Analysis of Algorithms and Formal Language Theory

Cyril Nicaud

LIGM, Paris-Est, Marne-la-Vallée

AofA'13

I. Introduction

– Languages –

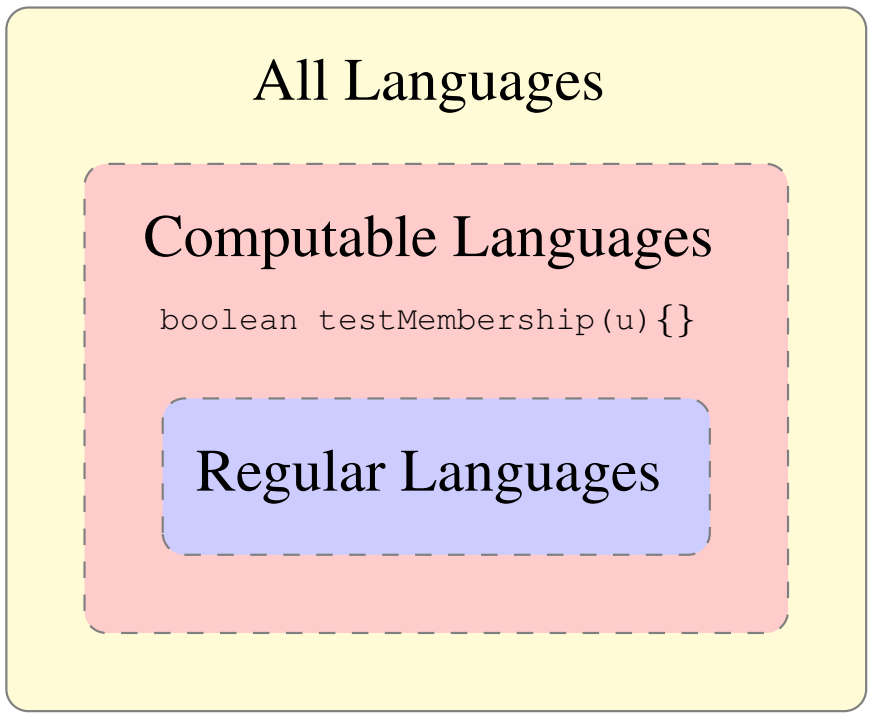
- Words are finite sequences of letters taken in a fixed (small) alphabet A
- Languages are sets of words

$$\mathcal{L} = \{u : u \text{ has more 0's than 1's}\} = \{0, 00, 001, 010, 100, \ldots\}$$

– Languages –

- Words are finite sequences of letters taken in a fixed (small) alphabet A
- Languages are sets of words

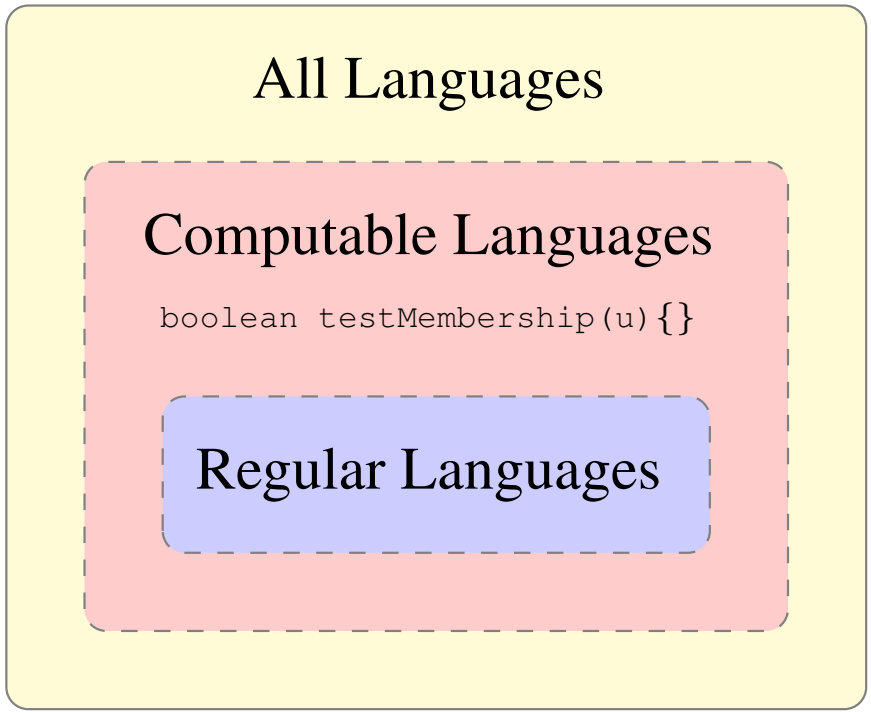
$$\mathcal{L} = \{u : u \text{ has more 0's than 1's}\} = \{0, 00, 001, 010, 100, \ldots\}$$



– Languages –

- Words are finite sequences of letters taken in a fixed (small) alphabet A
- Languages are sets of words

$$\mathcal{L} = \{u : u \text{ has more 0's than 1's}\} = \{0, 00, 001, 010, 100, \ldots\}$$



– Regular Languages –

- Computational properties
- Mathematical properties

– Regular Languages –

- Concatenation of two languages

$$X \cdot Y = \{uv : u \in X, v \in Y\}$$

- Kleene star of a language

$$X^* = \{\varepsilon\} \cup X \cup X \cdot X \cup X \cdot X \cdot X \cup \ldots$$

– Regular Languages –

- Concatenation of two languages

$$X \cdot Y = \{uv : u \in X, v \in Y\}$$

- Kleene star of a language

$$X^* = \{\varepsilon\} \cup X \cup X \cdot X \cup X \cdot X \cdot X \cup \dots$$

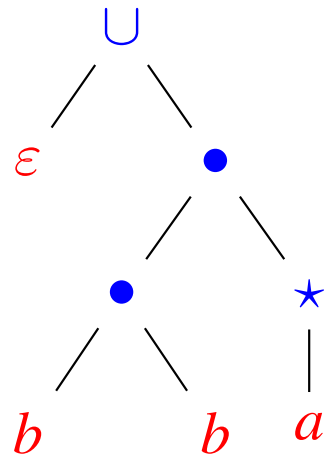
Regular Languages

The set $\mathcal{R}$ of regular languages over A is inductively defined by:

- $\emptyset$, $\{\varepsilon\}$ and $\{a\}$ are in $\mathcal{R}$, for every $a \in A$
- $\mathcal{R}$ is stable for union, concatenation and Kleene star

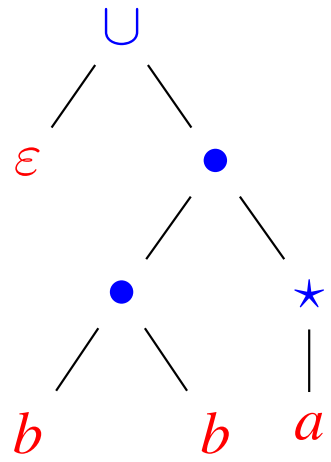
$$\begin{aligned}\mathcal{L} &= (\{a\} \cup \{b\})^* \cdot \{b\} \cdot \{a\} \cdot \{b\} \cdot (\{a\} \cup \{b\})^* \\ &= \{\text{words containing the factor } bab\}\end{aligned}$$

– Regular Expressions –



The language denoted by this regular expression is $\varepsilon \cup bba^*$

– Regular Expressions –



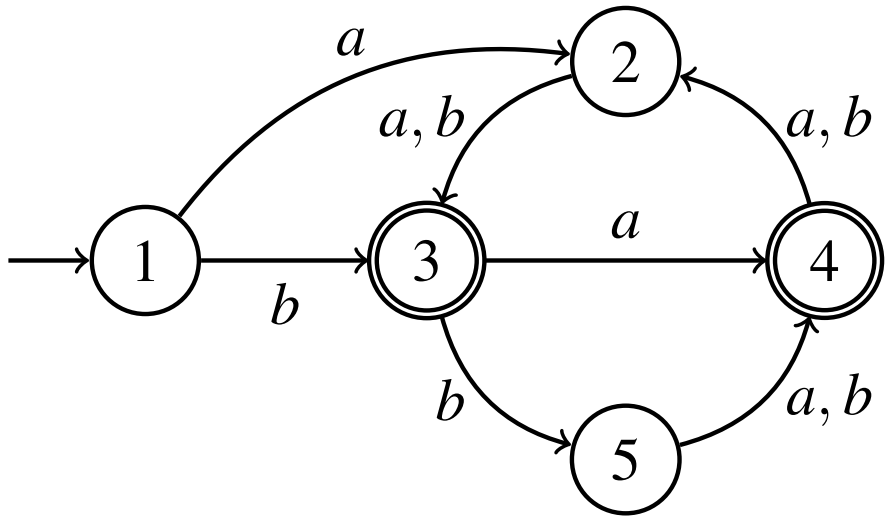
The language denoted by this regular expression is $\varepsilon \cup bba^*$

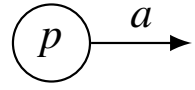
Universality for regular expressions

If $\mathcal{L}$ is given by a regular expression, testing whether $\mathcal{L} = A^*$ is PSPACE-complete.

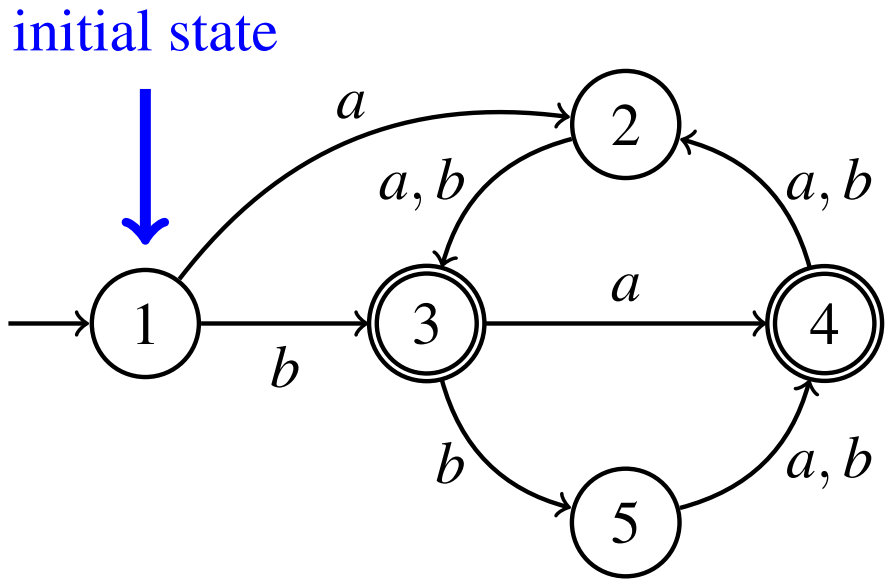
- it is a hard problem, since NP $\subset$ PSPACE

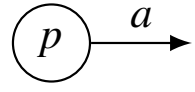
– Deterministic and complete automaton –



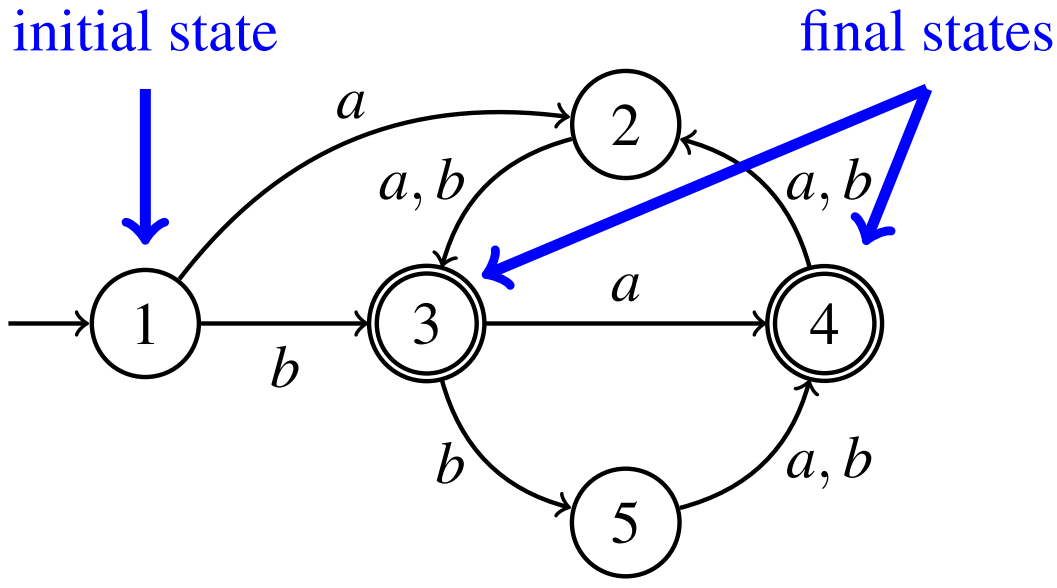
- Only one initial state
- For every state p and every letter a, exactly one 

