therefore useful to design examples or counterexamples. In this section we recall some basic facts about these unary automata and their average properties.

An automaton is a *unary automaton* when it is defined on a one-letter alphabet. We fix a one-letter alphabet $A = \{a\}$ and denote by $\mathcal{U}$ the set of complete automata on A, and by $\mathcal{U}_n$ the set of the n-state automata of $\mathcal{U}$. As described in [17], in the transition structure of a complete n-state unary automaton $\mathcal{U}$ over the alphabet $\{a\}$, the n states are $q_i = q_0 \cdot a^i$, for $i \in \{0, \ldots, n-1\}$, and the transition structure is determined by the choice of $q_c = q_{n-1} \cdot a$. The part $\{q_0, \ldots, q_{c-1}\}$, which can be empty if $c = 0$, is called the *queue* of the automaton, and the part $\{q_c, \ldots, q_{n-1}\}$ is its *cycle*.

Taking into account the $n!$ distinct ways to label the states, the number of distinct labelled transition structures in $\mathcal{U}_n$ is $n \cdot n!$. Thus $|\mathcal{U}_n| = n2^n n!$. For the uniform distribution on $\mathcal{U}$, the probability of any n-state automaton is therefore $\frac{1}{|\mathcal{U}_n|} = \frac{1}{n2^n n!}$.

We will use the following result:

Proposition 5 [17] *For the uniform distribution on $\mathcal{U}$, the probability for a n-state automaton to be minimal is asymptotically equivalent to $\frac{1}{2}$.*

5.3 Tightness for Unary Automata and Lower Bound for the Model

In this section we prove that the bound $\mathcal{O}(kn \log n)$ is optimal for the uniform distribution on $\mathcal{U}$, and for similar distributions when $k \geq 2$.

Proposition 6 *For the uniform distribution on deterministic accessible and complete unary n-state automata, the average time complexity of Moore's state minimization algorithm is $\Theta(n \log n)$.*

Proof By Theorem 2, and since $k = 1$, the average time complexity is $\mathcal{O}(n \log n)$.

By 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 minimal n-state unary automaton is at least $Cn \log n$. Let m_n denote the number of minimal n-state unary automata. Taking into account the contribution of minimal automata only, the average complexity of Moore's algorithm is bounded from below by

$$\frac{m_n}{|\mathcal{U}_n|} Cn \log n \sim \frac{1}{2} Cn \log n,$$

where the equivalence follows from Proposition 5. Hence, the average complexity is $\Omega(n \log n)$, concluding the proof. □

An important consequence of Proposition 6 is that Theorem 2 is optimal without further conditions on the probabilistic model: For $k = 1$, this is the case for the uniform distribution on $\mathcal{U}$, by Proposition 6. For $k \geq 2$, let $\mathcal{U}'$ be the set of complete automata on A, such that for every state p, for every letters $a, b \in A$, $p \cdot a = p \cdot b$. Hence, an element of $\mathcal{U}'$ is a complete unary automaton, whose transitions have been duplicated. By Lemma 3, the uniform distribution on $\mathcal{U}'$ is a Bernoulli model. Moreover, we get the following result as a consequence of Proposition 6:

Corollary 2 *For any k-letter alphabet A, with $k \geq 2$, the average time complexity of Moore's algorithm for the uniform distribution on $\mathcal{U}'$ is $\Theta(kn \log n)$.*

Proof Each automaton of $\mathcal{U}'$ can be seen as an element of $\mathcal{U}$ whose transitions have been duplicated. The process of Moore's algorithm applied to such an automaton uses exactly the same number of iterations as if performed on the associated element in $\mathcal{U}$. Since the induced distribution on associated automata is the uniform distribution on $\mathcal{U}$, Proposition 6 holds, the cost being multiplied by k because each iteration costs k times as much instructions as in the unary case.

Theorem 2 gives the upper bound, concluding the proof. □

5.4 An Example Where Transition Structures and Sets of Final States Are not Independent: Co-accessible Automata

In this section we consider the uniform distribution on complete and co-accessible automata. This probabilistic model is not a Bernoulli model, nor a simple version of a Bernoulli model, since transition structures and sets of final states are not independent: an example of two 2-state transition structures having different impossible sets of final states is depicted in Fig. 5.

