Descriptional complexity of automata and languages

Giovanni Pighizzini

Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano
ITALY

Milano – December 10th, 2008

Outline of the talk

Outline of the talk

Outline of the talk

Outline of the talk

Outline of the talk

Outline of the talk

Outline of the talk

1. Introduction
2. Regular languages and finite automata
3. The problem of Sakoda and Sipser
4. Unary automata
5. Regular languages vs context-free grammars and pushdown automata
6. State complexity of language operations
7. State complexity of the complementation

Outline of the talk

1. Introduction
2. Regular languages and finite automata
3. The problem of Sakoda and Sipser
4. Unary automata
5. Regular languages vs context-free grammars and pushdown automata
6. State complexity of language operations
7. State complexity of the complementation

Descriptional complexity

Given

- $\mathcal{C}$, a class of languages
- $\mathcal{S}$, a formal system (e.g., class of devices, class of grammars,..) able to represent all the languages in $\mathcal{C}$

What is the *size* of the representations of the languages in $\mathcal{C}$ by the system $\mathcal{S}$?

Usually, descriptional complexity compares different description for a same class of languages:

- given $\mathcal{S}'$, another formal system able to represent all the languages in $\mathcal{C}$

Descriptional complexity

Given

- $\mathcal{C}$, a class of languages
- $\mathcal{S}$, a formal system (e.g., class of devices, class of grammars,..) able to represent all the languages in $\mathcal{C}$

What is the *size* of the representations of the languages in $\mathcal{C}$ by the system $\mathcal{S}$?

Usually, descriptional complexity compares different description for a same class of languages:

- given $\mathcal{S}'$, another formal system able to represent all the languages in $\mathcal{C}$

Descriptional complexity

Given

- $\mathcal{C}$, a class of languages
- $\mathcal{S}$, a formal system (e.g., class of devices, class of grammars,..) able to represent all the languages in $\mathcal{C}$

What is the *size* of the representations of the languages in $\mathcal{C}$ by the system $\mathcal{S}$?

Usually, descriptional complexity compares different description for a same class of languages:

- given $\mathcal{S}'$, another formal system able to represent all the languages in $\mathcal{C}$

What is the *size* of the representations of the languages in $\mathcal{C}$ by the system $\mathcal{S}'$, with respect to the size of their representations by the system $\mathcal{S}$?

Descriptional complexity

Given

- $\mathcal{C}$, a class of languages
- $\mathcal{S}$, a formal system (e.g., class of devices, class of grammars,..) able to represent all the languages in $\mathcal{C}$

What is the *size* of the representations of the languages in $\mathcal{C}$ by the system $\mathcal{S}$?

Usually, descriptional complexity compares different description for a same class of languages:

- given $\mathcal{S}'$, another formal system able to represent all the languages in $\mathcal{C}$

What is the *size* of the representations of the languages in $\mathcal{C}$ by the system $\mathcal{S}'$, with respect to the size of their representations by the system $\mathcal{S}$?

Descriptional complexity

Given

- $\mathcal{C}$, a class of languages
- $\mathcal{S}$, a formal system (e.g., class of devices, class of grammars,..) able to represent all the languages in $\mathcal{C}$

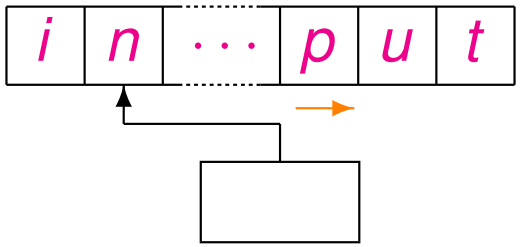
What is the *size* of the representations of the languages in $\mathcal{C}$ by the system $\mathcal{S}$?

Usually, descriptional complexity compares different description for a same class of languages:

- given $\mathcal{S}'$, another formal system able to represent all the languages in $\mathcal{C}$

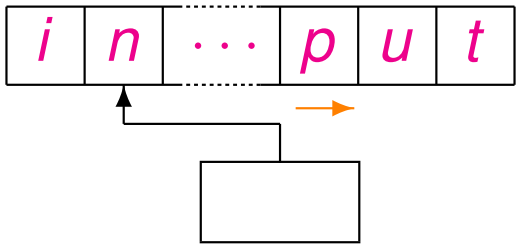
What is the *size* of the representations of the languages in $\mathcal{C}$ by the system $\mathcal{S}'$, with respect to the size of their representations by the system $\mathcal{S}$?

Classical example: finite state automata



Descriptional complexity

Classical example: finite state automata



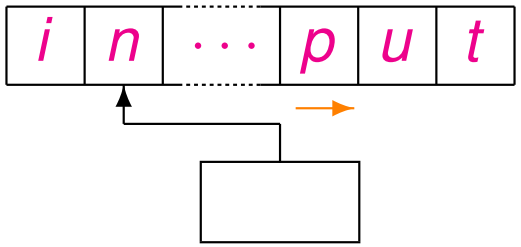
Base version:

One-way determistic finite automata (1dfa)

- one-way input tape
- deterministic

Descriptional complexity

Classical example: finite state automata

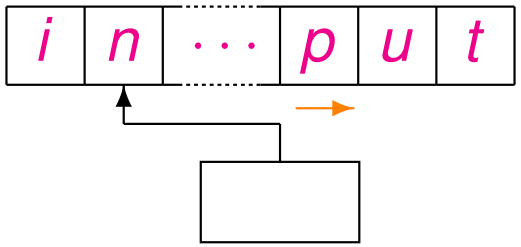


Some possibile variants introducing:

- non determinism
- two-way input head motion
- alternation
- ...

Descriptional complexity

Classical example: finite state automata

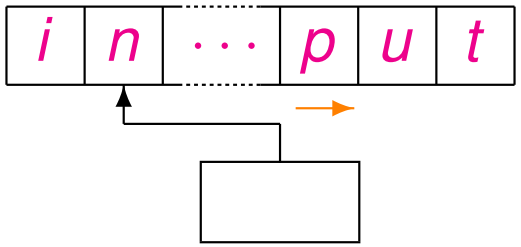


Some possibile variants introducing:

- non determinism
- two-way input head motion
- alternation
- ...

Descriptional complexity

Classical example: finite state automata

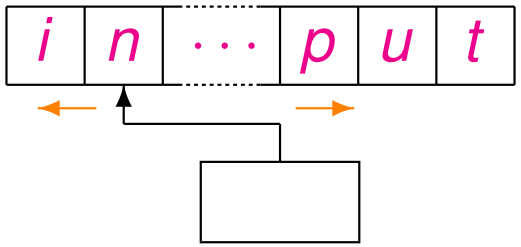


Some possibile variants introducing:

- non determinism
  - one-way nondetermistic finite automata (1nfa)
- two-way input head motion
- alternation
- ...

Descriptional complexity

Classical example: finite state automata

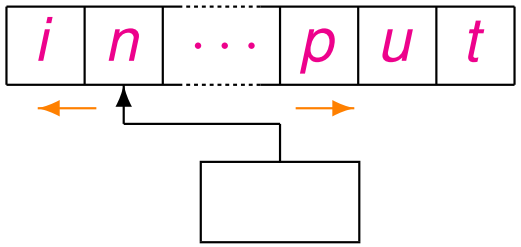


Some possibile variants introducing:

- non determinism (1nfa)
- two-way input head motion
- alternation
- ...

Descriptional complexity

Classical example: finite state automata

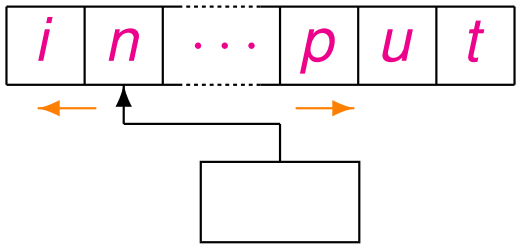


Some possibile variants introducing:

- non determinism (1nfa)
- two-way input head motion
  - two-way determistic finite automata (2dfa)
  - two-way nondetermistic finite automata (2nfa)
- alternation
- ...

Descriptional complexity

Classical example: finite state automata

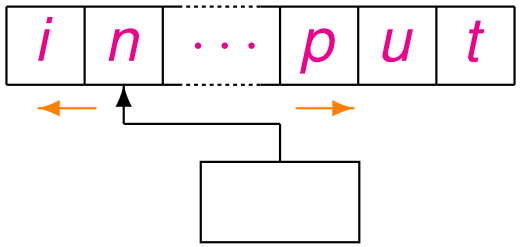


Some possibile variants introducing:

- non determinism (1nfa)
- two-way input head motion
  - two-way determistic finite automata (2dfa)
  - two-way nondetermistic finite automata (2nfa)
- alternation
- ...

Descriptional complexity

Classical example: finite state automata

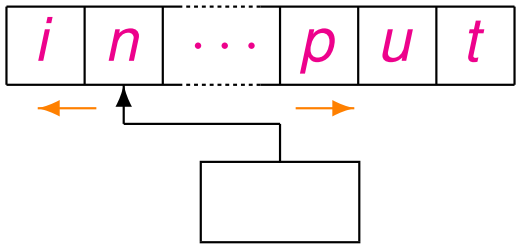


Some possibile variants introducing:

- non determinism (1nfa)
- two-way input head motion (2dfa, 2nfa)
- alternation
- ...

Descriptional complexity

Classical example: finite state automata



Some possibile variants introducing:

- non determinism (1nfa)
- two-way input head motion (2dfa, 2nfa)
- alternation
- ...

Finite state automata: different variants

1dfa, 1nfa, 2dfa, 2nfa, ...

- Formal language point of view:

  What about the *power* of these models?

  All of them characterize the class of *regular languages*...

- Descriptional complexity point of view:

  What about the *size* of their descriptions?

  ...however some of them are more succinct.

Finite state automata: different variants

1dfa, 1nfa, 2dfa, 2nfa, ...

- Formal language point of view:

  What about the *power* of these models?

  All of them characterize the class of *regular languages*...

- Descriptional complexity point of view:

  What about the *size* of their descriptions?

  ...however some of them are more succinct.

Finite state automata: different variants

1dfa, 1nfa, 2dfa, 2nfa, ...

- Formal language point of view:

  What about the *power* of these models?

  All of them characterize the class of *regular languages*...

- Descriptional complexity point of view:

  What about the *size* of their descriptions?

  ...however some of them are more succinct.

Finite state automata: different variants

1dfa, 1nfa, 2dfa, 2nfa, ...

- Formal language point of view:

  What about the *power* of these models?

  All of them characterize the class of *regular languages*...

- Descriptional complexity point of view:

  What about the *size* of their descriptions?

  ...however some of them are more succinct.

Finite state automata: different variants

1dfa, 1nfa, 2dfa, 2nfa, ...

- Formal language point of view:

  What about the *power* of these models?

  All of them characterize the class of *regular languages*...

- Descriptional complexity point of view:

  What about the *size* of their descriptions?

  ...however some of them are more succinct.

Finite state automata: different variants

1dfa, 1nfa, 2dfa, 2nfa, ...

- Formal language point of view:

  What about the *power* of these models?

  All of them characterize the class of *regular languages*...

- Descriptional complexity point of view:

  What about the *size* of their descriptions?

  ...however some of them are more succinct.

Example

- Each n-state 1nfa can be simulated by a 1dfa with 2^n states (subset construction) [Rabin and Scott '59], and:
- For each integer $n \geq 1$ there is a language which is accepted by a n-state 1nfa which requires 2^n states to be accepted by a 1dfa [Meyer and Fischer '71].

Finite state automata: different variants

1dfa, 1nfa, 2dfa, 2nfa, ...

- Formal language point of view:

  What about the *power* of these models?

  All of them characterize the class of *regular languages*...

- Descriptional complexity point of view:

  What about the *size* of their descriptions?

  ...however some of them are more succinct.

Example

- Each n-state 1nfa can be simulated by a 1dfa with 2^n states (subset construction) [Rabin and Scott '59], and:
- For each integer $n \geq 1$ there is a language which is accepted by a n-state 1nfa which requires 2^n states to be accepted by a 1dfa [Meyer and Fischer '71].

Descriptional complexity measures for finite automa

- Deterministic automata (dfa):
  number of states.
- Nondeterministic automata (nfa):
  number of states, or
  number of transitions (more precise).

Remarks:

- Each nfa with n states has $O(n^2)$ transitions.
- Many results have been obtained for the measure “number of the states”.
- The measure “number of transitions” was recently investigated by some authors,
  however it was already proposed as a more realistic measure in1971 by Meyer and Fischer.

Descriptional complexity measures for finite automa

- Deterministic automata (dfa):
  number of states.
- Nondeterministic automata (nfa):
  number of states, or
  number of transitions (more precise).

Remarks:

- Each nfa with n states has $O(n^2)$ transitions.
- Many results have been obtained for the measure “number of the states”.
- The measure “number of transitions” was recently investigated by some authors,
  however it was already proposed as a more realistic measure in1971 by Meyer and Fischer.

Descriptional complexity measures for finite automa

- Deterministic automata (dfa):
  number of states.
- Nondeterministic automata (nfa):
  number of states, or
  number of transitions (more precise).

Remarks:

- Each nfa with n states has $O(n^2)$ transitions.
- Many results have been obtained for the measure “number of the states”.
- The measure “number of transitions” was recently investigated by some authors,
  however it was already proposed as a more realistic measure in1971 by Meyer and Fischer.

Descriptional complexity measures for finite automa

- Deterministic automata (dfa):
  number of states.
- Nondeterministic automata (nfa):
  number of states, or
  number of transitions (more precise).

Remarks:

- Each nfa with n states has $O(n^2)$ transitions.
- Many results have been obtained for the measure “number of the states”.
- The measure “number of transitions” was recently investigated by some authors,
  however it was already proposed as a more realistic measure in1971 by Meyer and Fischer.

Descriptional complexity measures for finite automa

- Deterministic automata (dfa):
  number of states.
- Nondeterministic automata (nfa):
  number of states, or
  number of transitions (more precise).

Remarks:

- Each nfa with n states has $O(n^2)$ transitions.
- Many results have been obtained for the measure “number of the states”.
- The measure “number of transitions” was recently investigated by some authors,
  however it was already proposed as a more realistic measure in1971 by Meyer and Fischer.

Descriptional complexity measures for finite automa

- Deterministic automata (dfa):
  number of states.
- Nondeterministic automata (nfa):
  number of states, or
  number of transitions (more precise).

Remarks:

- Each nfa with n states has $O(n^2)$ transitions.
- Many results have been obtained for the measure “number of the states”.
- The measure “number of transitions” was recently investigated by some authors,
  however it was already proposed as a more realistic measure in1971 by Meyer and Fischer.

Descriptional complexity measures for finite automa

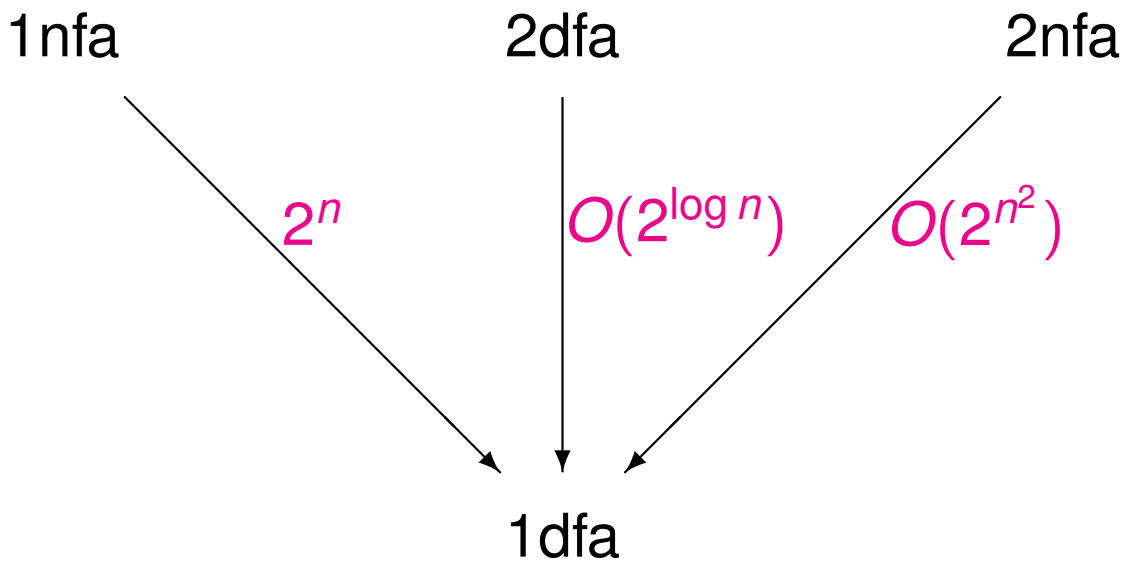
- Deterministic automata (dfa):
  number of states.
- Nondeterministic automata (nfa):
  number of states, or
  number of transitions (more precise).

Remarks:

- Each nfa with n states has $O(n^2)$ transitions.
- Many results have been obtained for the measure “number of the states”.
- The measure “number of transitions” was recently investigated by some authors,
  however it was already proposed as a more realistic measure in1971 by Meyer and Fischer.