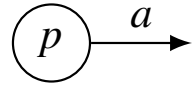
– Deterministic and complete automaton –



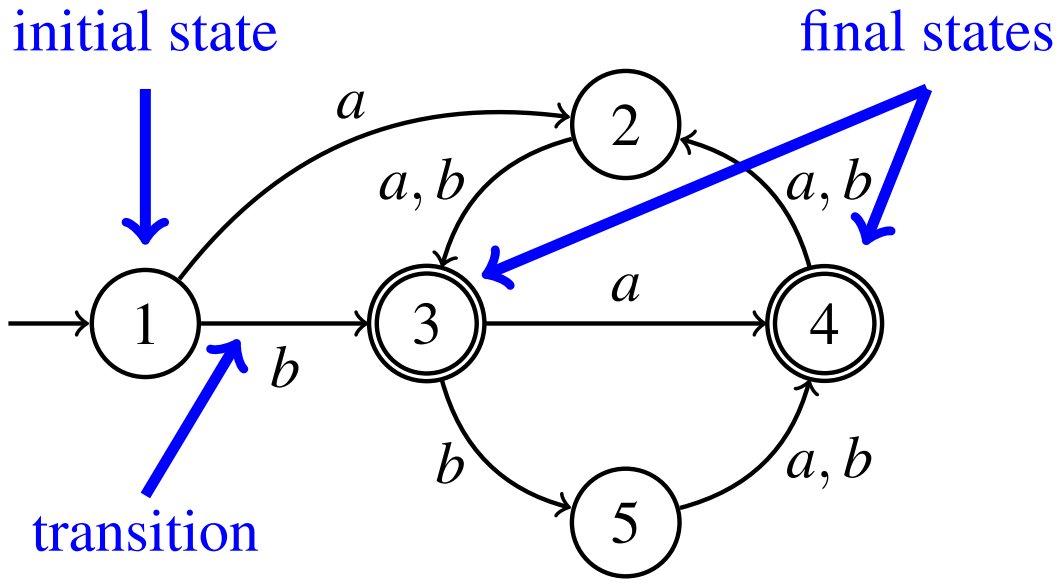
- Only one initial state
- For every state p and every letter a, exactly one 

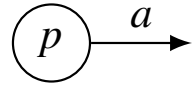
– Deterministic and complete automaton –



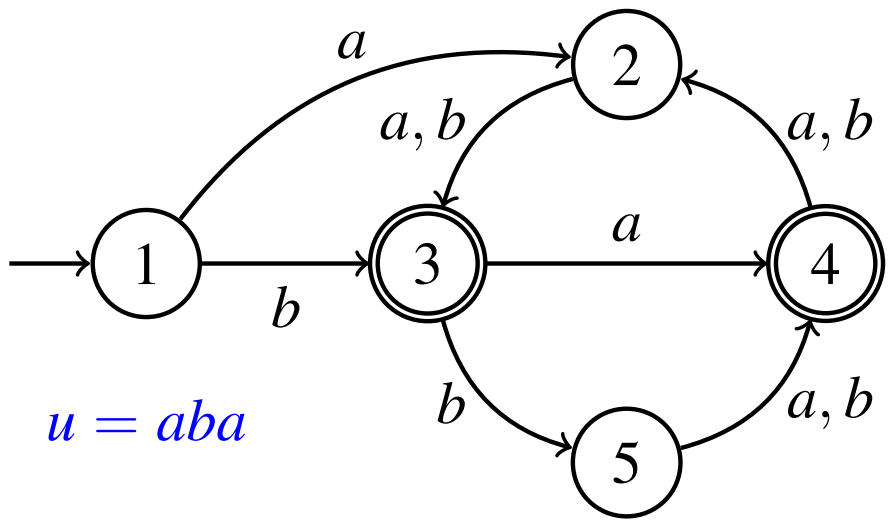
- Only one initial state
- For every state p and every letter a, exactly one 

– Deterministic and complete automaton –

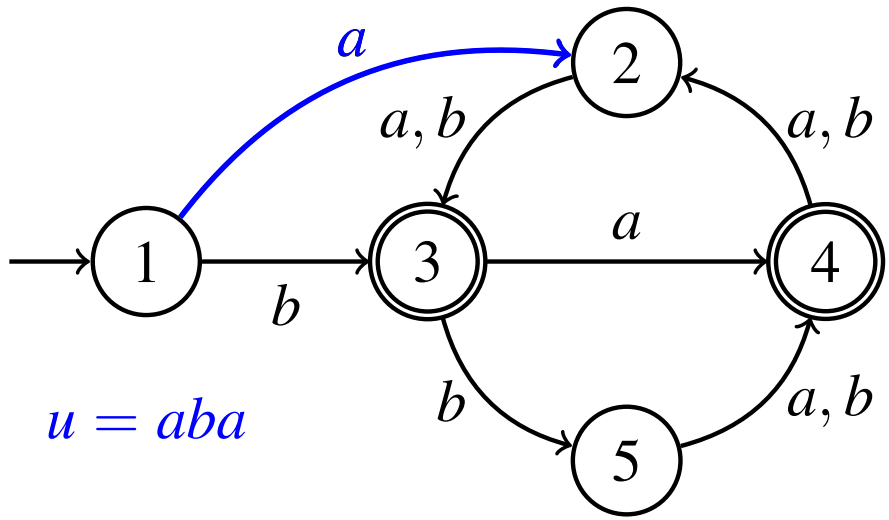


- Only one initial state
- For every state p and every letter a, exactly one 

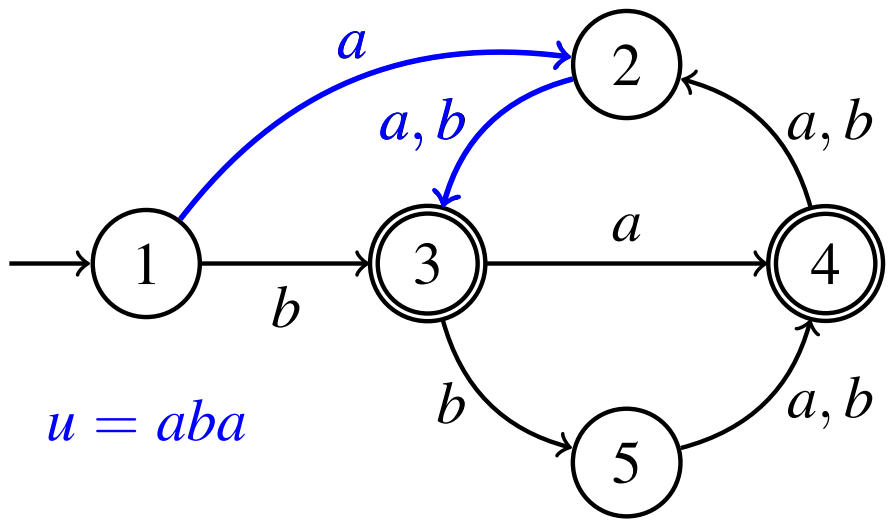
– Deterministic and complete automaton –



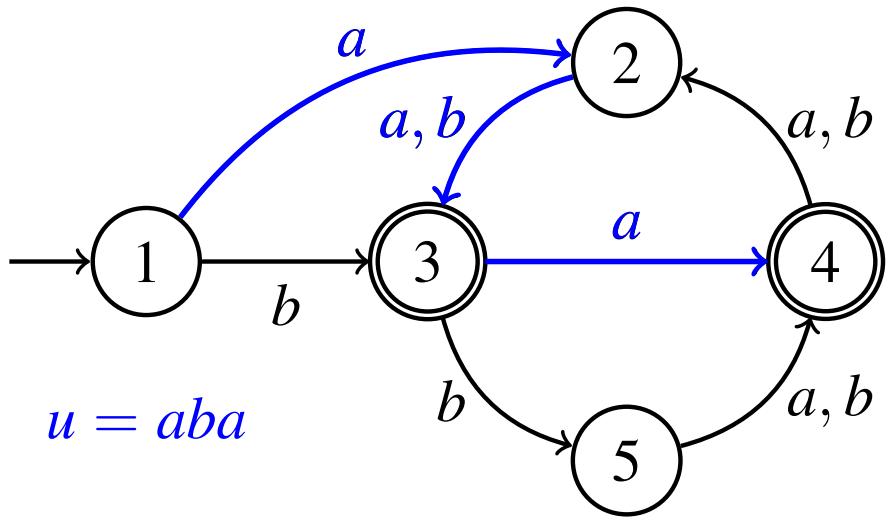
– Deterministic and complete automaton –



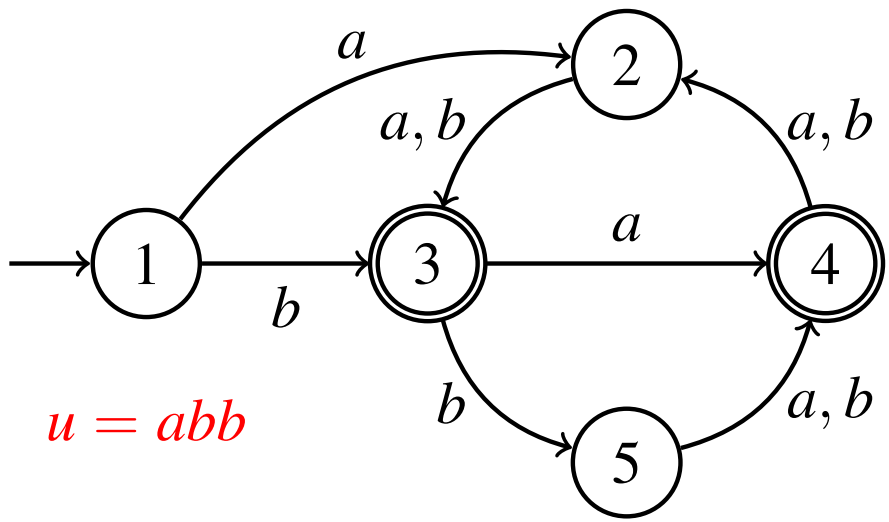
– Deterministic and complete automaton –



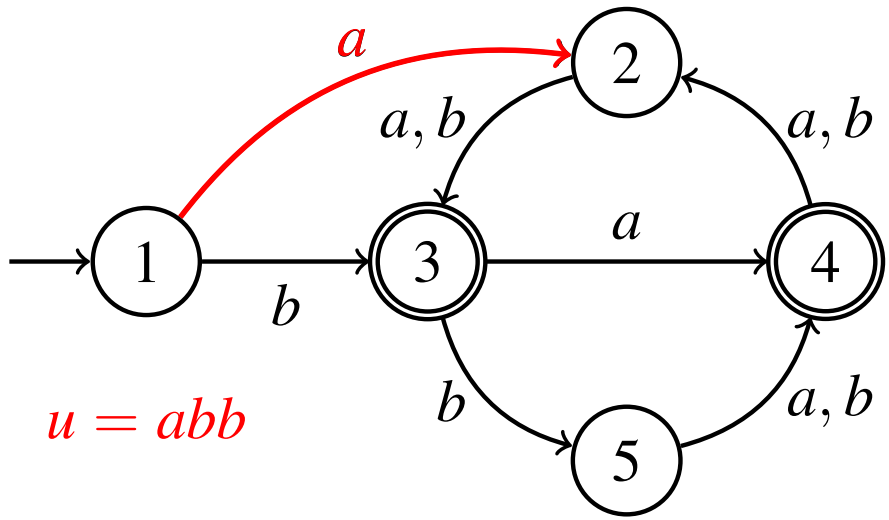
– Deterministic and complete automaton –



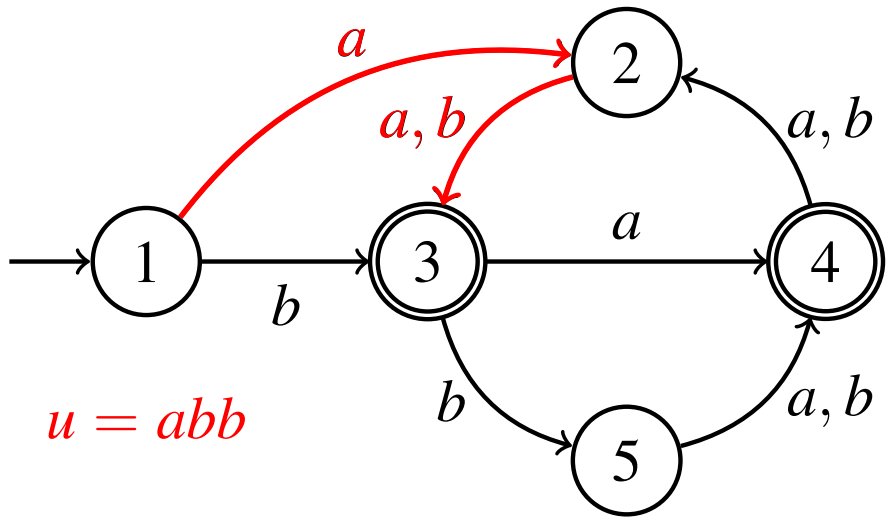
– Deterministic and complete automaton –



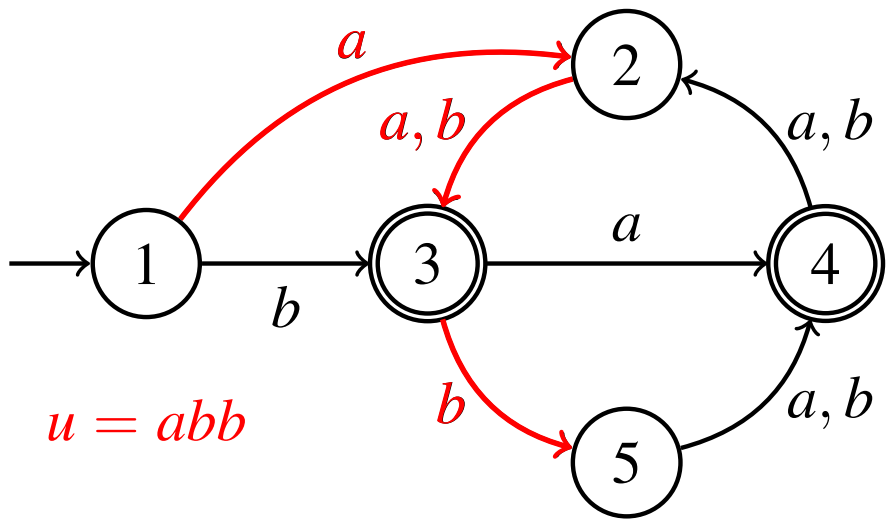
– Deterministic and complete automaton –



– Deterministic and complete automaton –



– Deterministic and complete automaton –



– Kleene's theorem –

Kleene's theorem

A language is regular if and only if it is recognized by a (deterministic) automaton.