However, the average complexity of Moore's algorithm is still $\mathcal{O}(kn \log n)$. The proof we propose below is based on the specific shape of a typical transition structure under the uniform distribution on $\mathcal{T}_n$. The key argument is that the proportion of co-accessible automata amongst complete ones is not negligible, which is a consequence of a result by Korshunov [25]:

Theorem 3 [25] *For any alphabet A of size $k \geq 2$, there exists a real constant $0 < c_k < 1$, which only depends on k, such that for the uniform distribution on complete transition structures, the probability for a transition structure to be strongly connected tends to c_k as the number of states tends to infinity.*

The result of this section is the following.

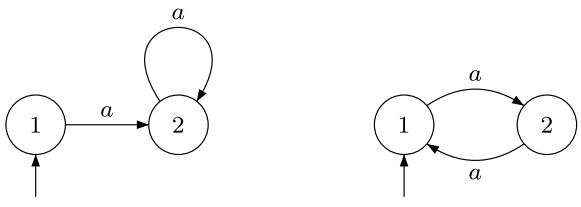

Fig. 5 On the *left*, the sets of final states ∅ and {1} are not possible for the automaton to be co-accessible. On the *right*, only the empty set is not allowed

Proposition 7 *For the uniform distribution on complete and co-accessible automata, the average time complexity of Moore's state minimization algorithm is* $\mathcal{O}(kn\log n)$.

Proof Let $\mathcal{C}_n \subseteq \mathcal{D}_n$ denote the set of complete n-state automata, and let $\mathcal{A}_n \subseteq \mathcal{C}_n$ denote the set of complete and co-accessible n-state automata.

For $\ell \geq 1$, let $\mathcal{A}_n^{<\ell}$ (resp. $\mathcal{A}_n^{\geq \ell}$) denote the subset of $\mathcal{A}_n$ consisting of the automata $\mathcal{A}$ such that $\textsc{Moore}(\mathcal{A}) \leq \ell$ (resp. $\textsc{Moore}(\mathcal{A}) > \ell$), as in the proof of Proposition 3.

First, note that $|\mathcal{A}_n| = \Theta(|\mathcal{C}_n|)$:

- For $k = 1$, an element of $\mathcal{C}_n = \mathcal{U}_n$ is in $\mathcal{A}_n$ if and only if its cycle contains at least one final state. Therefore, if $q_{n-1} \cdot a = q_c$, the automaton is not co-accessible if and only if the $n - c$ states in the cycle are not final. Hence,

$$|\mathcal{C}_n \setminus \mathcal{A}_n| = n! \sum_{c=0}^{n-1} 2^c = (2^n - 1)n! = o(|\mathcal{C}_n|).$$

- For $k \geq 2$, this is a consequence of Theorem 3: if a transition structure is strongly connected then any choice of set of final states but the empty set leads to a co-accessible automaton. Hence, if S_n denotes the number of strongly connected elements of $\mathcal{C}_n$, then there are at least $(1 - 2^{-n})S_n$ co-accessible elements in $\mathcal{C}_n$, the statement follows since $S_n = \Theta(|\mathcal{C}_n|)$ by Theorem 3.

The uniform distribution on $\mathcal{C}_n$ is a Bernoulli model on automata which are complete, hence the result of Proposition 3 holds. In the proof, it is shown that for such a distribution and for $\ell = \lceil 5\log_2 n \rceil$, the probability that $\textsc{Moore}(\mathcal{A}) > \ell$ is $\mathcal{O}(\frac{1}{n})$. Since we are considering a uniform distribution, probabilities are directly related to cardinals, so that the number of n-state automata $\mathcal{A}$ of $\mathcal{C}_n$ such that $\textsc{Moore}(\mathcal{A}) > \ell$ is in $\mathcal{O}(\frac{1}{n}|\mathcal{C}_n|)$, for $\ell = \lceil 5\log_2 n \rceil$. Hence $|\mathcal{A}_n^{\geq \ell}|$ is $\mathcal{O}(\frac{1}{n}|\mathcal{C}_n|)$.

Therefore, since every element of $\mathcal{A}_n$ has probability $\frac{1}{|\mathcal{A}_n|}$ under the uniform distribution, and since $\textsc{Moore}(\mathcal{A}) \leq n$, the average number of iterations for $\ell = \lceil 5\log_2 n \rceil$ is

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

Hence, since $|\mathcal{A}_n| = \Theta(|\mathcal{C}_n|)$ we get

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

concluding the proof. □

5.5 Unary Automata with Exactly One Final State

Under a Bernoulli model, automata tend to have a large number of final states. It is natural to wonder what happens when Moore's algorithm is applied to automata with a small number of final states; the proposition below proves that we cannot expect to always have a $O(kn \log n)$ bound for such models.

