Investigations on Automata and Languages over a Unary Alphabet

Giovanni Pighizzini

Dipartimento di Informatica
Università degli Studi di Milano, Italy

CIAA 2014 – Gießen, Germany
July 30 – August 2, 2014

UNIVERSITÀ DEGLI STUDI
DI MILANO

Unary or Tally Languages

- One letter alphabet $\Sigma = \{a\}$
- Many differences with the general case have been discovered

  First example:

  Theorem [Ginsurg&Rice '62]

  Each unary context-free languages is regular

- Structural complexity: classes of *tally sets*
  - Hartmanis, 1972
  - Book, 1974, 1979
  - ...

  Space complexity:
  - Alt&Mehlhorn, 1975
  - Geffert, 1993
  - ...

Unary or Tally Languages

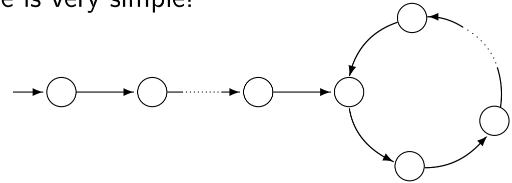
This talk:

- Focus mainly on *descriptional complexity aspects*
  - Optimal simulations between variants of unary automata
  - Unary two-way automata:
    connection with the question $L \stackrel{?}{=} NL$
  - Unary context-free grammars and pushdown automata
- Devices accepting nonregular languages

Unary Automata

Unary One-Way Deterministic Automata (1DFAs)

The structure is very simple!



Theorem

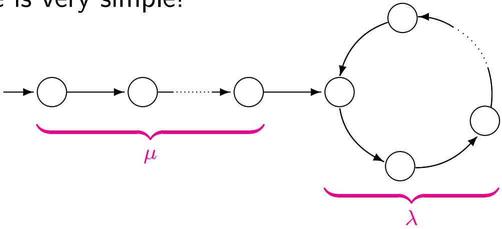
$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ s.t.

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

When $\mu = 0$ the language L is said to be *cyclic*

Unary One-Way Deterministic Automata (1DFAs)

The structure is very simple!



Theorem

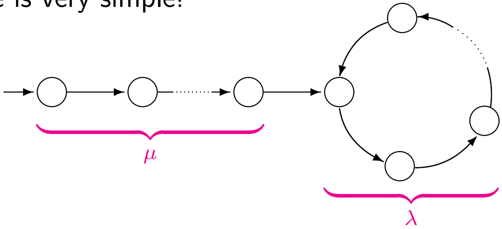
$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ s.t.

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

When $\mu = 0$ the language L is said to be *cyclic*

Unary One-Way Deterministic Automata (1DFAs)

The structure is very simple!



Theorem

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

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

When $\mu = 0$ the language L is said to be *cyclic*

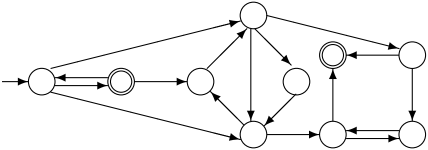
Unary One-Way Nondeterministic Automata (1NFAs)

The structure can be very complicate!

Each direct graph with

- *a vertex selected as initial state*
- *some vertices selected as final states*

is the transition diagram of a unary 1NFA!



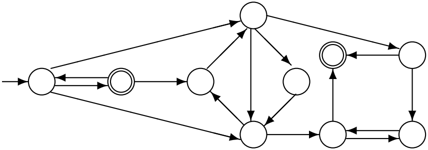
Unary One-Way Nondeterministic Automata (1NFAs)

The structure can be very complicate!

Each direct graph with

- *a vertex selected as initial state*
- *some vertices selected as final states*

is the transition diagram of a unary 1NFA!

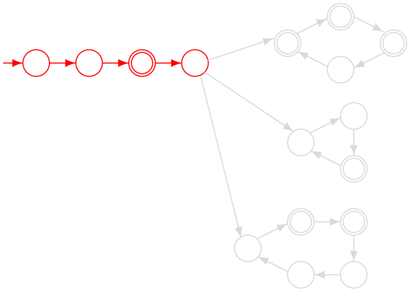


However, we can always obtain an equivalent 1NFA with a

- *simple* and
- *not too big*

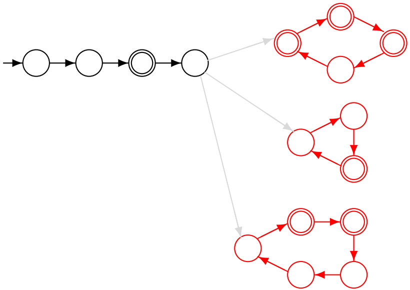
transition graph

Chrobak Normal Form for 1NFAs



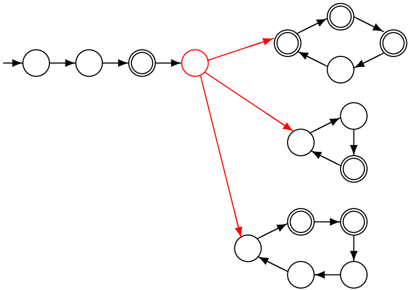
- An initial *deterministic path*
- Some disjoint *deterministic loops*
- *Only one nondeterministic decision*

Chrobak Normal Form for 1NFAs



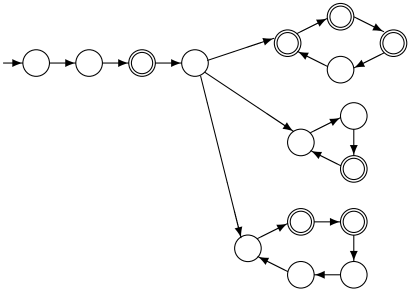
- An initial *deterministic path*
- Some disjoint *deterministic loops*
- *Only one nondeterministic decision*

Chrobak Normal Form for 1NFAs



- An initial *deterministic path*
- Some disjoint *deterministic loops*
- *Only one nondeterministic decision*

Chrobak Normal Form for 1NFAs



- An initial *deterministic path*
- Some disjoint *deterministic loops*
- *Only one nondeterministic decision*

Theorem ([Chrobak '86])

Each unary n-state 1NFA can be converted into an equivalent 1NFA in Chrobak normal form with

- *an initial path of* $O(n^2)$ *states*
- *total number of states in the loops* $\leq n$

Conversion to Chrobak Normal Form for 1NFAs

- Subtle error in the original proof fixed by To (2009)
- Different transformation proposed by Geffert (2007)
- Polynomial time conversion algorithms
  by Martinez (2004), Gawrychowski (2011), Sawa (2013)
- From the results by Geffert and Gawrychowski:
  - length of the initial path $\leq n^2 - n$
  - total number of states in the loops $\leq n - 1$
    (except when the given 1NFA is the trivial loop of n states)

Conversion to Chrobak Normal Form for 1NFAs

- Subtle error in the original proof fixed by To (2009)
- Different transformation proposed by Geffert (2007)
- Polynomial time conversion algorithms
  by Martinez (2004), Gawrychowski (2011), Sawa (2013)
- From the results by Geffert and Gawrychowski:
  - length of the initial path $\leq n^2 - n$
  - total number of states in the loops $\leq n - 1$
    (except when the given 1NFA is the trivial loop of n states)

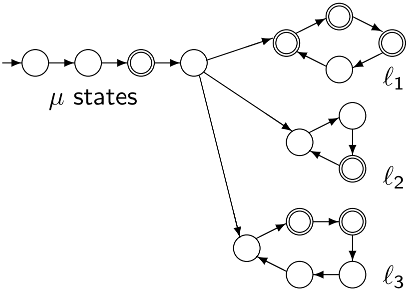
Conversion to Chrobak Normal Form for 1NFAs

- Subtle error in the original proof fixed by To (2009)
- Different transformation proposed by Geffert (2007)
- Polynomial time conversion algorithms
  by Martinez (2004), Gawrychowski (2011), Sawa (2013)
- From the results by Geffert and Gawrychowski:
  - length of the initial path $\leq n^2 - n$
  - total number of states in the loops $\leq n - 1$
    (except when the given 1NFA is the trivial loop of n states)

Conversion to Chrobak Normal Form for 1NFAs

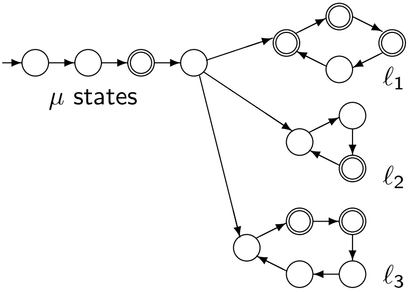
- Subtle error in the original proof fixed by To (2009)
- Different transformation proposed by Geffert (2007)
- Polynomial time conversion algorithms
  by Martinez (2004), Gawrychowski (2011), Sawa (2013)
- From the results by Geffert and Gawrychowski:
  - length of the initial path $\leq n^2 - n$
  - total number of states in the loops $\leq n - 1$
    (except when the given 1NFA is the trivial loop of n states)

