Random Automata

Cyril Nicaud

LIGM – Université Gustave Eiffel & CNRS

GASCOM 2024

Introduction

PRIMALITY: check whether a given n is a prime number

PATTERN-MATCHING: check whether a given string P is contained in another string T

- Both can be solved using a computer

Introduction

PRIMALITY: check whether a given n is a prime number

PATTERN-MATCHING: check whether a given string P is contained in another string T

- Both can be solved using a computer
- Only **PATTERN-MATCHING** can be solved using an *automaton*

Introduction

PRIMALITY: check whether a given n is a prime number

PATTERN-MATCHING: check whether a given string P is contained in another string T

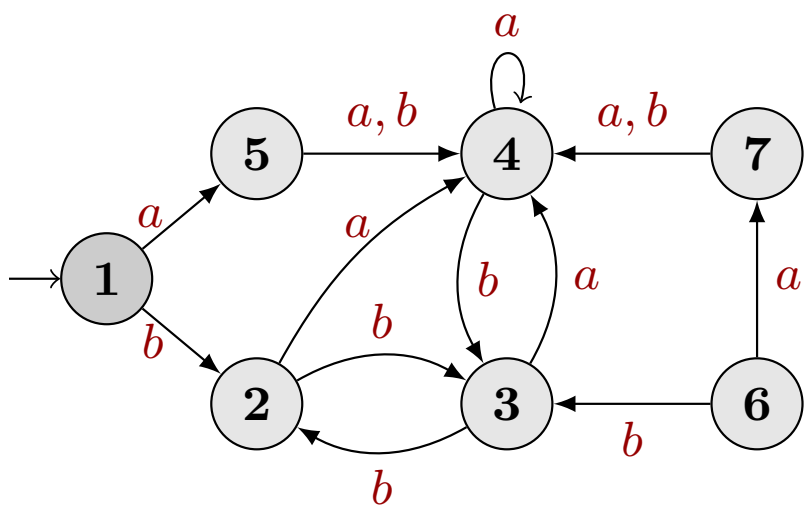
- Both can be solved using a computer
- Only **PATTERN-MATCHING** can be solved using an *automaton*

$\rightarrow$ *Problems that fall within automata theory possess many beneficial properties*

Deterministic and complete automata

A (deterministic and complete) automaton is a directed graph s.t.:

- vertices are called "states", edges are called "transitions"
- for each state and for each letter a of a fixed alphabet A, there is exactly one outgoing transition labeled by a
- there is a distinguished initial state

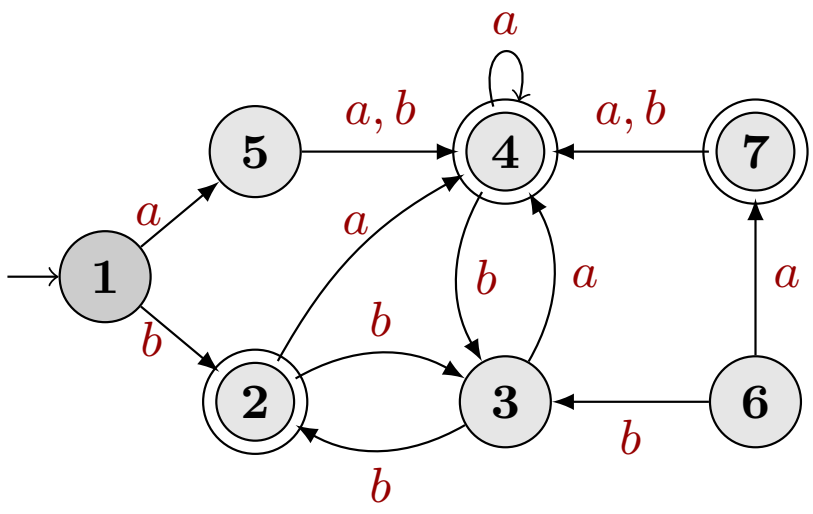


For every word $u \in A^*$, δ_u is the map from the set of states to itself, obtained by following the path labeled by u.

Deterministic and complete automata

A (deterministic and complete) automaton is a directed graph s.t.:

- One transition $p \xrightarrow{a} q$ outgoing from every state p for each letter a
- A distinguished initial state
- A *set of final states*

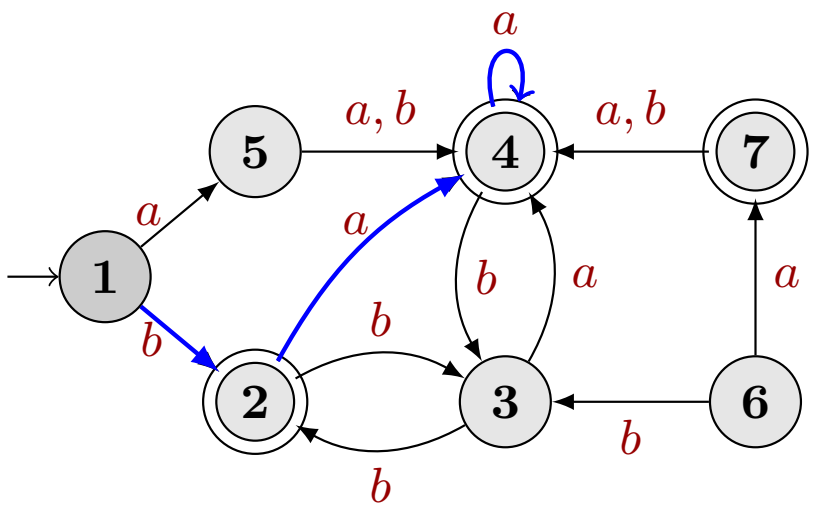


- A word is *recognized* if it labels a path from the initial state to a final state in the automaton $\mathcal{A}$
- The *language recognized* by $\mathcal{A}$ is the set $L(\mathcal{A})$ of recognized words

Deterministic and complete automata

A (deterministic and complete) automaton is a directed graph s.t.:

- One transition $p \xrightarrow{a} q$ outgoing from every state p for each letter a
- A distinguished initial state
- A *set of final states*



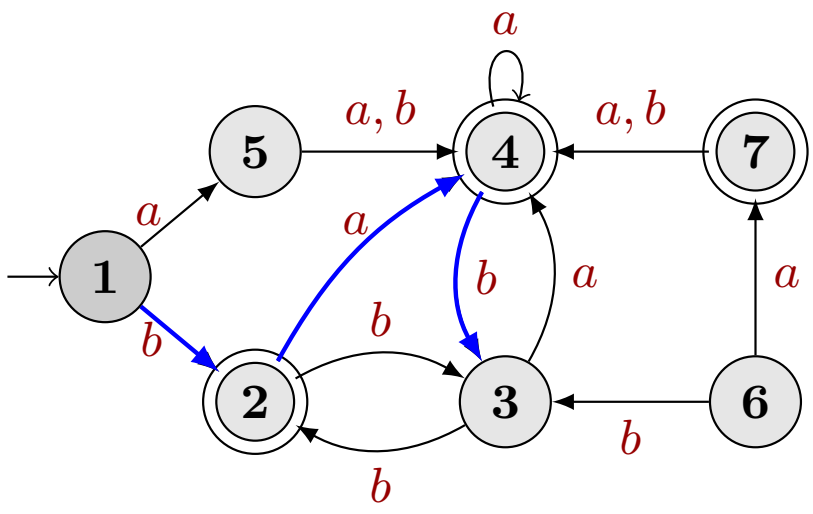
$baa \in L(\mathcal{A})$

- A word is *recognized* if it labels a path from the initial state to a final state in the automaton $\mathcal{A}$
- The *language recognized* by $\mathcal{A}$ is the set $L(\mathcal{A})$ of recognized words

Deterministic and complete automata

A (deterministic and complete) automaton is a directed graph s.t.:

- One transition $p \xrightarrow{a} q$ outgoing from every state p for each letter a
- A distinguished initial state
- A *set of final states*

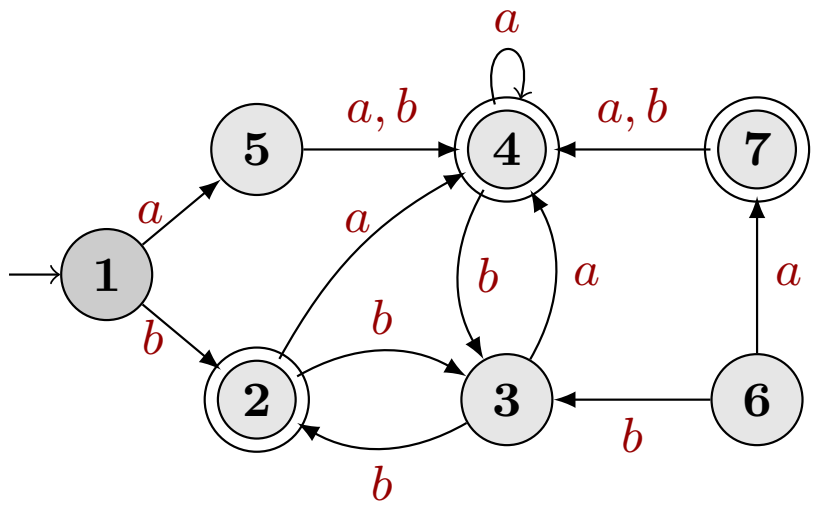


$bab \notin L(\mathcal{A})$

- A word is *recognized* if it labels a path from the initial state to a final state in the automaton $\mathcal{A}$
- The *language recognized* by $\mathcal{A}$ is the set $L(\mathcal{A})$ of recognized words

Accessible states, accessible automata

- State 6 and state 7 are useless (not accessible from the initial state)



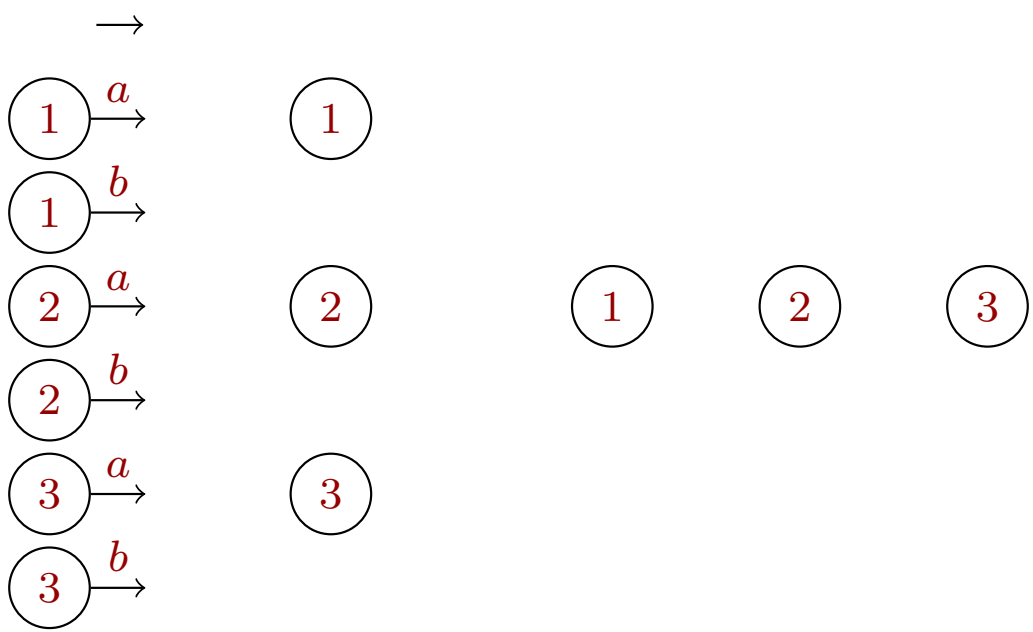
- An automaton is *accessible* when all its states are accessible

Question

Can we design an efficient algorithm to generate accessible automata with n states uniformly at random?