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

Proof We use the notations of Sect. 5.2. Let $\mathcal{A}$ be an element of $\mathcal{U}_n$, with $q_c = q_{n-1} \cdot a$, that has only one final state q_f. First note that there are $n^2 \cdot n!$ such automata: n choices for c and f and $n!$ ways to label the states with $\{1, \ldots, n\}$. Assume that $f \geq 2$. Then the states q_0 and q_1 satisfy $q_0 \sim_{f-2} q_1$ but $q_0 \not\sim_{f-1} q_1$, since the states $q_0, \ldots, q_{f-1}$ are not final and q_f is. Hence, for such an automaton, $\textsc{Moore}(\mathcal{A}) \geq f$. Therefore, the average number of iterations for n-state automata with one final state, for the uniform distribution, is

$$\frac{1}{n^2 \cdot n!} \sum_{\substack{(T,F)\in\mathcal{U}_n \\ |F|=1}} \textsc{Moore}(\mathcal{A}) \geq \frac{1}{n^2} \sum_{c=0}^{n-1} \sum_{f=2}^{n-1} f = \Theta(n).$$

We used the fact that there are exactly $n!$ such automata for given c and f. □

6 Conclusion and Open Problems

We have seen that Moore's state minimization algorithm has an average complexity $\mathcal{O}(kn \log n)$ in many situations. Its average time complexity therefore matches the worst-case complexity of the best known minimization algorithm for complete automata, due to Hopcroft. One may wonder if we are in a situation like quicksort versus optimal sorting algorithms, where the best worst-case algorithms are not always the best choices. In our situation there is no evidence of such a phenomenon; Hopcroft's algorithm is closely related to Moore's algorithm, as stated in [9], and might therefore be fast when Moore's algorithm is.

We have also seen that our result cannot be improved without adding new conditions on the probabilistic model. Assuming uniformity on complete transition structures, the second author proved a $\mathcal{O}(kn \log \log n)$ behavior in [22]. An open question here is to prove that this result also holds for accessible and complete transition structures, the combinatorics behind this problem being much more difficult when the automata are accessible.

Another open question is raised by Fig. 6. Considering the uniform probabilistic model on complete automata with only one final state, on a two-letter alphabet, the average number of iterations seems to be sublinear, whereas it is $\Theta(n)$ for a one-letter alphabet (Proposition 8). This model is not a Bernoulli model, so it would be interesting to know whether there are $\mathcal{O}(\log n)$ average iterations in this case also.

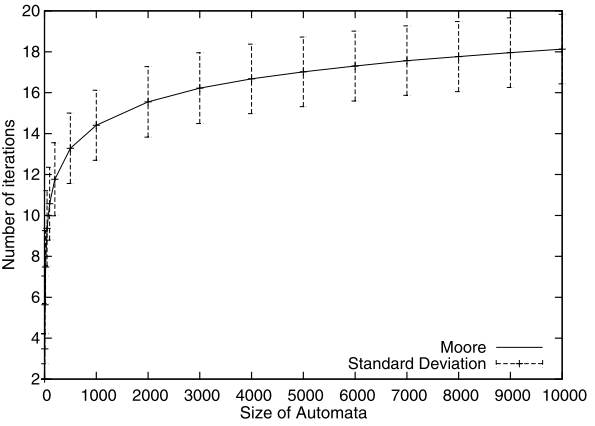

Fig. 6 Experimental study of the average number of iterations in Moore's algorithm for the uniform probabilistic model over deterministic accessible and complete automata with only one final state. For each size, the values are computed from 20 000 random automata on a 2-letter alphabet

Acknowledgements The authors wish to thank the anonymous referees for insightful comments and suggestions. They were supported by the ANR (GAMMA—project BLAN07-2_195422 and MAGNUM—project ANR-2010-BLAN-0204).