Removing Nondeterminism from Unary Automata



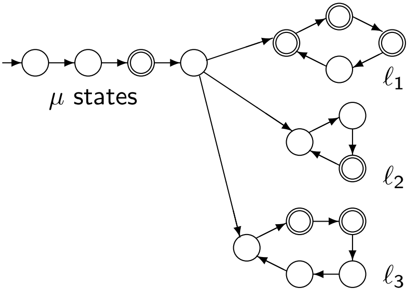
- Keep the same initial path
- Simulate all the loops "in parallel"
- A loop of $\mathrm{lcm}\{\ell_1, \ell_2, \ldots\}$ many states is enough
- Total number of states $\leq \mu + \mathrm{lcm}\{\ell_1, \ell_2, \ldots\}$
- From a n-state 1NFA: $\mu = O(n^2)$, $\ell_1 + \ell_2 + \cdots \leq n$

Removing Nondeterminism from Unary Automata



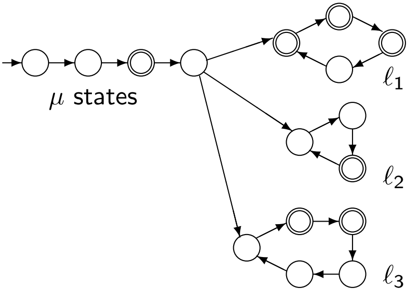
- Keep the same initial path
- Simulate all the loops "in parallel"
- A loop of $\mathrm{lcm}\{\ell_1, \ell_2, \ldots\}$ many states is enough
- Total number of states $\leq \mu + \mathrm{lcm}\{\ell_1, \ell_2, \ldots\}$
- From a n-state 1NFA: $\mu = O(n^2)$, $\ell_1 + \ell_2 + \cdots \leq n$

Removing Nondeterminism from Unary Automata



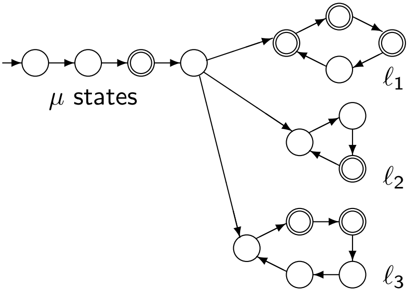
- Keep the same initial path
- Simulate all the loops "in parallel"
- A loop of $\text{lcm}\{\ell_1, \ell_2, \ldots\}$ many states is enough
- Total number of states $\leq \mu + \text{lcm}\{\ell_1, \ell_2, \ldots\}$
- From a n-state 1NFA: $\mu = O(n^2)$, $\ell_1 + \ell_2 + \cdots \leq n$

Removing Nondeterminism from Unary Automata



- Keep the same initial path
- Simulate all the loops "in parallel"
- A loop of $\text{lcm}\{\ell_1, \ell_2, \ldots\}$ many states is enough
- Total number of states $\leq \mu + \text{lcm}\{\ell_1, \ell_2, \ldots\}$
- From a *n*-state 1NFA: $\mu = O(n^2)$, $\ell_1 + \ell_2 + \cdots \leq n$

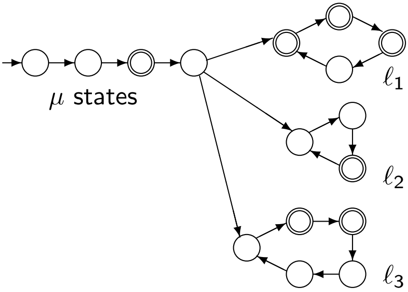
Removing Nondeterminism from Unary Automata



- Keep the same initial path
- Simulate all the loops "in parallel"
- A loop of $\text{lcm}\{\ell_1, \ell_2, \dots\}$ many states is enough
- Total number of states $\le \mu + \text{lcm}\{\ell_1, \ell_2, \dots\}$
- From a n-state 1NFA: $\mu = O(n^2)$, $\ell_1 + \ell_2 + \cdots \le n$

How large can be $\text{lcm}\{\ell_1, \ell_2, \dots\}$?

Removing Nondeterminism from Unary Automata

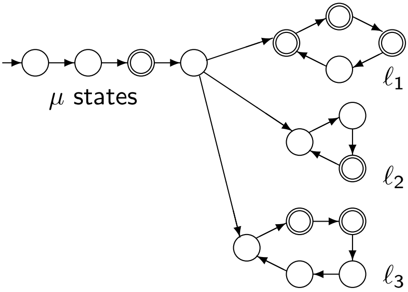


- Keep the same initial path
- Simulate all the loops “in parallel”
- A loop of $\text{lcm}\{\ell_1, \ell_2, \ldots\}$ many states is enough
- Total number of states $\leq \mu + \text{lcm}\{\ell_1, \ell_2, \ldots\}$
- From a n-state 1NFA: $\mu = O(n^2)$, $\ell_1 + \ell_2 + \cdots \leq n$

How large can be $\text{lcm}\{\ell_1, \ell_2, \ldots\}$?

$$F(n) = \max\{\text{lcm}\{\ell_1, \ell_2, \ldots, \ell_s\} \mid s \geq 1 \wedge \ell_1 + \ell_2 + \cdots + \ell_s \leq n\}$$

Removing Nondeterminism from Unary Automata



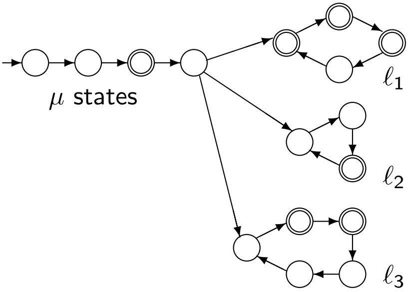
- Keep the same initial path
- Simulate all the loops "in parallel"
- A loop of $\text{lcm}\{\ell_1, \ell_2, \ldots\}$ many states is enough
- Total number of states $\leq \mu + \text{lcm}\{\ell_1, \ell_2, \ldots\}$
- From a n-state 1NFA: $\mu = O(n^2)$, $\ell_1 + \ell_2 + \cdots \leq n$

How large can be $\text{lcm}\{\ell_1, \ell_2, \ldots\}$?

$$F(n) = \max\{\text{lcm}\{\ell_1, \ell_2, \ldots, \ell_s\} \mid s \geq 1 \wedge \ell_1 + \ell_2 + \cdots + \ell_s \leq n\}$$

Landau's function (1903) $\qquad F(n) = e^{\Theta(\sqrt{n \ln n})}$ [Szalay '80]

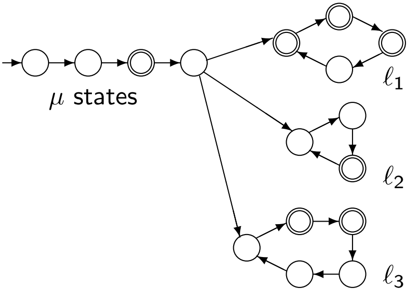
Removing Nondeterminism from Unary Automata



- Keep the same initial path
- Simulate all the loops "in parallel"
- A loop of $\mathrm{lcm}\{\ell_1, \ell_2, \ldots\}$ many states is enough
- Total number of states $\leq \mu + \mathrm{lcm}\{\ell_1, \ell_2, \ldots\}$
- From a n-state 1NFA: $\mu = O(n^2)$, $\ell_1 + \ell_2 + \cdots \leq n$

- $F(n)$ states are also necessary in the worst case [Chrobak '86]

Removing Nondeterminism from Unary Automata



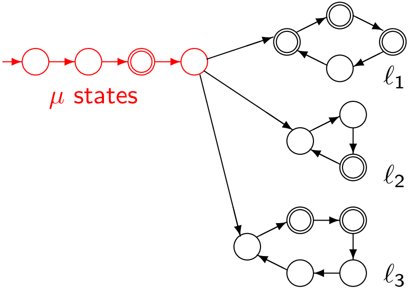
- Keep the same initial path
- Simulate all the loops "in parallel"
- A loop of $\text{lcm}\{\ell_1, \ell_2, \ldots\}$ many states is enough
- Total number of states $\leq \mu + \text{lcm}\{\ell_1, \ell_2, \ldots\}$
- From a n-state 1NFA: $\mu = O(n^2)$, $\ell_1 + \ell_2 + \cdots \leq n$

- $F(n)$ states are also necessary in the worst case [Chrobak '86]

Theorem ([Ljubič '64, Chrobak '86])