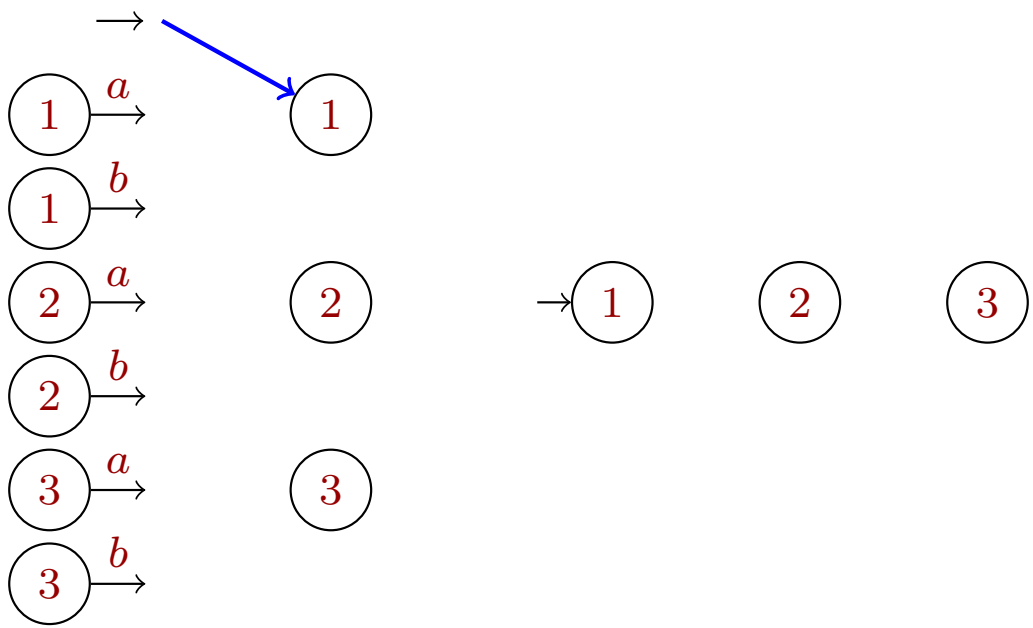
Korshunov's idea

- Consider automata where each state, except possibly the initial one, has an incoming transition
- This is a necessary condition for accessibility
- It's a surjection from the set of "arrows" (transition or initial $\to$) onto the set of states



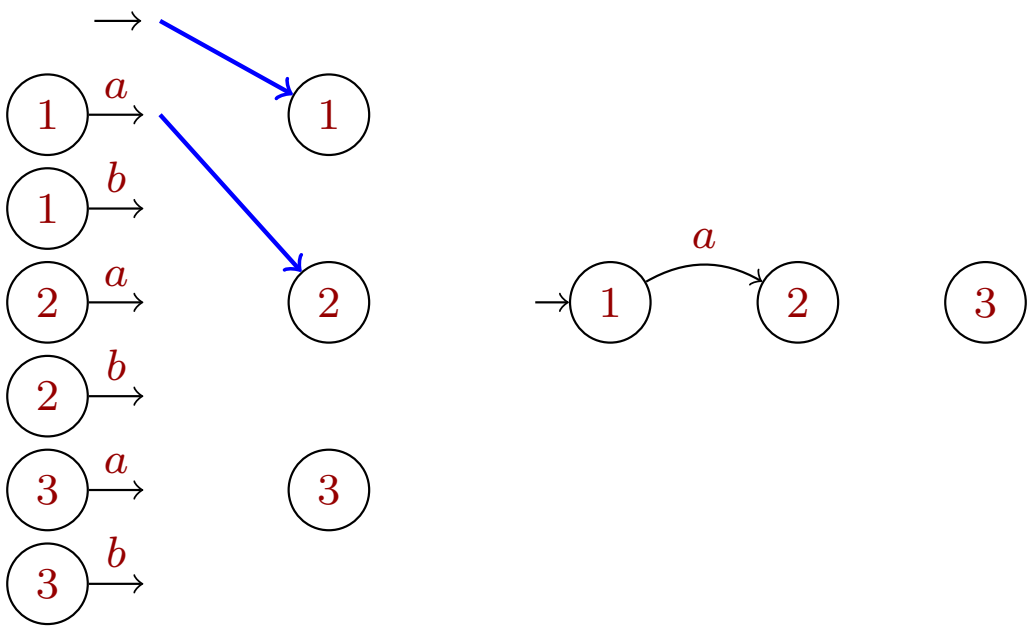
Korshunov's idea

- Consider automata where each state, except possibly the initial one, has an incoming transition
- This is a necessary condition for accessibility
- It's a surjection from the set of "arrows" (transition or initial $\to$) onto the set of states



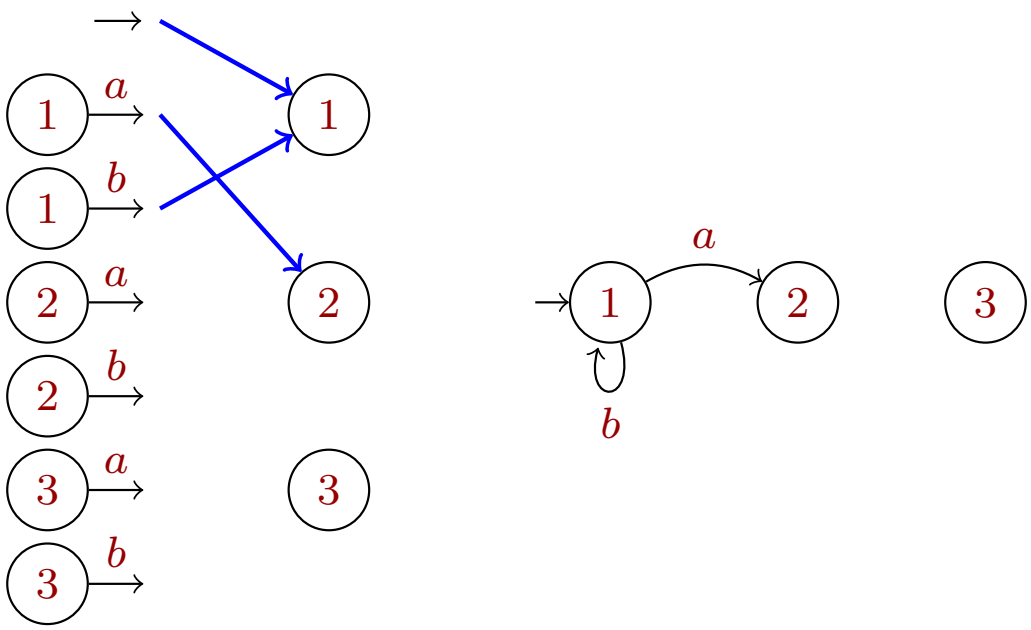
Korshunov's idea

- Consider automata where each state, except possibly the initial one, has an incoming transition
- This is a necessary condition for accessibility
- It's a surjection from the set of "arrows" (transition or initial $\to$) onto the set of states



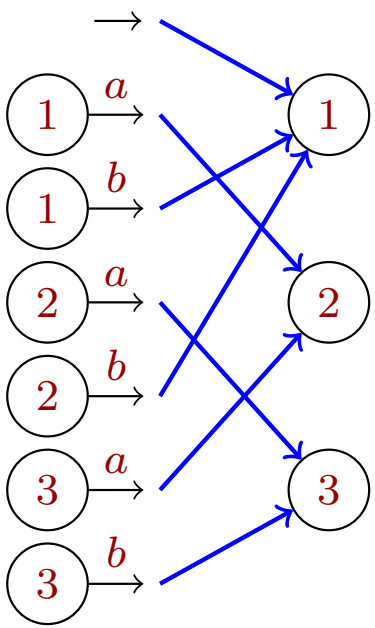
Korshunov's idea

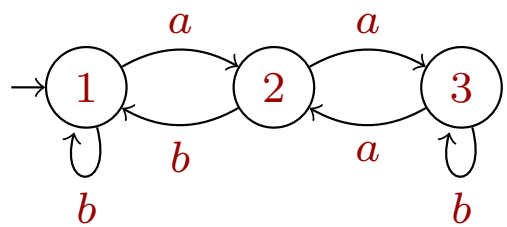
- Consider automata where each state, except possibly the initial one, has an incoming transition
- This is a necessary condition for accessibility
- It's a surjection from the set of "arrows" (transition or initial $\to$) onto the set of states



Korshunov's idea

- Consider automata where each state, except possibly the initial one, has an incoming transition
- This is a necessary condition for accessibility
- It's a surjection from the set of "arrows" (transition or initial $\to$) onto the set of states





Korshunov's theorem

Theorem [Korshunov 1978]

An asymptotically constant proportion of surjections of $[kn+1]$ onto $[n]$ produces accessible automata ($\approx 74.5\%$ for $k=2$)

We can generate such surjections :

- using the recursive method : precomputation in $\Theta(n^2)$ and generation in $\Theta(n)$ arithmetic operations [N. 2000]
- using Boltzmann sampler : $\Theta(n^{3/2})$ time for exact sampling [Bassino, N. 2006]

Korshunov's theorem

Theorem [Korshunov 1978]

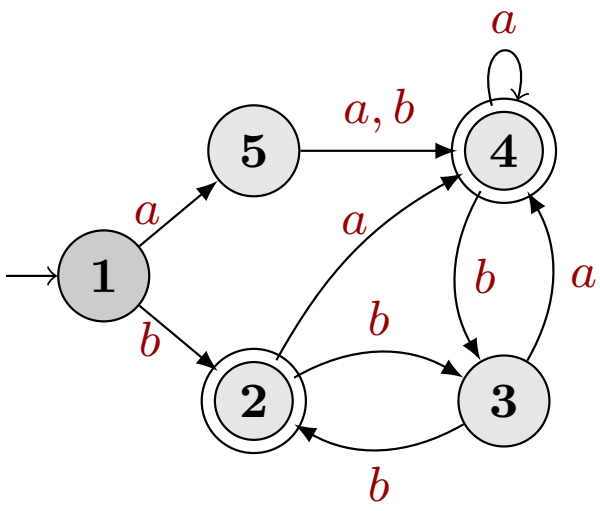
An asymptotically constant proportion of surjections of $[kn+1]$ onto $[n]$ produces accessible automata ($\approx 74.5\%$ for $k=2$)