References

1. Moore, E.F.: Gedanken experiments on sequential machines. In: Automata Studies, pp. 129–153. Princeton University Press, Princeton (1956)
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 Inform. **2**, 97–109 (1973)
4. Knuutila, T.: Re-describing an algorithm by Hopcroft. Theor. Comput. Sci. **250**(1–2), 333–363 (2001)
5. Berstel, J., Boasson, L., Carton, O.: Continuant polynomials and worst-case behavior of Hopcroft's minimization algorithm. Theor. Comput. Sci. **410**(30–32), 2811–2822 (2009)
6. Castiglione, G., Restivo, A., Sciortino, M.: Hopcroft's algorithm and cyclic automata. In: Martín-Vide, C., Otto, F., Fernau, H. (eds.) Second International Conference on Language and Automata Theory and Applications (LATA 2008). Lecture Notes in Computer Science, vol. 5196, pp. 172–183. Springer, Berlin (2008)
7. Castiglione, G., Restivo, A., Sciortino, M.: On extremal cases of Hopcroft's algorithm. Theor. Comput. Sci. **411**(38–39), 3414–3422 (2010)
8. Blum, N.: An $\mathcal{O}(n \log n)$ implementation of the standard method for minimizing n-state finite automata. Inf. Process. Lett. **57**(2), 65–69 (1996)
9. Lothaire, M.: Applied Combinatorics on Words. Encyclopedia of Mathematics and Its Applications, vol. 105. Cambridge University Press, Cambridge (2005)
10. Valmari, A., Lehtinen, P.: Efficient minimization of DFAs with partial transition functions. In: Albers, S., Weil, P. (eds.) 25th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2008),Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany. Dagstuhl Seminar Proceedings, vol. 08001, pp. 645–656 (2008)
11. Beal, M.P., Crochemore, M.: Minimizing incomplete automata. In: Seventh International Workshop on Finite-State Methods and Natural Language Processing (FSMNLP'08), pp. 9–16 (2008)
12. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading (1979)
13. Brzozowski, J.A.: Canonical regular expressions and minimal state graphs for definite events. In: Symposium on the Mathematical Theory of Automata. Polytechnic Institute of Brooklyn, vol. 12, pp. 529–561. Polytechnic Press, New York (1962)
14. Champarnaud, J.M., Khorsi, A., Paranthoen, T.: Split and join for minimizing: Brzozowski's algorithm. In: The Prague Stringology Conference'02, pp. 96–104 (2002)

15. 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)
16. Revuz, D.: Minimisation of acyclic deterministic automata in linear time. Theor. Comput. Sci. **92**(1), 181–189 (1992)
17. Nicaud, C.: Average state complexity of operations on unary automata. In: Kutylowski, M., Pacholski, L., Wierzbicki, T. (eds.) 24th International Symposium on Mathematical Foundations of Computer Science 1999 (MFCS'99). Lecture Notes in Computer Science, vol. 1672, pp. 231–240. Springer, Berlin (1999)
18. Beal, M.P., Crochemore, M.: Minimizing local automata. In: Caire, G., Fossorier, M. (eds.): IEEE International Symposium on Information Theory (ISIT'07), pp. 1376–1380 (2007)
19. Bassino, F., Nicaud, C.: Enumeration and random generation of accessible automata. Theor. Comput. Sci. **381**, 86–104 (2007)
20. Bassino, F., David, J., Nicaud, C.: REGAL: a library to randomly and exhaustively generate automata. In: 12th International Conference Implementation and Application of Automata (CIAA 2007). Lecture Notes in Computer Science, vol. 4783, pp. 303–305 (2007)
21. Bassino, F., David, J., Nicaud, C.: Enumeration and random generation of possibly incomplete deterministic automata. Pure Math. Appl. **19**, 1–16 (2010)
22. David, J.: The average complexity of Moore's state minimization algorithm is $\mathcal{O}(n \log \log n)$. In: Hlinený, P., Kucera, A. (eds.) 35th International Symposium on Mathematical Foundations of Computer Science (MFCS'10). Lecture Notes in Computer Science, vol. 6281, pp. 318–329. Springer, Berlin (2010)
23. Bassino, F., David, J., Nicaud, C.: On the average complexity of Moore's state minimization algorithm. In: Albers, S., Marion, J.Y. (eds.) 26th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2009), Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, Germany Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany. Dagstuhl Seminar Proceedings, vol. 09001, pp. 123–134 (2009)
24. Nerode, A.: Linear automaton transformation. In: Proc. American Mathematical Society, pp. 541–544 (1958)
25. Korshunov, A.D.: On the number of non-isomorphic strongly connected finite automata. Elektron. Inf.verarb. Kybern. **22**(9), 459–462 (1986)