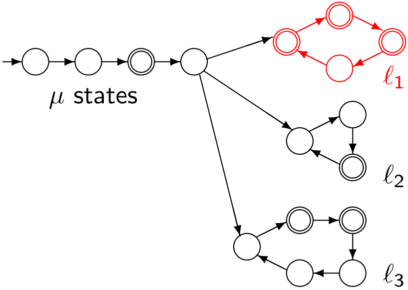
The state cost of the simulation of unary n-state 1NFAs by equivalent 1DFAs is $e^{\Theta(\sqrt{n \ln n})}$

From Chrobak Normal Form to Two-Way Automata



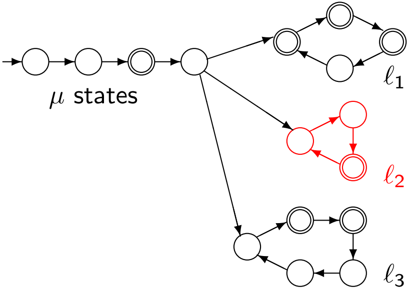
- Check if the input is "short" and accepted on the initial path — $\mu + 1$ states
- Check if the input is accepted on the first loop — ℓ_1 states
- Check if the input is accepted on the second loop — ℓ_2 states
- Check if the input is accepted on the third loop — ℓ_3 states

From Chrobak Normal Form to Two-Way Automata



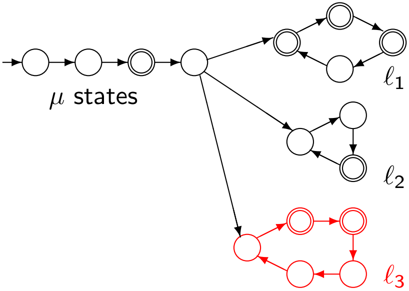
- Check if the input is "short" and accepted on the initial path — $\mu + 1$ states
- Check if the input is accepted on the first loop — ℓ_1 states
- Check if the input is accepted on the second loop — ℓ_2 states
- Check if the input is accepted on the third loop — ℓ_3 states

From Chrobak Normal Form to Two-Way Automata



- Check if the input is "short" and accepted on the initial path $\mu + 1$ states
- Check if the input is accepted on the first loop ℓ_1 states
- Check if the input is accepted on the second loop ℓ_2 states
- Check if the input is accepted on the third loop ℓ_3 states

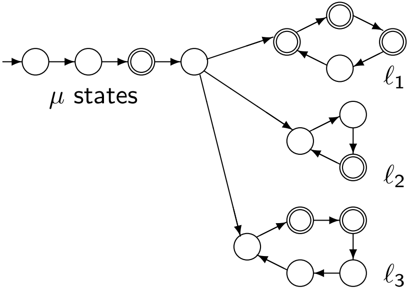
From Chrobak Normal Form to Two-Way Automata



- Check if the input is "short" and accepted on the initial path $\mu + 1$ states
- Check if the input is accepted on the first loop ℓ_1 states
- Check if the input is accepted on the second loop ℓ_2 states
- Check if the input is accepted on the third loop ℓ_3 states

$\mu + \ell_1 + \ell_2 + \cdots + 2$ states are sufficient!

From Chrobak Normal Form to Two-Way Automata

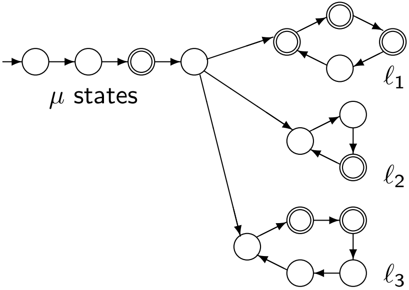


- Check if the input is "short" and accepted on the initial path $\mu + 1$ states
- Check if the input is accepted on the first loop ℓ_1 states
- Check if the input is accepted on the second loop ℓ_2 states
- Check if the input is accepted on the third loop ℓ_3 states

$\mu + \ell_1 + \ell_2 + \cdots + 2$ states are sufficient!

This number is also necessary in the worst case [Chrobak '86]

From Chrobak Normal Form to Two-Way Automata



- Check if the input is "short" and accepted on the initial path $\mu + 1$ states
- Check if the input is accepted on the first loop ℓ_1 states
- Check if the input is accepted on the second loop ℓ_2 states
- Check if the input is accepted on the third loop ℓ_3 states

$\mu + \ell_1 + \ell_2 + \cdots + 2$ states are sufficient!

This number is also necessary in the worst case [Chrobak '86]

Theorem

The state cost of the simulation of unary n-state 1NFAs by 2DFAs is $\Theta(n^2)$

Optimal Simulations Between Unary Automata

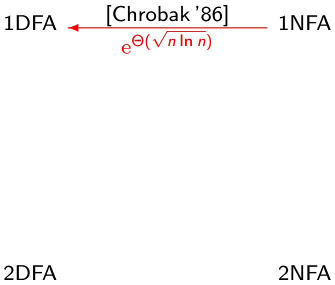
1DFA

1NFA

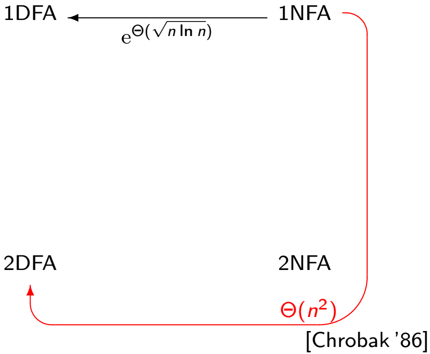
2DFA

2NFA

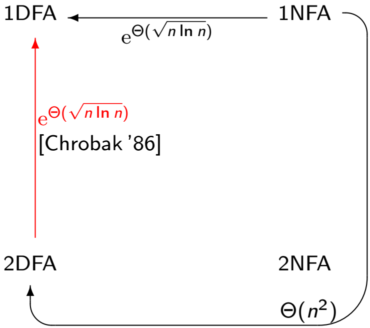
Optimal Simulations Between Unary Automata



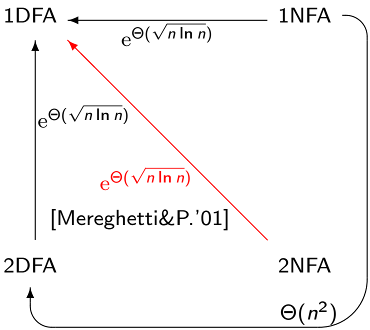
Optimal Simulations Between Unary Automata



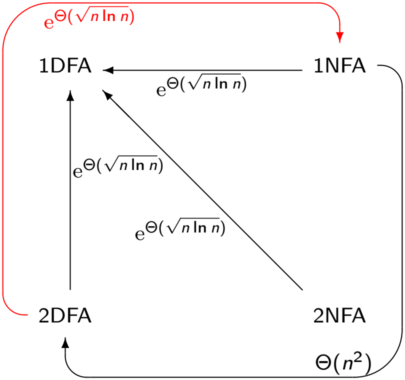
Optimal Simulations Between Unary Automata



Optimal Simulations Between Unary Automata

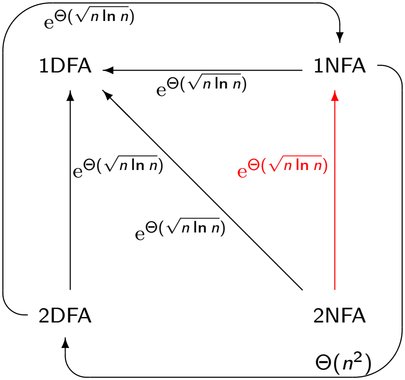


Optimal Simulations Between Unary Automata



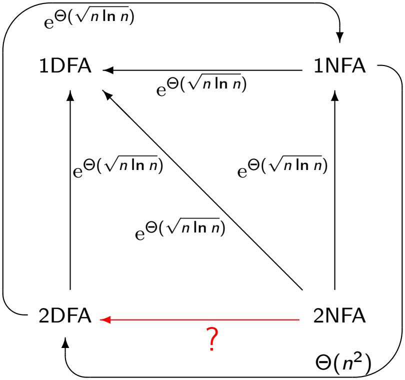
from 2DFA $\to$ 1DFA

Optimal Simulations Between Unary Automata



from 2NFA $\rightarrow$ 1DFA

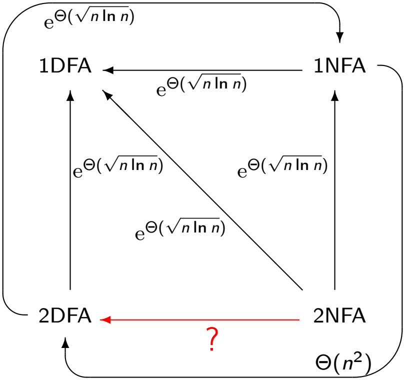
Optimal Simulations Between Unary Automata



2NFA → 2DFA Open!