– Kleene's theorem –

Kleene's theorem

A language is regular if and only if it is recognized by a (deterministic) automaton.

Corollary

The set of regular languages is stable by complement, union, intersection, quotients, . . .

– Some Complexity Results –

If languages are given by deterministic automata:

- Emptyness ($\mathcal{L} = \emptyset$) in constant time
- Universality ($\mathcal{L} = A^*$) in linear time
- Membership ($u \in \mathcal{L}$) in linear time in $|u|$

– Some Complexity Results –

If languages are given by deterministic automata:

- Emptyness ($\mathcal{L} = \emptyset$) in constant time
- Universality ($\mathcal{L} = A^*$) in linear time
- Membership ($u \in \mathcal{L}$) in linear time in $|u|$

- Equality ($\mathcal{L} = \mathcal{L}'$) in $\mathcal{O}(n\,\alpha(n))$
- Complement in linear time

– Some Complexity Results –

If languages are given by deterministic automata:

- Emptyness ($\mathcal{L} = \emptyset$) in constant time
- Universality ($\mathcal{L} = A^*$) in linear time
- Membership ($u \in \mathcal{L}$) in linear time in $|u|$

- Equality ($\mathcal{L} = \mathcal{L}'$) in $\mathcal{O}(n\,\alpha(n))$
- Complement in linear time

- Union and intersection in quadratic time
- Star and concatenation in exponential time

– Minimal Automata –

- The size of an automaton is its number of states

Minimal automaton

For every regular language $\mathcal{L}$ there exists a unique smallest deterministic automaton that recognizes $\mathcal{L}$. It is called the minimal automaton of $\mathcal{L}$.

– Minimal Automata –

- The size of an automaton is its number of states

Minimal automaton

For every regular language $\mathcal{L}$ there exists a unique smallest deterministic automaton that recognizes $\mathcal{L}$. It is called the minimal automaton of $\mathcal{L}$.

- We define the size of a regular language as the number of states of its minimal automaton

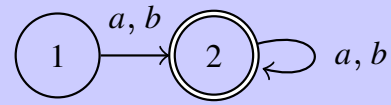
- We aim at investigating the properties of random regular languages and random automata: typical shape, asymptotics, random generation, average case analysis of algorithms, ...

– Characterizations –

Regular expressions

$a^*b \cup a \cdot (c \cup d^*)$

Deterministic automata



Logical formulas

$\forall x\,(a(x) \rightarrow (\exists y\,(y = x + 1) \wedge b(y)))$

Finite Monoids

$\phi : A^* \longrightarrow M$

$\mathcal{L} = \phi^{-1}(S),\ S \subseteq M$

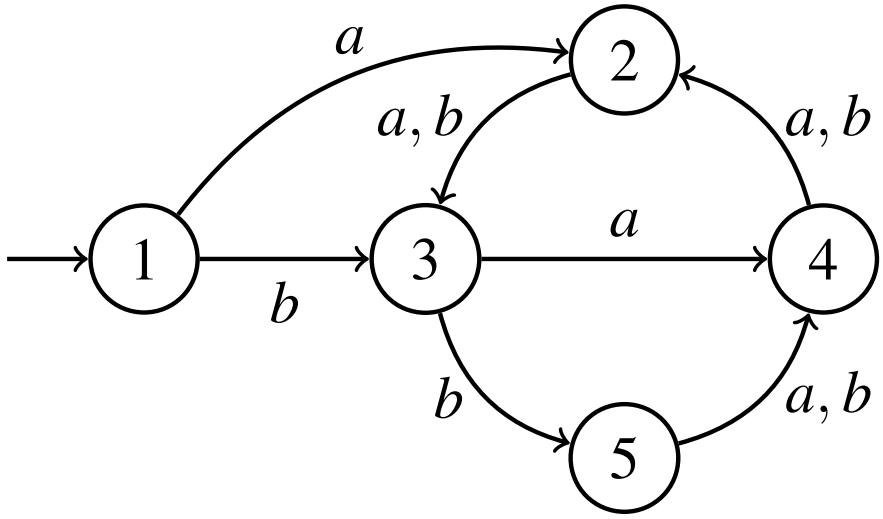
II. Combinatorics

– Counting automata –

- n is the size of the automaton, its number of states
- k is the size of the (fixed) alphabet
- 1 is the inital state, no final states for now

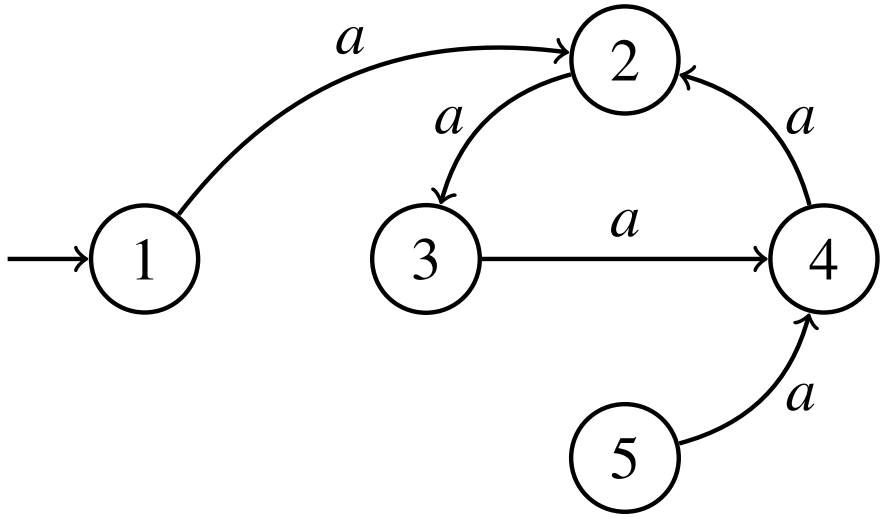
– Counting automata –

- n is the size of the automaton, its number of states
- k is the size of the (fixed) alphabet
- 1 is the inital state, no final states for now
- The action of each letter a is a total map from the set of states to itself:

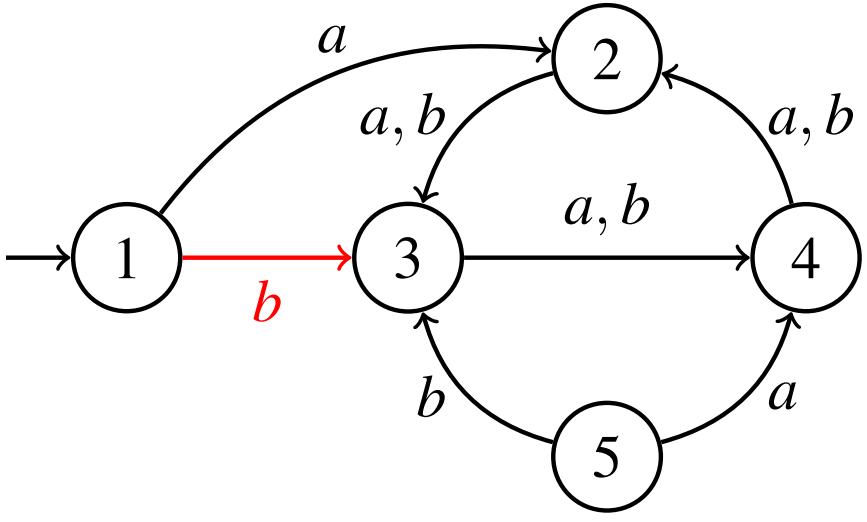


– Counting automata –

- n is the size of the automaton, its number of states
- k is the size of the (fixed) alphabet
- 1 is the inital state
- The action of each letter a is a total map from the set of states to itself:



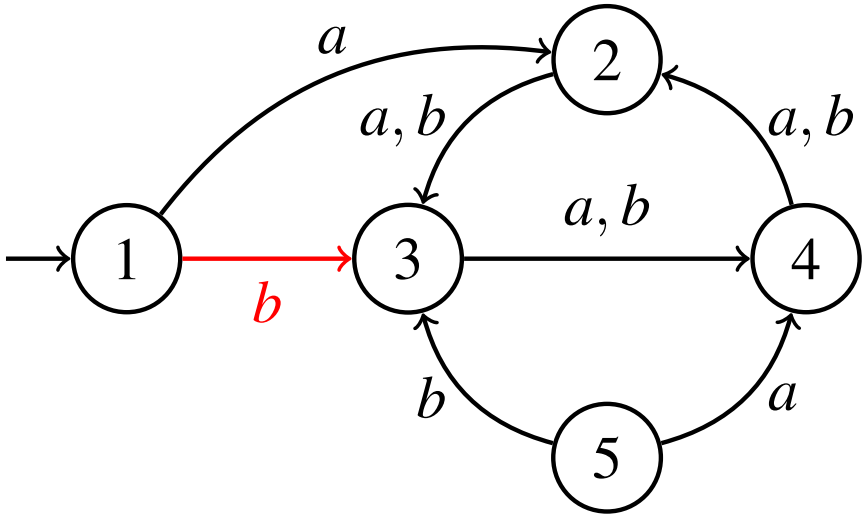
– Counting automata –



	1	2	3	4	5
a	2	3	4	2	4
b	3	3	4	2	3

- $n^{kn}\, 2^n$ automata (counting final states)

– Counting automata –



	1	2	3	4	5
a	2	3	4	2	4
b	3	3	4	2	3

- $n^{kn}\,2^n$ automata (counting final states)

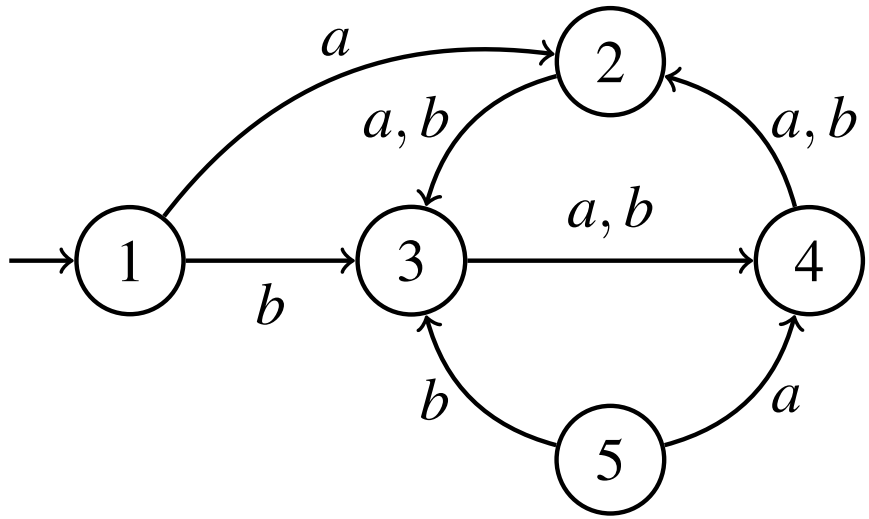
- States that are not accessible are useless: the automaton cannot be minimal

Lemma

The probability that an automaton is accessible is exponentially small.

– Accessible Automata –

- What is the asymptotic number of accessible automata?
- Can we design an efficient random generator for accessible automata?