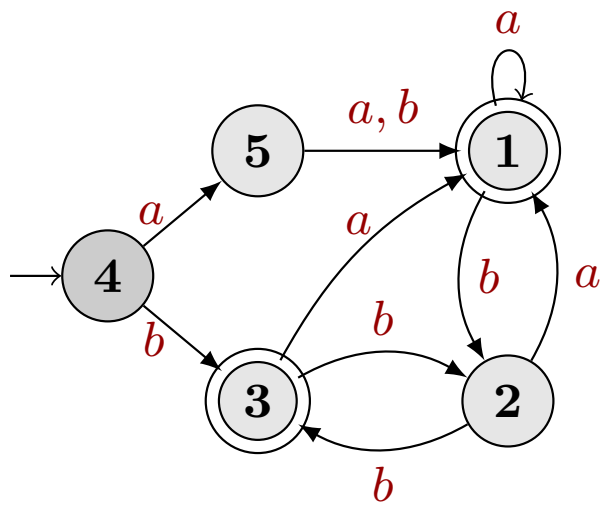
We can generate such surjections :

- using the recursive method : precomputation in $\Theta(n^2)$ and generation in $\Theta(n)$ arithmetic operations [N. 2000]
- using Boltzmann sampler : $\Theta(n^{3/2})$ time for exact sampling [Bassino, N. 2006]
- Mixed method in $\Theta(n)$ [Bassino, Sportiello, 2013]

$\Rightarrow$ This yields efficient algorithm to generate uniform accessible n-state automata using a simple rejection algorithm

Remark: no symmetry





- Changing state labels doesn't change the recognized language
- Each state is uniquely identified by its minlex shortest word that labels a path from the initial state
- There are $n!$ different ways to label the states (no symmetry)
- *not true for non-accessible automata*

So we could have worked on *partitions* instead of surjections

Counting accessible automata

Theorem [Korshunov 1978]

the number of n-state accessible automata is asymptotically equivalent to

$$E_k \cdot n! \cdot \left\{ {kn+1 \atop n} \right\} \cdot 2^n$$

- $\left\{ {n \atop k} \right\}$ is the number of partitions of $[n]$ into k parts, the *Stirling numbers of the second kind*: $\left\{ {n \atop k} \right\} = \left\{ {n-1 \atop k-1} \right\} + k \left\{ {n-1 \atop k} \right\}$

Theorem [Good 1961]

We have the asymptotic equivalent for fixed k

$$\left\{ {kn \atop n} \right\} \sim \alpha_k \cdot \beta_k^n \cdot n^{(k-1)n-1/2}$$

Proof. Saddle point method

Regular languages

- The *alphabet* is Σ, e.g. $\Sigma = \{a, b\}$
- A *language* is a subset of Σ^*
- *concatenation*: $u \cdot v = u_0 \cdots u_{\ell-1} \cdot v_0 \cdots v_{m-1}$, extended to languages $K \cdot L = \{u \cdot v : \ u \in K, \ v \in L\}$
- *Kleene star*: concatenations of an arbitrary number of words

$$L^\star = \{\varepsilon\} \cup L \cup L \cdot L \cup L \cdot L \cdot L \cup \ldots$$

The set $\mathcal{R}$ of *regular languages* is inductively defined by $\emptyset$, $\{\varepsilon\}$ and $\{a\}$ are in $\mathcal{R}$ for $a \in \Sigma$ and closed by union, concatenation and Kleene star

Kleene's Theorem

A language is recognized by an automaton iff it is regular

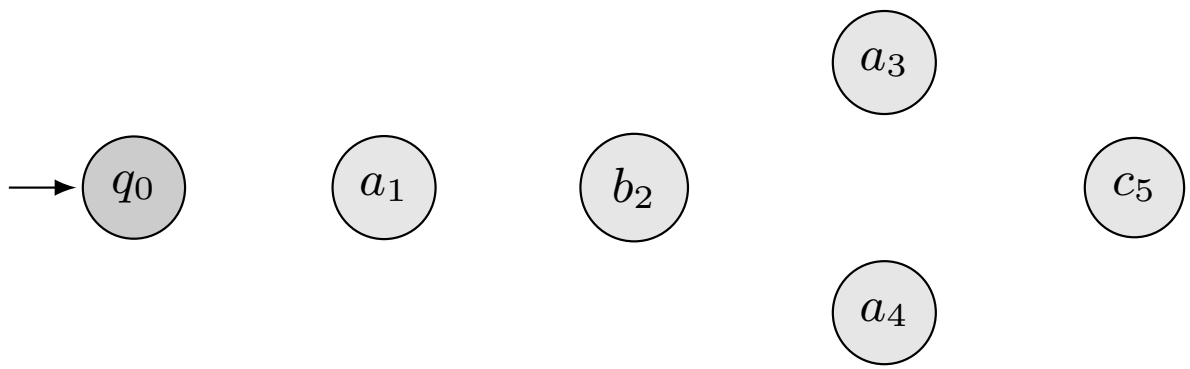
Regular expressions, Glushkov automaton

A *regular expression* is a formula that follows the inductive definition:

$$a \cdot b \cdot (a + a \cdot c)^\star + \varepsilon$$

The *Glushkov automaton* of a regular expression is obtained by:

- Distinguishing the letters $a_1 \cdot b_2 \cdot (a_3 + a_4 \cdot c_5)^\star + \varepsilon$
- Use an initial state q_0 and one state by letter



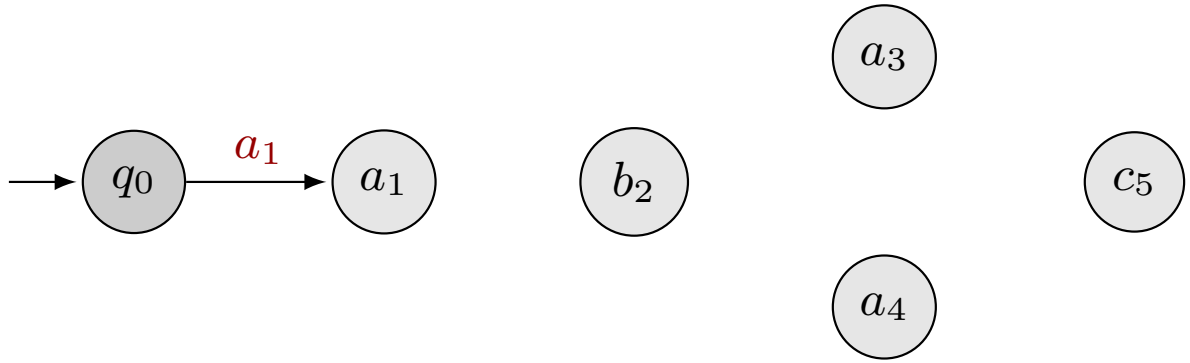
Regular expressions, Glushkov automaton

A *regular expression* is a formula that follows the inductive definition:

$$a \cdot b \cdot (a + a \cdot c)^\star + \varepsilon$$

The *Glushkov automaton* of a regular expression is obtained by:

- Distinguishing the letters $a_1 \cdot b_2 \cdot (a_3 + a_4 \cdot c_5)^\star + \varepsilon$
- Use an initial state q_0 and one state by letter
- Add $q_0 \xrightarrow{\alpha} \alpha$ if α starts a word of the language



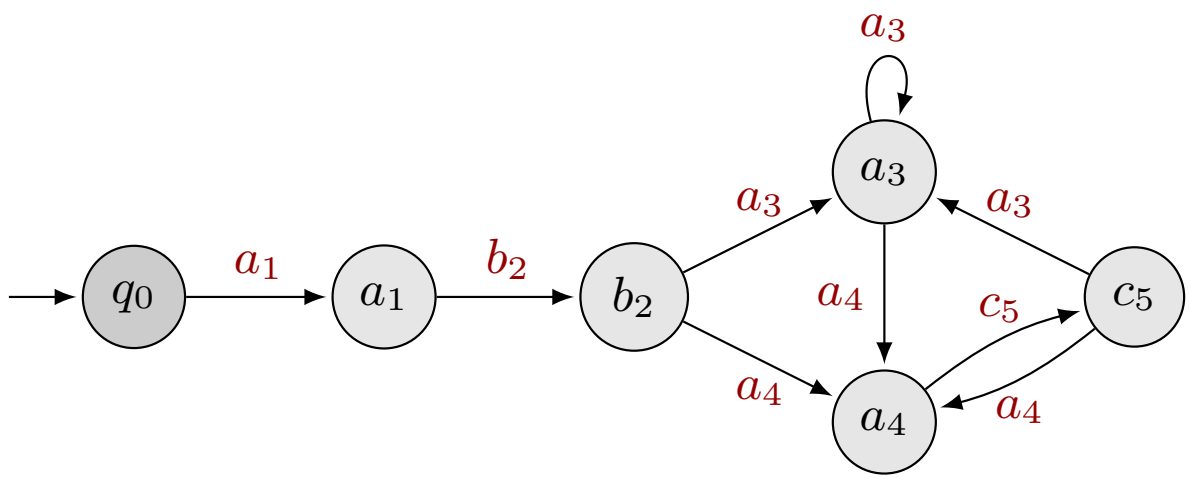
Regular expressions, Glushkov automaton

A *regular expression* is a formula that follows the inductive definition:

$$a \cdot b \cdot (a + a \cdot c)^\star + \varepsilon$$

The *Glushkov automaton* of a regular expression is obtained by:

- Distinguishing the letters $a_1 \cdot b_2 \cdot (a_3 + a_4 \cdot c_5)^\star + \varepsilon$
- Use an initial state q_0 and one state by letter
- Add $\alpha \xrightarrow{\beta} \beta$ if $\alpha\beta$ is a factor of a word of the language



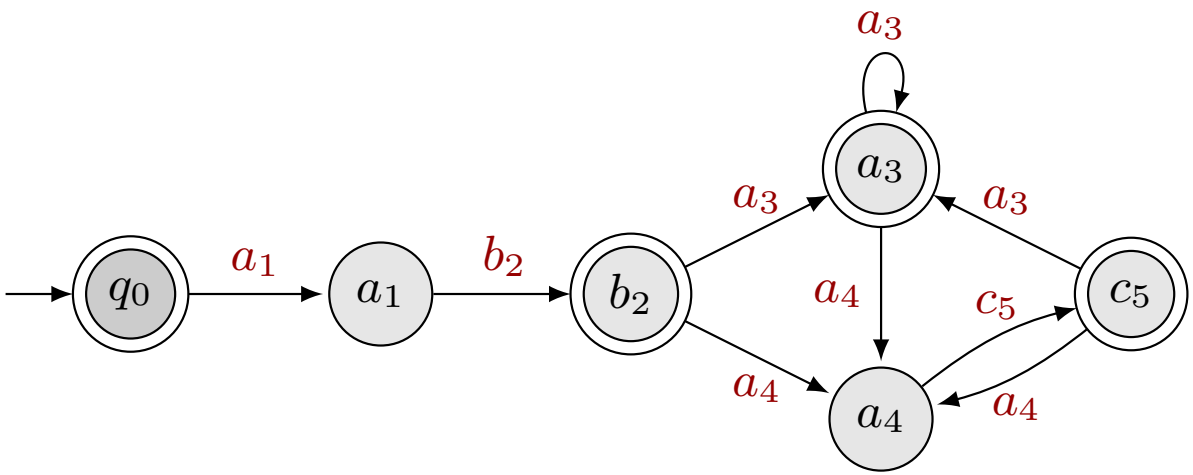
Regular expressions, Glushkov automaton

A *regular expression* is a formula that follows the inductive definition:

$$a \cdot b \cdot (a + a \cdot c)^\star + \varepsilon$$

The *Glushkov automaton* of a regular expression is obtained by:

- Distinguishing the letters $a_1 \cdot b_2 \cdot (a_3 + a_4 \cdot c_5)^\star + \varepsilon$
- Use an initial state q_0 and one state by letter
- α is final if it ends a word of the language



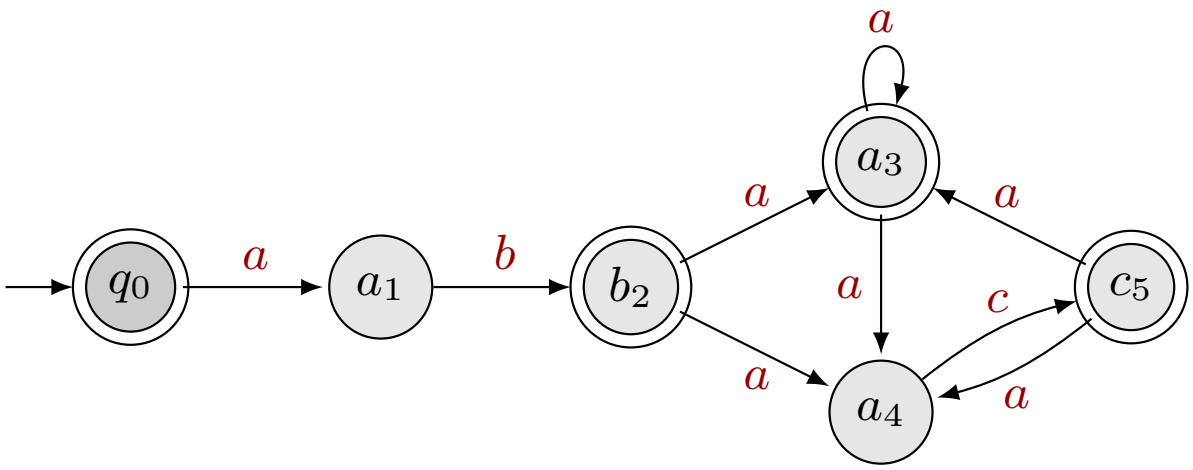
Regular expressions, Glushkov automaton

A *regular expression* is a formula that follows the inductive definition:

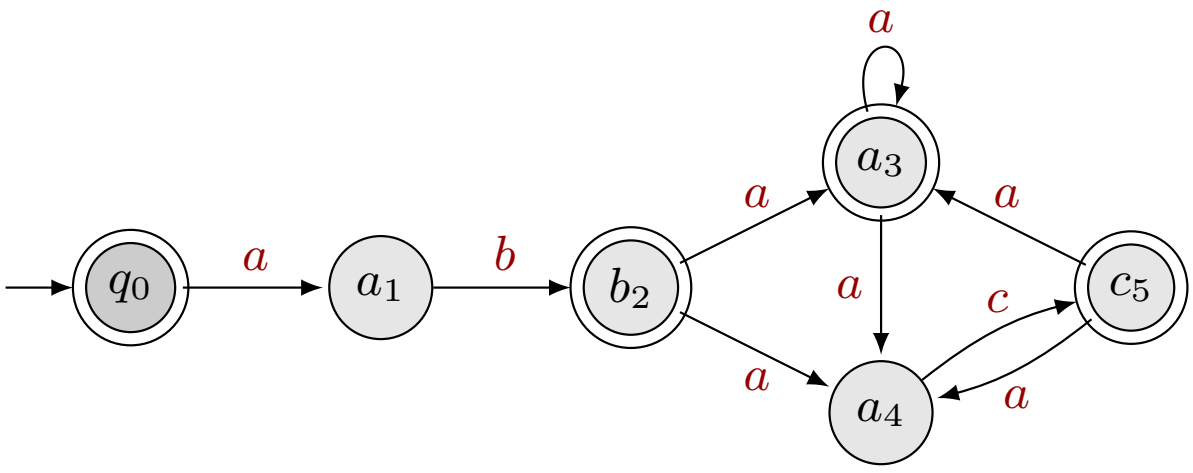
$$a \cdot b \cdot (a + a \cdot c)^\star + \varepsilon$$

The *Glushkov automaton* of a regular expression is obtained by:

- Distinguishing the letters $a_1 \cdot b_2 \cdot (a_3 + a_4 \cdot c_5)^\star + \varepsilon$
- Use an initial state q_0 and one state by letter
- remove the letter indices



Non-deterministic automata

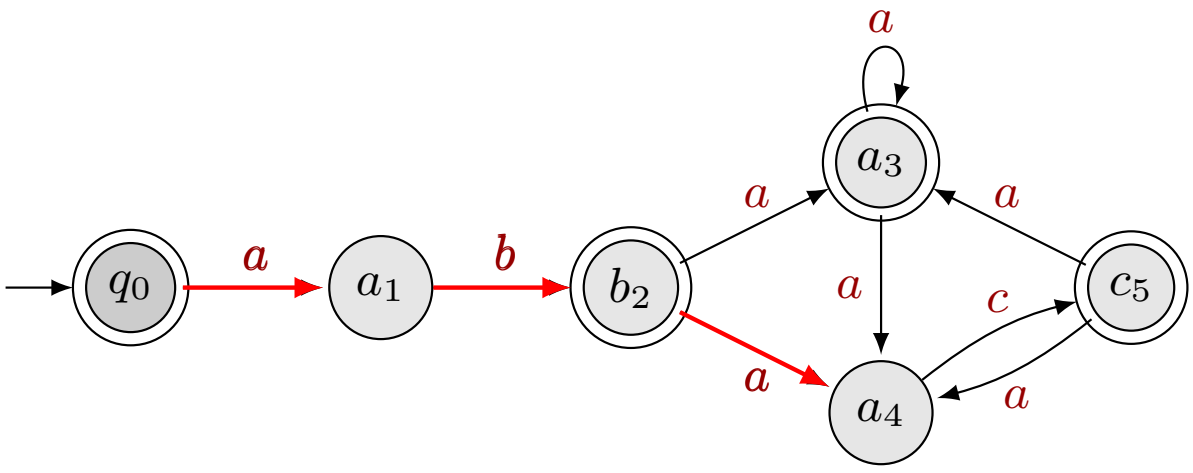


- This automaton is neither deterministic nor complete
- A word u is *accepted* by such an automaton if there is a path from an initial state to a final state labeled by u

Theorem

Non-deterministic automata and deterministic automata recognize the same languages (regular languages)

Non-deterministic automata

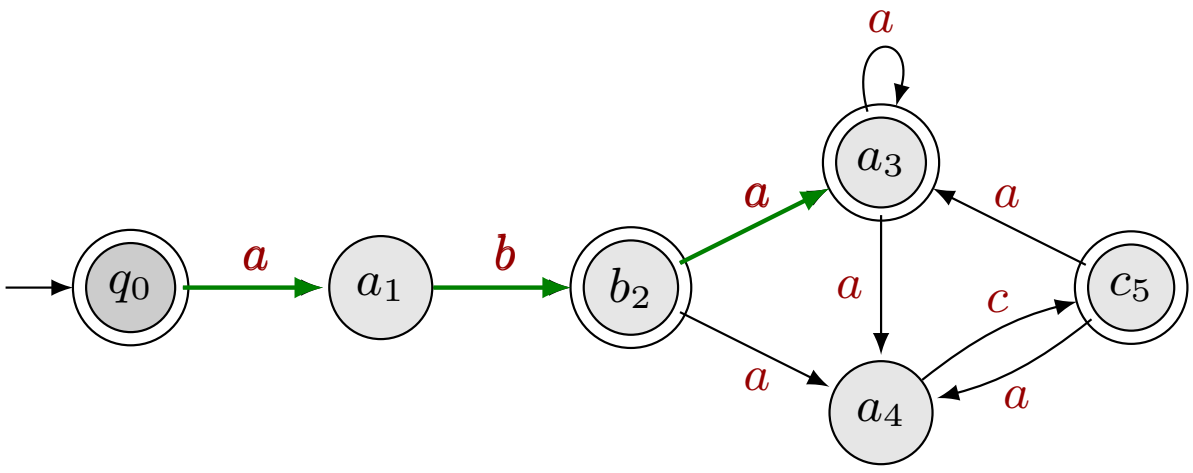


- This automaton is neither deterministic nor complete
- A word u is *accepted* by such an automaton if there is a path from an initial state to a final state labeled by u

Theorem

Non-deterministic automata and deterministic automata recognize the same languages (regular languages)

Non-deterministic automata

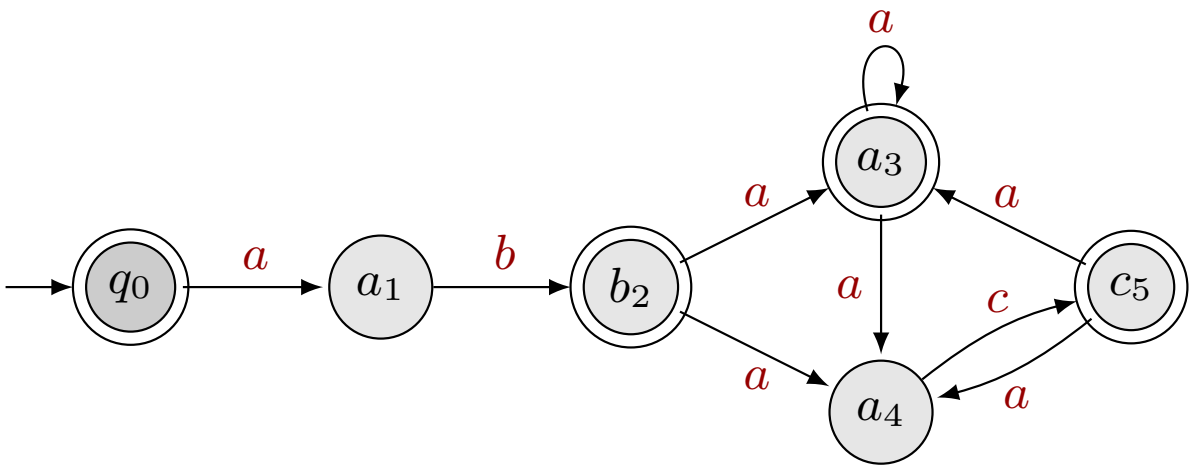


- This automaton is neither deterministic nor complete
- A word u is *accepted* by such an automaton if there is a path from an initial state to a final state labeled by u

Theorem

Non-deterministic automata and deterministic automata recognize the same languages (regular languages)

Back to Glushkov automaton



$$a \cdot b \cdot (a + a \cdot c)^\star + \varepsilon$$

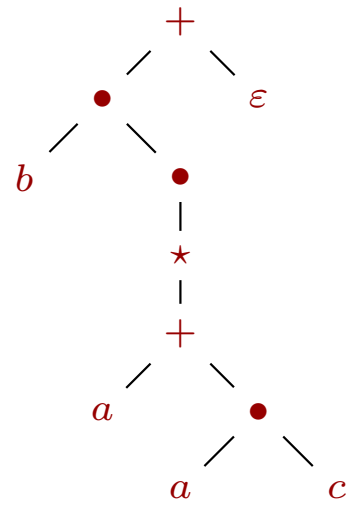
Theorem [Glushkov 1961]

A regular expression and its Glushkov automaton describe the same language

It has up to a quadratic number of transitions

Average case analysis of Glushkov's construction

- A regular expression can be seen as a *tree*, i.e. for $a \cdot b \cdot (a + a \cdot c)^\star + \varepsilon$



Theorem [N. 2009]

In expectation, the Glushkov automaton of a size-n regular expression taken uniformly at random has $\mathcal{O}(n)$ transitions

$\rightarrow$ followed by several results on other similar constructions

Wait a minute . . .