- upper bound $e^{\Theta(\sqrt{n \ln n})}$ (from 2NFA → 1DFA)
- lower bound $\Omega(n^2)$ (from 1NFA → 2DFA)

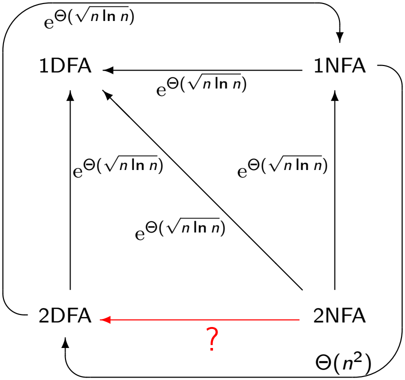
Optimal Simulations Between Unary Automata



2NFA → 2DFA Open!

- upper bound $e^{\Theta(\sqrt{n \ln n})}$
  (from 2NFA → 1DFA)
- lower bound $\Omega(n^2)$
  (from 1NFA → 2DFA)

Optimal Simulations Between Unary Automata



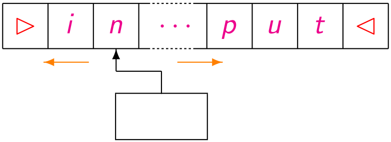
2NFA → 2DFA Open!

- upper bound $e^{\Theta(\sqrt{n \ln n})}$
  (from 2NFA → 1DFA)
- lower bound $\Omega(n^2)$
  (from 1NFA → 2DFA)

Better upper bound $e^{O(\ln^2 n)}$
[Geffert&Mereghetti&P.'03]

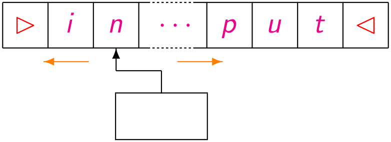
Unary Two-Way Automata

Two-Way Automata: Few Technical Details



- Input surrounded by the end-markers $\triangleright$ and $\triangleleft$
- $w \in \Sigma^*$ is accepted iff there is a computation
  - with input tape $\triangleright w \triangleleft$
  - starting with the head on $\triangleright$ in the initial state
  - reaching a final state (with the head on $\triangleright$)

Two-Way Automata: Few Technical Details



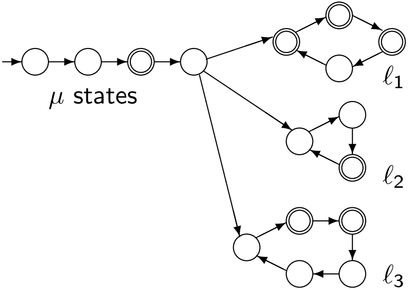
- Input surrounded by the end-markers $\triangleright$ and $\triangleleft$
- $w \in \Sigma^*$ is accepted iff there is a computation
  - with input tape $\triangleright w \triangleleft$
  - starting with the head on $\triangleright$ in the initial state
  - reaching a final state (with the head on $\triangleright$)

Almost Equivalent Automata

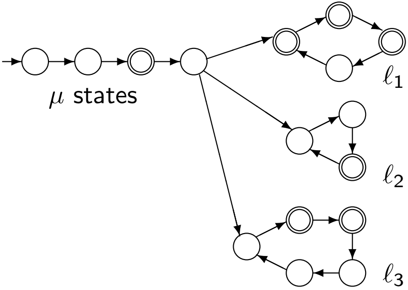
Definition

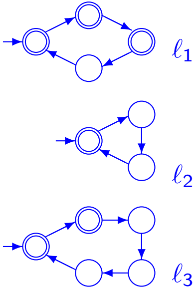
Two automata A and B are *almost equivalent*
if $L(A)$ and $L(B)$ differ for finitely many strings

Chrobak Normal Form Revisited

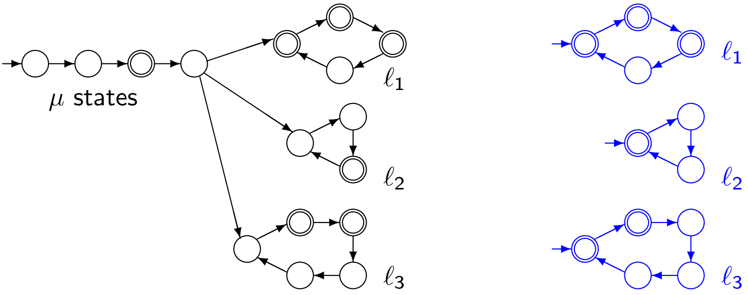


Chrobak Normal Form Revisited





Chrobak Normal Form Revisited



Each unary n-state 1NFA A is almost equivalent to a 1NFA B:

- s disjoint loops of lengths $\ell_1, \ldots, \ell_s$, with $\ell_1 + \cdots + \ell_s \leq n$
- at the beginning of the computation,
  B nondeterministically selects a loop $i \in \{1, \ldots, s\}$
- then B counts the input length modulo ℓ_i
- $L(A)$ and $L(B)$ can differ only on strings of length at most $n^2 - n$

A Normal Form for Unary 2NFAs

Theorem ([Geffert&Mereghetti&P.'03])

For each unary n-state 2NFA A there exists an almost equivalent 2NFA M s.t.

- *M makes nondeterministic choices and changes the head direction only visiting the end-markers*
- *M has $N \leq 2n+2$ many states*
- *$L(A)$ and $L(M)$ can differ only on strings of length $\leq 5n^2$*

A Normal Form for Unary 2NFAs

More details on M:

- State set: $\{q_I, q_F\} \cup Q_1 \cup \cdots \cup Q_s$
  - q_I initial state
  - q_F accepting state
  - Q_i deterministic loop of length ℓ_i
- A computation is a sequence of traversals of the input
- In each traversal M counts the input length modulo one ℓ_i

A Normal Form for Unary 2NFAs

More details on M:

- State set: $\{q_I, q_F\} \cup Q_1 \cup \cdots \cup Q_s$
  - q_I initial state
  - q_F accepting state
  - Q_i deterministic loop of length ℓ_i
- A computation is a sequence of traversals of the input
- In each traversal M counts the input length modulo one ℓ_i

A Normal Form for Unary 2NFAs

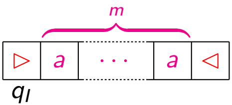
More details on M:

- State set: $\{q_I, q_F\} \cup Q_1 \cup \cdots \cup Q_s$
  - q_I initial state
  - q_F accepting state
  - Q_i deterministic loop of length ℓ_i
- A computation is a sequence of traversals of the input
- In each traversal M counts the input length modulo one ℓ_i

A Normal Form for Unary 2NFAs

More details on M:

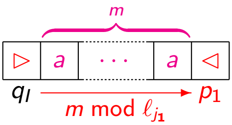
- State set: $\{q_I, q_F\} \cup Q_1 \cup \cdots \cup Q_s$
  - q_I initial state
  - q_F accepting state
  - Q_i deterministic loop of length ℓ_i
- A computation is a sequence of traversals of the input
- In each traversal M counts the input length modulo one ℓ_i



A Normal Form for Unary 2NFAs

More details on M:

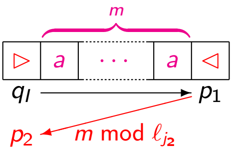
- State set: $\{q_I, q_F\} \cup Q_1 \cup \cdots \cup Q_s$
  - q_I initial state
  - q_F accepting state
  - Q_i deterministic loop of length ℓ_i
- A computation is a sequence of traversals of the input
- In each traversal M counts the input length modulo one ℓ_i



A Normal Form for Unary 2NFAs

More details on M:

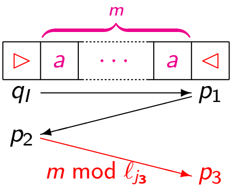
- State set: $\{q_I, q_F\} \cup Q_1 \cup \cdots \cup Q_s$
  - q_I initial state
  - q_F accepting state
  - Q_i deterministic loop of length ℓ_i
- A computation is a sequence of traversals of the input
- In each traversal M counts the input length modulo one ℓ_i



A Normal Form for Unary 2NFAs

More details on M:

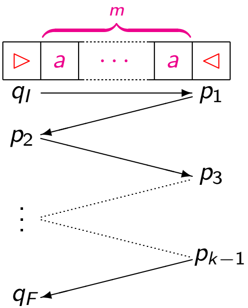
- State set: $\{q_I, q_F\} \cup Q_1 \cup \cdots \cup Q_s$
  - q_I initial state
  - q_F accepting state
  - Q_i deterministic loop of length ℓ_i
- A computation is a sequence of traversals of the input
- In each traversal M counts the input length modulo one ℓ_i