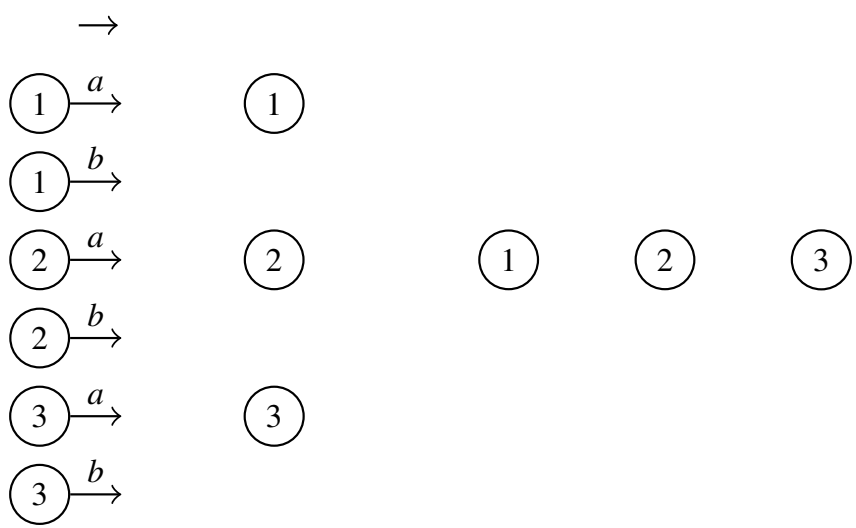
- Accessible automata are rigid: there are exactly $(n-1)!$ ways to label them

– Korshunov's idea (1978) –

- Consider automata where each state, except possibly the initial one, has an incoming transition

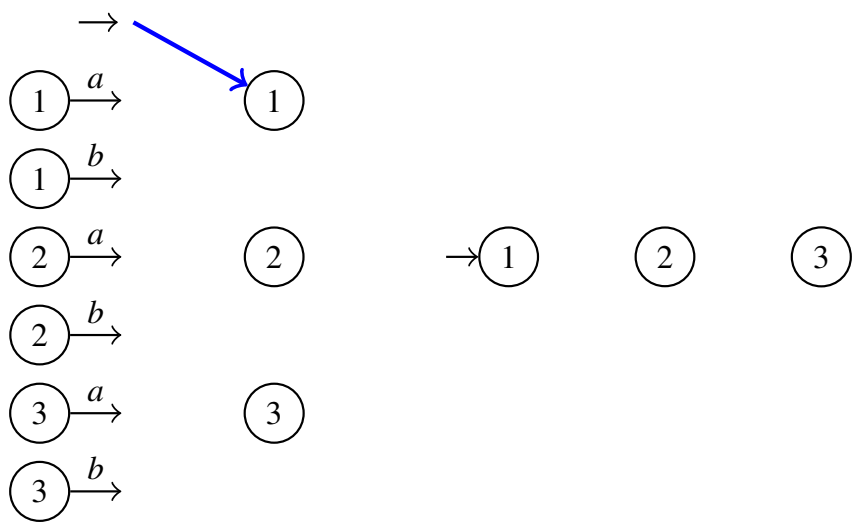
– Korshunov's idea (1978) –

- Consider automata where each state, except possibly the initial one, has an incoming transition
- What we want is a surjection from the set of "arrows" onto the set of states, s.t. $\rightarrow$ is mapped to 1



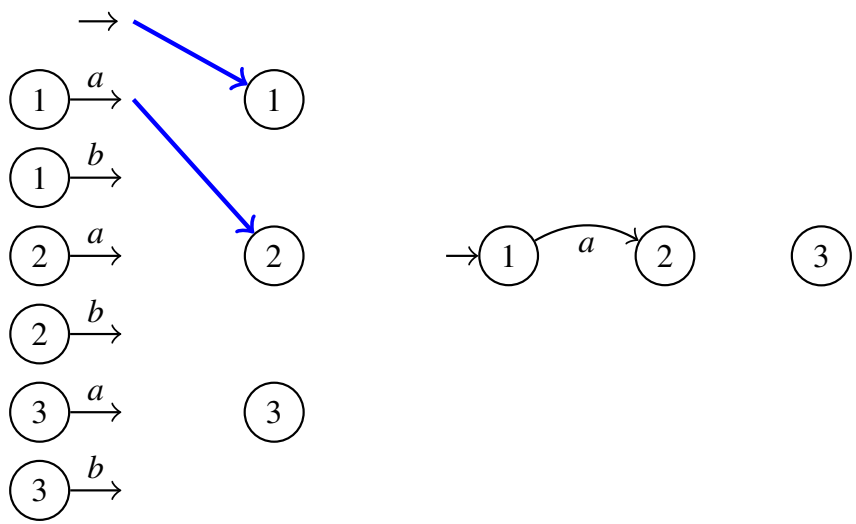
– Korshunov's idea (1978) –

- Consider automata where each state, except possibly the initial one, has an incoming transition
- What we want is a surjection from the set of "arrows" onto the set of states, s.t. $\rightarrow$ is mapped to 1



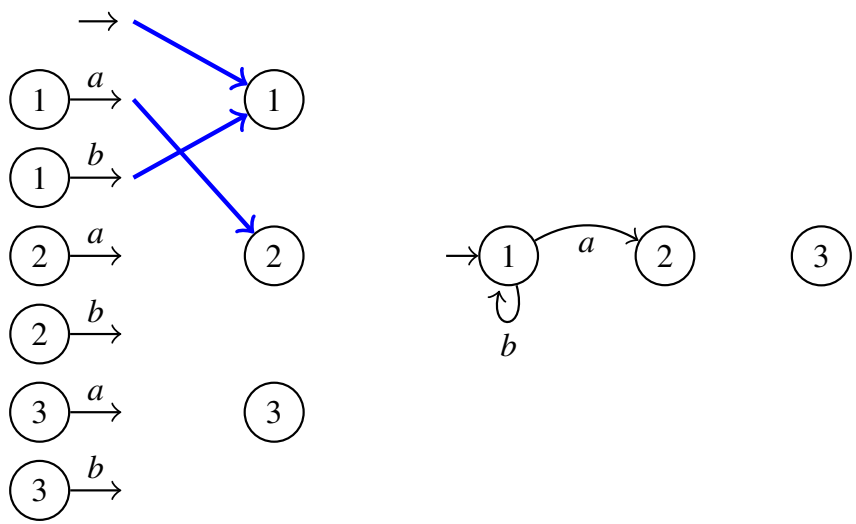
– Korshunov's idea (1978) –

- Consider automata where each state, except possibly the initial one, has an incoming transition
- What we want is a surjection from the set of "arrows" onto the set of states, s.t. $\rightarrow$ is mapped to 1



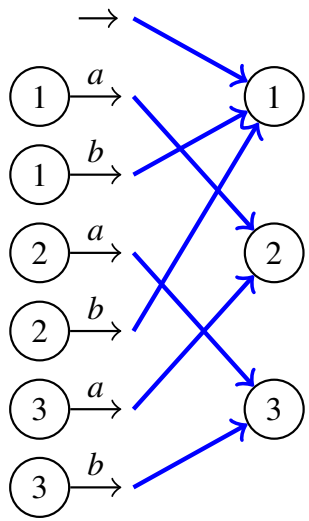
– Korshunov's idea (1978) –

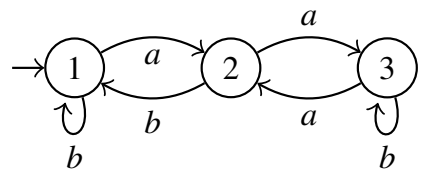
- Consider automata where each state, except possibly the initial one, has an incoming transition
- What we want is a surjection from the set of "arrows" onto the set of states, s.t. $\rightarrow$ is mapped to 1



– Korshunov's idea (1978) –

- Consider automata where each state, except possibly the initial one, has an incoming transition
- What we want is a surjection from the set of "arrows" onto the set of states, s.t. $\rightarrow$ is mapped to 1





– Korshunov's formula (1978) –

- Let $\mathcal{A}_n$ denote the set accessible automata with n states
- Let $S(x,y)$ denote the number of surjections from $[x]$ onto $[y]$

Theorem [Korshunov 78]

Asymptotically, a constant proportion of Korshunov's automata are accessible:

$$|\mathcal{A}_n| \sim E \cdot S(k\,n, n), \text{ with } E = \frac{1 + \sum_{r=1}^{\infty} \frac{1}{r}\binom{kr}{r-1}(e^{k-1}\lambda)^{-r}}{1 + \sum_{r=1}^{\infty} \binom{kr}{r}(e^{k-1}\lambda)^{-r}},$$

where λ is a computable constant.

– Korshunov's formula (1978) –

Theorem [Good 61]

For fixed k, we have

$$S(k\,n, n) \sim \alpha \cdot \beta^n \cdot n^{kn},$$

for some computable constants α and β, with $0 < \beta < 1$.

- Korshunov + Good yield

$$|\mathcal{A}_n| \sim E \cdot \alpha \cdot \beta^n n^{kn}$$

– Korshunov's formula (1978) –

Theorem [Good 61]

For fixed k, we have

$$S(k\,n, n) \sim \alpha \cdot \beta^n \cdot n^{kn},$$

for some computable constants α and β, with $0 < \beta < 1$.

- Korshunov + Good yield

$$|\mathcal{A}_n| \sim E \cdot \alpha \cdot \beta^n n^{kn}$$

- The proportion of accessible automata is exponentially small

– Random generation: a first algorithm –

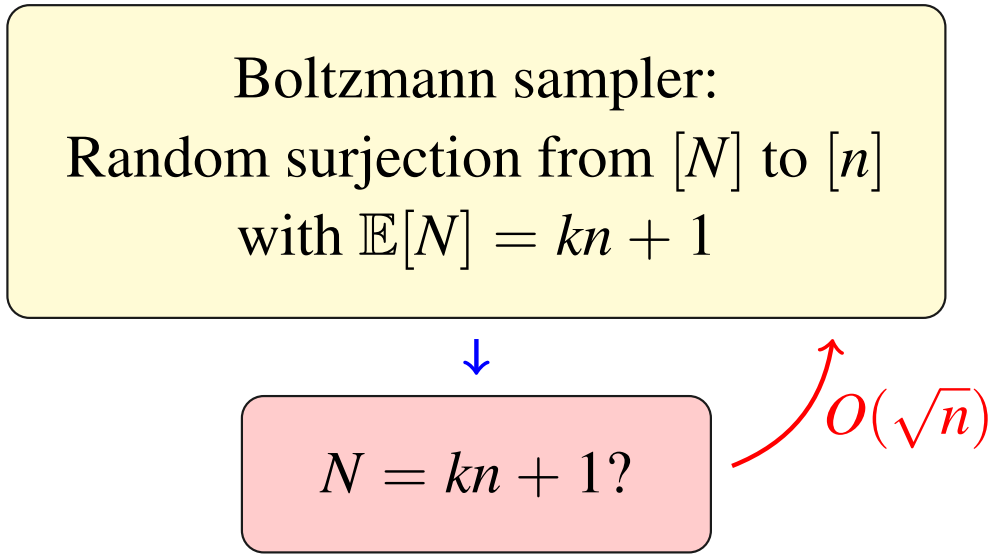
Boltzmann sampler:
Random surjection from $[N]$ to $[n]$
with $\mathbb{E}[N] = kn + 1$

– Random generation: a first algorithm –

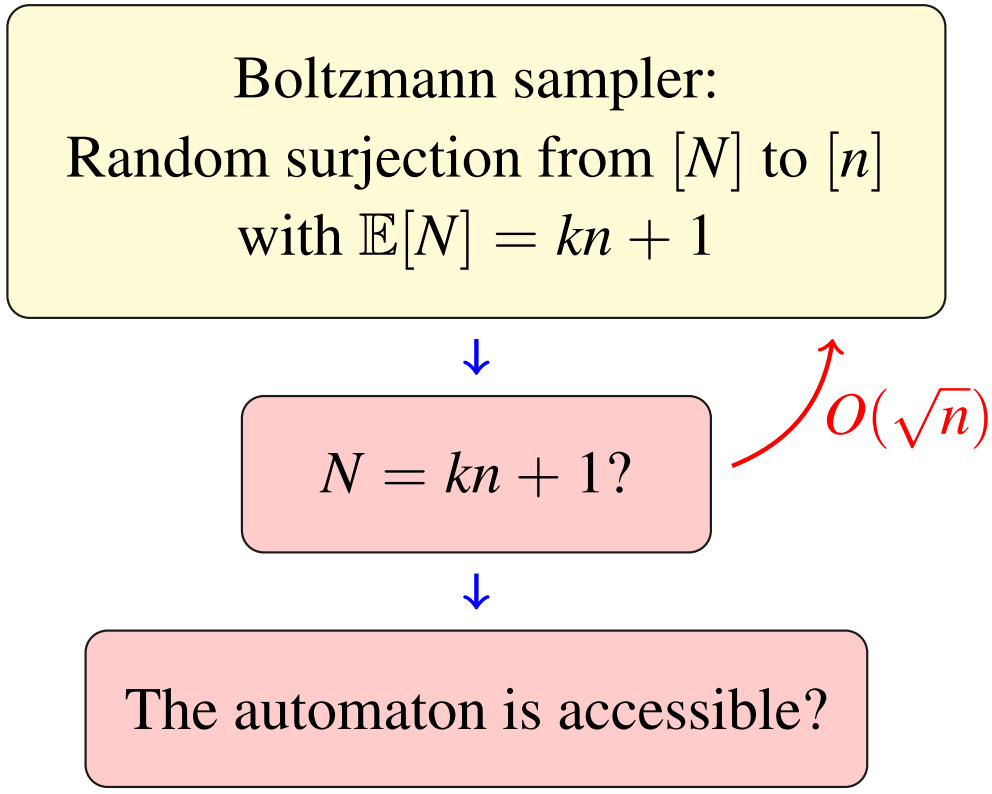
Boltzmann sampler:
Random surjection from $[N]$ to $[n]$
with $\mathbb{E}[N] = kn + 1$

$N = kn + 1$?