$(a+b)^\star$ is an absorbing pattern for the union $+$ on $\Sigma = \{a, b\}$:

$$E + (a+b)^\star \equiv (a+b)^\star + E \equiv (a+b)^\star$$

Theorem [Koechlin, N., Rotondo 2021]

The expected size of a random regular expression after applying the bottom-up simplification algorithm is bounded by a constant

Wait a minute ...

$(a+b)^\star$ is an absorbing pattern for the union $+$ on $\Sigma = \{a, b\}$:

$$E + (a+b)^\star \equiv (a+b)^\star + E \equiv (a+b)^\star$$

Theorem [Koechlin, N., Rotondo 2021]

The expected size of a random regular expression after applying the bottom-up simplification algorithm is bounded by a constant

- Works for many kind of expression, when there is an absorbing pattern
- Works in very general uniform settings

Wait a minute ...

$(a+b)^\star$ is an absorbing pattern for the union $+$ on $\Sigma = \{a, b\}$:

$$E + (a+b)^\star \equiv (a+b)^\star + E \equiv (a+b)^\star$$

Theorem [Koechlin, N., Rotondo 2021]

The expected size of a random regular expression after applying the bottom-up simplification algorithm is bounded by a constant

- Works for many kind of expression, when there is an absorbing pattern
- Works in very general uniform settings

$\rightarrow$ Uniform random expressions induce a degenerate distribution on regular languages

$\rightarrow$ *What about the languages recognized by random automata?*

Exercice: multiples of 6 in binary

We take $\Sigma = \{0, 1\}$ and L_6 is the language of the binary representations of multiples of 6 and ε:

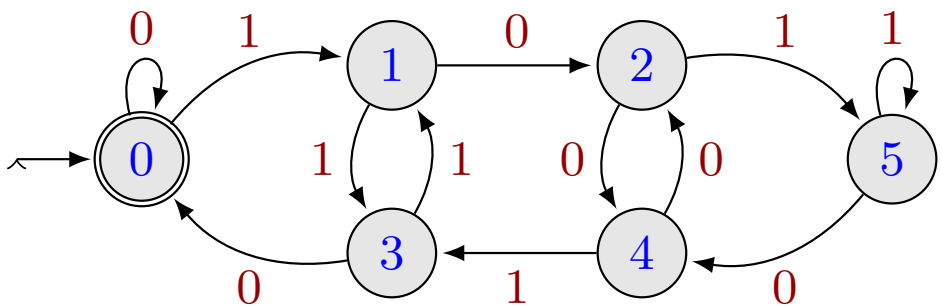
$$L_6 = \{\varepsilon, 0, 110, 0110, 1100, \ldots\}$$

Exercice: multiples of 6 in binary

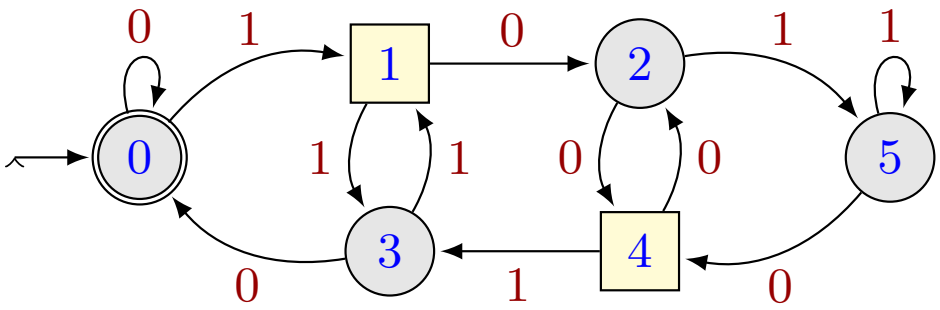
We take $\Sigma = \{0, 1\}$ and L_6 is the language of the binary representations of multiples of 6 and ε:

$$L_6 = \{\varepsilon, 0, 110, 0110, 1100, \ldots\}$$

- adding a 0 on the right = multiply by 2
- adding a 1 on the right = multiply by 2 and add 1



Equivalent states

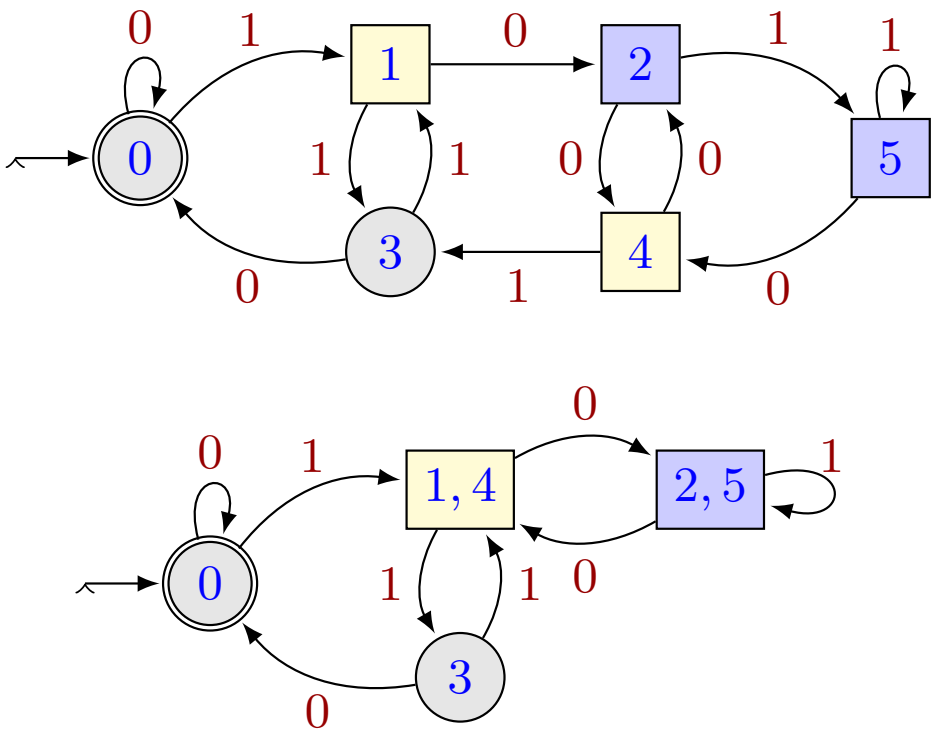


- State 1 and state 4 have the same "future"
- We can merge them without changing the recognized language
- Same for state 2 and state 5

Two states are *equivalent* when, placing the initial state on either of them, we recognize the same language

Minimal automaton

If we merge equivalent states, we obtain the *minimal automaton*



Theorem

The minimal automaton is the smallest deterministic and complete automaton that recognizes $L(\mathcal{A})$. It is unique up to the labels of the states.

State complexity

Theorem

The minimal automaton is the smallest deterministic and complete automaton that recognizes $L(\mathcal{A})$. It is unique up to the labels of the states.

There is a bijection between regular languages and their (normalized) minimal automata.

The *state complexity* of a regular language is the number of states of its minimal automaton.

Moore's state minimization algorithm

- The algorithm computes the minimal automaton of an automaton by approaching the state equivalence
- Two states p and q are i-equivalent, $p \sim_i q$, if they recognize the same words of length at most i
- $\sim_0$ is easily computed
- $\sim_{i+1}$ is computed from $\sim_i$ in linear time
- $\sim_n$ is the equivalence of states

Theorem [Moore 1956]

Moore's state minimization algorithm computes the minimal automaton of $L(\mathcal{A})$ in $\mathcal{O}(n^2)$ time

- There is a $\mathcal{O}(n \log n)$ time algorithm [Hopcroft 1971]

Average case analysis

Theorem [Bassino, David, N. 2009]

The average running time of Moore's state minimization algorithm is $\mathcal{O}(n \log n)$

- The result is very robust on the shape of the automata
- For uniform random automata (not necessarily connected) it is $\mathcal{O}(n \log \log n)$ [David 2010]
- The algorithm is used in practice (Hopcroft's algorithm is way more complicated to implement)

Average case analysis

Theorem [Bassino, David, N. 2009]

The average running time of Moore's state minimization algorithm is $\mathcal{O}(n \log n)$

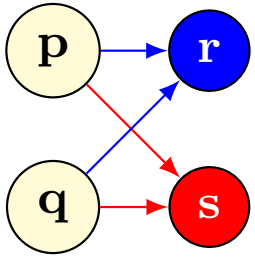
- The result is very robust on the shape of the automata
- For uniform random automata (not necessarily connected) it is $\mathcal{O}(n \log \log n)$ [David 2010]
- The algorithm is used in practice (Hopcroft's algorithm is way more complicated to implement)

$\rightarrow$ *Is it good news, or is it because the distribution on regular languages is degenerated too?*

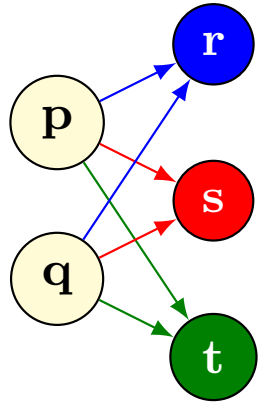
Proportion of minimal automata

Theorem (Bassino, David, Sportiello 12)

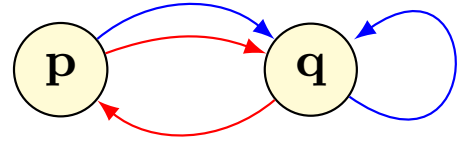
The probability that an accessible automaton taken uniformly at random is minimal tends to a positive constant if $k = 2$ and to 1 if $k \geq 3$.