A Normal Form for Unary 2NFAs

More details on M:

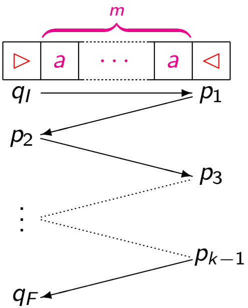
- State set: $\{q_I, q_F\} \cup Q_1 \cup \cdots \cup Q_s$
  - q_I initial state
  - q_F accepting state
  - Q_i deterministic loop of length ℓ_i
- A computation is a sequence of traversals of the input
- In each traversal M counts the input length modulo one ℓ_i



A Normal Form for Unary 2NFAs

More details on M:

- State set: $\{q_I, q_F\} \cup Q_1 \cup \cdots \cup Q_s$
  - q_I initial state
  - q_F accepting state
  - Q_i deterministic loop of length ℓ_i
- A computation is a sequence of traversals of the input
- In each traversal M counts the input length modulo one ℓ_i



Remark

If a string is accepted by M then it is accepted by a computation which visits the left end-marker at most N times

Converting Unary 2NFAs into 2DFAs

[Geffert&Mereghetti&P.'03]

M unary N-state 2NFA *in normal form*

a^m input string

- For $p, q \in Q$, $k \geq 1$, we consider the predicate
  reachable$(p, q, k) \equiv$
  $\exists$computation path on a^m which
  - starts in the state p on $\triangleright$
  - ends in the state q on $\triangleright$
  - visits $\triangleright$ at most k times

  Then:
  $a^m \in L(M)$ iff *reachable*(q_I, q_F, N) is true
- *reachable*(p, q, k) can be computed by a recursive procedure
- Implemented by a 2DFA with $e^{O(\ln^2 N)}$ states

Converting Unary 2NFAs into 2DFAs
[Geffert&Mereghetti&P.'03]

M unary N-state 2NFA *in normal form*

a^m input string

- For $p, q \in Q$, $k \geq 1$, we consider the predicate
  $reachable(p, q, k) \equiv$
  $\exists$computation path on a^m which
  - starts in the state p on $\triangleright$
  - ends in the state q on $\triangleright$
  - visits $\triangleright$ at most k times

  Then:
  $a^m \in L(M)$ iff $reachable(q_I, q_F, N)$ is true
- $reachable(p, q, k)$ can be computed by a recursive procedure
- Implemented by a 2DFA with $e^{O(\ln^2 N)}$ states

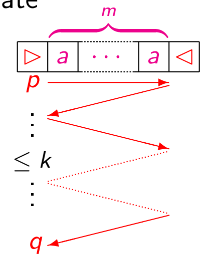
Converting Unary 2NFAs into 2DFAs
[Geffert&Mereghetti&P.'03]

M unary N-state 2NFA *in normal form*

a^m input string

- For $p, q \in Q$, $k \geq 1$, we consider the predicate
  $\mathit{reachable}(p, q, k) \equiv$
  $\exists$computation path on a^m which
  - starts in the state p on $\triangleright$
  - ends in the state q on $\triangleright$
  - visits $\triangleright$ at most k times

  Then:
  $a^m \in L(M)$ iff $\mathit{reachable}(q_I, q_F, N)$ is true



- $\mathit{reachable}(p, q, k)$ can be computed by a recursive procedure
- Implemented by a 2DFA with $e^{O(\ln^2 N)}$ states

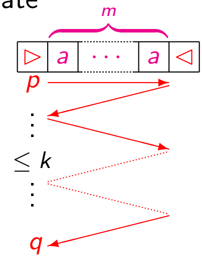
Converting Unary 2NFAs into 2DFAs
[Geffert&Mereghetti&P.'03]

M unary N-state 2NFA *in normal form*

a^m input string

- For $p, q \in Q$, $k \geq 1$, we consider the predicate
  $reachable(p, q, k) \equiv$
  $\exists$computation path on a^m which
  - starts in the state p on $\triangleright$
  - ends in the state q on $\triangleright$
  - visits $\triangleright$ at most k times

  Then:
  $a^m \in L(M)$ iff $reachable(q_I, q_F, N)$ is true



- $reachable(p, q, k)$ can be computed by a recursive procedure
- Implemented by a 2DFA with $e^{O(\ln^2 N)}$ states

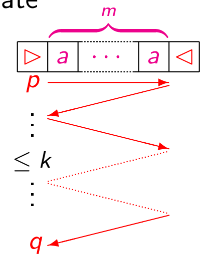
Converting Unary 2NFAs into 2DFAs
[Geffert&Mereghetti&P.'03]

M unary N-state 2NFA *in normal form*

a^m input string

- For $p, q \in Q$, $k \geq 1$, we consider the predicate
  $reachable(p, q, k) \equiv$
  $\exists$computation path on a^m which
  - starts in the state p on $\triangleright$
  - ends in the state q on $\triangleright$
  - visits $\triangleright$ at most k times

  Then:
  $a^m \in L(M)$ iff $reachable(q_I, q_F, N)$ is true



- $reachable(p, q, k)$ can be computed by a recursive procedure
- Implemented by a 2DFA with $e^{O(\ln^2 N)}$ states

Converting Unary 2NFAs into 2DFAs

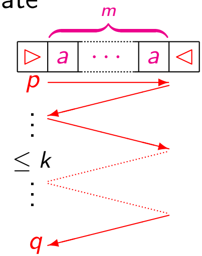
[Geffert&Mereghetti&P.'03]

M unary N-state 2NFA *in normal form*

a^m input string

- For $p, q \in Q$, $k \geq 1$, we consider the predicate
  $reachable(p, q, k) \equiv$
  $\exists$computation path on a^m which
  - starts in the state p on $\triangleright$
  - ends in the state q on $\triangleright$
  - visits $\triangleright$ at most k times

  Then:
  $a^m \in L(M)$ iff $reachable(q_I, q_F, N)$ is true



- $reachable(p, q, k)$ can be computed by a recursive procedure
- Implemented by a 2DFA with $e^{O(\ln^2 N)}$ states

From Unary 2NFAs to 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		
M	almost equivalent 2NFA	$N \leq 2n + 2$ states
$\Downarrow$		
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

From Unary 2NFAs to 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Conversion into normal form
M	almost equivalent 2NFA	$N \leq 2n+2$ states
$\Downarrow$		
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

From Unary 2NFAs to 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Conversion into normal form
M	almost equivalent 2NFA	$N \leq 2n + 2$ states
$\Downarrow$		Deterministic simulation
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

From Unary 2NFAs to 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Conversion into normal form
M	almost equivalent 2NFA	$N \leq 2n+2$ states
$\Downarrow$		Deterministic simulation
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		Preliminary scan to accept/reject inputs of length $\leq 5n^2$ then simulation of B for longer inputs
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

From Unary 2NFAs to 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Conversion into normal form
M	almost equivalent 2NFA	$N \leq 2n + 2$ states
$\Downarrow$		Deterministic simulation
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		Preliminary scan to accept/reject inputs of length $\leq 5n^2$ then simulation of B for longer inputs
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

Theorem ([Geffert&Mereghetti&P.'03])

Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ many states

From Unary 2NFAs to 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Conversion into normal form
M	almost equivalent 2NFA	$N \leq 2n+2$ states
$\Downarrow$		Deterministic simulation
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		Preliminary scan to accept/reject inputs of length $\leq 5n^2$ then simulation of B for longer inputs
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

Theorem ([Geffert&Mereghetti&P.'03])

Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ *many states*

Upper bound

- superpolynomial
- subexponential

From Unary 2NFAs to 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Conversion into normal form
M	almost equivalent 2NFA	$N \leq 2n + 2$ states
$\Downarrow$		Deterministic simulation
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		Preliminary scan to accept/reject inputs of length $\leq 5n^2$ then simulation of B for longer inputs
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

Theorem ([Geffert&Mereghetti&P.'03])

Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ *many states*

Can this upper bound be reduced to a polynomial?