Finite state automata

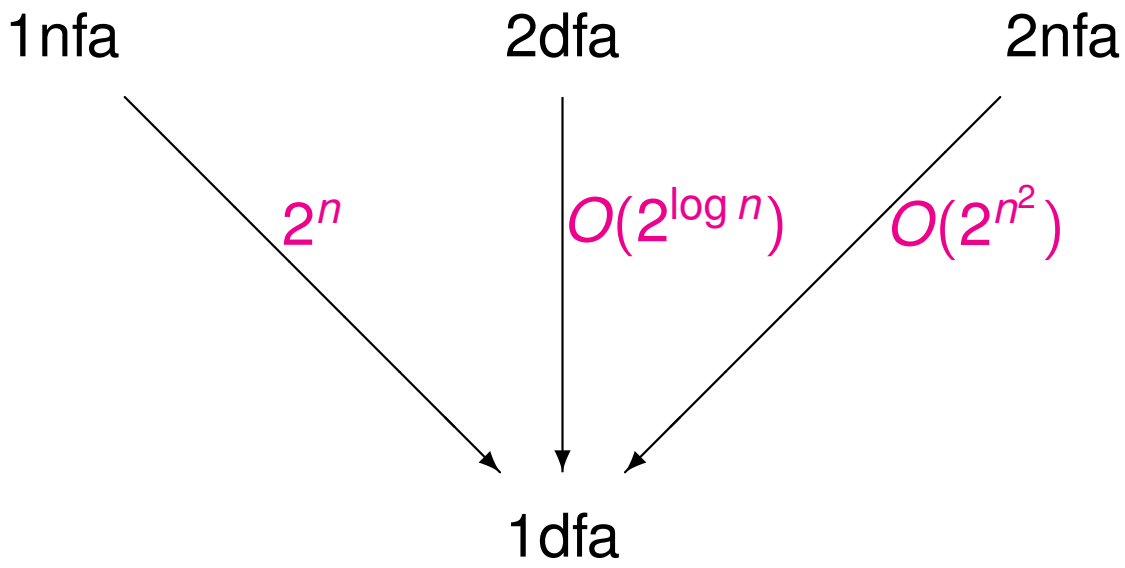
Costs of the optimal simulations by 1dfa (in terms of states):



[Rabin and Scott '59, Shepardson '59,Meyer and Fischer '71,...]

Finite state automata

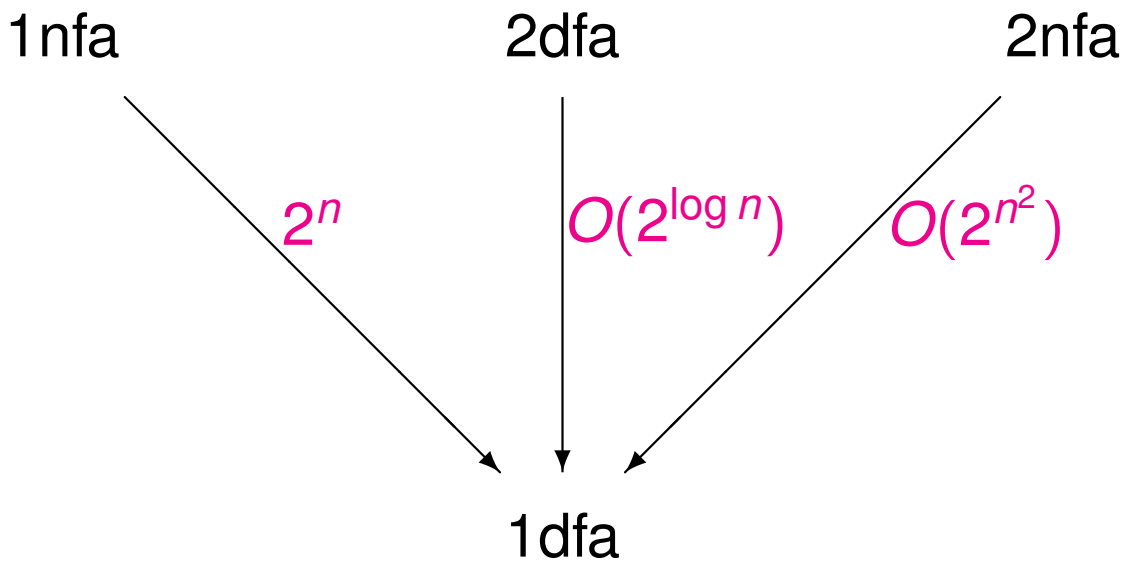
Costs of the optimal simulations by 1dfa (in terms of states):



How much two-way motion can be useful in order to eliminate the nondeterminism?

Finite state automata

Costs of the optimal simulations by 1dfa (in terms of states):



Problem ([Sakoda and Sipser 1978])

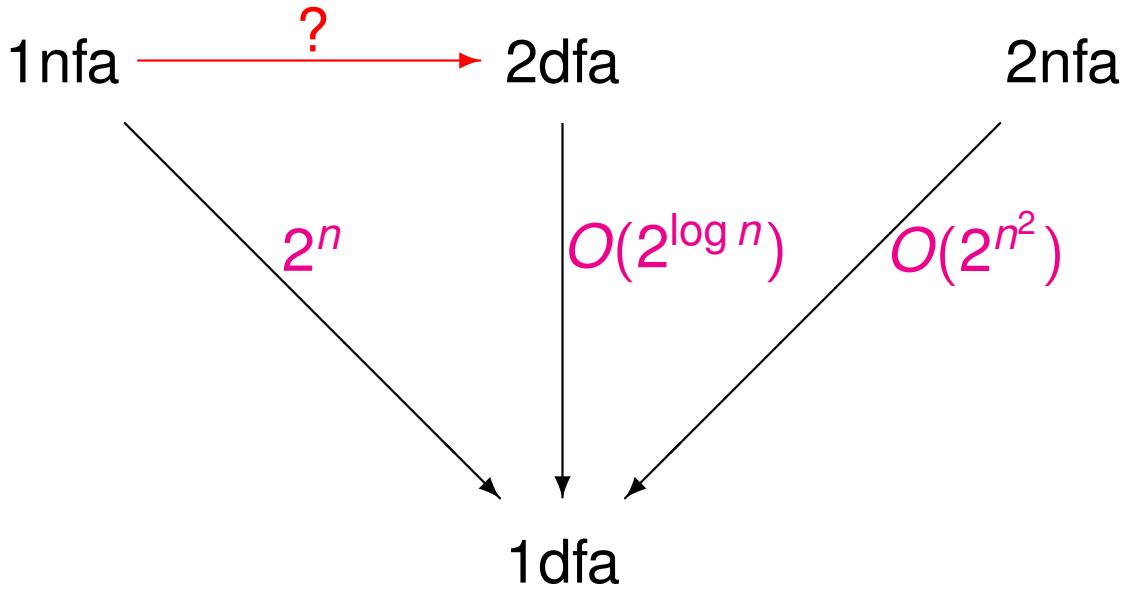
Find the costs, in terms of states, of the optimal simulations of

- *1nfa by 2dfa*
- *2nfa by 2dfa*

Conjecture: these costs are exponential

Finite state automata

Costs of the optimal simulations by 1dfa (in terms of states):



Problem ([Sakoda and Sipser 1978])

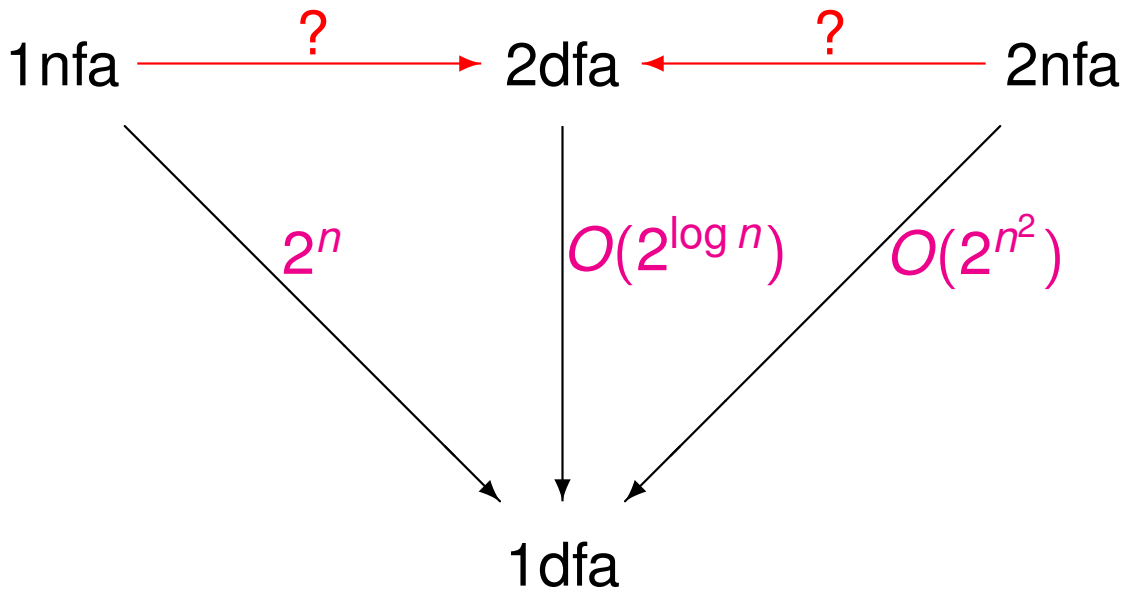
Find the costs, in terms of states, of the optimal simulations of

- *1nfa by 2dfa*
- *2nfa by 2dfa*

Conjecture: these costs are exponential

Finite state automata

Costs of the optimal simulations by 1dfa (in terms of states):



Problem ([Sakoda and Sipser 1978])

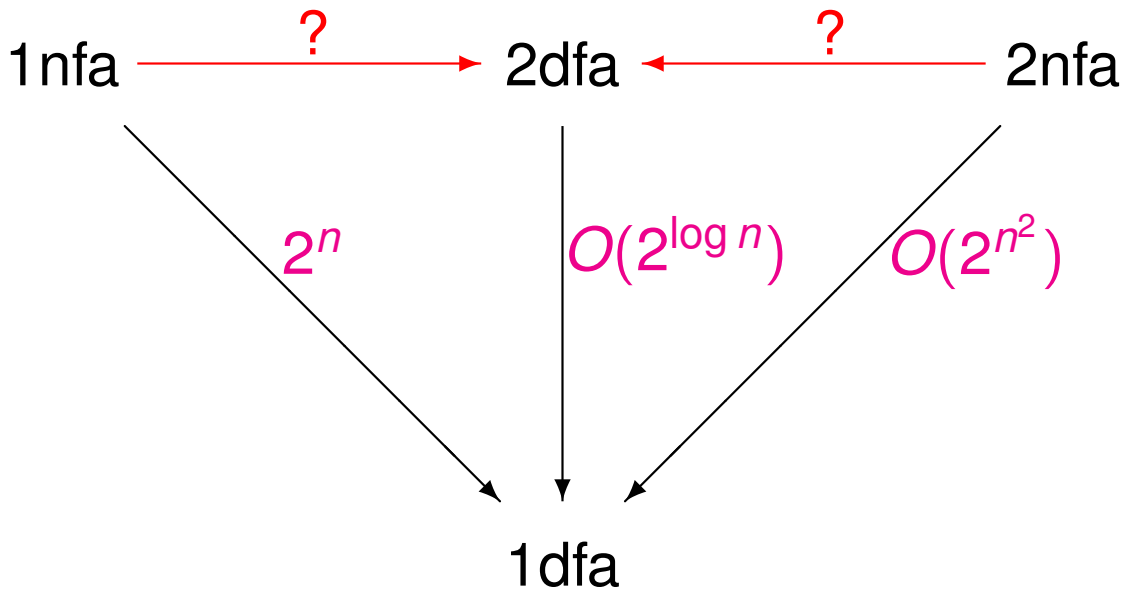
Find the costs, in terms of states, of the optimal simulations of

- *1nfa by 2dfa*
- *2nfa by 2dfa*

Conjecture: these costs are exponential

Finite state automata

Costs of the optimal simulations by 1dfa (in terms of states):



Problem ([Sakoda and Sipser 1978])

Find the costs, in terms of states, of the optimal simulations of

- *1nfa by 2dfa*
- *2nfa by 2dfa*

Conjecture: these costs are exponential

Sakoda and Sipser question: complete languages

Theorem (Complete languages for 1nfa vs 2dfa)

There exists a sequence of languages $< B_1, B_2, \ldots, B_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- B_n *is accepted by a 1nfa with* n *states, and*
- *among all languages accepted by* n*-state 1nfa,* B_n *requires the largest 2dfa.*

Remark: the second condition implies that
the simulation of 1nfa by 2dfa is polynomial iff each B_n is accepted by a 2dfa with a polynomial (in n) number of states.

In a similar way:

Theorem (Complete languages for 2nfa vs 2dfa)

There exists a sequence of languages $< C_1, C_2, \ldots, C_n, \ldots >$ *complete for the reduction of 2nfa to 2dfa.*

Sakoda and Sipser question: complete languages

Theorem (Complete languages for 1nfa vs 2dfa)

There exists a sequence of languages $< B_1, B_2, \ldots, B_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- B_n *is accepted by a 1nfa with* n *states, and*
- *among all languages accepted by* n*-state 1nfa,* B_n *requires the largest 2dfa.*

Remark: the second condition implies that
the simulation of 1nfa by 2dfa is polynomial iff each B_n is accepted by a 2dfa with a polynomial (in n) number of states.

In a similar way:

Theorem (Complete languages for 2nfa vs 2dfa)

There exists a sequence of languages $< C_1, C_2, \ldots, C_n, \ldots >$ *complete for the reduction of 2nfa to 2dfa.*

Sakoda and Sipser question: complete languages

Theorem (Complete languages for 1nfa vs 2dfa)

There exists a sequence of languages $< B_1, B_2, \ldots, B_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- B_n *is accepted by a 1nfa with* n *states, and*
- *among all languages accepted by* n*-state 1nfa,* B_n *requires the largest 2dfa.*

Remark: the second condition implies that
the simulation of 1nfa by 2dfa is polynomial iff each B_n is accepted by a 2dfa with a polynomial (in n) number of states.

In a similar way:

Theorem (Complete languages for 2nfa vs 2dfa)

There exists a sequence of languages $< C_1, C_2, \ldots, C_n, \ldots >$ *complete for the reduction of 2nfa to 2dfa.*

Sakoda and Sipser question: complete languages

Theorem (Complete languages for 1nfa vs 2dfa)

There exists a sequence of languages $< B_1, B_2, \ldots, B_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- B_n *is accepted by a 1nfa with* n *states, and*
- *among all languages accepted by* n*-state 1nfa,* B_n *requires the largest 2dfa.*

Remark: the second condition implies that
the simulation of 1nfa by 2dfa is polynomial iff each B_n is accepted by a 2dfa with a polynomial (in n) number of states.

In a similar way:

Theorem (Complete languages for 2nfa vs 2dfa)

There exists a sequence of languages $< C_1, C_2, \ldots, C_n, \ldots >$ *complete for the reduction of 2nfa to 2dfa.*

Sakoda and Sipser question: complete languages

Theorem (Complete languages for 1nfa vs 2dfa)

There exists a sequence of languages $< B_1, B_2, \ldots, B_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- B_n *is accepted by a 1nfa with* n *states, and*
- *among all languages accepted by* n*-state 1nfa,* B_n *requires the largest 2dfa.*

Remark: the second condition implies that
the simulation of 1nfa by 2dfa is polynomial iff each B_n is accepted by a 2dfa with a polynomial (in n) number of states.

In a similar way:

Theorem (Complete languages for 2nfa vs 2dfa)

There exists a sequence of languages $< C_1, C_2, \ldots, C_n, \ldots >$ *complete for the reduction of 2nfa to 2dfa.*

Sakoda and Sipser question: lower bounds

Polynomial lower bounds have been proved for the cost $c(n)$ of simulation of 1nfa by 2dfa. In particular:

- $c(n) \in \Omega(\frac{n^2}{\log n})$ [Berman and Lingas 1977]
- $c(n) \in \Omega(n^2)$ [Chrobak 1986]

Exponential lower bounds have been proved, if the resulting machines are required to satisfy some special conditions. e.g.,

- *sweeping automata* [Sipser 1980]
- *oblivious automata* [Hromkovič and Schnitger 2003]

Sakoda and Sipser question: lower bounds

Polynomial lower bounds have been proved for the cost $c(n)$ of simulation of 1nfa by 2dfa. In particular:

- $c(n) \in \Omega(\frac{n^2}{\log n})$ [Berman and Lingas 1977]
- $c(n) \in \Omega(n^2)$ [Chrobak 1986]

Exponential lower bounds have been proved, if the resulting machines are required to satisfy some special conditions. e.g.,

- *sweeping automata* [Sipser 1980]
- *oblivious automata* [Hromkovič and Schnitger 2003]

Sakoda and Sipser question: lower bounds

Polynomial lower bounds have been proved for the cost $c(n)$ of simulation of 1nfa by 2dfa. In particular:

- $c(n) \in \Omega(\frac{n^2}{\log n})$ [Berman and Lingas 1977]
- $c(n) \in \Omega(n^2)$ [Chrobak 1986]

Exponential lower bounds have been proved, if the resulting machines are required to satisfy some special conditions. e.g.,

- *sweeping automata* [Sipser 1980]
- *oblivious automata* [Hromkovič and Schnitger 2003]

Sakoda and Sipser question: lower bounds

Polynomial lower bounds have been proved for the cost $c(n)$ of simulation of 1nfa by 2dfa. In particular:

- $c(n) \in \Omega(\frac{n^2}{\log n})$ [Berman and Lingas 1977]
- $c(n) \in \Omega(n^2)$ [Chrobak 1986]

Exponential lower bounds have been proved, if the resulting machines are required to satisfy some special conditions. e.g.,

- *sweeping automata* [Sipser 1980]
- *oblivious automata* [Hromkovič and Schnitger 2003]

Sakoda and Sipser question: lower bounds

Polynomial lower bounds have been proved for the cost $c(n)$ of simulation of 1nfa by 2dfa. In particular:

- $c(n) \in \Omega(\frac{n^2}{\log n})$ [Berman and Lingas 1977]
- $c(n) \in \Omega(n^2)$ [Chrobak 1986]

Exponential lower bounds have been proved, if the resulting machines are required to satisfy some special conditions. e.g.,

- *sweeping automata* [Sipser 1980]
- *oblivious automata* [Hromkovič and Schnitger 2003]

Sakoda and Sipser question: lower bounds

Polynomial lower bounds have been proved for the cost $c(n)$ of simulation of 1nfa by 2dfa. In particular:

- $c(n) \in \Omega(\frac{n^2}{\log n})$ [Berman and Lingas 1977]
- $c(n) \in \Omega(n^2)$ [Chrobak 1986]

Exponential lower bounds have been proved, if the resulting machines are required to satisfy some special conditions. e.g.,

- *sweeping automata* [Sipser 1980]
- *oblivious automata* [Hromkovič and Schnitger 2003]

Sweeping automata

Theorem ([Sipser 1980])

There exists a family of languages $< B_1, B_2, \ldots, B_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- B_n *is accepted by a 1nfa with* n *states, and*
- B_n *cannot be accepted by any sweeping automaton with less than* 2^n *states.*

However, 2dfa can be exponentially more succinct than sweeping automata:

Theorem ([Berman 1981, Micali 1981])

There exists a family of languages $< A_1, A_2, \ldots, A_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- A_n *is accepted by a 2dfa with* n *states, and*
- A_n *cannot be accepted by any sweeping automaton with less than* $2^n - 1$ *states.*

Sweeping automata

Theorem ([Sipser 1980])

There exists a family of languages $< B_1, B_2, \ldots, B_n, \ldots >$ s.t. for each integer $n \geq 1$:

- *B_n is accepted by a 1nfa with n states, and*
- *B_n cannot be accepted by any sweeping automaton with less than 2^n states.*

However, 2dfa can be exponentially more succinct than sweeping automata:

Theorem ([Berman 1981, Micali 1981])

There exists a family of languages $< A_1, A_2, \ldots, A_n, \ldots >$ s.t. for each integer $n \geq 1$:

- *A_n is accepted by a 2dfa with n states, and*
- *A_n cannot be accepted by any sweeping automaton with less than $2^n - 1$ states.*

Sweeping automata

Theorem ([Sipser 1980])

There exists a family of languages $< B_1, B_2, \ldots, B_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- B_n *is accepted by a 1nfa with* n *states, and*
- B_n *cannot be accepted by any sweeping automaton with less than* 2^n *states.*

However, 2dfa can be exponentially more succinct than sweeping automata:

Theorem ([Berman 1981, Micali 1981])

There exists a family of languages $< A_1, A_2, \ldots, A_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- A_n *is accepted by a 2dfa with* n *states, and*
- A_n *cannot be accepted by any sweeping automaton with less than* $2^n - 1$ *states.*

Sweeping automata

Theorem ([Sipser 1980])

There exists a family of languages $< B_1, B_2, \ldots, B_n, \ldots >$ s.t. for each integer $n \geq 1$:

- *B_n is accepted by a 1nfa with n states, and*
- *B_n cannot be accepted by any sweeping automaton with less than 2^n states.*

However, 2dfa can be exponentially more succinct than sweeping automata:

Theorem ([Berman 1981, Micali 1981])

There exists a family of languages $< A_1, A_2, \ldots, A_n, \ldots >$ s.t. for each integer $n \geq 1$:

- *A_n is accepted by a 2dfa with n states, and*
- *A_n cannot be accepted by any sweeping automaton with less than $2^n - 1$ states.*

Sweeping automata

Theorem ([Sipser 1980])

There exists a family of languages $< B_1, B_2, \ldots, B_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- B_n *is accepted by a 1nfa with* n *states, and*
- B_n *cannot be accepted by any sweeping automaton with less than* 2^n *states.*

However, 2dfa can be exponentially more succinct than sweeping automata:

Theorem ([Berman 1981, Micali 1981])

There exists a family of languages $< A_1, A_2, \ldots, A_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- A_n *is accepted by a 2dfa with* n *states, and*
- A_n *cannot be accepted by any sweeping automaton with less than* $2^n - 1$ *states.*

Sweeping automata

Theorem ([Sipser 1980])

There exists a family of languages $< B_1, B_2, \ldots, B_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- B_n *is accepted by a 1nfa with* n *states, and*
- B_n *cannot be accepted by any sweeping automaton with less than* 2^n *states.*

However, 2dfa can be exponentially more succinct than sweeping automata:

Theorem ([Berman 1981, Micali 1981])

There exists a family of languages $< A_1, A_2, \ldots, A_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- A_n *is accepted by a 2dfa with* n *states, and*
- A_n *cannot be accepted by any sweeping automaton with less than* $2^n - 1$ *states.*

Sweeping automata

Theorem ([Sipser 1980])

There exists a family of languages $< B_1, B_2, \ldots, B_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- B_n *is accepted by a 1nfa with* n *states, and*
- B_n *cannot be accepted by any sweeping automaton with less than* 2^n *states.*

However, 2dfa can be exponentially more succinct than sweeping automata:

Theorem ([Berman 1981, Micali 1981])

There exists a family of languages $< A_1, A_2, \ldots, A_n, \ldots >$ *s.t. for each integer* $n \geq 1$*:*

- A_n *is accepted by a 2dfa with* n *states, and*
- A_n *cannot be accepted by any sweeping automaton with less than* $2^n - 1$ *states.*

Sakoda&Sipser question

Current knowledge

- Upper bounds

	1nfa $\longrightarrow$ 2dfa	2nfa $\longrightarrow$ 2dfa
general case	exponential	exponential
unary case	$O(n^2)$ [1] optimal	$n^{O(\log n)}$ [2]

[1: Chrobak 1986]

[2: Geffert, Mereghetti, Pighizzini 2003]

- Lower bounds

  For all the cases, the best known lower bound is $\Omega(n^2)$ [1]

Sakoda&Sipser question

Current knowledge

- Upper bounds

	1nfa $\longrightarrow$ 2dfa	2nfa $\longrightarrow$ 2dfa
general case	exponential	exponential
unary case	$O(n^2)$ [1] optimal	$n^{O(\log n)}$ [2]

[1: Chrobak 1986]
[2: Geffert, Mereghetti, Pighizzini 2003]

- Lower bounds
  For all the cases, the best known lower bound is $\Omega(n^2)$ [1]

Sakoda&Sipser question

Current knowledge

- Upper bounds

	1nfa $\longrightarrow$ 2dfa	2nfa $\longrightarrow$ 2dfa
general case	exponential	exponential
unary case	$O(n^2)$ [1] optimal	$n^{O(\log n)}$ [2]

[1: Chrobak 1986]

[2: Geffert, Mereghetti, Pighizzini 2003]

- Lower bounds

  For all the cases, the best known lower bound is $\Omega(n^2)$ [1]

Unary automata: 1dfa

Input alphabet $\Sigma = \{a\}$

Theorem

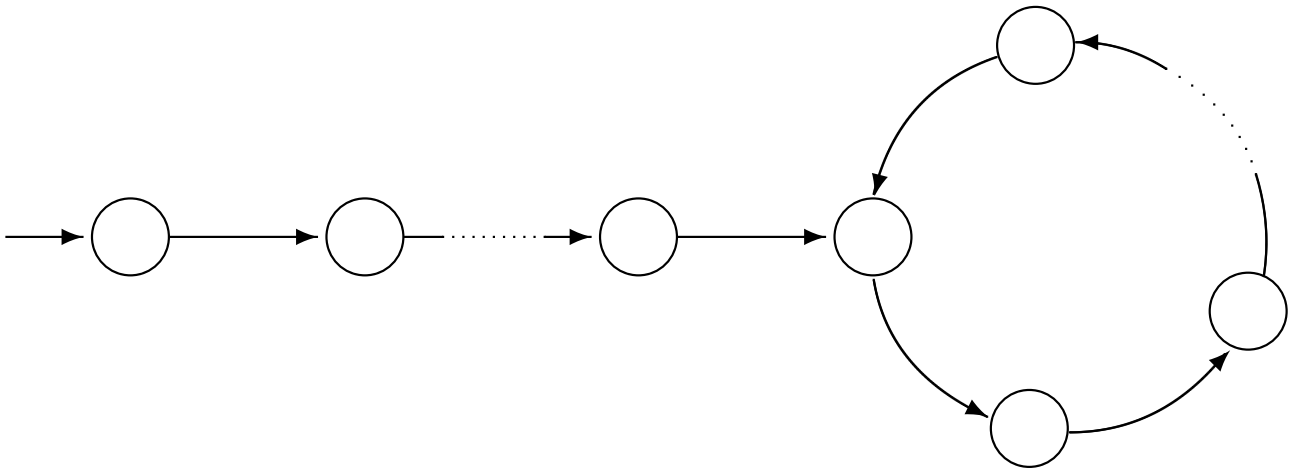
$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ s.t.

$$\forall n \geq \mu : a^n \in L \text{ iff } a^{n+\lambda} \in L.$$

Special case $\mu = 0$: the language is *periodic* or *cyclic*.

Unary automata: 1dfa

Input alphabet $\Sigma = \{a\}$



Theorem

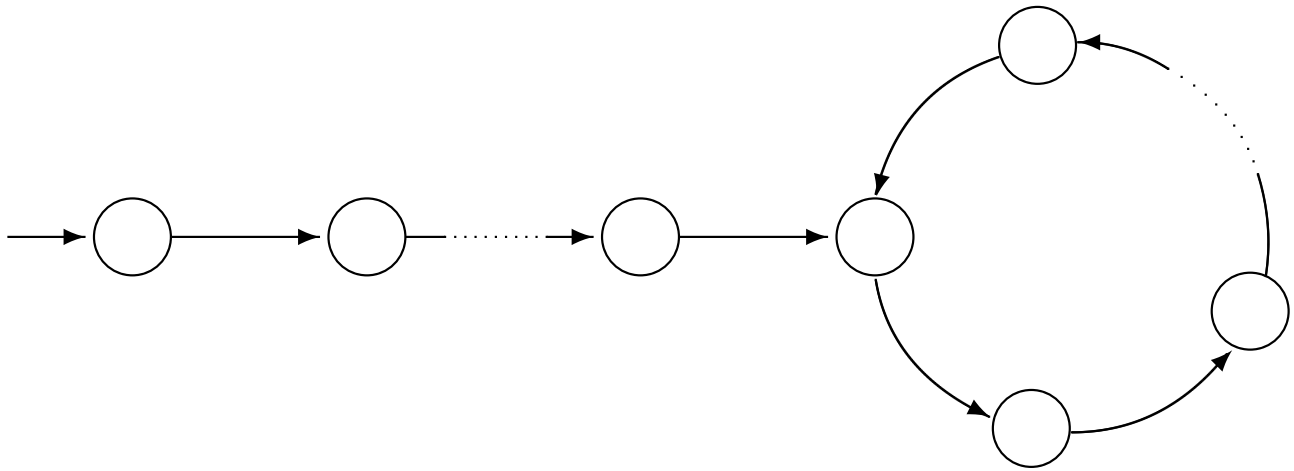
$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ s.t.

$$\forall n \geq \mu : a^n \in L \text{ iff } a^{n+\lambda} \in L.$$

Special case $\mu = 0$: the language is *periodic* or *cyclic*.

Unary automata: 1dfa

Input alphabet $\Sigma = \{a\}$



Theorem

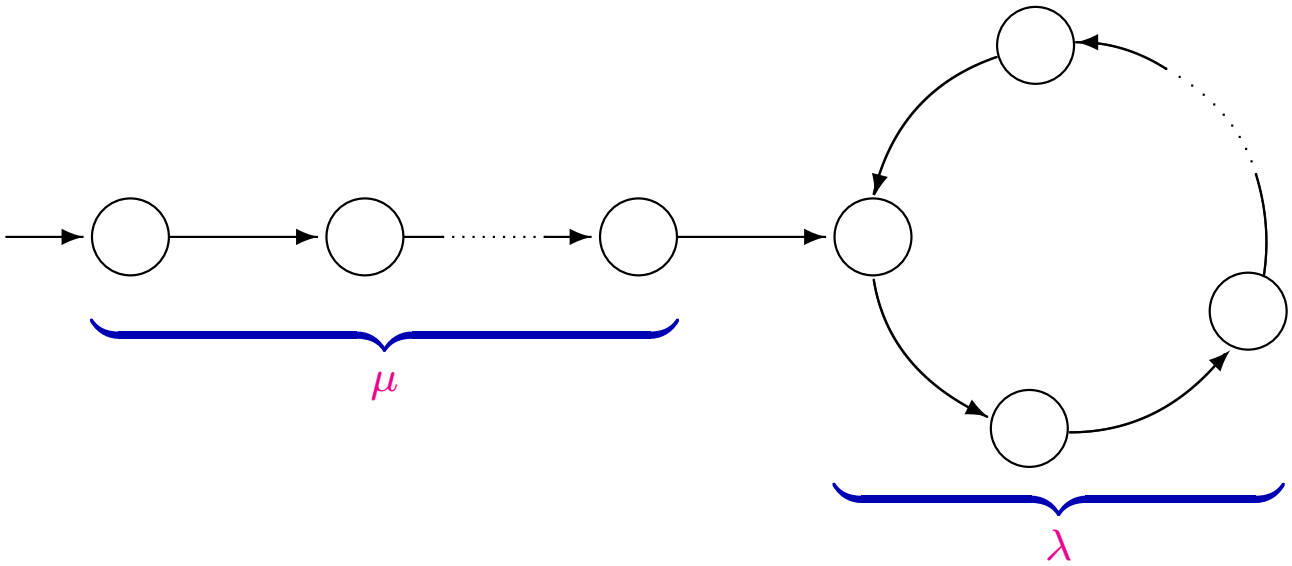
$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ s.t.

$$\forall n \geq \mu : a^n \in L \text{ iff } a^{n+\lambda} \in L.$$

Special case $\mu = 0$: the language is *periodic* or *cyclic*.

Unary automata: 1dfa

Input alphabet $\Sigma = \{a\}$



Theorem

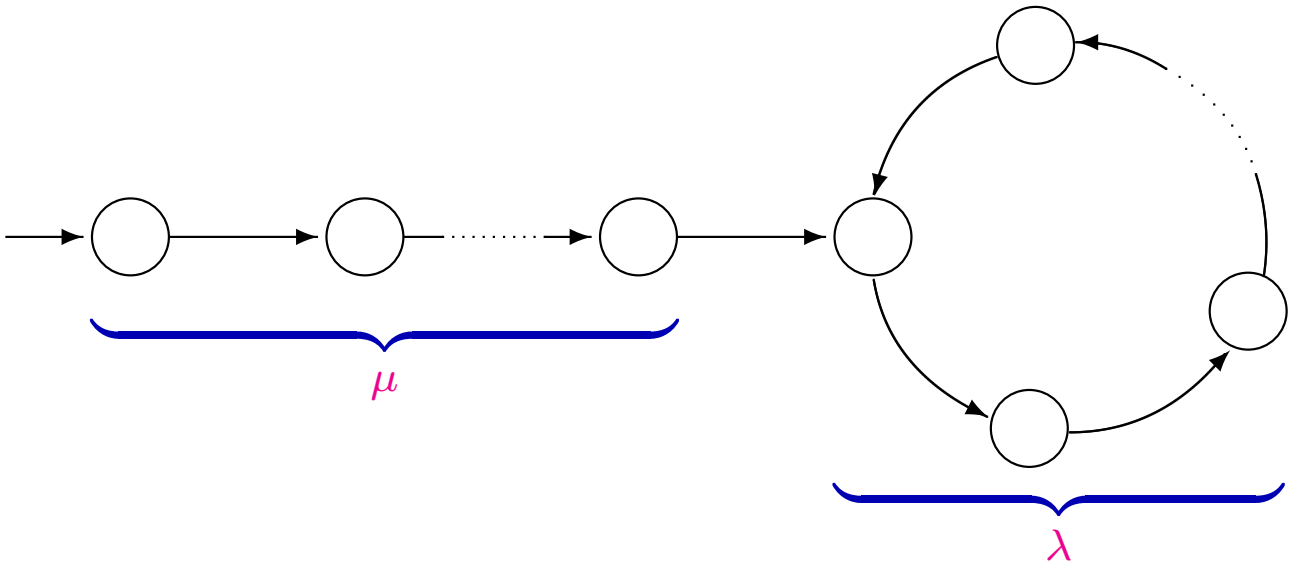
$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ s.t.

$$\forall n \geq \mu : a^n \in L \text{ iff } a^{n+\lambda} \in L.$$

Special case $\mu = 0$: the language is *periodic* or *cyclic*.

Unary automata: 1dfa

Input alphabet $\Sigma = \{a\}$



Theorem

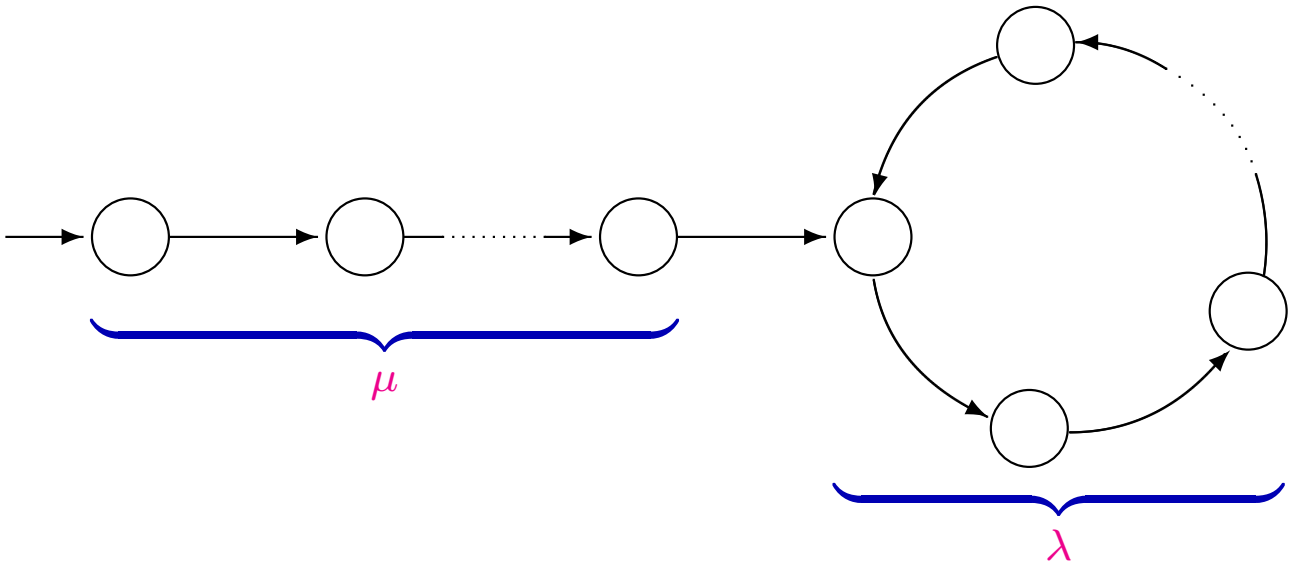
$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ s.t.

$$\forall n \geq \mu : a^n \in L \text{ iff } a^{n+\lambda} \in L.$$

Special case $\mu = 0$: the language is *periodic* or *cyclic*.