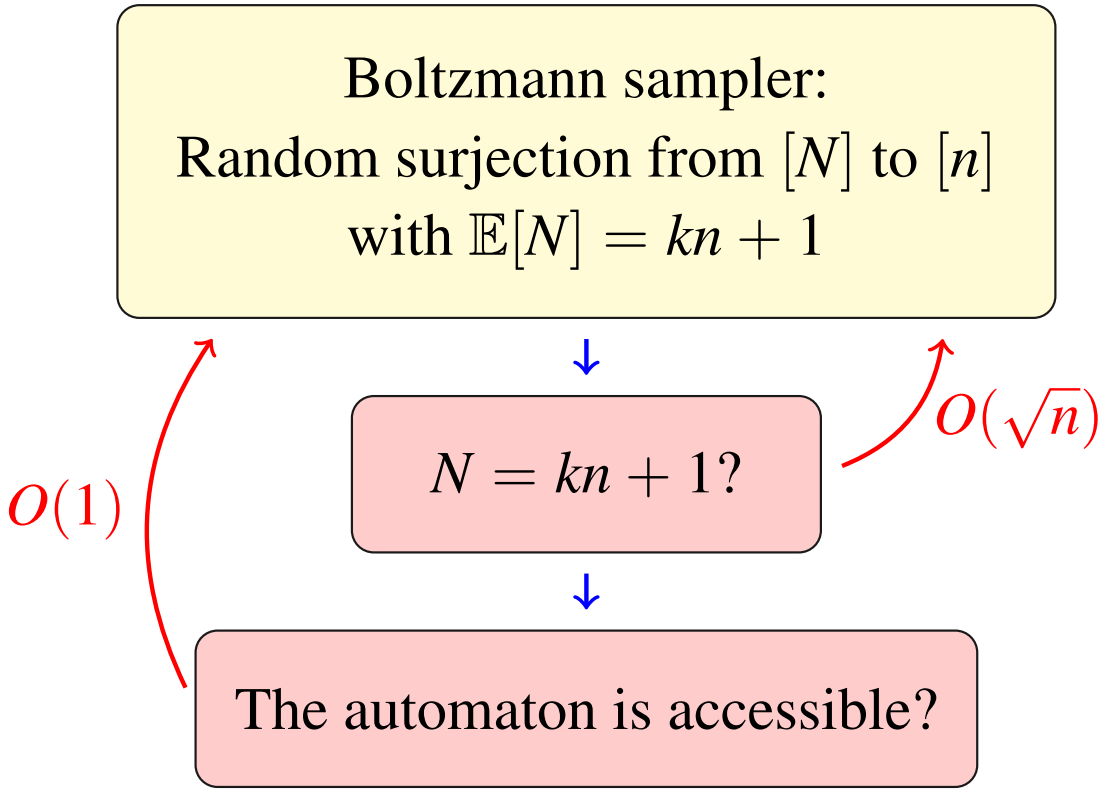
– Random generation: a first algorithm –



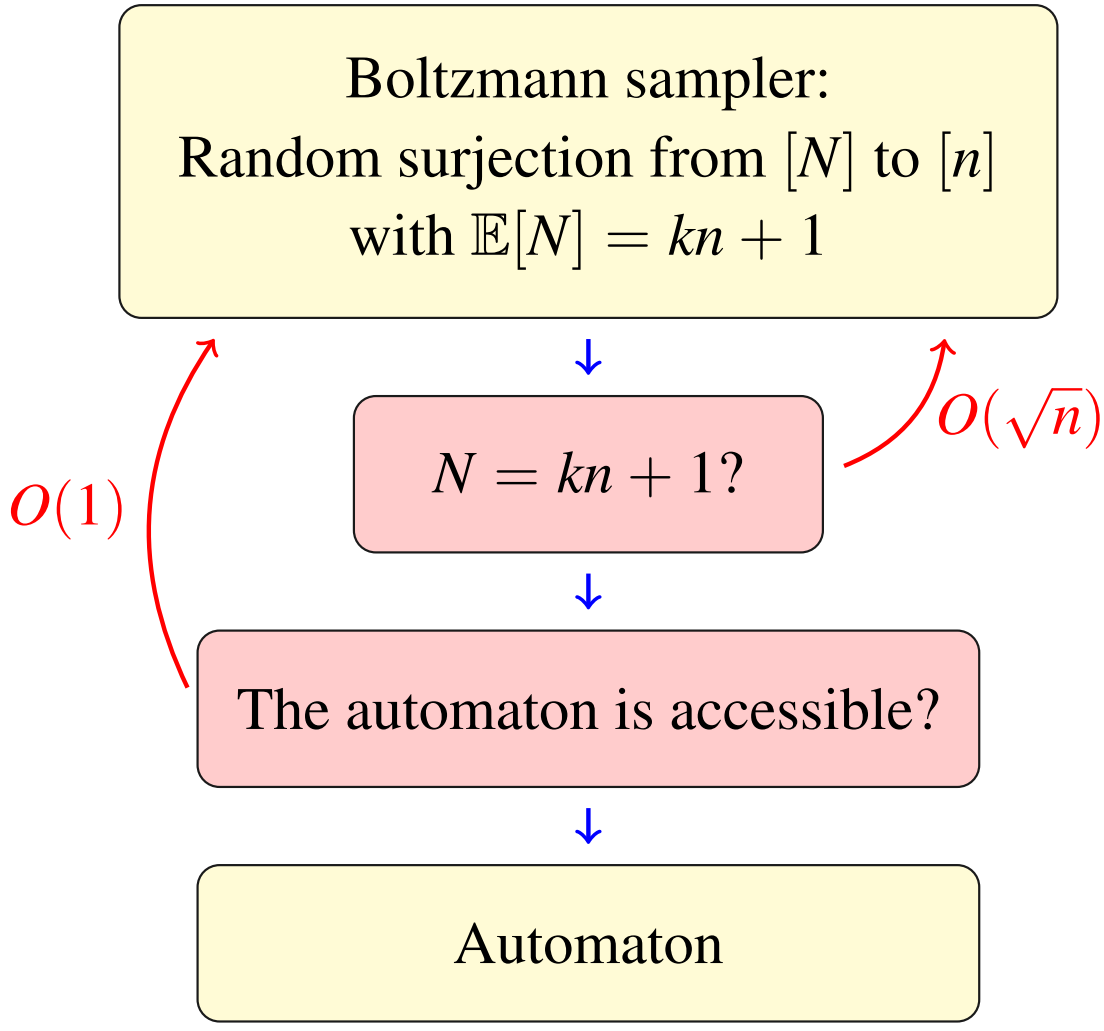
– Random generation: a first algorithm –



– Random generation: a first algorithm –



– Random generation: a first algorithm –



– Random generation: a first algorithm –

Boltzmann sampler [Bassino, N. 07]

Using a Boltzmann sampler, one can generate random accessible automata in average time $\Theta(n^{3/2})$.

- Variations: David, Héam, Schmitz

– Random generation: a first algorithm –

Boltzmann sampler [Bassino, N. 07]

Using a Boltzmann sampler, one can generate random accessible automata in average time $\Theta(n^{3/2})$.

- Variations: David, Héam, Schmitz

Recursive generator [N. 00; Champarnaud, Paranthoën 05]

Using the same kind of bijection and the recursive method, one can generate random accessible automata in linear time, at the cost of a $\Theta(n^2)$ preprocessing.

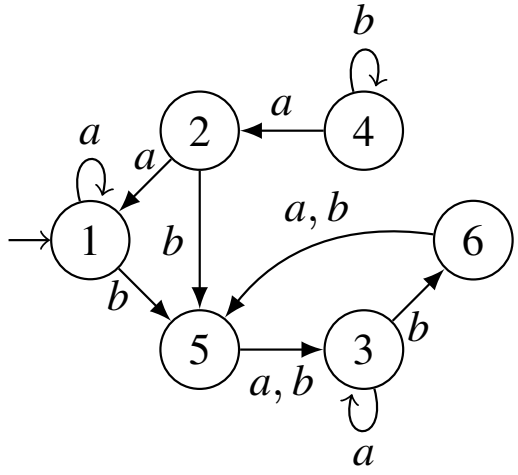
- The algorithm above uses large numbers, it is not really linear

III. Accessible part

– Another approach –

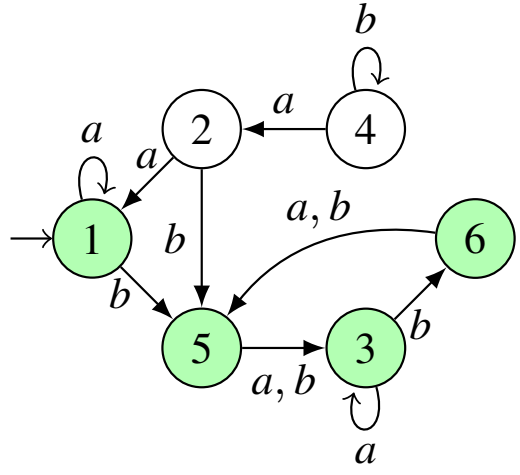
- We can extract the accessible part from a random automata

	1	2	3	4	5	6
a	1	1	3	2	3	5
b	5	5	6	4	3	5



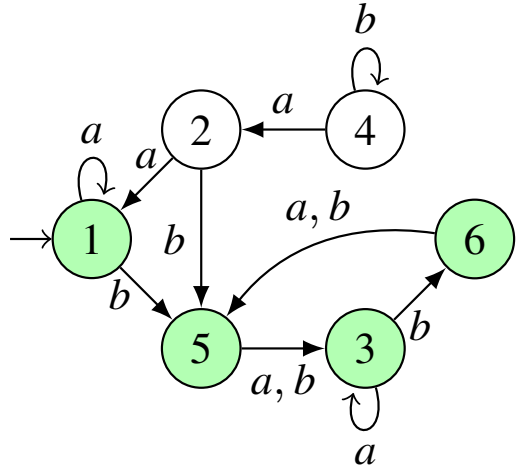
- We can extract the accessible part from a random automata

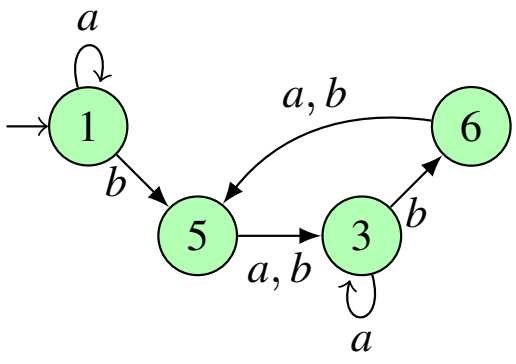
	1	2	3	4	5	6
a	1	1	3	2	3	5
b	5	5	6	4	3	5



- We can extract the accessible part from a random automata

	1	2	3	4	5	6
a	1	1	3	2	3	5
b	5	5	6	4	3	5

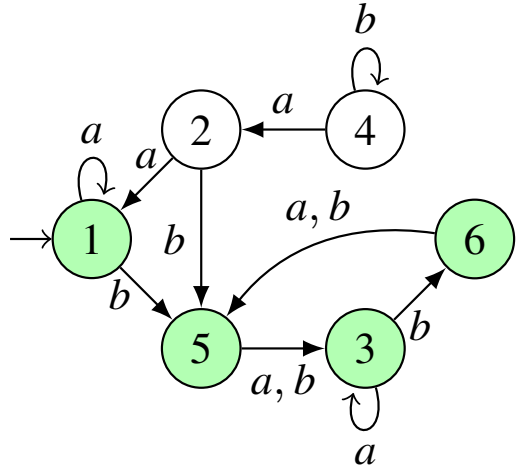


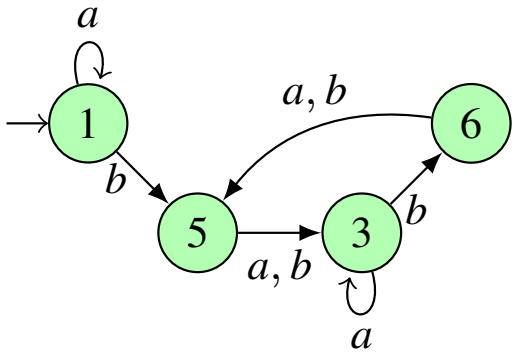


Extract

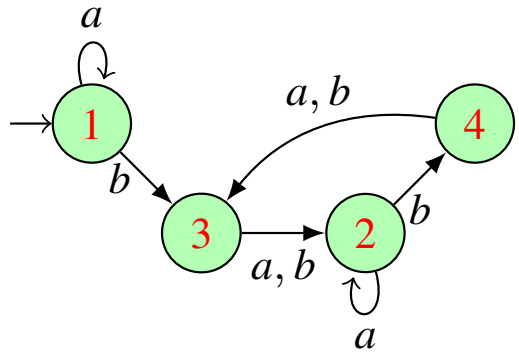
- We can extract the accessible part from a random automata