M-Pattern for $k = 2$



M-Pattern for $k = 3$

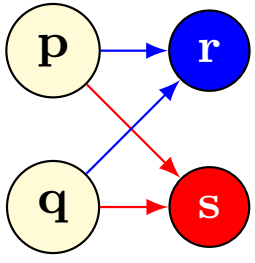


Unlikely for $k = 2$

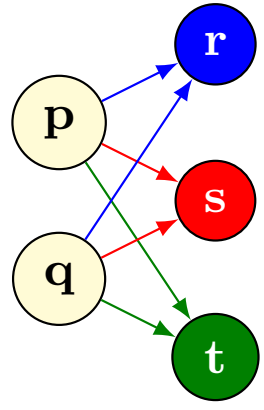
Proportion of minimal automata

Theorem (Bassino, David, Sportiello 12)

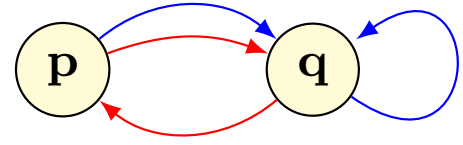
The probability that an accessible automaton taken uniformly at random is minimal tends to a positive constant if $k = 2$ and to 1 if $k \geq 3$.



M-Pattern for $k = 2$



M-Pattern for $k = 3$



Unlikely for $k = 2$

Quite wrong justification (for uniform and independent $p \xrightarrow{\alpha}$):

$$\binom{n}{4}\frac{1}{n^4} \sim \frac{1}{4!} \qquad \binom{n}{5}\frac{1}{n^6} = \mathcal{O}\left(\frac{1}{n}\right) \qquad \binom{n}{2}\frac{1}{n^4} = \mathcal{O}\left(\frac{1}{n^2}\right)$$

Proportion of minimal automata

Theorem (Bassino, David, Sportiello 2012)

The probability that an accessible automaton taken uniformly at random is minimal tends to a positive constant if $k = 2$ and to 1 if $k \geq 3$.

$\rightarrow$ The induced distribution on regular languages is *not degenerated*

Random (non-accessible) automata

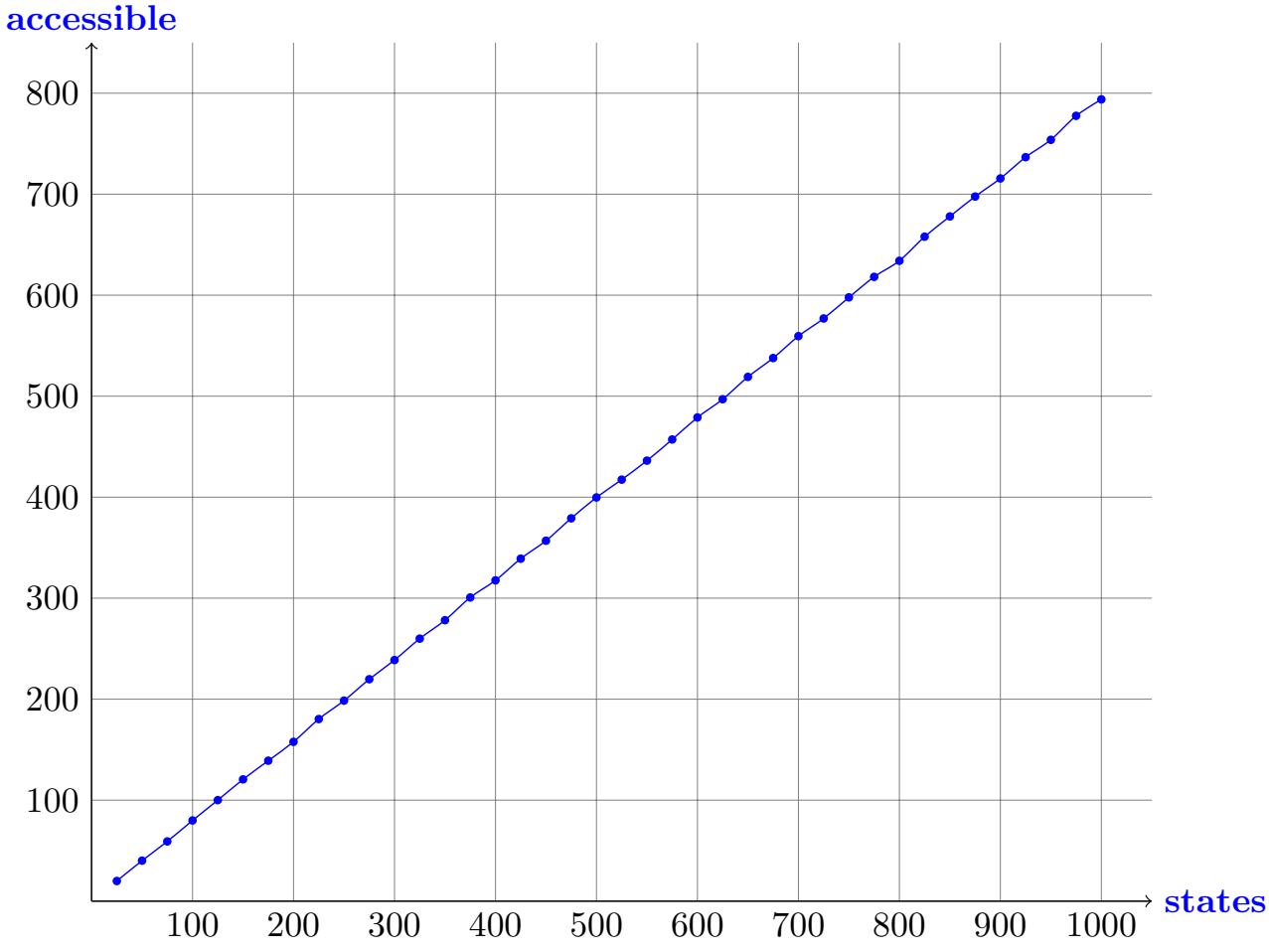
Probabilistic Settings

For given n and A, we consider the uniform distribution on all deterministic automata with n states on the alphabet A.

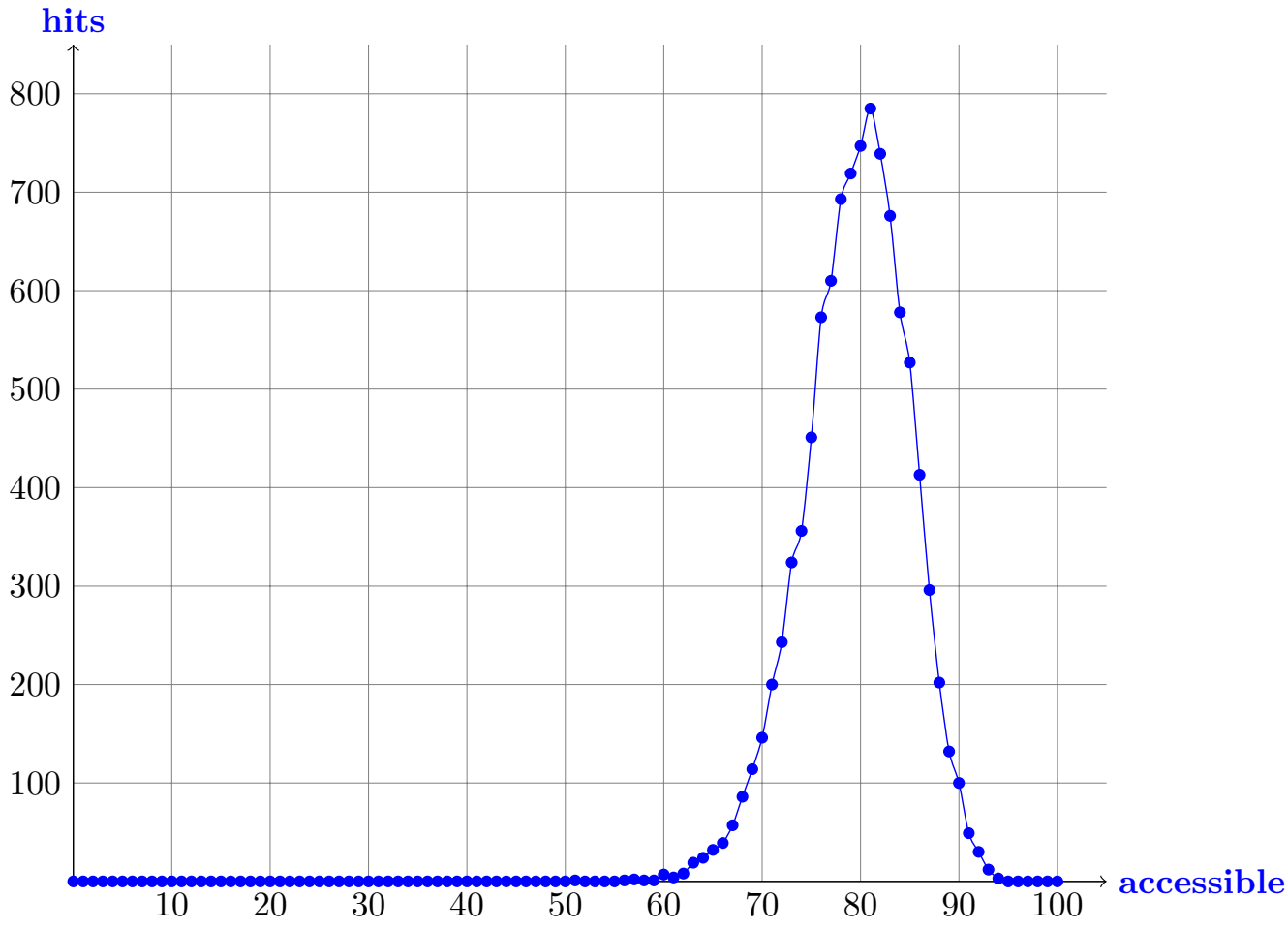
- 1 is the initial state
- There are exactly n^{kn} such automata, with $k = |A|$
- It is the same as choosing the image of every state by every letter uniformly and independently in Q

$\rightarrow$ *Are there many accessible states?*

Experiments – number of accessible states



Experiments – number of accessible states



The number of accessible states

Theorem

Let C_n be the number of accessible states in a uniform random automaton with n states. Then $\mathbb{E}[C_n] \sim \omega_k\, n$, where ω_k is the unique positive root of the equation $x = 1 - e^{-kx}$.

Moreover, C_n is asymptotically Gaussian, with standard deviation equivalent to $\sigma_k \sqrt{n}$.