Unary automata: 1dfa

Input alphabet $\Sigma = \{a\}$



Theorem

$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ s.t.

$$\forall n \geq \mu : a^n \in L \text{ iff } a^{n+\lambda} \in L.$$

Special case $\mu = 0$: the language is *periodic* or *cyclic*.

Unary automata: 1nfa

The transition graph describing the automaton can have a whatever structure, however...

...we can restrict to 1nfa with the following form *(Chrobak normal form)*:

- an initial path
- a nondeterministic choice
- a set of cycles

Unary automata: 1nfa

The transition graph describing the automaton can have a whatever structure, however...

...we can restrict to 1nfa with the following form *(Chrobak normal form)*:

- an initial path
- a nondeterministic choice
- a set of cycles

Unary automata: 1nfa

The transition graph describing the automaton can have a whatever structure, however...

...we can restrict to 1nfa with the following form *(Chrobak normal form)*:

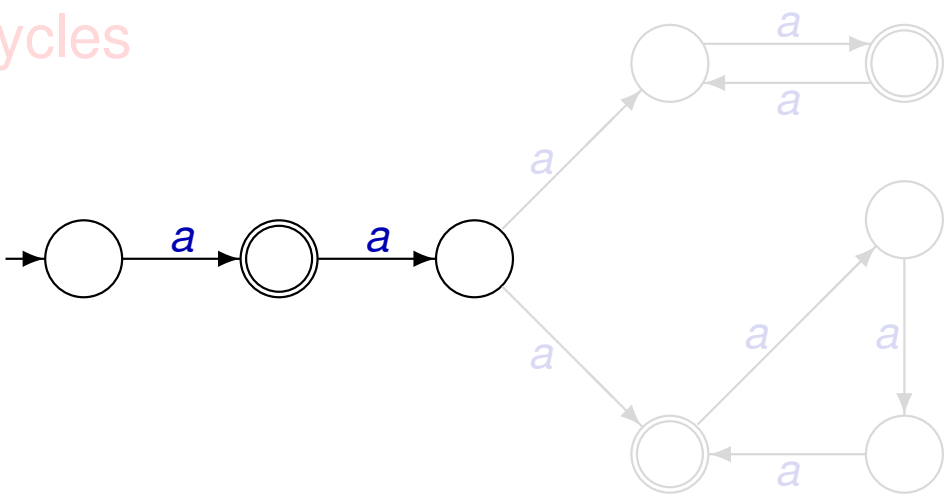
- an initial path
- a nondeterministic choice
- a set of cycles

Unary automata: 1nfa

The transition graph describing the automaton can have a whatever structure, however...

...we can restrict to 1nfa with the following form *(Chrobak normal form)*:

- an initial path
- a nondeterministic choice
- a set of cycles

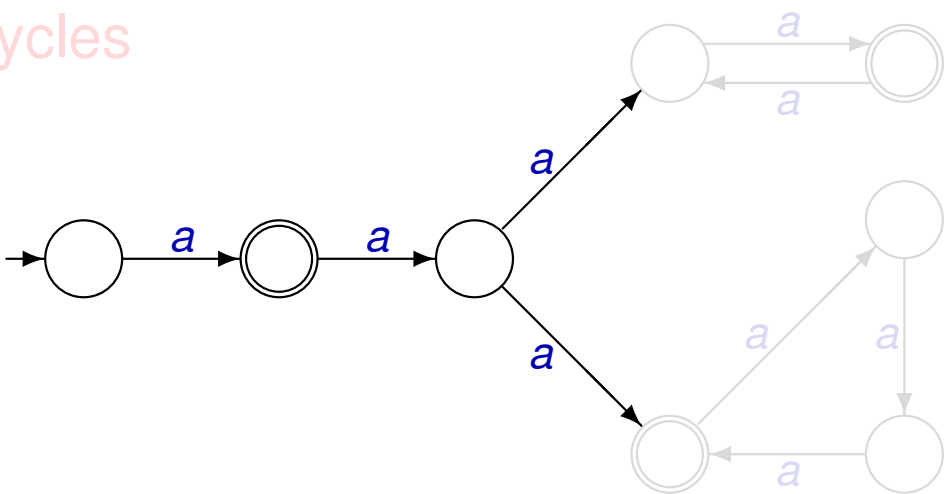


Unary automata: 1nfa

The transition graph describing the automaton can have a whatever structure, however...

...we can restrict to 1nfa with the following form *(Chrobak normal form)*:

- an initial path
- a nondeterministic choice
- a set of cycles

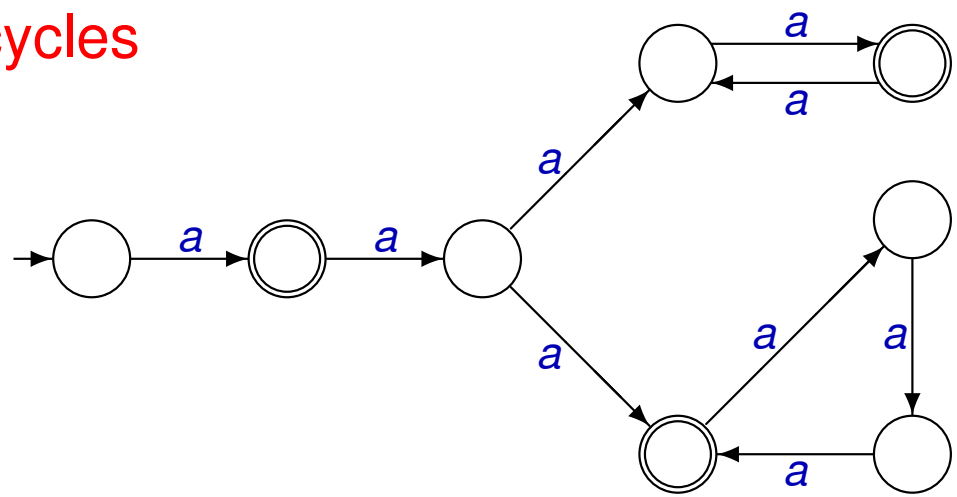


Unary automata: 1nfa

The transition graph describing the automaton can have a whatever structure, however...

...we can restrict to 1nfa with the following form *(Chrobak normal form)*:

- an initial path
- a nondeterministic choice
- a set of cycles



Unary automata: Chrobak normal form

Theorem

For any unary n-state 1nfa there exists an equivalent 1nfa in Chrobak normal form s.t.

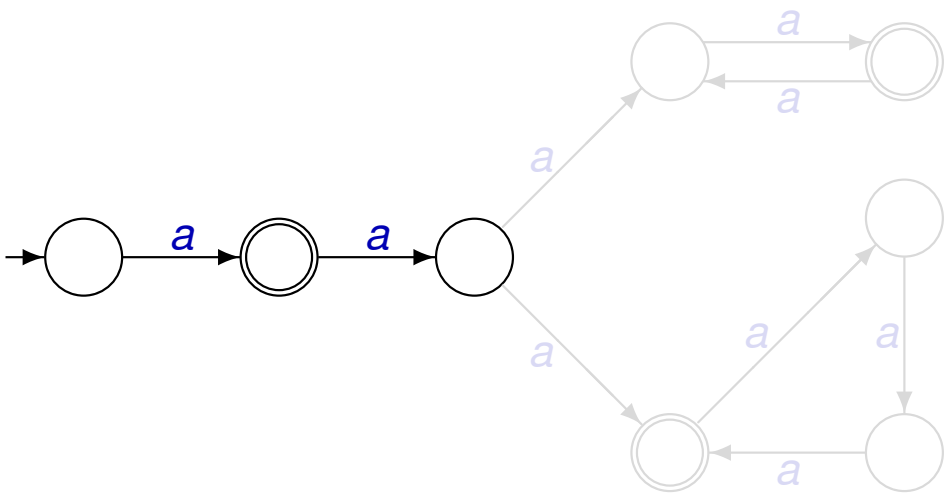
- *there are $O(n^2)$ states on the initial path*
- *the total number of states on the cycles is at most n*

Unary automata: Chrobak normal form

Theorem

For any unary n-state 1nfa there exists an equivalent 1nfa in Chrobak normal form s.t.

- *there are $O(n^2)$ states on the initial path*
- *the total number of states on the cycles is at most n*

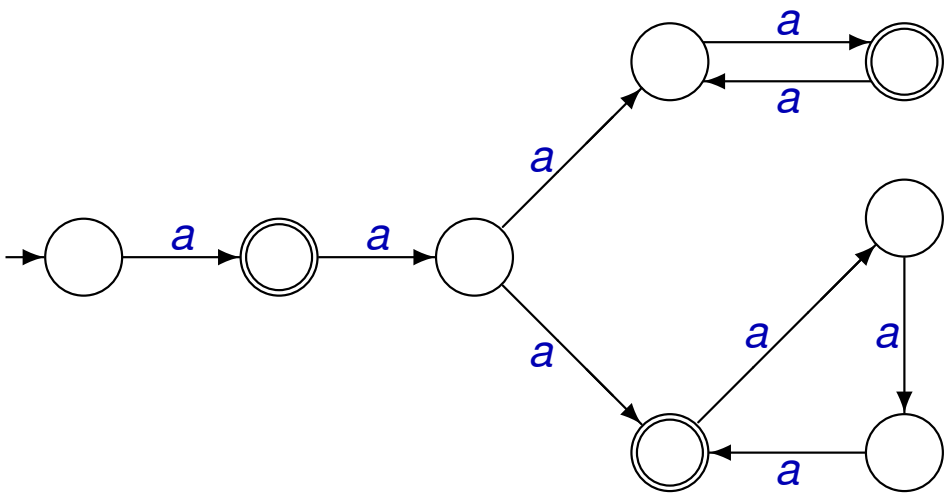


Unary automata: Chrobak normal form

Theorem

For any unary n-state 1nfa there exists an equivalent 1nfa in Chrobak normal form s.t.

- *there are $O(n^2)$ states on the initial path*
- *the total number of states on the cycles is at most n*



Unary nondeterministic automata

How to eliminate the nondeterminism from a unary 1nfa?

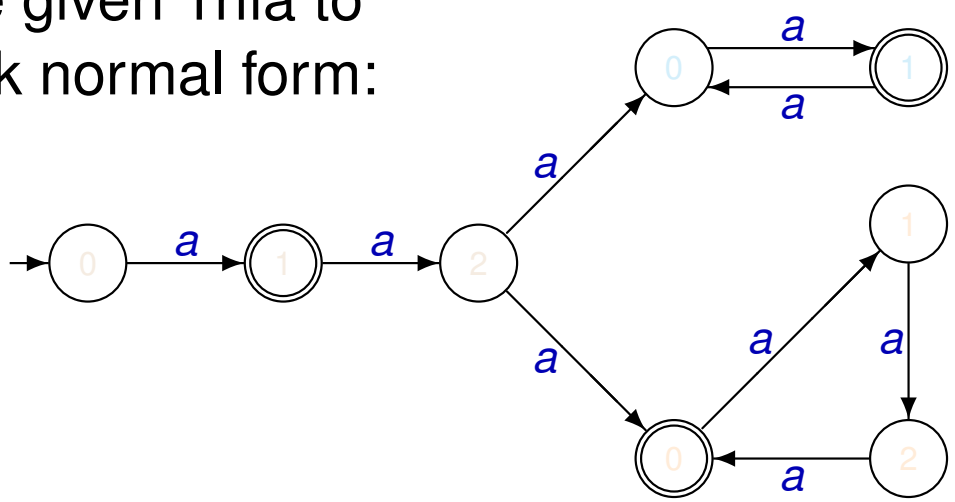
1. Convert the given 1nfa to the Chrobak normal form:

2. Copy the initial path
3. Replace the set of cycles with a unique cycle:

Unary nondeterministic automata

How to eliminate the nondeterminism from a unary 1nfa?

1. Convert the given 1nfa to the Chrobak normal form:

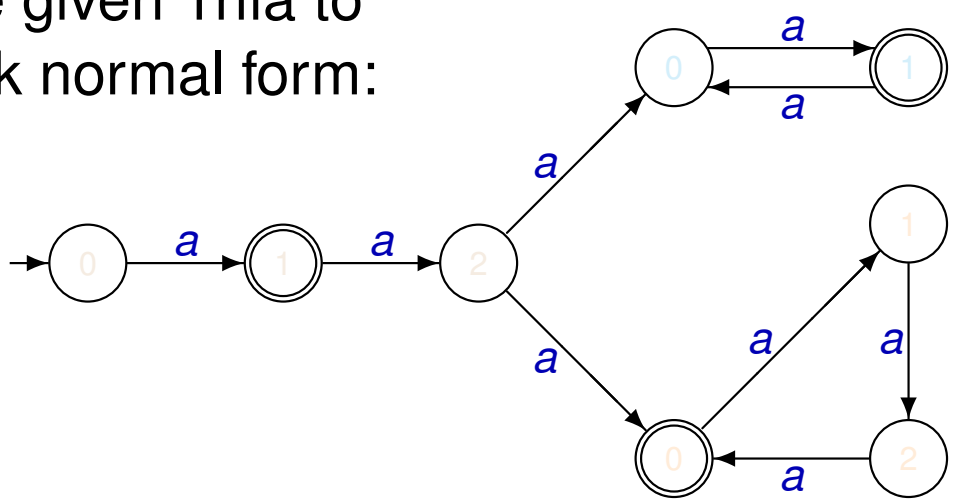


2. Copy the initial path
3. Replace the set of cycles with a unique cycle:

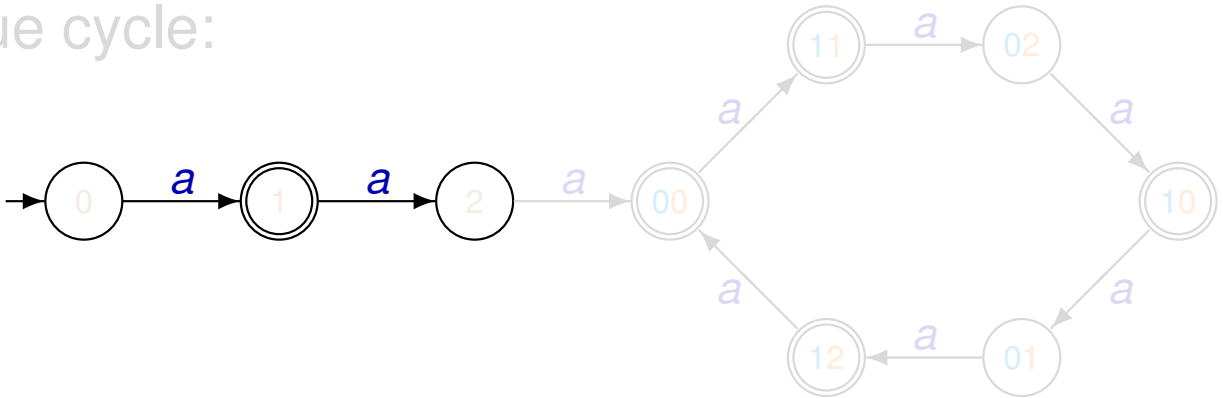
Unary nondeterministic automata

How to eliminate the nondeterminism from a unary 1nfa?

1. Convert the given 1nfa to the Chrobak normal form:



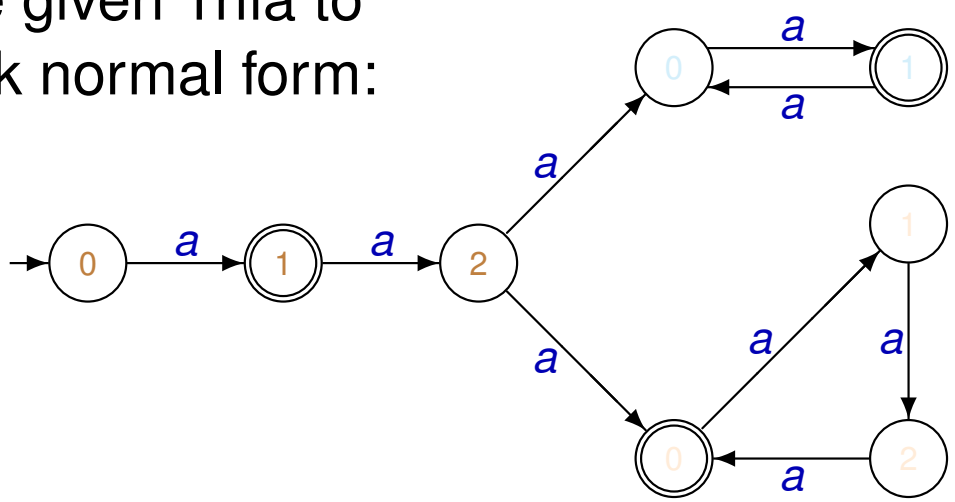
2. Copy the initial path
3. Replace the set of cycles with a unique cycle:



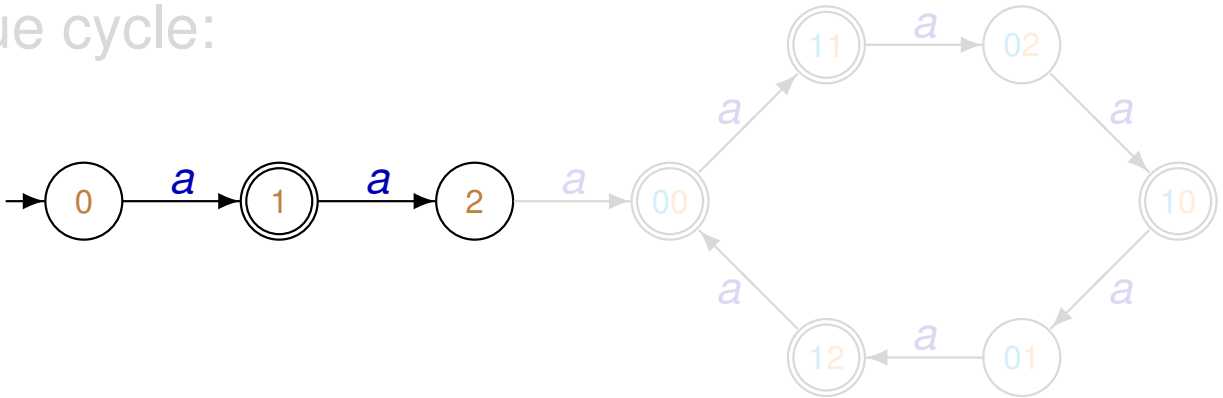
Unary nondeterministic automata

How to eliminate the nondeterminism from a unary 1nfa?