	1	2	3	4	5	6
a	1	1	3	2	3	5
b	5	5	6	4	3	5





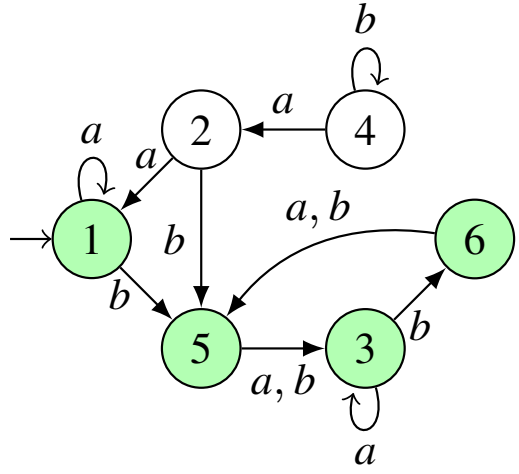
Extract

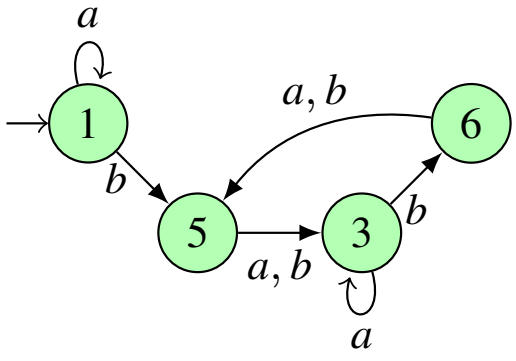


Normalize

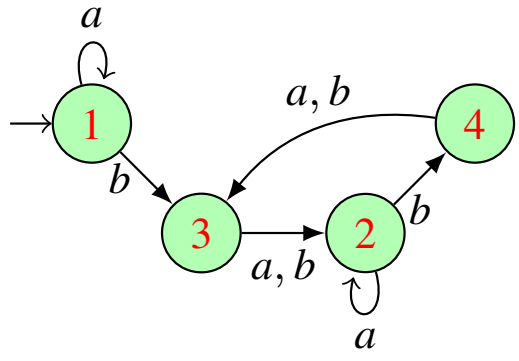
- We can extract the accessible part from a random automata

	1	2	3	4	5	6
a	1	1	3	2	3	5
b	5	5	6	4	3	5





Extract



Normalize

- We keep the relative order:

$$1 < 3 < 5 < 6$$
$$1 < 2 < 3 < 4$$

Two natural questions:

- What is the size of the accessible part?
- Is the induced distribution on accessible automata interesting?

Two natural questions:

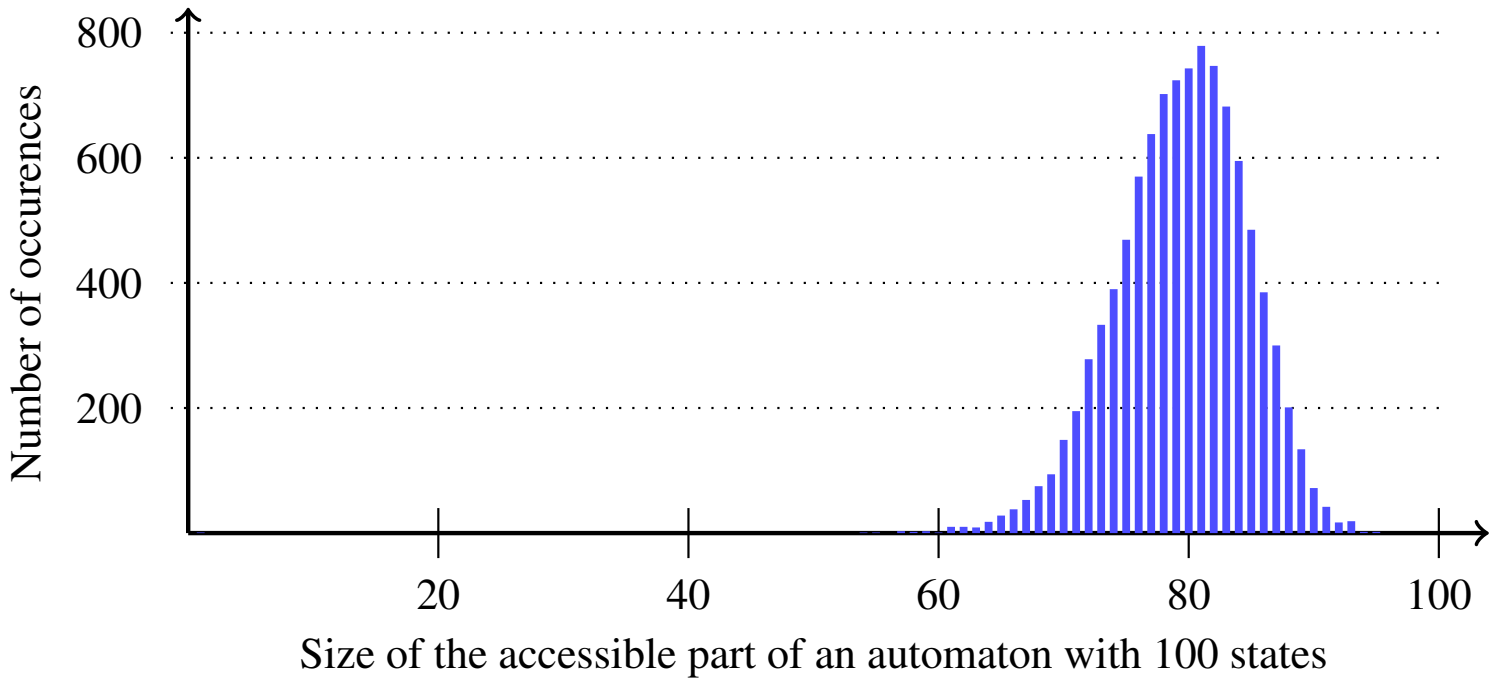
- What is the size of the accessible part?
- Is the induced distribution on accessible automata interesting?

- For the first question, we can do experiments

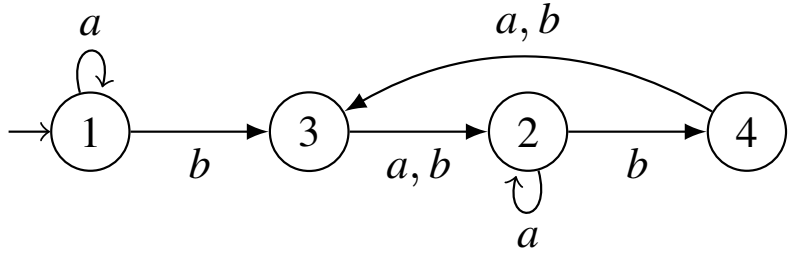
Two natural questions:

- What is the size of the accessible part?
- Is the induced distribution on accessible automata interesting?

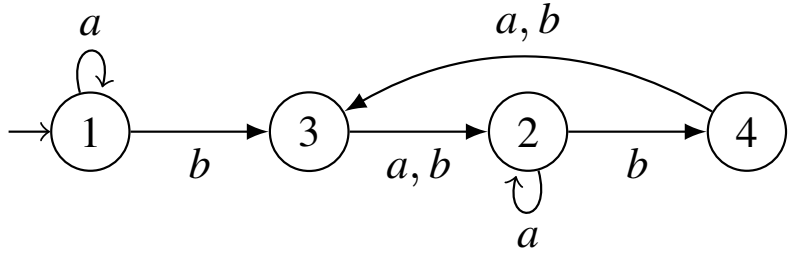
- For the first question, we can do experiments



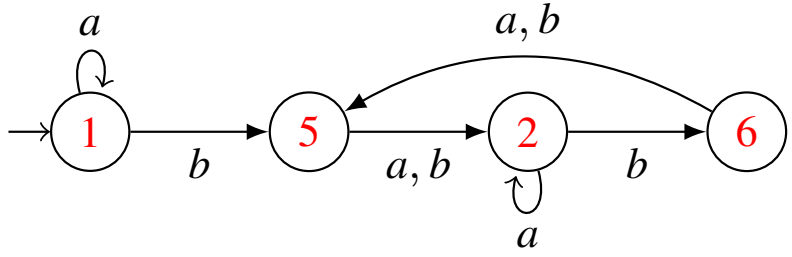
- Fix an accessible automaton $\mathcal{A}$ with i states. How many automata with n states produce $\mathcal{A}$? $n = 6$



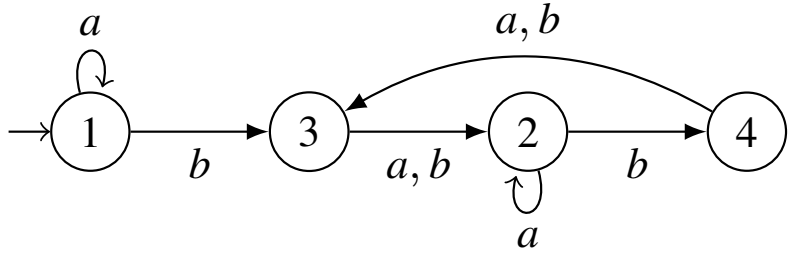
- Fix an accessible automaton $\mathcal{A}$ with i states. How many automata with n states produce $\mathcal{A}$? $n = 6$



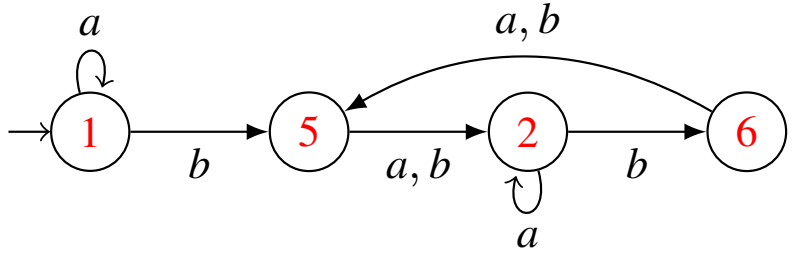
- Choose the labels of the states besides 1 and rename according to their relative order. $\{2, 5, 6\}$



- Fix an accessible automaton $\mathcal{A}$ with i states. How many automata with n states produce $\mathcal{A}$? $n = 6$



- Choose the labels of the states besides 1 and rename according to their relative order. $\{2, 5, 6\}$



- Remark that for the remaining states (for the example, states 3 and 4) any choice for their outgoing transitions is valid.