Upper bound

- superpolynomial
- subexponential

Logspace Classes and Graph Accessibility Problem

L: class of languages accepted in logarithmic space by *deterministic* machines

NL: class of languages accepted in logarithmic space by *nondeterministic* machines

Problem

$L \stackrel{?}{=} NL$

Logspace Classes and Graph Accessibility Problem

L: class of languages accepted in logarithmic space by *deterministic* machines

NL: class of languages accepted in logarithmic space by *nondeterministic* machines

Problem

$L \stackrel{?}{=} NL$

Logspace Classes and Graph Accessibility Problem

L: class of languages accepted in logarithmic space by *deterministic* machines

NL: class of languages accepted in logarithmic space by *nondeterministic* machines

Problem

$L \stackrel{?}{=} NL$

Logspace Classes and Graph Accessibility Problem

L: class of languages accepted in logarithmic space by *deterministic* machines

NL: class of languages accepted in logarithmic space by *nondeterministic* machines

Problem

$\mathsf{L} \stackrel{?}{=} \mathsf{NL}$

Graph Accessibility Problem GAP

- Given $G = (V, E)$ oriented graph, $s, t \in V$
- Decide whether or not G contains a path from s to t

Logspace Classes and Graph Accessibility Problem

L: class of languages accepted in logarithmic space by *deterministic* machines

NL: class of languages accepted in logarithmic space by *nondeterministic* machines

Problem

$L \stackrel{?}{=} NL$

Graph Accessibility Problem GAP

- Given $G = (V, E)$ oriented graph, $s, t \in V$
- Decide whether or not G contains a path from s to t

Theorem ([Jones '75])

GAP *is complete for* NL

Hence GAP $\in$ L iff L $=$ NL

Reduction to GAP

[Geffert&P.'11]

M unary 2NFA in normal form, with N states

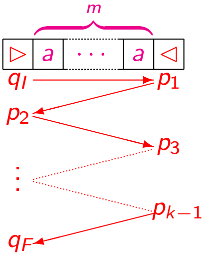
- Accepting computation on a^m
  - sequence of traversals of the input
    - starting in q_I on ▷
    - ending in q_F on ▷
- Graph $G(m)$
  - vertices ≡ states
  - edges ≡ traversals on a^m
- a^m is accepted iff $G(m)$ contains a path from q_I to q_F

Reduction to GAP

[Geffert&P.'11]

M unary 2NFA in normal form, with N states

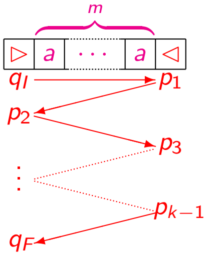
- Accepting computation on a^m
  - sequence of traversals of the input
    - starting in q_I on ▷
    - ending in q_F on ▷
- Graph $G(m)$
  - vertices ≡ states
  - edges ≡ traversals on a^m
- a^m is accepted iff $G(m)$ contains a path from q_I to q_F



Reduction to GAP
[Geffert&P.'11]

M unary 2NFA in normal form, with N states

- Accepting computation on a^m
  - sequence of traversals of the input
    - starting in q_I on $\triangleright$
    - ending in q_F on $\triangleright$
- Graph $G(m)$
  - vertices $\equiv$ states
  - edges $\equiv$ traversals on a^m

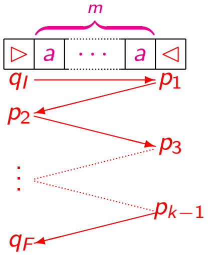


- a^m is accepted iff $G(m)$ contains a path from q_I to q_F

Reduction to GAP
[Geffert&P.'11]

M unary 2NFA in normal form, with N states

- Accepting computation on a^m
  - sequence of traversals of the input
    - starting in q_I on $\triangleright$
    - ending in q_F on $\triangleright$
- Graph $G(m)$
  - vertices $\equiv$ states
  - edges $\equiv$ traversals on a^m

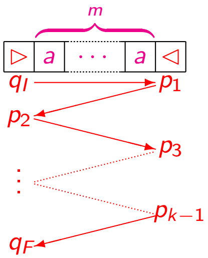


- a^m is accepted iff $G(m)$ contains a path from q_I to q_F

Reduction to GAP
[Geffert&P.'11]

M unary 2NFA in normal form, with N states

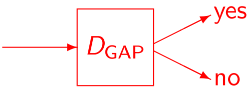
- Accepting computation on a^m
  - sequence of traversals of the input
    - starting in q_I on $\triangleright$
    - ending in q_F on $\triangleright$
- Graph $G(m)$
  - vertices $\equiv$ states
  - edges $\equiv$ traversals on a^m



- a^m is accepted iff $G(m)$ contains a path from q_I to q_F

To decide whether or not $a^m \in L(M)$ reduces to decide GAP for $G(m)$

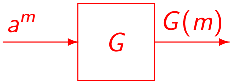
$L = NL \Rightarrow$ Polynomial Deterministic Simulation!
[Geffert&P.'11]



D_{GAP} logspace bounded *deterministic* machine solving GAP

L = NL $\Rightarrow$ Polynomial Deterministic Simulation!

[Geffert&P.'11]

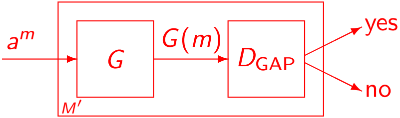


D_{GAP} logspace bounded *deterministic* machine solving GAP

$G(m)$ graph associated with a^m

L = NL $\Rightarrow$ Polynomial Deterministic Simulation!

[Geffert&P.'11]



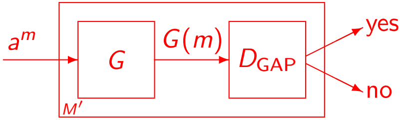
D_{GAP} logspace bounded *deterministic* machine solving GAP

$G(m)$ graph associated with a^m

M' resulting 2DFA

L = NL ⇒ Polynomial Deterministic Simulation!

[Geffert&P.'11]



D_{GAP} logspace bounded *deterministic* machine solving GAP

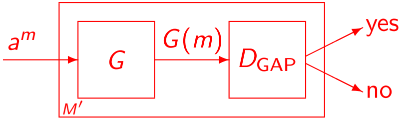
- $O(\log N)$ space N=#states of the given 2NFA M
- $poly(N)$ different configurations

$G(m)$ graph associated with a^m

M' resulting 2DFA

L = NL $\Rightarrow$ Polynomial Deterministic Simulation!

[Geffert&P.'11]



D_{GAP} logspace bounded *deterministic* machine solving GAP

- $O(\log N)$ space — N=#states of the given 2NFA M
- $poly(N)$ different configurations

$G(m)$ graph associated with a^m

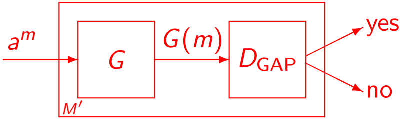
- $O(N^2)$ bits
- $exp(N)$ different configurations

Too many!!!

M' resulting 2DFA

$L = NL \Rightarrow$ Polynomial Deterministic Simulation!

[Geffert&P.'11]



D_{GAP} logspace bounded *deterministic* machine solving GAP

- $O(\log N)$ space $N = \#$states of the given 2NFA M
- $poly(N)$ different configurations

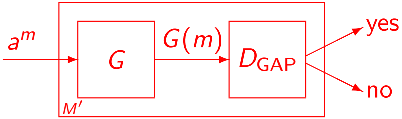
$G(m)$ graph associated with a^m

- $O(N^2)$ bits
- $exp(N)$ different configurations Too many!!!
- bits computed on demand:
  an N-state 1DFA $A_{p,q}$ tests the existence of the edge (p, q)
  trying to simulate a traversal of M from p to q

M' resulting 2DFA

$L = NL \;\Rightarrow\;$ Polynomial Deterministic Simulation!

[Geffert&P.'11]



D_{GAP} logspace bounded *deterministic* machine solving GAP

- $O(\log N)$ space — N=#states of the given 2NFA M
- $poly(N)$ different configurations

$G(m)$ graph associated with a^m

- $O(N^2)$ bits
- $exp(N)$ different configurations — Too many!!!
- bits computed on demand:
  an N-state 1DFA $A_{p,q}$ tests the existence of the edge (p, q) trying to simulate a traversal of M from p to q

M' resulting 2DFA — $poly(N)$ many states!!!