1. Convert the given 1nfa to the Chrobak normal form:



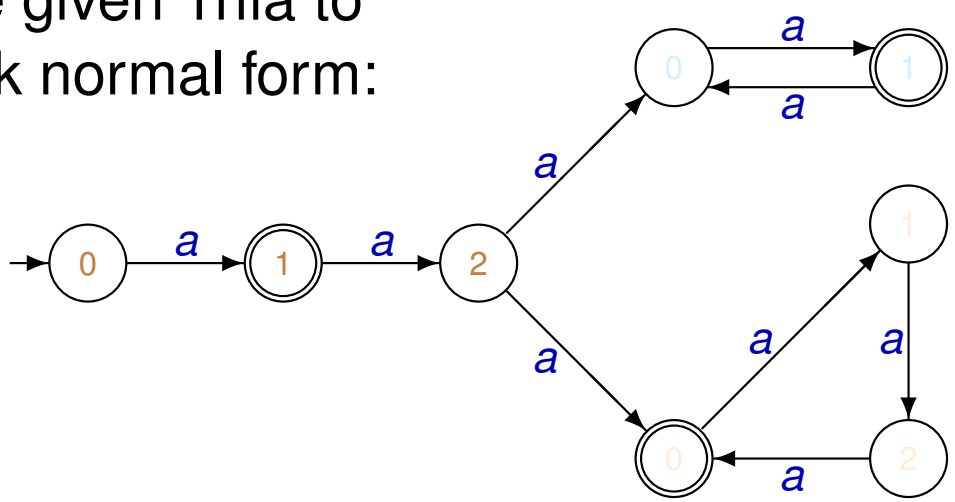
2. Copy the initial path
3. Replace the set of cycles with a unique cycle:



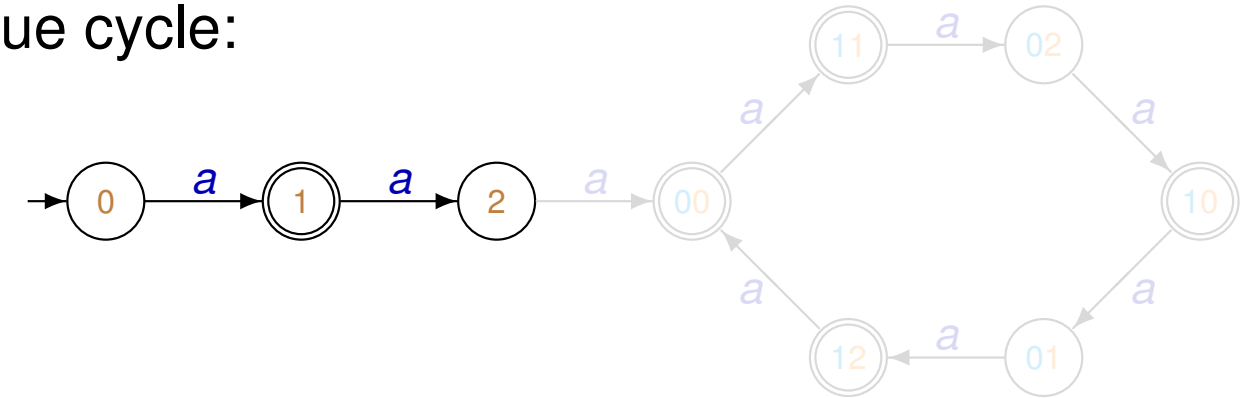
Unary nondeterministic automata

How to eliminate the nondeterminism from a unary 1nfa?

1. Convert the given 1nfa to the Chrobak normal form:



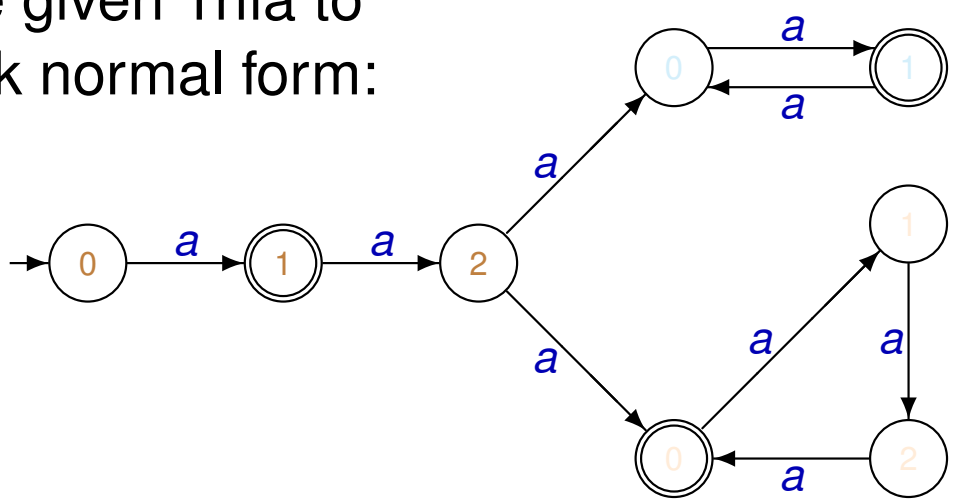
2. Copy the initial path
3. Replace the set of cycles with a unique cycle:



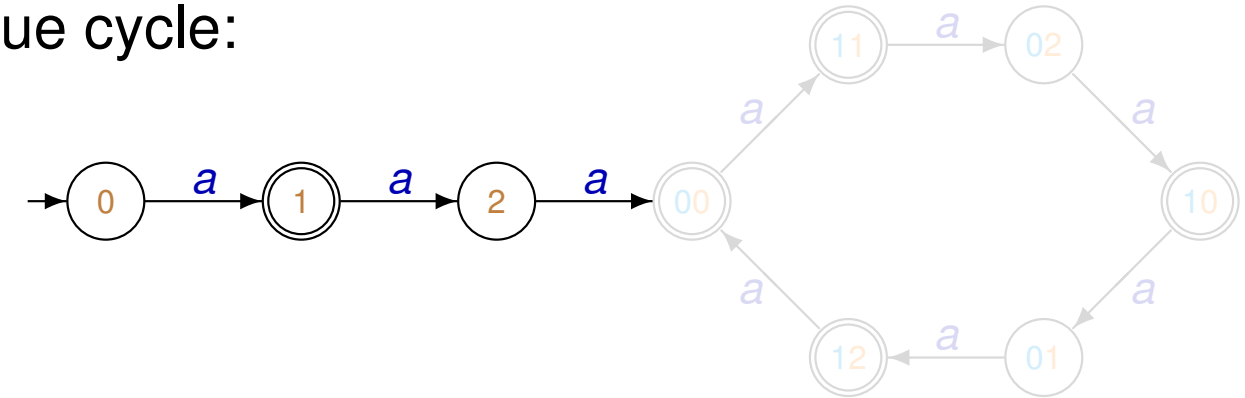
Unary nondeterministic automata

How to eliminate the nondeterminism from a unary 1nfa?

1. Convert the given 1nfa to the Chrobak normal form:



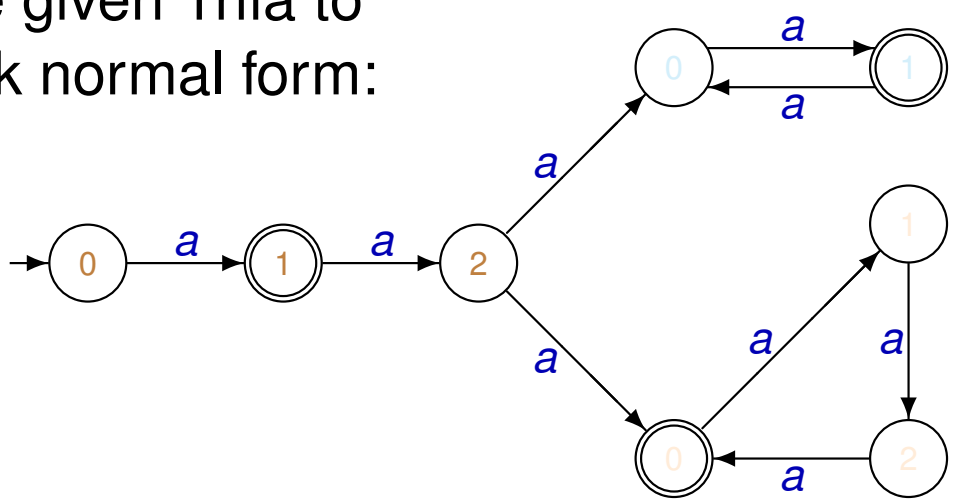
2. Copy the initial path
3. Replace the set of cycles with a unique cycle:



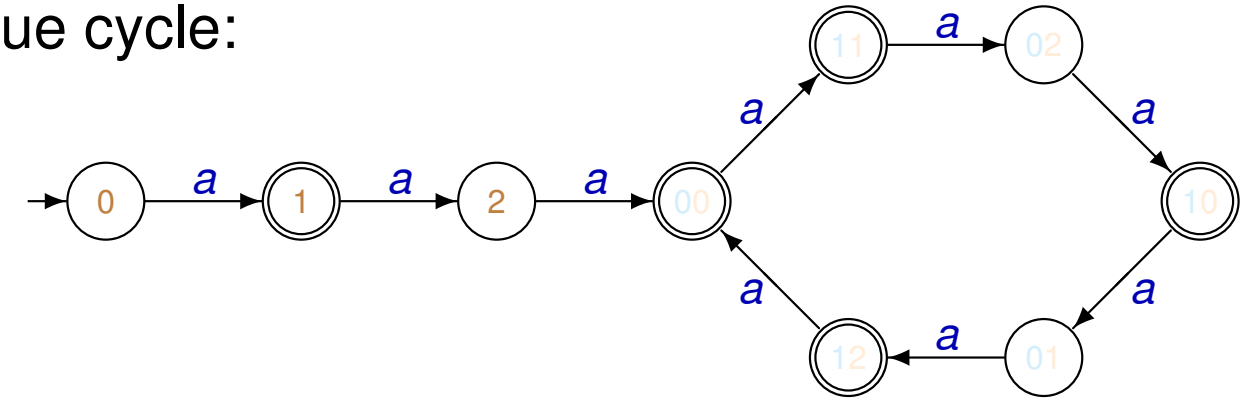
Unary nondeterministic automata

How to eliminate the nondeterminism from a unary 1nfa?

1. Convert the given 1nfa to the Chrobak normal form:



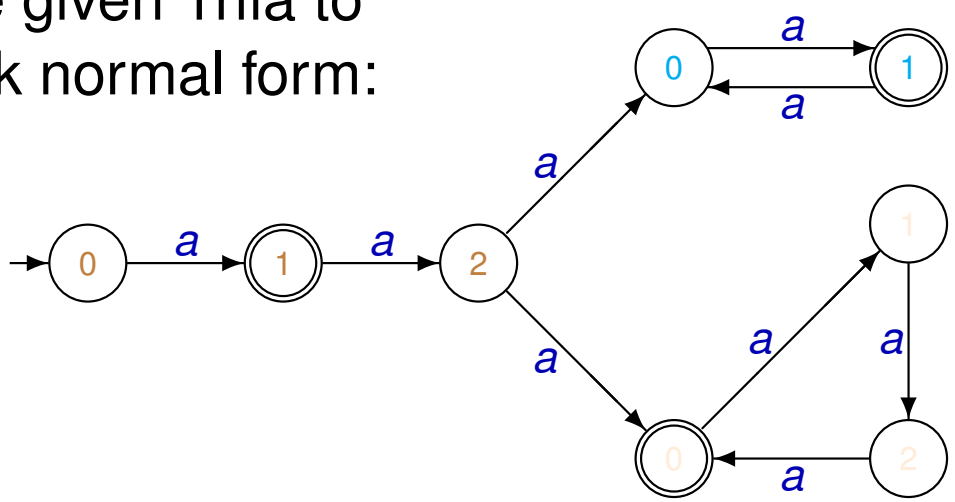
2. Copy the initial path
3. Replace the set of cycles with a unique cycle:



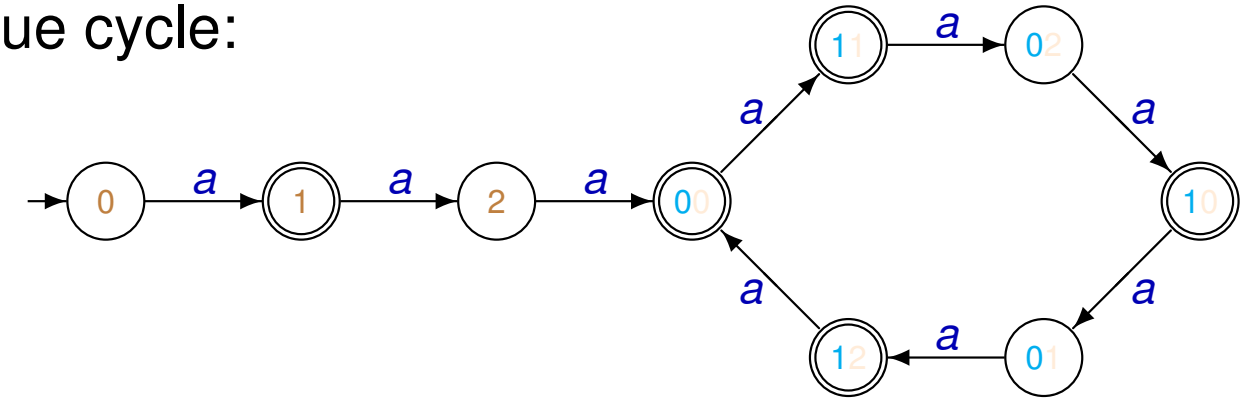
Unary nondeterministic automata

How to eliminate the nondeterminism from a unary 1nfa?

1. Convert the given 1nfa to the Chrobak normal form:



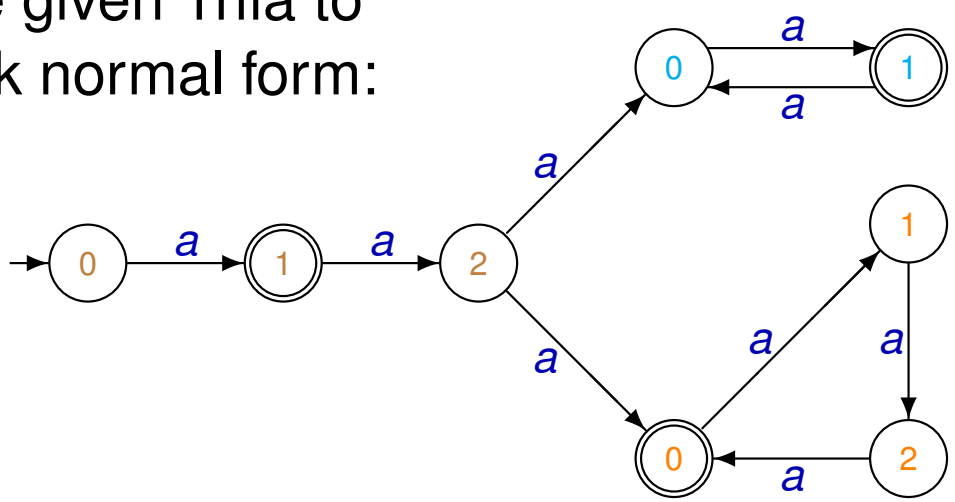
2. Copy the initial path
3. Replace the set of cycles with a unique cycle:



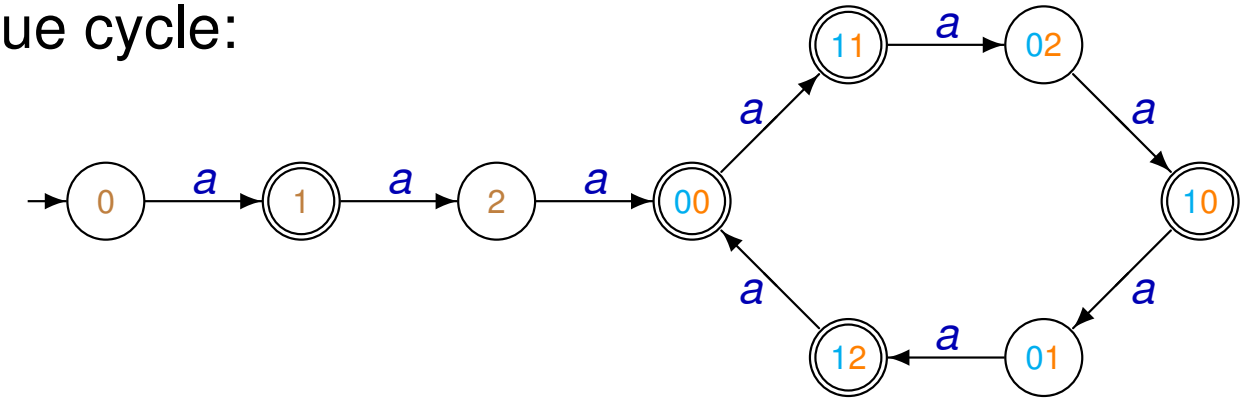
Unary nondeterministic automata

How to eliminate the nondeterminism from a unary 1nfa?