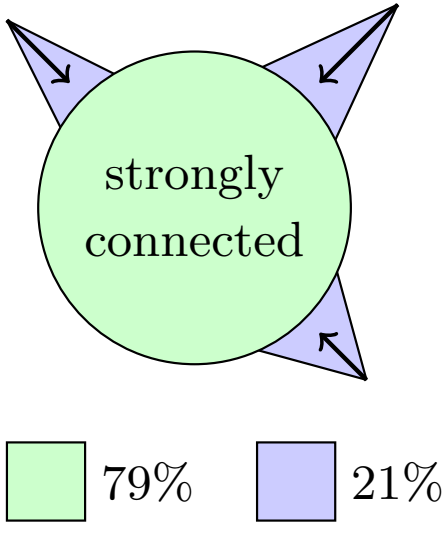
- Probabilistic proof [Grusho 1973]
- Combinatorial proof, with a local limit [Carayol & N. 2012]
- Large deviations [Berend & Kontorovich 2016]
- Refined probabilistic study [Cai, Devroye 2017]

Random automata vs random digraphs

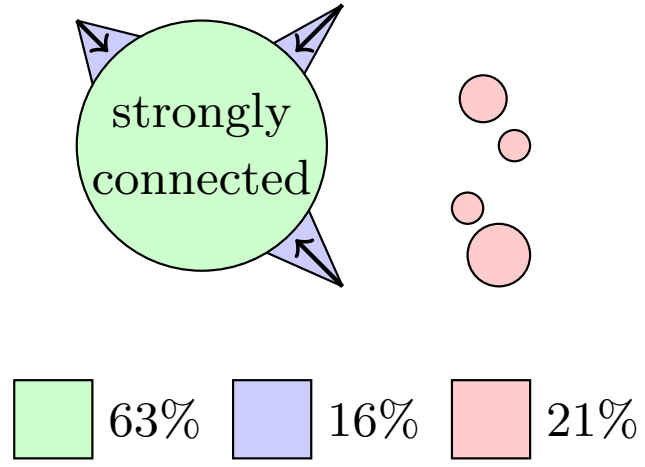
For $A = \{a, b\}$.

- **Random automata:** each state has 2 outgoing transitions
- **Random digraph (Erdős-Rényi):** each edge has probability $\frac{2}{n}$
- Let ω be the unique positive real solution of $1 - x = e^{-2x}$ ($\omega \approx 0.79$)

Random automaton



Random digraph [Karp 1990]

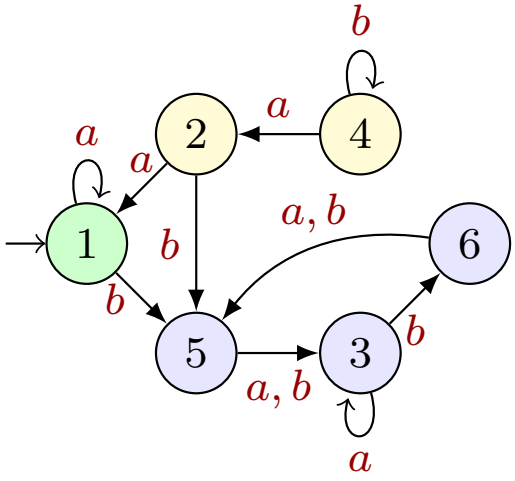


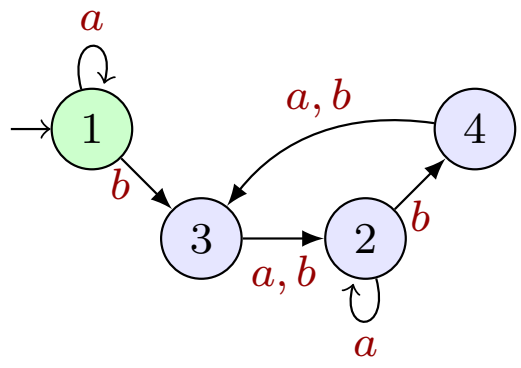
Another random generation algorithm

Question

Is there an efficient algorithm to generate random **accessible** automata uniformly at random from this result?

- **Idea:** extract the accessible part from a random automaton
- Two accessible automata of the same size are generated with the same probability



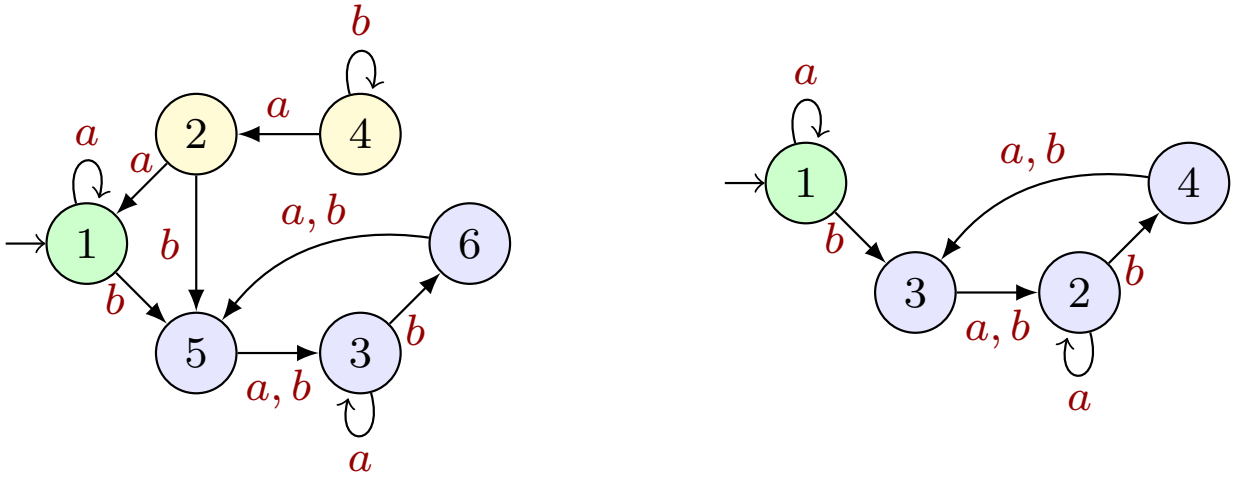


Another random generation algorithm

Question

Is there an efficient algorithm to generate random **accessible** automata uniformly at random from this result?

- **Idea:** extract the accessible part from a random automaton
- Two accessible automata of the same size are generated with the same probability



Each accessible automaton with 4 states is obtained from exactly $\binom{5}{3}6^{2\times 2}$ automata, as 1 is the initial state

Random generation of accessible automata

Theorem

The expected number of accessible states in a uniform random automaton with n states is asymptotically $\sim \omega_k\, n$, with a standard deviation $\sim \sigma_k \sqrt{n}$.

- Compute ω_k
- Repeat
  - $\mathcal{A} = \text{accessible}(\text{random automaton}(n/\omega_k))$
- Until $|\mathcal{A}| = n$
- Return $\mathcal{A}$

Average running time: $\mathcal{O}(n^{3/2})$

Random generation of accessible automata

Theorem

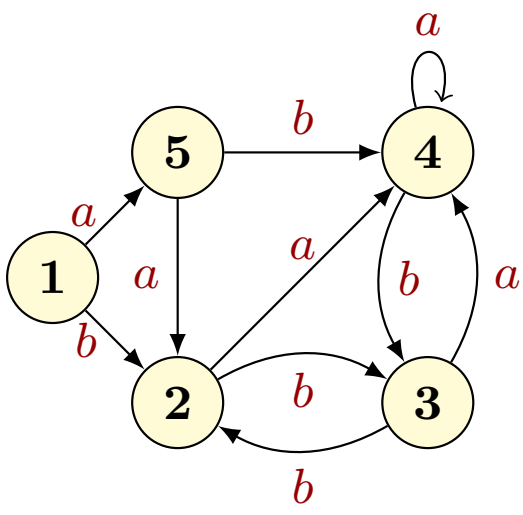
The expected number of accessible states in a uniform random automaton with n states is asymptotically $\sim \omega_k\, n$, with a standard deviation $\sim \sigma_k \sqrt{n}$.

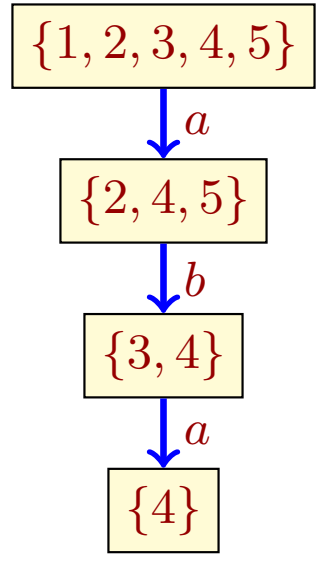
- Compute α_k
- Repeat
  - $\mathcal{A} = \text{accessible}(\text{random automaton}(n/\alpha_k))$
- Until $|\mathcal{A}| = n \pm 1\%$
- Return $\mathcal{A}$

Average running time: $\mathcal{O}(n)$

Synchronizing automata

- An automaton is **synchronizing** when there exists a word that brings every state to one and the same state
- Such a word is a **synchronizing word**

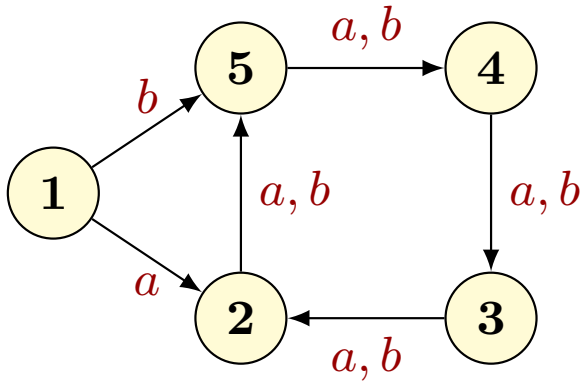




- *aaaa* is a synchronizing word
- *aba* is a smaller synchronizing word

Synchronizing automata

- An automaton is **synchronizing** when there exists a word that brings every state to one and the same state
- Such a word is a **synchronizing word**



- This automaton is not synchronizing.

The Černý conjecture

Conjecture [Černý 1964]

A synchronizing automaton with n states admits a synchronizing word of length at most $(n-1)^2$.

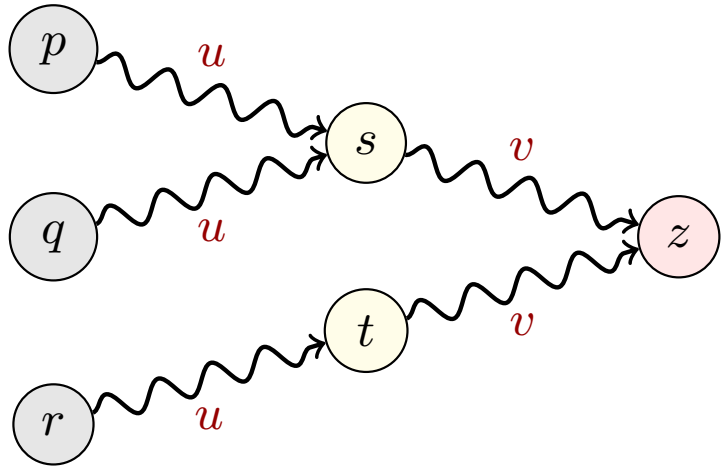
- $(n-1)^2$ is best possible [Černý 1964]
- n^3 is trivial
- better bound of $\frac{1}{6}(n^3 - n)$ [Frankl 1983] [Pin 1983]
- $\approx 0.1664\, n^3$ [Szykuła 2017], $\approx 0.1654\, n^3$ [Shitov 2019]
- the conjecture holds for many families of automata

Pairwise synchronized = synchronizing

Two states p and q are *synchronized* if there exists a word u such that $\delta_u(p) = \delta_u(q)$

Lemma

If every pair of states is synchronized by a word of length at most ℓ then $\mathcal{A}$ admits a synchronizing word of length at most $(n-1)\ell$



$u \cdot v$ synchronizes all three states

Synchronization of random automata

Question

Is a random automaton synchronizing with high probability?