From Unary 2NFAs to 2DFAs (under L = NL)

A	given unary 2NFA	n states
$\Downarrow$		
M	almost equivalent 2NFA	$N \leq 2n+2$ states
$\Downarrow$		
B	2DFA equivalent to M	$poly(N)$ states
$\Downarrow$		
C	2DFA equivalent to A	$poly(n)$ states

From Unary 2NFAs to 2DFAs (under L = NL)

A	given unary 2NFA	n states
$\Downarrow$		Conversion into normal form
M	almost equivalent 2NFA	$N \le 2n+2$ states
$\Downarrow$		
B	2DFA equivalent to M	$poly(N)$ states
$\Downarrow$		
C	2DFA equivalent to A	$poly(n)$ states

From Unary 2NFAs to 2DFAs (under L = NL)

A	given unary 2NFA	n states
$\Downarrow$		Conversion into normal form
M	almost equivalent 2NFA	$N \leq 2n + 2$ states
$\Downarrow$		Deterministic simulation
B	2DFA equivalent to M	$poly(N)$ states
$\Downarrow$		
C	2DFA equivalent to A	$poly(n)$ states

From Unary 2NFAs to 2DFAs (under L = NL)

A	given unary 2NFA	n states
$\Downarrow$		Conversion into normal form
M	almost equivalent 2NFA	$N \leq 2n + 2$ states
$\Downarrow$		Deterministic simulation
B	2DFA equivalent to M	$poly(N)$ states
$\Downarrow$		Preliminary scan to accept/reject inputs of length $\leq 5n^2$ then simulation of B for longer inputs
C	2DFA equivalent to A	$poly(n)$ states

From Unary 2NFAs to 2DFAs (under L = NL)

A	given unary 2NFA	n states
$\Downarrow$		Conversion into normal form
M	almost equivalent 2NFA	$N \leq 2n+2$ states
$\Downarrow$		Deterministic simulation
B	2DFA equivalent to M	$poly(N)$ states
$\Downarrow$		Preliminary scan to accept/reject inputs of length $\leq 5n^2$ then simulation of B for longer inputs
C	2DFA equivalent to A	$poly(n)$ states

Theorem ([Geffert&P.'11])

If L = NL *then each unary n-state 2NFA can be simulated by a 2DFA with* $poly(n)$ *many states*

From Unary 2NFAs to 2DFAs (under $\mathsf{L} = \mathsf{NL}$)

A	given unary 2NFA	n states
$\Downarrow$		Conversion into normal form
M	almost equivalent 2NFA	$N \le 2n + 2$ states
$\Downarrow$		Deterministic simulation
B	2DFA equivalent to M	$poly(N)$ states
$\Downarrow$		Preliminary scan to accept/reject inputs of length $\le 5n^2$ then simulation of B for longer inputs
C	2DFA equivalent to A	$poly(n)$ states

Theorem ([Geffert&P.'11])

If $\mathsf{L} = \mathsf{NL}$ *then each unary* n*-state 2NFA can be simulated by a 2DFA with* $poly(n)$ *many states*

Proving that the best known upper bound $e^{O(\ln^2 n)}$ *is tight would separate* L *and* NL

From Unary 2NFAs to 2DFAs (under L = NL)

A	given unary 2NFA	n states
$\Downarrow$		Conversion into normal form
M	almost equivalent 2NFA	$N \leq 2n + 2$ states
$\Downarrow$		Deterministic simulation
B	2DFA equivalent to M	$poly(N)$ states
$\Downarrow$		Preliminary scan to accept/reject inputs of length $\leq 5n^2$ then simulation of B for longer inputs
C	2DFA equivalent to A	$poly(n)$ states

Theorem ([Geffert&P.'11])

If L = NL *then each unary n-state 2NFA can be simulated by a 2DFA with* $poly(n)$ *many states*

Theorem ([Kapoutsis&P.'12])

L/*poly* $\supseteq$ NL *iff each unary n-state 2NFA can be simulated by a 2DFA with* $poly(n)$ *many states*

Normal Form for Unary 2NFAs: Consequences

(i) Subexponential simulation of unary 2NFAs by 2DFAs
[Geffert&Mereghetti&P.'03]

(ii) Polynomial simulation of unary 2NFAs by 2DFAs
under the condition L = NL [Geffert&P.'11]

(iii) Polynomial simulation of unary 2NFAs by unambiguous 2NFAs
(unconditional) [Geffert&P.'11]

(iv) Polynomial complementation of unary 2NFAs
Inductive counting argument [Geffert&Mereghetti&P.'07]

Normal Form for Unary 2NFAs: Consequences

(i) Subexponential simulation of unary 2NFAs by 2DFAs
[Geffert&Mereghetti&P.'03]

(ii) Polynomial simulation of unary 2NFAs by 2DFAs
under the condition $\mathsf{L} = \mathsf{NL}$ [Geffert&P.'11]

(iii) Polynomial simulation of unary 2NFAs by unambiguous 2NFAs
(unconditional) [Geffert&P.'11]

(iv) Polynomial complementation of unary 2NFAs
Inductive counting argument [Geffert&Mereghetti&P.'07]

Normal Form for Unary 2NFAs: Consequences

(i) Subexponential simulation of unary 2NFAs by 2DFAs
[Geffert&Mereghetti&P.'03]

(ii) Polynomial simulation of unary 2NFAs by 2DFAs
under the condition L = NL [Geffert&P.'11]

(iii) Polynomial simulation of unary 2NFAs by unambiguous 2NFAs
(unconditional) [Geffert&P.'11]

(iv) Polynomial complementation of unary 2NFAs
Inductive counting argument [Geffert&Mereghetti&P.'07]

Normal Form for Unary 2NFAs: Consequences

(i) Subexponential simulation of unary 2NFAs by 2DFAs
[Geffert&Mereghetti&P.'03]

(ii) Polynomial simulation of unary 2NFAs by 2DFAs
under the condition L = NL [Geffert&P.'11]

(iii) Polynomial simulation of unary 2NFAs by unambiguous 2NFAs
(unconditional) [Geffert&P.'11]

(iv) Polynomial complementation of unary 2NFAs
Inductive counting argument [Geffert&Mereghetti&P.'07]

Pushdown Automata and Other Devices

Unary Context-Free Languages

Theorem [Ginsurg&Rice '62]

Each unary context-free languages is regular

How large should be a finite automata equivalent to a given unary context-free grammar or pushdown automaton?

Unary Context-Free Languages

Theorem [Ginsurg&Rice '62]

Each unary context-free languages is regular

How large should be a finite automata equivalent to a given unary context-free grammar or pushdown automaton?

Unary Pushdown Automata

From PDAs of size s, accepting regular languages, to equivalent 1DFAs

	unary input	general input
PDAs	$2^{poly(s)}$ [P.&Shallit&Wang '02]	

All the bounds are tight!

Unary Pushdown Automata

From PDAs of size s, accepting regular languages,
to equivalent 1DFAs

	unary input	general input
PDAs	$2^{poly(s)}$ [P.&Shallit&Wang '02]	non recursive [Meyer&Fischer '71]

All the bounds are tight!

Unary Pushdown Automata

From PDAs of size s, accepting regular languages,
to equivalent 1DFAs

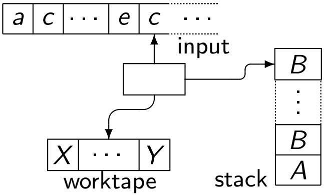
	unary input	general input
PDAs	$2^{poly(s)}$ [P.&Shallit&Wang '02]	non recursive [Meyer&Fischer '71]
deterministic PDAs	$2^{O(s)}$ [P.'09]	$2^{2^{O(s)}}$ [Valiant '75]

All the bounds are tight!

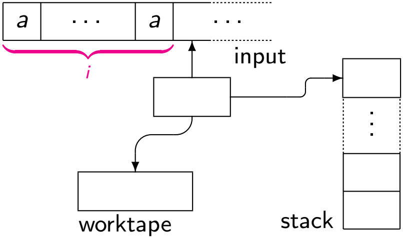
Auxiliary Pushdown Automata (AuxPDAs)

PDAs augmented with an auxiliary worktape

'SPACE' ≡ worktape

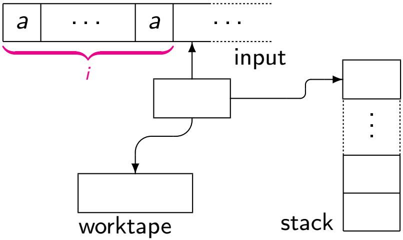


1AuxPDAs: How to Count the Input Length



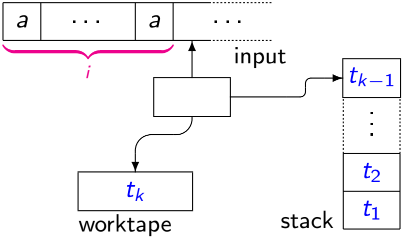
i

1AuxPDAs: How to Count the Input Length



$i = (1\ 1\ 0\ \cdots\ 1\ 0\ 0\ 0\ 1\ 0)_2$

1AuxPDAs: How to Count the Input Length

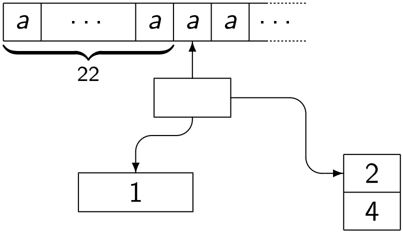


$$i = (1\ 1\ 0\ \cdots\ 1\ 0\ 0\ 0\ 1\ 0)_2 = 2^{t_1} + 2^{t_2} + \cdots + 2^{t_{k-1}} + 2^{t_k}$$