1. Convert the given 1nfa to the Chrobak normal form:



2. Copy the initial path
3. Replace the set of cycles with a unique cycle:



Simulation of unary 1nfa by 1dfa

Given an n-state 1nfa:

- Convert it in Chrobak normal form:
  - Initial path of $O(n^2)$ states
  - Cycles of lenghts $\lambda_1, \ldots, \lambda_k$, with $\lambda_1 + \ldots + \lambda_k \leq n$
- The cycles are replaced by a unique cycle of length $\mathrm{lcm}(\lambda_1, \ldots, \lambda_k)$

Hence:

- The number of states in the resulting cycle is bouned by $F(n) = \max\{\mathrm{lcm}(x_1, \ldots, x_k) \mid x_1 + \ldots + x_k = n\}$

and:

- $F(n) = e^{O(\sqrt{n \log n})}$ [Landau 1903]

Theorem ([Chrobak 1986])

Each unary n-state 1nfa can be simulated by a 1dfa with $e^{O(\sqrt{n \log n})}$ states.

Simulation of unary 1nfa by 1dfa

Given an n-state 1nfa:

- Convert it in Chrobak normal form:
  - Initial path of $O(n^2)$ states
  - Cycles of lenghts $\lambda_1, \ldots, \lambda_k$, with $\lambda_1 + \ldots + \lambda_k \leq n$
- The cycles are replaced by a unique cycle of length $\mathrm{lcm}(\lambda_1, \ldots, \lambda_k)$

Hence:

- The number of states in the resulting cycle is bouned by $F(n) = \max\{\mathrm{lcm}(x_1, \ldots, x_k) \mid x_1 + \ldots + x_k = n\}$

and:

- $F(n) = e^{O(\sqrt{n \log n})}$ [Landau 1903]

Theorem ([Chrobak 1986])

Each unary n-state 1nfa can be simulated by a 1dfa with $e^{O(\sqrt{n \log n})}$ states.

Simulation of unary 1nfa by 1dfa

Given an n-state 1nfa:

- Convert it in Chrobak normal form:
  - Initial path of $O(n^2)$ states
  - Cycles of lenghts $\lambda_1, \ldots, \lambda_k$, with $\lambda_1 + \ldots + \lambda_k \leq n$
- The cycles are replaced by a unique cycle of length $\mathrm{lcm}(\lambda_1, \ldots, \lambda_k)$

Hence:

- The number of states in the resulting cycle is bouned by $F(n) = \max\{\mathrm{lcm}(x_1, \ldots, x_k) \mid x_1 + \ldots + x_k = n\}$

and:

- $F(n) = e^{O(\sqrt{n \log n})}$ [Landau 1903]

Theorem ([Chrobak 1986])

Each unary n-state 1nfa can be simulated by a 1dfa with $e^{O(\sqrt{n \log n})}$ states.

Simulation of unary 1nfa by 1dfa

Given an n-state 1nfa:

- Convert it in Chrobak normal form:
  - Initial path of $O(n^2)$ states
  - Cycles of lenghts $\lambda_1, \ldots, \lambda_k$, with $\lambda_1 + \ldots + \lambda_k \leq n$
- The cycles are replaced by a unique cycle of length $\mathrm{lcm}(\lambda_1, \ldots, \lambda_k)$

Hence:

- The number of states in the resulting cycle is bouned by $F(n) = \max\{\mathrm{lcm}(x_1, \ldots, x_k) \mid x_1 + \ldots + x_k = n\}$

and:

- $F(n) = e^{O(\sqrt{n \log n})}$ [Landau 1903]

Theorem ([Chrobak 1986])

Each unary n-state 1nfa can be simulated by a 1dfa with $e^{O(\sqrt{n \log n})}$ states.

Simulation of unary 1nfa by 1dfa

Given an n-state 1nfa:

- Convert it in Chrobak normal form:
  - Initial path of $O(n^2)$ states
  - Cycles of lenghts $\lambda_1, \ldots, \lambda_k$, with $\lambda_1 + \ldots + \lambda_k \leq n$
- The cycles are replaced by a unique cycle of length $\mathrm{lcm}(\lambda_1, \ldots, \lambda_k)$

Hence:

- The number of states in the resulting cycle is bouned by $F(n) = \max\{\mathrm{lcm}(x_1, \ldots, x_k) \mid x_1 + \ldots + x_k = n\}$

and:

- $F(n) = e^{O(\sqrt{n \log n})}$ [Landau 1903]

Theorem ([Chrobak 1986])

Each unary n-state 1nfa can be simulated by a 1dfa with $e^{O(\sqrt{n \log n})}$ states.

Simulation of unary 1nfa by 1dfa

Given an n-state 1nfa:

- Convert it in Chrobak normal form:
  - Initial path of $O(n^2)$ states
  - Cycles of lenghts $\lambda_1, \ldots, \lambda_k$, with $\lambda_1 + \ldots + \lambda_k \leq n$
- The cycles are replaced by a unique cycle of length $\mathrm{lcm}(\lambda_1, \ldots, \lambda_k)$

Hence:

- The number of states in the resulting cycle is bouned by $F(n) = \max\{\mathrm{lcm}(x_1, \ldots, x_k) \mid x_1 + \ldots + x_k = n\}$

and:

- $F(n) = e^{O(\sqrt{n \log n})}$ [Landau 1903]

Theorem ([Chrobak 1986])

Each unary n-state 1nfa can be simulated by a 1dfa with $e^{O(\sqrt{n \log n})}$ states.

Simulation of unary 1nfa by 1dfa

Given an n-state 1nfa:

- Convert it in Chrobak normal form:
  - Initial path of $O(n^2)$ states
  - Cycles of lenghts $\lambda_1, \ldots, \lambda_k$, with $\lambda_1 + \ldots + \lambda_k \leq n$
- The cycles are replaced by a unique cycle of length $\mathrm{lcm}(\lambda_1, \ldots, \lambda_k)$

Hence:

- The number of states in the resulting cycle is bouned by $F(n) = \max\{\mathrm{lcm}(x_1, \ldots, x_k) \mid x_1 + \ldots + x_k = n\}$

and:

- $F(n) = e^{O(\sqrt{n \log n})}$ [Landau 1903]

Theorem ([Chrobak 1986])

Each unary n-state 1nfa can be simulated by a 1dfa with $e^{O(\sqrt{n \log n})}$ states.

Simulation of unary 1nfa by 1dfa

Given an n-state 1nfa:

- Convert it in Chrobak normal form:
  - Initial path of $O(n^2)$ states
  - Cycles of lenghts $\lambda_1, \ldots, \lambda_k$, with $\lambda_1 + \ldots + \lambda_k \leq n$
- The cycles are replaced by a unique cycle of length $\mathrm{lcm}(\lambda_1, \ldots, \lambda_k)$

Hence:

- The number of states in the resulting cycle is bouned by $F(n) = \max\{\mathrm{lcm}(x_1, \ldots, x_k) \mid x_1 + \ldots + x_k = n\}$

and:

- $F(n) = e^{O(\sqrt{n \log n})}$ [Landau 1903]

Theorem ([Chrobak 1986])

Each unary n-state 1nfa can be simulated by a 1dfa with $e^{O(\sqrt{n \log n})}$ states.

Simulation of unary 1nfa by 1dfa

Given an n-state 1nfa:

- Convert it in Chrobak normal form:
  - Initial path of $O(n^2)$ states
  - Cycles of lenghts $\lambda_1, \ldots, \lambda_k$, with $\lambda_1 + \ldots + \lambda_k \leq n$
- The cycles are replaced by a unique cycle of length $\mathrm{lcm}(\lambda_1, \ldots, \lambda_k)$

Hence:

- The number of states in the resulting cycle is bouned by $F(n) = \max\{\mathrm{lcm}(x_1, \ldots, x_k) \mid x_1 + \ldots + x_k = n\}$

and:

- $F(n) = e^{O(\sqrt{n \log n})}$ [Landau 1903]

Theorem ([Chrobak 1986])

Each unary n-state 1nfa can be simulated by a 1dfa with $e^{O(\sqrt{n \log n})}$ states.

Simulation of unary 1nfa by 1dfa

Given an n-state 1nfa:

- Convert it in Chrobak normal form:
  - Initial path of $O(n^2)$ states
  - Cycles of lenghts $\lambda_1, \ldots, \lambda_k$, with $\lambda_1 + \ldots + \lambda_k \leq n$
- The cycles are replaced by a unique cycle of length $\mathrm{lcm}(\lambda_1, \ldots, \lambda_k)$

Hence:

- The number of states in the resulting cycle is bouned by $F(n) = \max\{\mathrm{lcm}(x_1, \ldots, x_k) \mid x_1 + \ldots + x_k = n\}$

and:

- $F(n) = e^{O(\sqrt{n \log n})}$ [Landau 1903]

Theorem ([Chrobak 1986])

Each unary n-state 1nfa can be simulated by a 1dfa with $e^{O(\sqrt{n \log n})}$ states.

Unary automata

The costs of the optimal simulations between automata are different in the unary and in the general case!

Unary case: [Chrobak 1986, Mereghetti and Pighizzini 2001]

1dfa 1nfa

2dfa 2nfa

Unary automata

The costs of the optimal simulations between automata are different in the unary and in the general case!

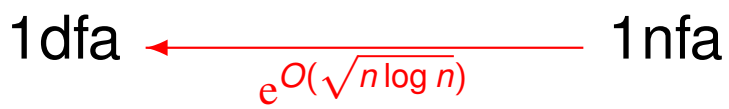
Unary case: [Chrobak 1986, Mereghetti and Pighizzini 2001]

1dfa	1nfa
2dfa	2nfa

Unary automata

The costs of the optimal simulations between automata are different in the unary and in the general case!

Unary case: [Chrobak 1986, Mereghetti and Pighizzini 2001]

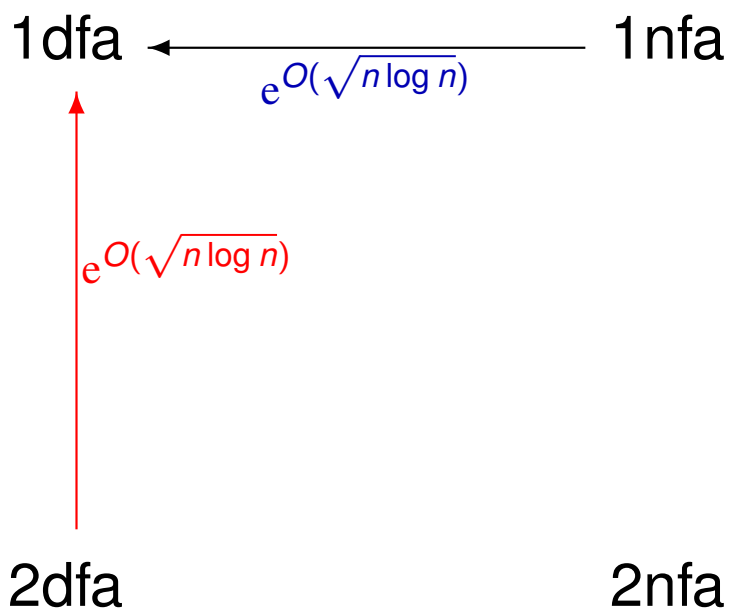


2dfa 2nfa

Unary automata

The costs of the optimal simulations between automata are different in the unary and in the general case!

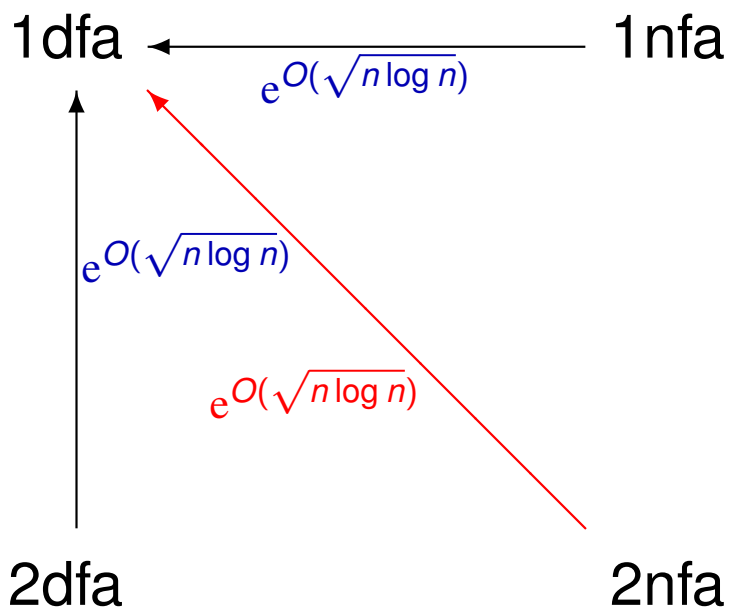
Unary case: [Chrobak 1986, Mereghetti and Pighizzini 2001]



Unary automata

The costs of the optimal simulations between automata are different in the unary and in the general case!

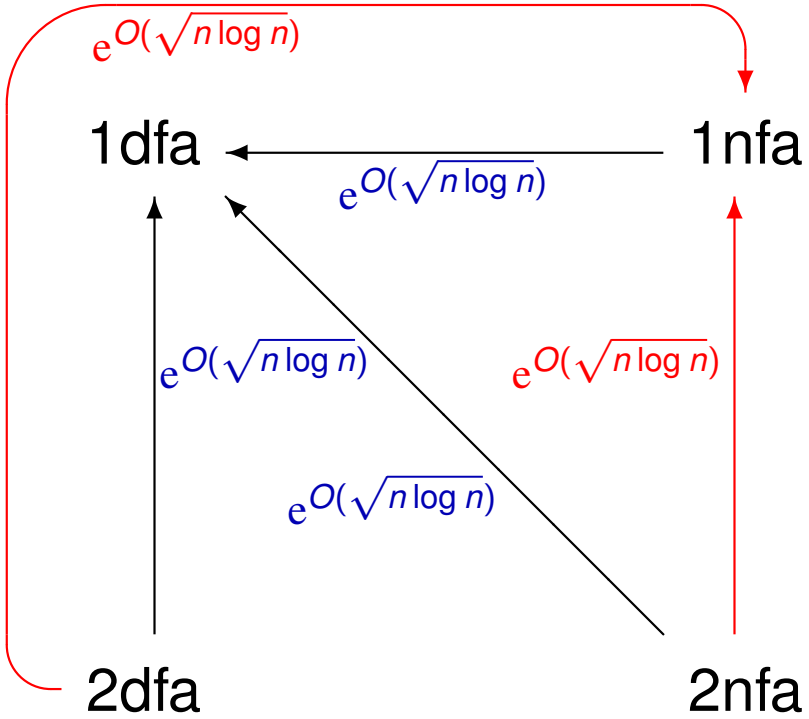
Unary case: [Chrobak 1986, Mereghetti and Pighizzini 2001]



Unary automata

The costs of the optimal simulations between automata are different in the unary and in the general case!

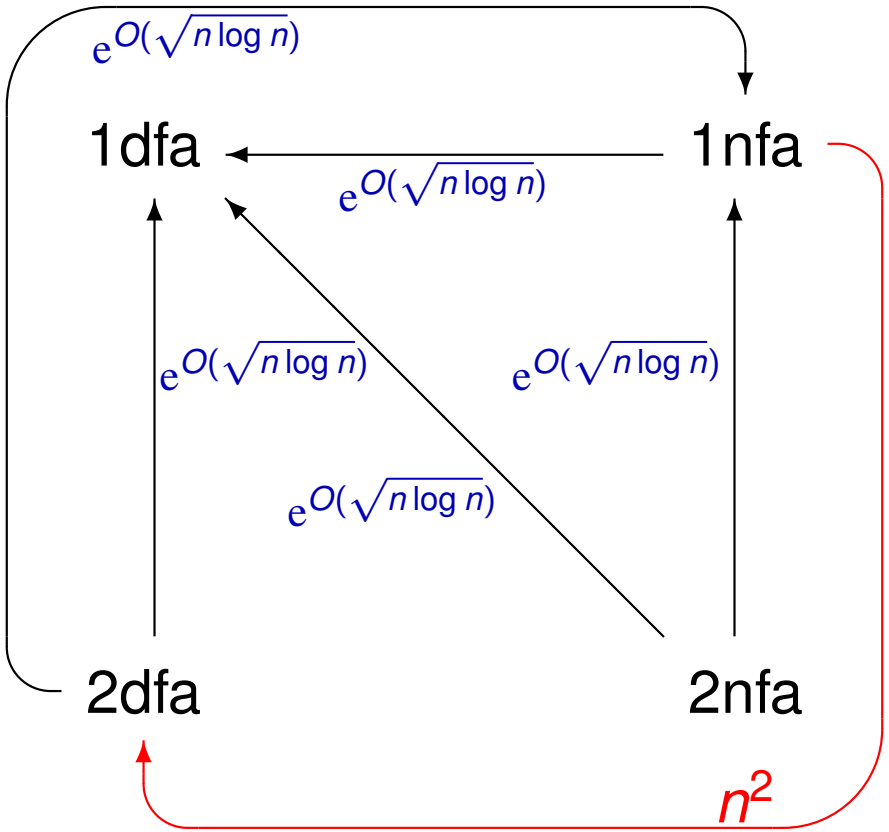
Unary case: [Chrobak 1986, Mereghetti and Pighizzini 2001]



Unary automata

The costs of the optimal simulations between automata are different in the unary and in the general case!