- The number of automata with n states that produce $\mathcal{A}$ is therefore:

$$\underbrace{\binom{n-1}{i-1}}_{\text{state labels}} \times \underbrace{n^{k(n-i)}}_{\text{remaining transitions}}$$

- It only depends on i, not on $\mathcal{A}$: two accessible automata with i states have the same probability of being generated

- Let X_n be the random variable associated with the size of the accessible part of a random automaton with n states. We have[1]

$$P(X_n = i) = |\mathcal{A}_i| \binom{n-1}{i-1} n^{-ki}$$

- First noticed in [Liskovets 69]

[1] Recall that $|\mathcal{A}_n|$ is the number of accessible automata with n states

– Limit distribution –

Theorem [Carayol, N. 12]

X_n is asymptotically normal, with mean and standard deviation respectively equivalent to vn and $\sigma\sqrt{n}$, with

$$v = 1 + \frac{1}{k} W_0(-k\,e^{-k}) \quad \text{and} \quad \sigma = \sqrt{\frac{v(1-v)}{kv - k + 1}}$$

– Limit distribution –

Theorem [Carayol, N. 12]

X_n is asymptotically normal, with mean and standard deviation respectively equivalent to vn and $\sigma\sqrt{n}$, with

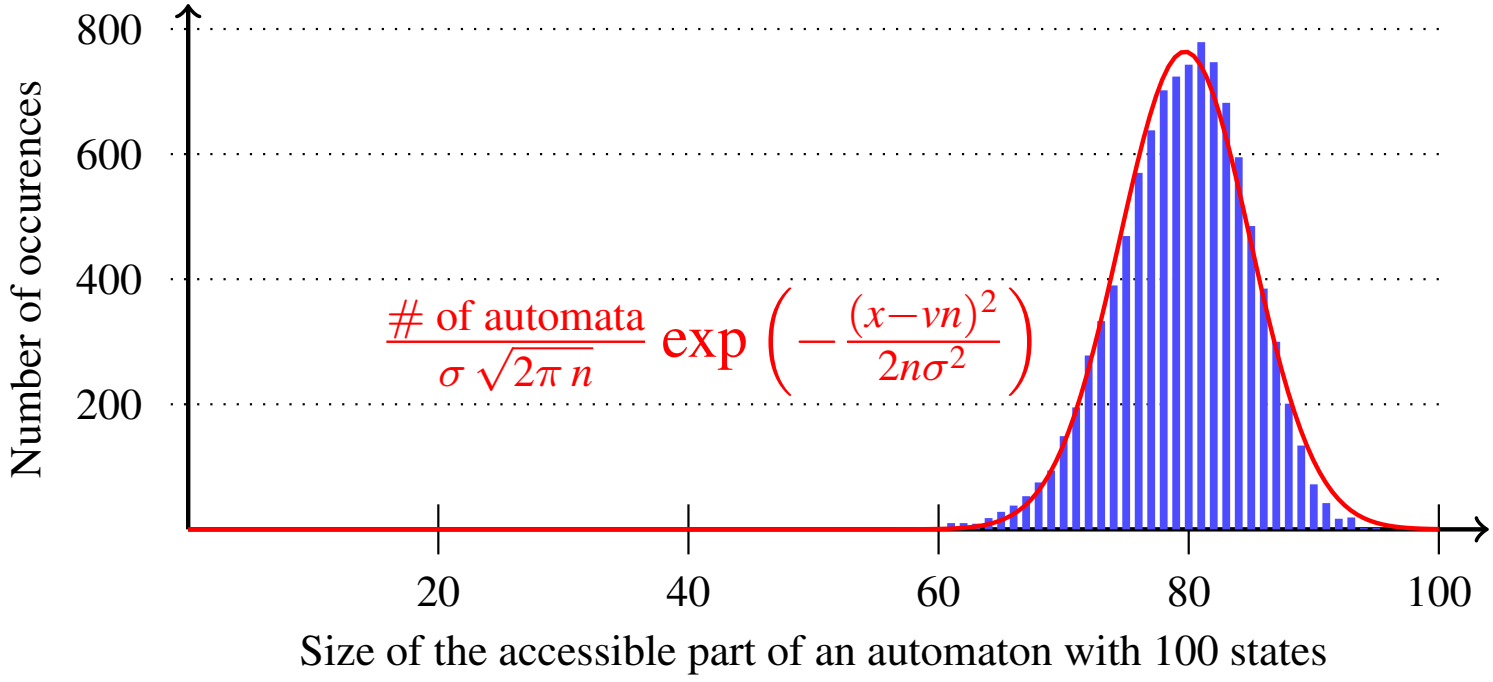
$$v = 1 + \frac{1}{k} W_0(-k\,e^{-k}) \quad \text{and} \quad \sigma = \sqrt{\frac{v(1-v)}{kv-k+1}}$$

- Approximating $|\mathcal{A}_i|$ using Korshunov's equivalent and the binomial coefficient with Stirling's yields

$$P(X_n = i) = |\mathcal{A}_i| \binom{n-1}{i-1} n^{-ki} \approx \frac{E \cdot \alpha}{\sqrt{2\pi n}}\, g\left(\frac{i}{n}\right) \left[f\left(\frac{i}{n}\right)\right]^n$$

with

$$f(x) = \frac{x^{(k-1)x}\beta^x}{(1-x)^{1-x}} \quad \text{and} \quad g(x) = \sqrt{\frac{x}{1-x}}$$


800
600
400
200
Number of occurences
$\frac{\#\text{ of automata}}{\sigma\sqrt{2\pi n}} \exp\left(-\frac{(x-vn)^2}{2n\sigma^2}\right)$
20
40
60
80
100
Size of the accessible part of an automaton with 100 states

– A simple yet efficient random generator –

- We have a very simple rejection algorithm to generate accessible automata uniformly at random:
  1. Generate a random automata $\mathcal{A}$ with $\frac{1}{v}n$ states
  2. If the accessible part of $\mathcal{A}$ does not have n states, go back to step 1
  3. Return the accessible part of $\mathcal{A}$

– A simple yet efficient random generator –

- We have a very simple rejection algorithm to generate accessible automata uniformly at random:
  1. Generate a random automata $\mathcal{A}$ with $\frac{1}{v}n$ states
  2. If the accessible part of $\mathcal{A}$ does not have n states, go back to step 1
  3. Return the accessible part of $\mathcal{A}$

- Each iteration of the loop is done in linear time
- The average number of iterations is $\Theta(\sqrt{n})$ since

$$P(X_{n/v} = n) \approx \frac{1}{\sigma\sqrt{n}}$$

– A simple yet efficient random generator –

- We have a very simple rejection algorithm to generate accessible automata uniformly at random:
  1. Generate a random automata $\mathcal{A}$ with $\frac{1}{v}n$ states
  2. If the accessible part of $\mathcal{A}$ does not have n states, go back to step 1
  3. Return the accessible part of $\mathcal{A}$

- Each iteration of the loop is done in linear time
- The average number of iterations is $\Theta(\sqrt{n})$ since

$$P(X_{n/v} = n) \approx \frac{1}{\sigma\sqrt{n}}$$

- The average complexity of this algorithm is $\Theta(n\sqrt{n})$
- It is the same complexity as before, but the algorithm is simpler

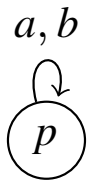
– A linear approximate sampler –

- We can do efficient approximate sampling
  1. Generate a random automata $\mathcal{A}$ with $\frac{1}{\nu}n$ states
  2. If the number of states of the accessible part of $\mathcal{A}$ is not in $[(1-\epsilon)n, (1+\epsilon)n)]$, go back to step 1
  3. Return the accessible part of $\mathcal{A}$

- Each iteration of the loop is done in linear time
- The average number of iterations tends to 1 as n tends to infinity

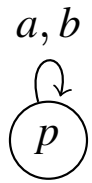
- The average complexity of this algorithm is linear

– $o(\frac{1}{\sqrt{n}})$-trick –

- An automaton of size m has a 

with probability $\leq \frac{1}{m}$
- Let $\mathcal{A}_n$ be the set of accessible automata of size n and $\mathcal{T}_m$ be the set of automata of size m

– $o(\frac{1}{\sqrt{n}})$-trick –

- An automaton of size m has a 

with probability $\leq \frac{1}{m}$
- Let $\mathcal{A}_n$ be the set of accessible automata of size n and $\mathcal{T}_m$ be the set of automata of size m

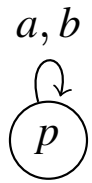
$$\frac{\left|\{\mathcal{A} \in \mathcal{A}_n : P(\mathcal{A})\}\right|}{|\mathcal{A}_n|} = \frac{\left|\{\mathcal{T} \in \mathcal{T}_m : |\mathcal{A}^{\mathcal{T}}| = n \text{ and } P(\mathcal{A}^{\mathcal{T}})\}\right|}{|\{\mathcal{T} \in \mathcal{T}_m : |\mathcal{A}^{\mathcal{T}}| = n\}|}$$

$$\leq \frac{\left|\{\mathcal{T} \in \mathcal{T}_m : P(\mathcal{T})\}\right|}{|\{\mathcal{T} \in \mathcal{T}_m : |\mathcal{A}^{\mathcal{T}}| = n\}|}$$

$$\leq \frac{\left|\{\mathcal{T} \in \mathcal{T}_m : P(\mathcal{T})\}\right|}{|\mathcal{T}_m|} \times \frac{|\mathcal{T}_m|}{|\{\mathcal{T} \in \mathcal{T}_m : |\mathcal{A}^{\mathcal{T}}| = n\}|}$$

$$\leq \frac{1}{m} \times \frac{1}{Pr(X_m = n)}$$

– $o(\frac{1}{\sqrt{n}})$-trick –

- An automaton of size m has a 

with probability $\leq \frac{1}{m}$
- Let $\mathcal{A}_n$ be the set of accessible automata of size n and $\mathcal{T}_m$ be the set of automata of size m

$$\frac{\left|\{\mathcal{A} \in \mathcal{A}_n : P(\mathcal{A})\}\right|}{|\mathcal{A}_n|} = \frac{\left|\{\mathcal{T} \in \mathcal{T}_m : |\mathcal{A}^{\mathcal{T}}| = n \text{ and } P(\mathcal{A}^{\mathcal{T}})\}\right|}{|\{\mathcal{T} \in \mathcal{T}_m : |\mathcal{A}^{\mathcal{T}}| = n\}|}$$

$$\leq \frac{\left|\{\mathcal{T} \in \mathcal{T}_m : P(\mathcal{T})\}\right|}{|\{\mathcal{T} \in \mathcal{T}_m : |\mathcal{A}^{\mathcal{T}}| = n\}|}$$

$$\leq \frac{\left|\{\mathcal{T} \in \mathcal{T}_m : P(\mathcal{T})\}\right|}{|\mathcal{T}_m|} \times \frac{|\mathcal{T}_m|}{|\{\mathcal{T} \in \mathcal{T}_m : |\mathcal{A}^{\mathcal{T}}| = n\}|}$$

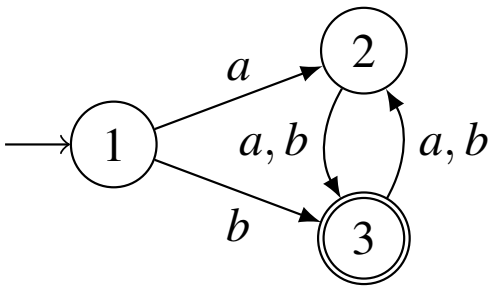
$$\leq \frac{1}{m} \times \frac{1}{Pr(X_m = n)}$$

- For $m = \frac{1}{\nu}n$, $Pr(X_m = n) = \Theta(\frac{1}{\sqrt{n}})$ and thus the probability is $O(\frac{1}{\sqrt{n}})$: accessible automata almost never have sinks

IV. Minimization algorithms

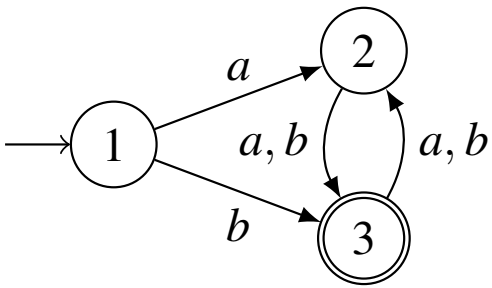
– Minimal automata –

- L_p is the language recognized by the automaton when the initial state is p
- p and q are equivalent ($p \sim q$) when $L_p = L_q$
- a deterministic automaton is minimal when there are no $p \neq q$ such that $p \sim q$.



– Minimal automata –

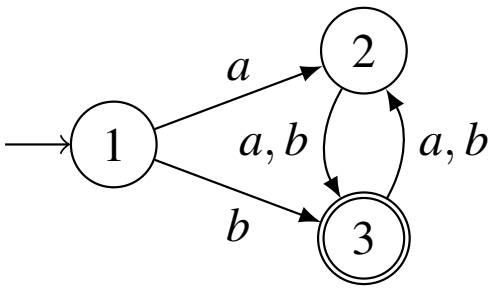
- L_p is the language recognized by the automaton when the initial state is p
- p and q are equivalent ($p \sim q$) when $L_p = L_q$
- a deterministic automaton is minimal when there are no $p \neq q$ such that $p \sim q$.



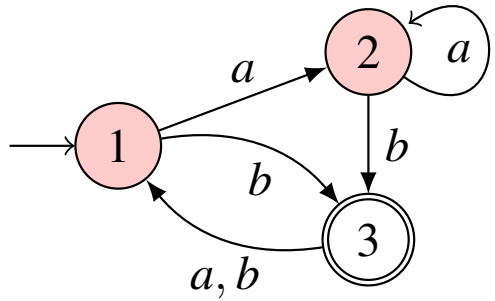
minimal

– Minimal automata –

- L_p is the language recognized by the automaton when the initial state is p
- p and q are equivalent ($p \sim q$) when $L_p = L_q$
- a deterministic automaton is minimal when there are no $p \neq q$ such that $p \sim q$.

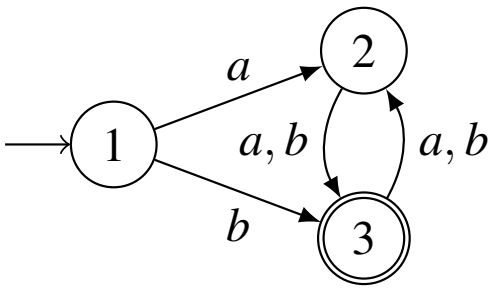


minimal

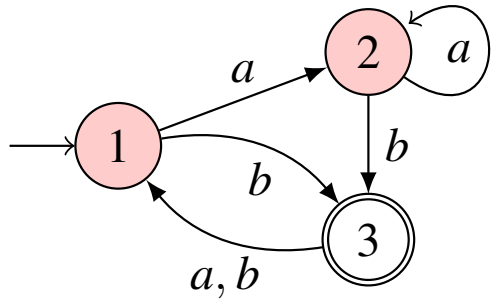


– Minimal automata –

- L_p is the language recognized by the automaton when the initial state is p
- p and q are equivalent ($p \sim q$) when $L_p = L_q$
- a deterministic automaton is minimal when there are no $p \neq q$ such that $p \sim q$.