Synchronization of random automata

Question

Is a random automaton synchronizing with high probability?

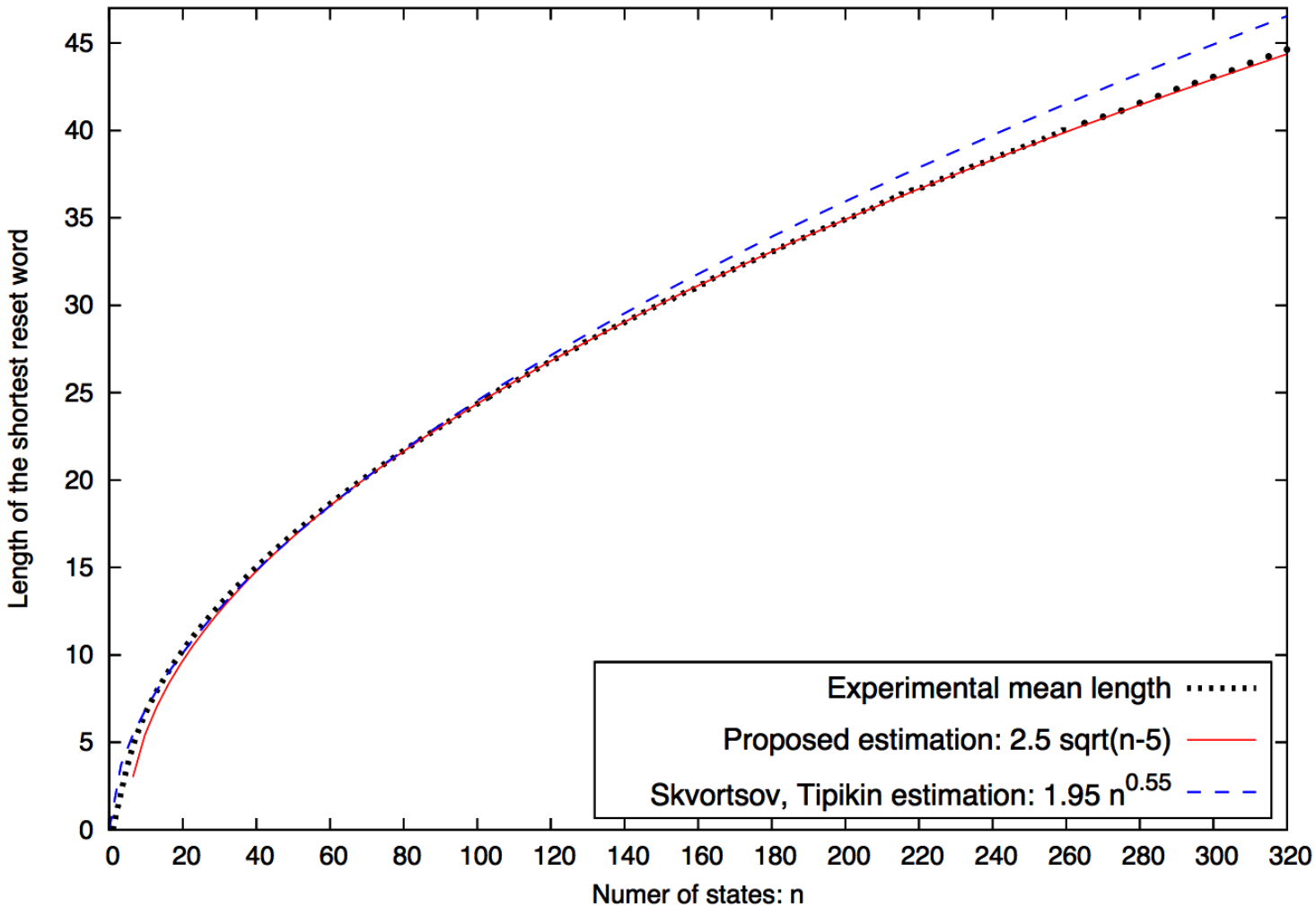
Theorem (Berlinkov 2016)

For alphabets with at least two letters, deterministic automata are synchronizing with high probability.

More precisely, a random automaton is not synchronizing with probability $\mathcal{O}(\frac{1}{n^{k/2}})$.

Experiments (Kisielewicz, Kowalski and Szykula 13)

:



The graphic comes from (Kisielewicz, Kowalski and Szykula 13)

Fast synchronization of random automata

Question

What is the length of the shortest synchronizing word of a random synchronizing automaton?

Fast synchronization of random automata

Question

What is the length of the shortest synchronizing word of a random synchronizing automaton?

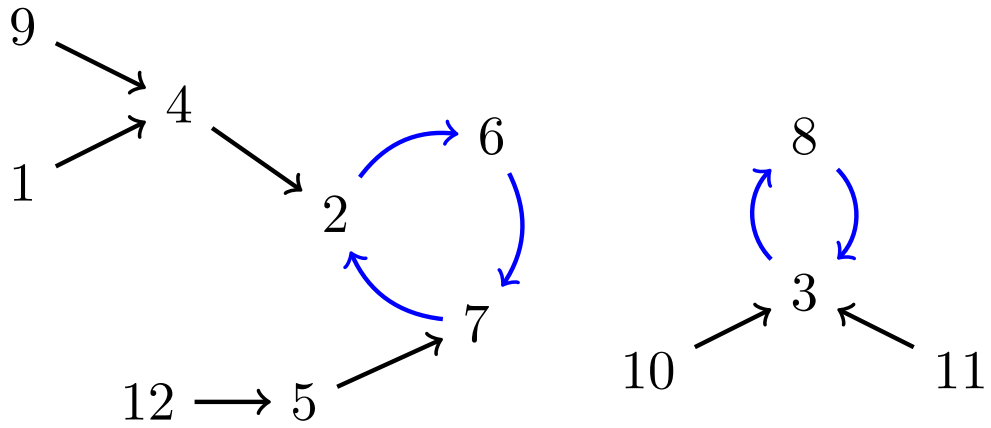
Theorem (N. 2016)

With at least two letters, with high probability a random automaton is synchronized by a word of length $\mathcal{O}(n \log^3 n)$.

$\rightarrow$ The Černý conjecture holds with high probability

Proof idea 1/2

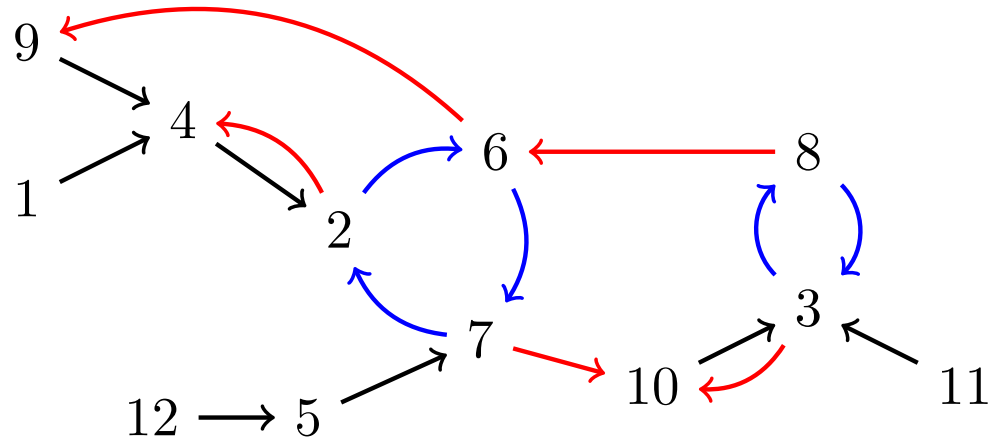
If we only consider the action of letter a it is a random mapping



- The cyclic part has size $\approx \sqrt{n}$
- The height is $\approx \sqrt{n}$
- Hence $u = a^{\sqrt{n}}$ maps the n states to a set of size $\approx \sqrt{n}$

Proof idea 2/2

If we only consider the action of letter a it is a random mapping



- $u = a^{\sqrt{n}}$ maps the n states to a set of size $\approx \sqrt{n}$
- From the a-cyclic part $\mathcal{C}_a$ generate the b-transitions
- $ba^{\sqrt{n}}$ is a (non-uniform) random mapping on $\mathcal{C}_a$
- hence $v = a^{\sqrt{n}}(ba^{\sqrt{n}})^{n^{1/4}}$ maps the n states to a set of size $\approx n^{1/4}$

. . . continue until the image has size $\approx n^{1/8}$ then pairwise-synchronize the states with high probability

A better result

Theorem (Chapuy, Perarnau 2023)

With high probability a random automaton is synchronized by a word of length $\mathcal{O}(\sqrt{n}\log n)$.

- A random mapping is synchronizing iff it is a rooted tree
- It happens with probability $\frac{1}{n}$, by Cayley formula
- The action of the words of length $(1+\epsilon)\log_2 n$ behave as independent uniform random mappings
- There are sufficiently many of them to get the result (second moment method)

$\rightarrow$ *the technical details are complicated*

A simpler proof

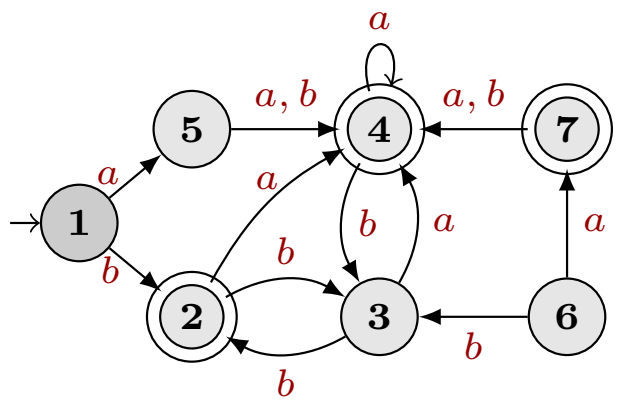
Theorem (Martinsson 2023)

With high probability at least $1 - \epsilon$ a random automaton is synchronized by a word of length $\mathcal{O}(\epsilon^{-1}\sqrt{n}\log n)$.

- $v = a^{\sqrt{n}}(ba^{\sqrt{n}})^{\log n}$ maps the n states to a set of size $\approx \sqrt{n}/\log n$
- States are pairwise synchronized by words of length $\mathcal{O}(\log n)$ with high probability

Open question: probabilistic lower bound?

That's all



Thank you!