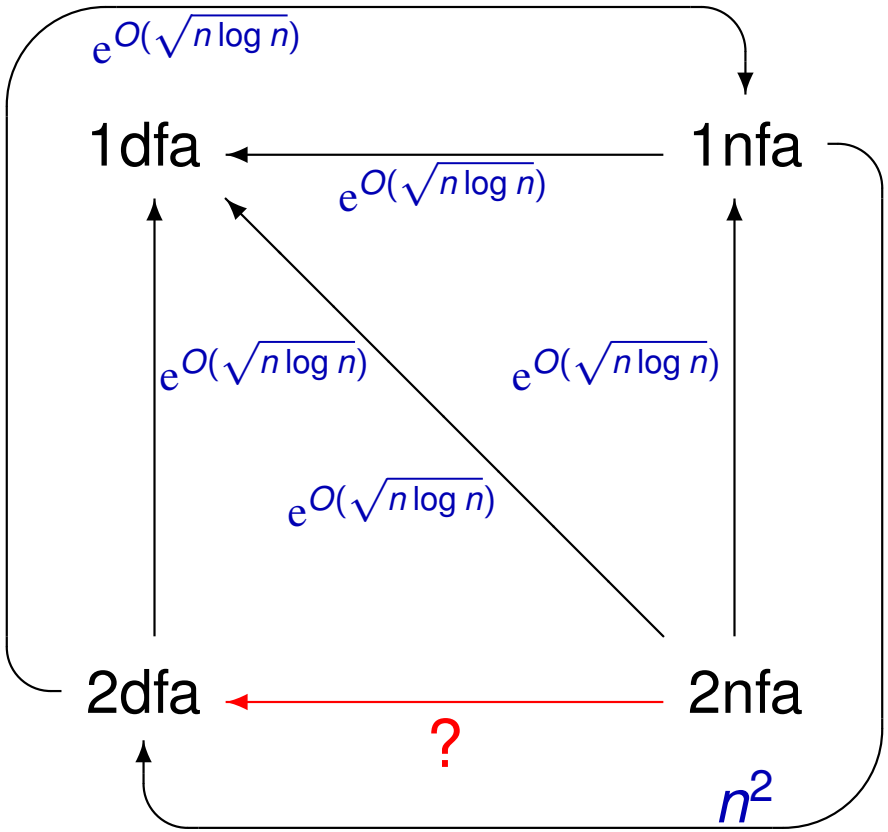
Unary case: [Chrobak 1986, Mereghetti and Pighizzini 2001]



Unary automata

The costs of the optimal simulations between automata are different in the unary and in the general case!

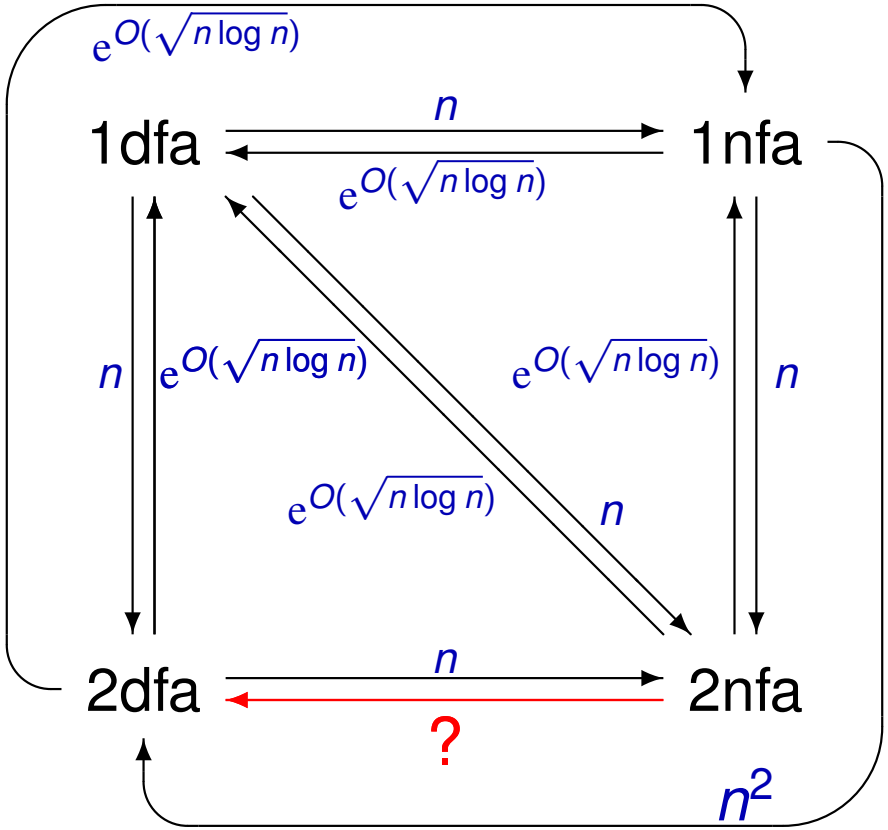
Unary case: [Chrobak 1986, Mereghetti and Pighizzini 2001]



Unary automata

The costs of the optimal simulations between automata are different in the unary and in the general case!

Unary case: [Chrobak 1986, Mereghetti and Pighizzini 2001]



Descriptional complexity of regular languages

- Different variant of finite automata characterize regular languages.
- However, we can describe regular languages using more powerful formalisms or devices, as context-free grammars and pushdown automata.

What about the sizes of cfg's or pda's describing regular languages vs the sizes of finite automata?

Descriptional complexity of regular languages

- Different variant of finite automata characterize regular languages.
- However, we can describe regular languages using more powerful formalisms or devices, as context-free grammars and pushdown automata.

What about the sizes of cfg's or pda's describing regular languages vs the sizes of finite automata?

Descriptional complexity of regular languages

- Different variant of finite automata characterize regular languages.
- However, we can describe regular languages using more powerful formalisms or devices, as context-free grammars and pushdown automata.

What about the sizes of cfg's or pda's describing regular languages vs the sizes of finite automata?

Descriptional complexity measures

- Context-free grammars:
  number of variables?

  For $n \geq 1$, consider the language $L_n = (a^n)^*$:
  - L_n requires n states to be accepted by dfa or nfa
  - L_n is generated by the grammar with one variable S and the productions

$$S \to a^n \qquad S \to a^n S \qquad S \to \epsilon$$

Thus, the number of variables cannot be a descriptional complexity measure for context-free grammars.
However, for grammars in *Chomsky Normal Form* the number of variables is a "reasonable" measure of complexity [Gruska, 1973].

Descriptional complexity measures

- Context-free grammars:
  number of variables?

  For $n \geq 1$, consider the language $L_n = (a^n)^*$:
  - L_n requires n states to be accepted by dfa or nfa
  - L_n is generated by the grammar with one variable S and the productions

$$S \to a^n \qquad S \to a^n S \qquad S \to \epsilon$$

  Thus, the number of variables cannot be a descriptional complexity measure for context-free grammars.
  However, for grammars in *Chomsky Normal Form* the number of variables is a "reasonable" measure of complexity [Gruska, 1973].

Descriptional complexity measures

- Context-free grammars:
  number of variables?

  For $n \geq 1$, consider the language $L_n = (a^n)^*$:
  - L_n requires n states to be accepted by dfa or nfa
  - L_n is generated by the grammar with one variable S and the productions

$$S \to a^n \qquad S \to a^n S \qquad S \to \epsilon$$

  Thus, the number of variables cannot be a descriptional complexity measure for context-free grammars.
  However, for grammars in *Chomsky Normal Form* the number of variables is a "reasonable" measure of complexity [Gruska, 1973].

Descriptional complexity measures

- Context-free grammars:
  number of variables?

  For $n \geq 1$, consider the language $L_n = (a^n)^*$:
  - L_n requires n states to be accepted by dfa or nfa
  - L_n is generated by the grammar with one variable S and the productions

$$S \to a^n \qquad S \to a^n S \qquad S \to \epsilon$$

  Thus, the number of variables cannot be a descriptional complexity measure for context-free grammars.
  However, for grammars in *Chomsky Normal Form* the number of variables is a "reasonable" measure of complexity [Gruska, 1973].

Descriptional complexity measures

- Context-free grammars:
  number of variables?

  For $n \geq 1$, consider the language $L_n = (a^n)^*$:
  - L_n requires n states to be accepted by dfa or nfa
  - L_n is generated by the grammar with one variable S and the productions

$$S \to a^n \qquad S \to a^n S \qquad S \to \epsilon$$

Thus, the number of variables cannot be a descriptional complexity measure for context-free grammars.
However, for grammars in *Chomsky Normal Form* the number of variables is a "reasonable" measure of complexity [Gruska, 1973].

Descriptional complexity measures

- Context-free grammars:
  number of variables?

  For $n \geq 1$, consider the language $L_n = (a^n)^*$:
  - L_n requires n states to be accepted by dfa or nfa
  - L_n is generated by the grammar with one variable S and the productions

$$S \to a^n \qquad S \to a^n S \qquad S \to \epsilon$$

Thus, the number of variables cannot be a descriptional complexity measure for context-free grammars.
However, for grammars in *Chomsky Normal Form* the number of variables is a "reasonable" measure of complexity [Gruska, 1973].

Descriptional complexity measures

- Context-free grammars:
  number of variables?

  For $n \geq 1$, consider the language $L_n = (a^n)^*$:
  - L_n requires n states to be accepted by dfa or nfa
  - L_n is generated by the grammar with one variable S and the productions

$$S \to a^n \qquad S \to a^n S \qquad S \to \epsilon$$

Thus, the number of variables cannot be a descriptional complexity measure for context-free grammars.
However, for grammars in *Chomsky Normal Form* the number of variables is a "reasonable" measure of complexity [Gruska, 1973].

Descriptional complexity measures

- Context-free grammars:
  number of variables?

  For $n \geq 1$, consider the language $L_n = (a^n)^*$:
  - L_n requires n states to be accepted by dfa or nfa
  - L_n is generated by the grammar with one variable S and the productions

$$S \to a^n \qquad S \to a^n S \qquad S \to \epsilon$$

Thus, the number of variables cannot be a descriptional complexity measure for context-free grammars.
However, for grammars in *Chomsky Normal Form* the number of variables is a "reasonable" measure of complexity [Gruska, 1973].

Descriptional complexity measures

- Pushdown automata:
  W.l.o.g., we can consider only pda's s.t. push operations add exactly one symbol on the pushdown store.

  The total size of the description of a pda satisfying this restriction is a polynomial function of two parameters:
  - the number of the states
  - the cardinality of the pushdown alphabet.

Descriptional complexity measures

- Pushdown automata:

  W.l.o.g., we can consider only pda's s.t. push operations add exactly one symbol on the pushdown store.

  The total size of the description of a pda satisfying this restriction is a polynomial function of two parameters:

  - the number of the states
  - the cardinality of the pushdown alphabet.

Context-free vs regular: descriptional complexity

Given a context-free grammar (or a pushdown automaton) of size n, generating a regular language, how much is big an equivalent finite automaton, wrt n ?

Theorem ([Meyer and Fischer, 1971])

For any recursive function f and arbitrarily large integers n, there exists a cfg G of size n generating a regular language L, s.t. any dfa accepting L must have at least $f(n)$ states.

As a consequence, the trade-off between context-grammars and finite automata is not recursive.
However... The witness language L is defined over a unary alphabet.

Context-free vs regular: descriptional complexity

Given a context-free grammar (or a pushdown automaton) of size n, generating a regular language, how much is big an equivalent finite automaton, wrt n ?

Theorem ([Meyer and Fischer, 1971])

For any recursive function f and arbitrarily large integers n, there exists a cfg G of size n generating a regular language L, s.t. any dfa accepting L must have at least $f(n)$ states.

As a consequence, the trade-off between context-grammars and finite automata is not recursive.
However... The witness language L is defined over a binary alphabet.

Context-free vs regular: descriptional complexity

Given a context-free grammar (or a pushdown automaton) of size n, generating a regular language, how much is big an equivalent finite automaton, wrt n ?

Theorem ([Meyer and Fischer, 1971])

For any recursive function f and arbitrarily large integers n, there exists a cfg G of size n generating a regular language L, s.t. any dfa accepting L must have at least $f(n)$ states.

As a consequence, the trade-off between context-grammars and finite automata is not recursive.
However... The witness language L is defined over a binary alphabet.

What about languages over a one letter alphabet?

Context-free vs regular: descriptional complexity

Given a context-free grammar (or a pushdown automaton) of size n, generating a regular language, how much is big an equivalent finite automaton, wrt n ?

Theorem ([Meyer and Fischer, 1971])

For any recursive function f and arbitrarily large integers n, there exists a cfg G of size n generating a regular language L, s.t. any dfa accepting L must have at least $f(n)$ states.

As a consequence, the trade-off between context-grammars and finite automata is not recursive.
However... The witness language L is defined over a binary alphabet.

What about languages over a one letter alphabet?

Context-free vs regular: descriptional complexity

Given a context-free grammar (or a pushdown automaton) of size n, generating a regular language, how much is big an equivalent finite automaton, wrt n ?

Theorem ([Meyer and Fischer, 1971])

For any recursive function f and arbitrarily large integers n, there exists a cfg G of size n generating a regular language L, s.t. any dfa accepting L must have at least $f(n)$ states.

As a consequence, the trade-off between context-grammars and finite automata is not recursive.
However... The witness language L is defined over a binary alphabet.

What about languages over a one letter alphabet?

Unary languages

$\Sigma = \{a\}$

Theorem ([Ginsurg and Rice, 1962])

Every unary context-free language is regular.

Hence the classes of unary regular languages and unary *context-free* languages coincide!

Problem

Study the equivalence between unary context-free and regular languages from the descriptional complexity point of view.

Unary languages

$\Sigma = \{a\}$

Theorem ([Ginsurg and Rice, 1962])

Every unary context-free language is regular.

Hence the classes of unary regular languages and unary *context-free* languages coincide!

Problem

Study the equivalence between unary context-free and regular languages from the descriptional complexity point of view.

Unary cfg's vs finite automata

The following bounds have been proved in [Pighizzini, Shallit, Wang, 2002]:

Given a unary cfg in Chomsky normal form with h variables, there exist:

- an equivalent 1nfa with at most $2^{2h-1} + 1$ states
- an equivalent 1dfa with at most 2^{h^2} states

These bounds are tight, namely, matching lower bound have been discovered.

Unary cfg's vs finite automata

The following bounds have been proved in [Pighizzini, Shallit, Wang, 2002]:

Given a unary cfg in Chomsky normal form with h variables, there exist:

- an equivalent 1nfa with at most $2^{2^{h-1}} + 1$ states
- an equivalent 1dfa with at most 2^{h^2} states

These bounds are tight, namely, matching lower bound have been discovered.

Unary cfg's vs finite automata

The following bounds have been proved in [Pighizzini, Shallit, Wang, 2002]:

Given a unary cfg in Chomsky normal form with h variables, there exist:

- an equivalent 1nfa with at most $2^{2h-1} + 1$ states
- an equivalent 1dfa with at most 2^{h^2} states

These bounds are tight, namely, matching lower bound have been discovered.

Unary cfg's vs finite automata

The following bounds have been proved in [Pighizzini, Shallit, Wang, 2002]:

Given a unary cfg in Chomsky normal form with h variables, there exist:

- an equivalent 1nfa with at most $2^{2h-1} + 1$ states
- an equivalent 1dfa with at most 2^{h^2} states

These bounds are tight, namely, matching lower bound have been discovered.

Extension to bounded languages

Bounded languages:

Subsets of $w_1^* w_2^* \dots w_n^*$, for given words $w_1, \dots, w_n$ (*letter bounded* if $w_1, \dots, w_n \in \Sigma$).

- The class bounded regular languages is *properly* included in that of bounded cfl's, e.g., $\{a^n b^n \mid n \geq 0\}$.
- Bounded cfl's can be accepted by *finite-turn* pda's.

Theorem (Maleher, Pighizzini, 2007)

Each bounded context-free language generated by a cfg with h variables in Chomsky normal form is accepted by a finite-turn pda with 2^h states and $O(1)$ stack symbols.

Even this upper bound is tight.

Extension to bounded languages

Bounded languages:
Subsets of $w_1^* w_2^* \dots w_n^*$, for given words $w_1, \dots, w_n$
(*letter bounded* if $w_1, \dots, w_n \in \Sigma$).

- The class bounded regular languages is *properly* included in that of bounded cfl's, e.g., $\{a^n b^n \mid n \geq 0\}$.
- Bounded cfl's can be accepted by *finite turn* pda's.

Theorem ([Malcher, Pighizzini, 2007])

Each bounded context-free language generated by a cfg with h variables in Chomsky normal form is accepted by a finite-turn pda with 2^h states and $O(1)$ stack symbols.

Even this upper bound is tight!

Extension to bounded languages

Bounded languages:
Subsets of $w_1^* w_2^* \dots w_n^*$, for given words $w_1, \dots, w_n$
(*letter bounded* if $w_1, \dots, w_n \in \Sigma$).

- The class bounded regular languages is *properly* included in that of bounded cfl's, e.g., $\{a^n b^n \mid n \geq 0\}$.
- Bounded cfl's can be accepted by *finite turn* pda's.

Theorem ([Malcher, Pighizzini, 2007])

Each bounded context-free language generated by a cfg with h variables in Chomsky normal form is accepted by a finite-turn pda with 2^h states and $O(1)$ stack symbols.

Even this upper bound is tight!

Extension to bounded languages

Bounded languages:
Subsets of $w_1^* w_2^* \dots w_n^*$, for given words $w_1, \dots, w_n$
(*letter bounded* if $w_1, \dots, w_n \in \Sigma$).

- The class bounded regular languages is *properly* included in that of bounded cfl's, e.g., $\{a^n b^n \mid n \geq 0\}$.
- Bounded cfl's can be accepted by *finite turn* pda's.

Theorem ([Malcher, Pighizzini, 2007])

Each bounded context-free language generated by a cfg with h variables in Chomsky normal form is accepted by a finite-turn pda with 2^h states and $O(1)$ stack symbols.