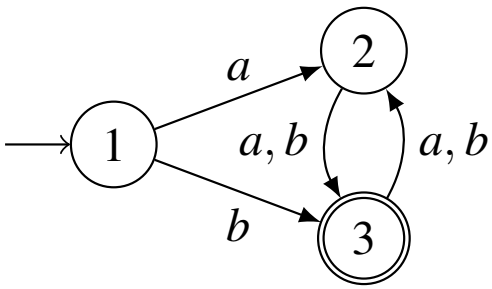
minimal



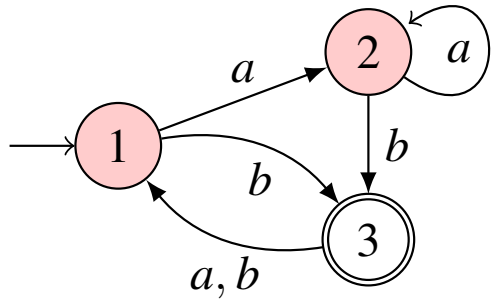
not minimal

– Minimal automata –

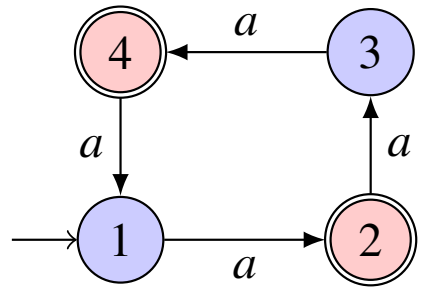
- L_p is the language recognized by the automaton when the initial state is p
- p and q are equivalent ($p \sim q$) when $L_p = L_q$
- a deterministic automaton is minimal when there are no $p \neq q$ such that $p \sim q$.



minimal

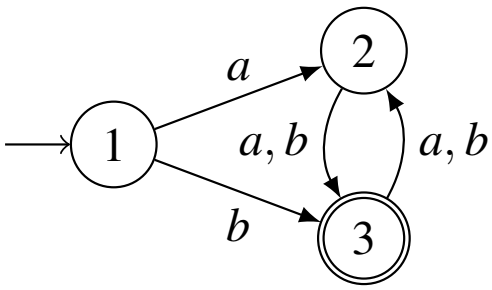


not minimal

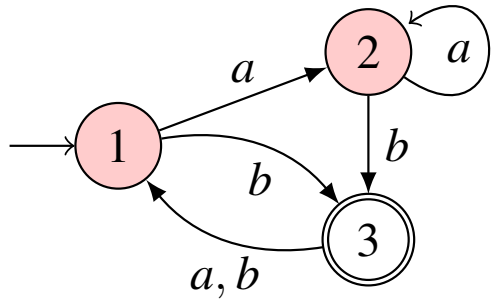


– Minimal automata –

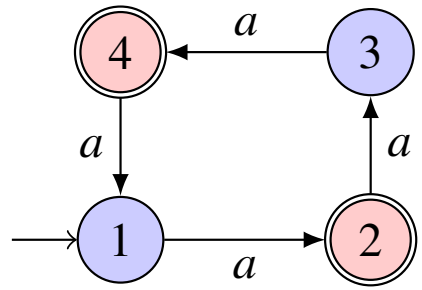
- L_p is the language recognized by the automaton when the initial state is p
- p and q are equivalent ($p \sim q$) when $L_p = L_q$
- a deterministic automaton is minimal when there are no $p \neq q$ such that $p \sim q$.



minimal



not minimal



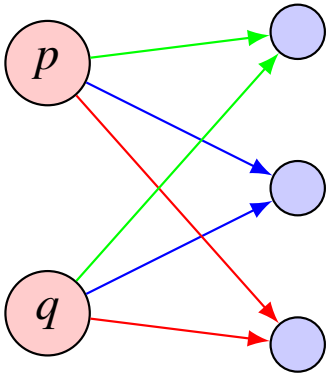
not minimal

– Counting Minimal Automata –

- The size of a regular language $\mathcal{L}$ is the number of states of its (unique) minimal automaton
- What is the ratio of minimal automata amongst accessible automata?

Theorem [Bassino, David, Sportiello 12]

For two letter alphabets, there exists a (computable) constant $c \in (0, 1)$ such that the ratio of minimal automata tends to c. For alphabets greater than two, the ratio tends to zero.



- Main pattern to avoid
- There are $\approx n^2$ choices for p and q
- The pattern appears for p and q with probability $\approx n^{-k}$

– Computing $\sim$ –

- $p \sim_\ell q$ when L_p and L_q contain the same words of lengths at most ℓ

– Computing $\sim$ –

- $p \sim_\ell q$ when L_p and L_q contain the same words of lengths at most ℓ
- $p \sim_0 q$ iff p and q are both final or both non-final
- A recursive formula:

$$p \sim_{\ell+1} q \Leftrightarrow \begin{cases} p \sim_\ell q \\ p \cdot a \sim_\ell q \cdot a, \text{ for every letter } a \end{cases}$$

– Computing $\sim$ –

- $p \sim_\ell q$ when L_p and L_q contain the same words of lengths at most ℓ
- $p \sim_0 q$ iff p and q are both final or both non-final
- A recursive formula:

$$p \sim_{\ell+1} q \Leftrightarrow \begin{cases} p \sim_\ell q \\ p \cdot a \sim_\ell q \cdot a, \text{ for every letter } a \end{cases}$$

- Moore's algorithm in $\mathcal{O}(n^2)$
- Hopcroft's algorithm in $\mathcal{O}(n \log n)$

– Computing $\sim$ –

- $p \sim_\ell q$ when L_p and L_q contain the same words of lengths at most ℓ
- $p \sim_0 q$ iff p and q are both final or both non-final
- A recursive formula:

$$p \sim_{\ell+1} q \Leftrightarrow \begin{cases} p \sim_\ell q \\ p \cdot a \sim_\ell q \cdot a, \text{ for every letter } a \end{cases}$$

- Moore's algorithm in $\mathcal{O}(n^2)$ $\leftarrow$ efficient in practice
- Hopcroft's algorithm in $\mathcal{O}(n \log n)$

– Moore's algorithm –

Moore($\mathcal{A}$)

1. Compute $\sim_0$
2. While $\sim_{i-1} \neq \sim_i$
   3. $i := i + 1$
   4. Compute $\sim_{i+1}$
5. Merge using $\sim_i$

- Moore's algorithm computes the minimal automaton
- Its complexity is $\Theta(n\ell)$, where ℓ is the number of iterations of the "while" loop
- In the worst case, $\ell = n$, and the complexity is quadratic

– Average case analysis of Moore's algorithm –

Theorem [Bassino, David, N. 09]

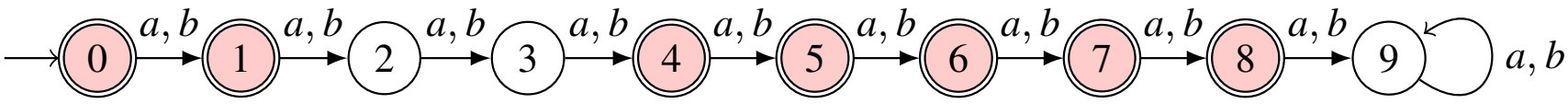
Let $\mathcal{A}$ be an accessible automaton with n states and no final state. For the uniform distribution on sets of final states, the average complexity of Moore's algorithm is $\mathcal{O}(n \log n)$.

- The $\mathcal{O}$ is uniform, the result holds for any distribution on automata's shapes.

Theorem [David 10]

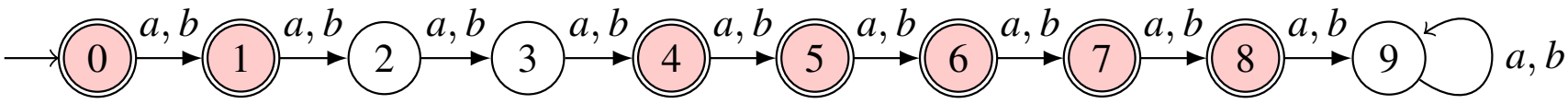
For the uniform distribution on (accessible) automata with n states, the average complexity of Moore's algorithm is $\mathcal{O}(n \log \log n)$.

– Proof on a very simple case –



- 2 and 5 are separated at the beginning
- 0 and 5 are separated after 2 iterations
- 4 and 5 are separated after 4 iterations

– Proof on a very simple case –



- 2 and 5 are separated at the beginning
- 0 and 5 are separated after 2 iterations
- 4 and 5 are separated after 4 iterations

- The number of iterations of the algorithm is rougthly the length of the longest run of final or non-final states
- This is $\mathcal{O}(\log n)$ in average

– Random Generation of Minimal Automata –

Random Minimal Automata

Using rejections, one can sample minimal automata of size n with average complexity $\mathcal{O}(n\sqrt{n})$.

- Checking minimality is done in $\mathcal{O}(n \log n)$

– Random Generation of Minimal Automata –

Random Minimal Automata

Using rejections, one can sample minimal automata of size n with average complexity $\mathcal{O}(n\sqrt{n})$.

- Checking minimality is done in $\mathcal{O}(n \log n)$

Approximate size

For any $\epsilon > 0$, one can sample minimal automata of size in $[(1-\epsilon)n, (1+\epsilon)n]$ with average complexity $\mathcal{O}(n \log\log n)$.

- Extraction of the accessible part
- Moore's algorithm + David's result for checking minimality

Perspectives

– Only one final state –

- In several “real life” applications automata only have a few final states
- Most results seen in this talk cannot be extended directly to automata with just one final state

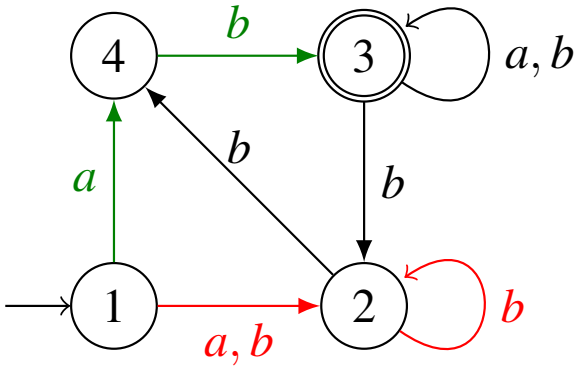
– Only one final state –

- In several "real life" applications automata only have a few final states
- Most results seen in this talk cannot be extended directly to automata with just one final state

- Experimentally the ratio of minimal automata still tends to a constant
- For Moore's algorithm:

	uniform shape	any shape
uniform final states	$\mathcal{O}(n \log \log n)$	$\mathcal{O}(n \log n)$
one final state	???	$\mathcal{O}(n^2)$

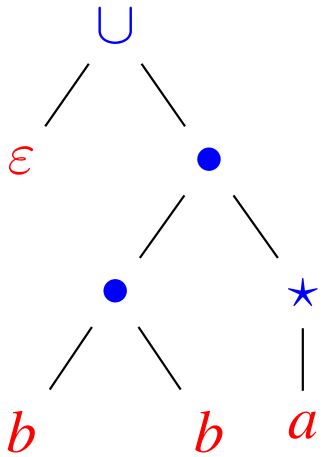
– Non-Deterministic Automata –



- A word is recognized when there exists a correct path
- Non-deterministic automata with n states can be turned into deterministic automata with 2^n states

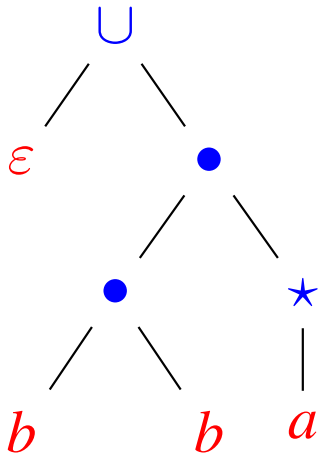
- The uniform distribution is not interesting
- Some results on codeterministic automata
- Experimental results for other classical distributions on graphs

– Distributions on Expressions –



- Regular expression of size n can be turned into non-deterministic automata of quadratic size
- For the uniform distribution, the average size of the automaton is linear
- For a BST-like distribution, the average size of the automaton is quadratic

– Distributions on Expressions –



- Regular expression of size n can be turned into non-deterministic automata of quadratic size
- For the uniform distribution, the average size of the automaton is linear
- For a BST-like distribution, the average size of the automaton is quadratic

- The distributions are somehow degenerated
- Difficult to find a "good" distribution for expressions
- Similar problem for logical formulas that denote regular expressions

Conclusion

Thank you