Even this upper bound is tight!

Extension to bounded languages

Bounded languages:
Subsets of $w_1^* w_2^* \dots w_n^*$, for given words $w_1, \dots, w_n$
(*letter bounded* if $w_1, \dots, w_n \in \Sigma$).

- The class bounded regular languages is *properly* included in that of bounded cfl's, e.g., $\{a^n b^n \mid n \geq 0\}$.
- Bounded cfl's can be accepted by *finite turn* pda's.

Theorem ([Malcher, Pighizzini, 2007])

Each bounded context-free language generated by a cfg with h variables in Chomsky normal form is accepted by a finite-turn pda with 2^h states and $O(1)$ stack symbols.

Even this upper bound is tight!

Unary dpda's vs finite automata

Let M be a unary pda with n states and m stack symbols, s.t. each push adds exactly one symbol.

Using the simulation of unary cfg's by finite automata and the standard transformation of pda's into cfg's, it can be shown that M can be simulated by a 1dfa with with $2^{O(n^4 m^2)}$ states.

What about the deterministic case?

Theorem ([Pighizzini, 2008])

If M is deterministic then it can be simulated by a 1dfa with $2^{O(nm)}$ states.
Furthermore, such a simulation is tight.
Its cost cannot be reduced even if we simulate M by a 2nfa.

Unary dpda's vs finite automata

Let M be a unary pda with n states and m stack symbols, s.t. each push adds exactly one symbol.

Using the simulation of unary cfg's by finite automata and the standard transformation of pda's into cfg's, it can be shown that M can be simulated by a 1dfa with with $2^{O(n^4 m^2)}$ states.

What about the deterministic case?

Theorem ([Pighizzini, 2008])

If M is deterministic then it can be simulated by a 1dfa with $2^{O(nm)}$ states.
Furthermore, such a simulation is tight.
Its cost cannot be reduced even if we simulate M by a 2nfa.

Unary dpda's vs finite automata

Let M be a unary pda with n states and m stack symbols, s.t. each push adds exactly one symbol.

Using the simulation of unary cfg's by finite automata and the standard transformation of pda's into cfg's, it can be shown that M can be simulated by a 1dfa with with $2^{O(n^4 m^2)}$ states.

What about the deterministic case?

Theorem ([Pighizzini, 2008])

If M is deterministic then it can be simulated by a 1dfa with $2^{O(nm)}$ states.
Furthermore, such a simulation is tight.
Its cost cannot be reduced even if we simulate M by a 2nfa.

Languages with "complex" dpda's

Unary dpda's can be exponentially more succinct than dfa's.
Does this is true for *each* unary regular language?

Problem

For $m \geq 0$, let $L_m \subseteq a^*$ be a language accepted by a dfa with 2^m states.
Does there exists an equivalent dpda with $O(m)$ states?

The answer to this question is negative: [Pighizzini, 2008]

For each $m > 0$ there exists a language $L_m \subseteq a^*$ s.t.:

- L_m is accepted by a dfa with 2^m states.
- The size of any dpda accepting L_m is at least $d\frac{2^m}{m^2}$, for a constant d.

Languages with "complex" dpda's

Unary dpda's can be exponentially more succinct than dfa's.
Does this is true for *each* unary regular language?

Problem

For $m \geq 0$, let $L_m \subseteq a^*$ be a language accepted by a dfa with 2^m states.
Does there exists an equivalent dpda with $O(m)$ states?

The answer to this question is negative: [Pighizzini, 2008]

For each $m > 0$ there exists a language $L_m \subseteq a^*$ s.t.:

- L_m is accepted by a dfa with 2^m states.
- The size of any dpda accepting L_m is at least $d\frac{2^m}{m^2}$, for a constant d.

Languages with "complex" dpda's

Unary dpda's can be exponentially more succinct than dfa's.
Does this is true for *each* unary regular language?

Problem

For $m \geq 0$, let $L_m \subseteq a^*$ be a language accepted by a dfa with 2^m states.
Does there exists an equivalent dpda with $O(m)$ states?

The answer to this question is negative: [Pighizzini, 2008]

For each $m > 0$ there exists a language $L_m \subseteq a^*$ s.t.:

- L_m is accepted by a dfa with 2^m states.
- The size of any dpda accepting L_m is at least $d\frac{2^m}{m^2}$, for a constant d.

State complexity of language operations

- Let L_1 and L_2 two regular languages accepted by two automata with s_1 and s_2 states, and op a binary operation preserving the regularity.
- State a tight upper bound $f_{op}(s_1, s_2)$ for the number of the states of an automaton accepting the language $L_1 \, op \, L_2$.
- A same question can be formulated for other kinds of operations.

These problems have been extensively studied in the case of one-way deterministic and nondeterministic automata for "classical" regular operations.

There are interesting results and problems for the complement of languages in the case of nondeterministic machines.

State complexity of language operations

- Let L_1 and L_2 two regular languages accepted by two automata with s_1 and s_2 states, and op a binary operation preserving the regularity.
- State a tight upper bound $f_{op}(s_1, s_2)$ for the number of the states of an automaton accepting the language $L_1 \, op \, L_2$.
- A same question can be formulated for other kinds of operations.

These problems have been extensively studied in the case of one-way deterministic and nondeterministic automata for "classical" regular operations.

There are interesting results and problems for the complement of languages in the case of nondeterministic machines.

State complexity of language operations

- Let L_1 and L_2 two regular languages accepted by two automata with s_1 and s_2 states, and op a binary operation preserving the regularity.
- State a tight upper bound $f_{op}(s_1, s_2)$ for the number of the states of an automaton accepting the language $L_1 \, op \, L_2$.
- A same question can be formulated for other kinds of operations.

These problems have been extensively studied in the case of one-way deterministic and nondeterministic automata for "classical" regular operations.

There are interesting results and problems for the complement of languages in the case of nondeterministic machines.

State complexity of language operations

- Let L_1 and L_2 two regular languages accepted by two automata with s_1 and s_2 states, and op a binary operation preserving the regularity.
- State a tight upper bound $f_{op}(s_1, s_2)$ for the number of the states of an automaton accepting the language $L_1 \, op \, L_2$.
- A same question can be formulated for other kinds of operations.

These problems have been extensively studied in the case of one-way deterministic and nondeterministic automata for "classical" regular operations.

There are interesting results and problems for the complement of languages in the case of nondeterministic machines.

State complexity of language operations

- Let L_1 and L_2 two regular languages accepted by two automata with s_1 and s_2 states, and op a binary operation preserving the regularity.
- State a tight upper bound $f_{op}(s_1, s_2)$ for the number of the states of an automaton accepting the language $L_1 \, op \, L_2$.
- A same question can be formulated for other kinds of operations.

These problems have been extensively studied in the case of one-way deterministic and nondeterministic automata for "classical" regular operations.

There are interesting results and problems for the complement of languages in the case of nondeterministic machines.

Complementation of two-way automata

Problem

Given a two-way automaton with n states accepting a language L, find the cost in term of states, of an automaton accepting the complement of L.

- Deterministic case:
  The cost is $4n$, for *any input alphabet*.
  [Geffert, Mereghetti, Pighizzini 2007]
  (note that 2dfa's can have infinite computations)
- Nondeterministic case:
  The cost is polynomial for a *unary alphabet*.
  [Geffert, Mereghetti, Pighizzini 2007]

What about the complementation of 2nfa over *nonunary* alphabets?

Complementation of two-way automata

Problem

Given a two-way automaton with n states accepting a language L, find the cost in term of states, of an automaton accepting the complement of L.

- Deterministic case:
  The cost is $4n$, for *any input alphabet.*
  [Geffert, Mereghetti, Pighizzini 2007]
  (note that 2dfa's can have infinite computations)
- Nondeterministic case:
  The cost is polynomial for a *unary alphabet.*
  [Geffert, Mereghetti, Pighizzini 2007]

What about the complementation of 2nfa over *nonunary* alphabets?

Complementation of two-way automata

Problem

Given a two-way automaton with n states accepting a language L, find the cost in term of states, of an automaton accepting the complement of L.

- Deterministic case:
  The cost is $4n$, for *any input alphabet.*
  [Geffert, Mereghetti, Pighizzini 2007]
  (note that 2dfa's can have infinite computations)
- Nondeterministic case:
  The cost is polynomial for a *unary alphabet.*
  [Geffert, Mereghetti, Pighizzini 2007]

What about the complementation of 2nfa over *nonunary* alphabets?

Complementation of two-way automata

Problem

Given a two-way automaton with n states accepting a language L, find the cost in term of states, of an automaton accepting the complement of L.

- Deterministic case:
  The cost is $4n$, for *any input alphabet.*
  [Geffert, Mereghetti, Pighizzini 2007]
  (note that 2dfa's can have infinite computations)
- Nondeterministic case:
  The cost is polynomial for a *unary alphabet.*
  [Geffert, Mereghetti, Pighizzini 2007]

What about the complementation of 2nfa over *nonunary* alphabets?

Complementation of two-way automata

Relationship with the Sakoda&Sipser question

$f(n) :=$ the cost of the simulation of a n-state 2nfa by a 2dfa.

Given an n-state 2nfa accepting L we can find:

- a 2dfa accepting L with $f(n)$ states

Complementation of two-way automata

Relationship with the Sakoda&Sipser question

$f(n) :=$ the cost of the simulation of a n-state 2nfa by a 2dfa.

Given an n-state 2nfa accepting L we can find:

- a 2dfa accepting L with $f(n)$ states
- a 2nfa (actually a 2dfa) accepting L^R with $4f(n)$ states

Complementation of two-way automata

Relationship with the Sakoda&Sipser question

$f(n) :=$ the cost of the simulation of a n-state 2nfa by a 2dfa.

Given an n-state 2nfa accepting L we can find:

- a 2dfa accepting L with $f(n)$ states
- a 2nfa (actually a 2dfa) accepting L^c with $4f(n)$ states

Hence:

the complementation of 2nfa's costs no more than the determinization.

Theorem

If the complementation of 2nfa's requires an exponential number of states then the gap between 2nfa's and 2dfa's is exponential.

Complementation of two-way automata

Relationship with the Sakoda&Sipser question

$f(n) :=$ the cost of the simulation of a n-state 2nfa by a 2dfa.

Given an n-state 2nfa accepting L we can find:

- a 2dfa accepting L with $f(n)$ states
- a 2nfa (actually a 2dfa) accepting L^c with $4f(n)$ states

Hence:
the complementation of 2nfa's costs no more than their determinization.

Theorem

If the complementation of 2nfa's requires an exponential number of states then the gap between 2nfa's and 2dfa's is exponential.

Complementation of two-way automata

Relationship with the Sakoda&Sipser question

$f(n) :=$ the cost of the simulation of a n-state 2nfa by a 2dfa.

Given an n-state 2nfa accepting L we can find:

- a 2dfa accepting L with $f(n)$ states
- a 2nfa (actually a 2dfa) accepting L^c with $4f(n)$ states

Hence:
the complementation of 2nfa's costs no more than their determinization.

Theorem

If the complementation of 2nfa's requires an exponential number of states then the gap between 2nfa's and 2dfa's is exponential.

Complementation of two-way automata

Relationship with the Sakoda&Sipser question

$f(n) :=$ the cost of the simulation of a n-state 2nfa by a 2dfa.

Given an n-state 2nfa accepting L we can find:

- a 2dfa accepting L with $f(n)$ states
- a 2nfa (actually a 2dfa) accepting L^c with $4f(n)$ states

Hence:
the complementation of 2nfa's costs no more than their determinization.

Theorem

If the complementation of 2nfa's requires an exponential number of states then the gap between 2nfa's and 2dfa's is exponential.

Complementation of one-way automata

- Deterministic case: trivial
- Nondeterministic case:

  Worst case: [Jiraskova, 2005]

  For each integer $n \geq 1$ there exists a language L accepted by a 1nfa with n states such that each 1nfa accepting L^c needs 2^n states.

  Best case: [Mera, Pighizzini, 2005]

  For each integer $n \geq 1$ there exists a language L such that:
  – the smallest 1nfa accepting L has n states
  – the minimum 1dfa's accepting L and L^c have 2^n states
  – the smallest 1nfa accepting L^c has at most $n + 1$ states.

  Remarks:

  L is a witness of the gap between 1nfa's and 1dfa's.
  Both L and L^c have "small" 1nfa but "large" 1dfa's.

Complementation of one-way automata

- Deterministic case: trivial
- Nondeterministic case:

  Worst case: [Jiraskova, 2005]

  For each integer $n \geq 1$ there exists a language L accepted by a 1nfa with n states such that each 1nfa accepting L^c needs 2^n states.

  Best case: [Mera, Pighizzini, 2005]

  For each integer $n \geq 1$ there exists a language L such that:
  – the smallest 1nfa accepting L has n states
  – the minimum 1dfa's accepting L and L^c have 2^n states
  – the smallest 1nfa accepting L^c has at most $n + 1$ states.

  Remarks:

  L is a witness of the gap between 1nfa's and 1dfa's.
  Both L and L^c have "small" 1nfa but "large" 1dfa's.

Complementation of one-way automata

- Deterministic case: trivial
- Nondeterministic case:

  Worst case: [Jiraskova, 2005]

  For each integer $n \geq 1$ there exists a language L accepted by a 1nfa with n states such that each 1nfa accepting L^c needs 2^n states.

  Best case: [Mera, Pighizzini, 2005]

  For each integer $n \geq 1$ there exists a language L such that:
  – the smallest 1nfa accepting L has n states
  – the minimum 1dfa's accepting L and L^c have 2^n states
  – the smallest 1nfa accepting L^c has at most $n+1$ states.

  Remarks:

  L is a witness of the gap between 1nfa's and 1dfa's.
  Both L and L^c have "small" 1nfa but "large" 1dfa's.

Complementation of one-way automata

- Deterministic case: trivial
- Nondeterministic case:

  Worst case: [Jiraskova, 2005]

  For each integer $n \geq 1$ there exists a language L accepted by a 1nfa with n states such that each 1nfa accepting L^c needs 2^n states.

  Best case: [Mera, Pighizzini, 2005]

  For each integer $n \geq 1$ there exists a language L such that:
  – the smallest 1nfa accepting L has n states
  – the minimum 1dfa's accepting L and L^c have 2^n states
  – the smallest 1nfa accepting L^c has at most $n + 1$ states.

  Remarks:

  L is a witness of the gap between 1nfa's and 1dfa's.

  Both L and L^c have "small" 1nfa but "large" 1dfa's.

Complementation of one-way automata

- Deterministic case: trivial
- Nondeterministic case:

  Worst case: [Jiraskova, 2005]

  For each integer $n \geq 1$ there exists a language L accepted by a 1nfa with n states such that each 1nfa accepting L^c needs 2^n states.

  Best case: [Mera, Pighizzini, 2005]

  For each integer $n \geq 1$ there exists a language L such that:
  – the smallest 1nfa accepting L has n states
  – the minimum 1dfa's accepting L and L^c have 2^n states
  – the smallest 1nfa accepting L^c has at most $n+1$ states.

  Remarks:

  L is a witness of the gap between 1nfa's and 1dfa's.

  Both L and L^c have "small" 1nfa but "large" 1dfa's.

Complementation of one-way automata

- Deterministic case: trivial
- Nondeterministic case:

  Worst case: [Jiraskova, 2005]

  For each integer $n \geq 1$ there exists a language L accepted by a 1nfa with n states such that each 1nfa accepting L^c needs 2^n states.

  Best case: [Mera, Pighizzini, 2005]

  For each integer $n \geq 1$ there exists a language L such that:
  – the smallest 1nfa accepting L has n states
  – the minimum 1dfa's accepting L and L^c have 2^n states
  – the smallest 1nfa accepting L^c has at most $n+1$ states.

  Remarks:

  L is a witness of the gap between 1nfa's and 1dfa's.
  Both L and L^c have "small" 1nfa but "large" 1dfa's.

Complementation of 1nfa: unary case

The last result does not hold in the case of unary languages:

For each unary language L such that:

- L is accepted by a 1nfa with n states
- L the minimum 1dfa accepting L has $e^{O(\sqrt{n \log n})}$ states

Then: [Mera, Pighizzini, 2005]

Each 1nfa accepting L^c should have at least n states.

Hence, if L has a "small" 1nfa and a "large" 1dfa, i.e., it is a witness of the gap between unary 1nfa's and 1dfa's, then the nondeterminism does not help in the recognition of L^c.

Complementation of 1nfa: unary case

The last result does not hold in the case of unary languages:

For each unary language L such that:

- L is accepted by a 1nfa with n states
- L the minimum 1dfa accepting L has $e^{O(\sqrt{n \log n})}$ states

Then: [Mera, Pighizzini, 2005]

Each 1nfa accepting L^c should have at least n states.

Hence, if L has a "small" 1nfa and a "large" 1dfa, i.e., it is a witness of the gap between unary 1nfa's and 1dfa's, then the nondeterminism does not help in the recognition of L^c.

Complementation of 1nfa: unary case

The last result does not hold in the case of unary languages:

For each unary language L such that:

- L is accepted by a 1nfa with n states
- L the minimum 1dfa accepting L has $e^{O(\sqrt{n \log n})}$ states

Then: [Mera, Pighizzini, 2005]

Each 1nfa accepting L^c should have at least n states.

Hence, if L has a "small" 1nfa and a "large" 1dfa, i.e., it is a witness of the gap between unary 1nfa's and 1dfa's, then the nondeterminism does not help in the recognition of L^c.

Complementation of 1nfa: unary case

The last result does not hold in the case of unary languages:

For each unary language L such that:

- L is accepted by a 1nfa with n states
- L the minimum 1dfa accepting L has $e^{O(\sqrt{n \log n})}$ states

Then: [Mera, Pighizzini, 2005]

Each 1nfa accepting L^c should have at least n states.

Hence, if L has a "small" 1nfa and a "large" 1dfa, i.e., it is a witness of the gap between unary 1nfa's and 1dfa's, then the nondeterminism does not help in the recognition of L^c.