$t_1\ t_2 \qquad t_{k-1} \qquad t_k$

1AuxPDAs: How to Count the Input Length

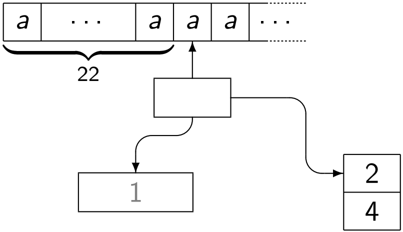
$22 = 2^4 + 2^2 + 2^1$



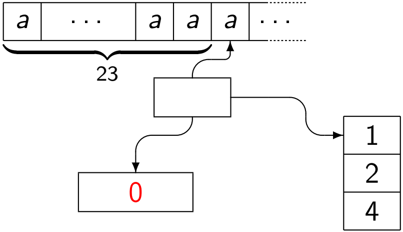
$23 = 2^4 + 2^2 + 2^1 + 2^0$

1AuxPDAs: How to Count the Input Length

$$22 = 2^4 + 2^2 + 2^1$$

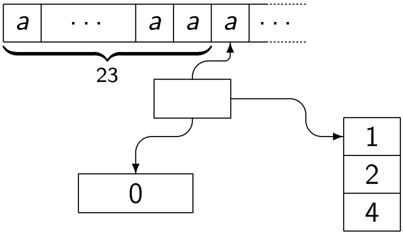


$$23 = 2^4 + 2^2 + 2^1 + 2^0$$



1AuxPDAs: How to Count the Input Length

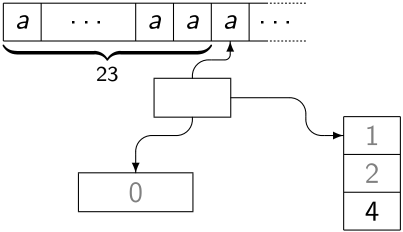
$23 = 2^4 + 2^2 + 2^1 + 2^0$



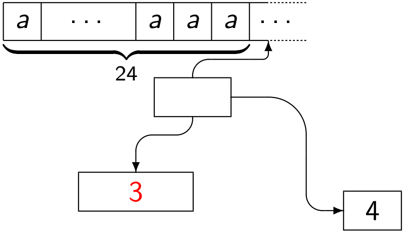
$24 = 2^4 + 2^3$

1AuxPDAs: How to Count the Input Length

$23 = 2^4 + 2^2 + 2^1 + 2^0$



$24 = 2^4 + 2^3$



Example: $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$

- $\mathcal{L}_p$ is nonregular
- $\mathcal{L}_p$ is accepted by a 1AuxPDA M which:
  - scans the input while counting its length
  - accepts iff the pushdown store is empty
    i.e., the binary representation of the input length contains exactly one digit 1
- On input a^n
  the largest integer stored on the worktape is $\lfloor \log_2 n \rfloor$, which is represented in $O(\log \log n)$ space

Example: $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$

- $\mathcal{L}_p$ is nonregular
- $\mathcal{L}_p$ is accepted by a 1AuxPDA M which:
  - scans the input while counting its length
  - accepts iff the pushdown store is empty
    i.e., the binary representation of the input length contains exactly one digit 1
- On input a^n
  the largest integer stored on the worktape is $\lfloor \log_2 n \rfloor$,
  which is represented in $O(\log \log n)$ space

Example: $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$

- $\mathcal{L}_p$ is nonregular
- $\mathcal{L}_p$ is accepted by a 1AuxPDA M which:
  - scans the input while counting its length
  - accepts iff the pushdown store is empty
    i.e., the binary representation of the input length contains exactly one digit 1
- On input a^n
  the largest integer stored on the worktape is $\lfloor \log_2 n \rfloor$, which is represented in $O(\log \log n)$ space

Example: $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$

- $\mathcal{L}_p$ is nonregular
- $\mathcal{L}_p$ is accepted by a 1AuxPDA M which:
  - scans the input while counting its length
  - accepts iff the pushdown store is empty
    i.e., the binary representation of the input length contains exactly one digit 1
- On input a^n
  the largest integer stored on the worktape is $\lfloor \log_2 n \rfloor$, which is represented in $O(\log \log n)$ space

$$\mathcal{L}_p \in \text{1AuxPDASpace}(\log \log n)$$

Space Bounds on 1AuxPDAs

$\mathcal{L}_p$ is accepted using the *minimum amount of space* for nonregular languages recognition:

Theorem ([P.&Shallit&Wang '02])

If a unary language L is accepted by a 1AuxPDA in $o(\log\log n)$ space then L is regular

In contrast

- with a binary alphabet,
- and space measured on the 'less' expensive accepting computation:

Theorem ([Chytil '86])

For each $k \geq 2$ there is a non context-free language L_k accepted by a 1AuxPDA in $O(\underbrace{\log \ldots \log}_{k} n)$ space

Space Bounds on 1AuxPDAs

$\mathcal{L}_p$ is accepted using the *minimum amount of space* for nonregular languages recognition:

Theorem ([P.&Shallit&Wang '02])

If a unary language L is accepted by a 1AuxPDA in $o(\log\log n)$ space then L is regular

In contrast

- with a binary alphabet,
- and space measured on the 'less' expensive accepting computation:

Theorem ([Chytil '86])

For each $k \geq 2$ there is a non context-free language L_k accepted by a 1AuxPDA in $O(\underbrace{\log \ldots \log}_{k} n)$ space

Two-way Pushdown Automata (2PDAs)

- More powerful than PDAs, e.g., $\{a^n b^n c^n \mid n \geq 0\}$
- 2DPDAs can be simulated by RAMs in *linear time* [Cook '71]

Main open problems:

- Power of nondeterminism, i.e., 2DPDAs vs 2PDA
- 2DPDAs vs linear bounded automata

Two-way Pushdown Automata (2PDAs)

- More powerful than PDAs, e.g., $\{a^n b^n c^n \mid n \geq 0\}$
- 2DPDAs can be simulated by RAMs in *linear time* [Cook '71]

Main open problems:

- Power of nondeterminism, i.e., 2DPDAs vs 2PDA
- 2DPDAs vs linear bounded automata

Unary 2PDAs

- Very powerful models, even in the deterministic version

Theorem ([Monien '84])

The unary encoding of each language in P *is accepted by a 2DPDA*

- With a *constant number* of input head reversals they accept only regular languages [Liu&Weiner '68]
- $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$ accepted by a 2DPDA making $\approx \log_2 n$ reversals

Unary 2PDAs

- Very powerful models, even in the deterministic version

Theorem ([Monien '84])

The unary encoding of each language in P *is accepted by a 2DPDA*

- With a *constant number* of input head reversals they accept only regular languages [Liu&Weiner '68]

- $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$ accepted by a 2DPDA making $\approx \log_2 n$ reversals

Unary 2PDAs

- Very powerful models, even in the deterministic version

Theorem ([Monien '84])

The unary encoding of each language in P *is accepted by a 2DPDA*

- With a *constant number* of input head reversals they accept only regular languages [Liu&Weiner '68]
- $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$ accepted by a 2DPDA making $\approx \log_2 n$ reversals

Unary 2PDAs

- Very powerful models, even in the deterministic version

Theorem ([Monien '84])

The unary encoding of each language in P *is accepted by a 2DPDA*

- With a *constant number* of input head reversals
they accept only regular languages [Liu&Weiner '68]
- $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$
accepted by a 2DPDA making $\approx \log_2 n$ reversals

Problem

Does there exist a unary nonregular language accepted by a 2PDA making $o(\log n)$ head reversals?

Multi-Head Finite Automata

- More powerful than one-head finite automata, even if the heads are *one-way*, e.g., $\{a^n b^n \mid n \geq 0\}$
- *Unary case:* with a *constant number* of head reversals they accept only regular languages [Sudborough '74]
- $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$ accepted by a 2-head automaton making $\approx \log_2 n$ reversals

Multi-Head Finite Automata

- More powerful than one-head finite automata, even if the heads are *one-way*, e.g., $\{a^n b^n \mid n \geq 0\}$
- *Unary case:* with a *constant number* of head reversals they accept only regular languages [Sudborough '74]
- $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$ accepted by a 2-head automaton making $\approx \log_2 n$ reversals

Multi-Head Finite Automata

- More powerful than one-head finite automata,
  even if the heads are *one-way*, e.g., $\{a^n b^n \mid n \geq 0\}$

- *Unary case:*
  with a *constant number* of head reversals
  they accept only regular languages [Sudborough '74]

- $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$
  accepted by a 2-head automaton making $\approx \log_2 n$ reversals

Multi-Head Finite Automata

- More powerful than one-head finite automata, even if the heads are *one-way*, e.g., $\{a^n b^n \mid n \geq 0\}$
- *Unary case:* with a *constant number* of head reversals they accept only regular languages [Sudborough '74]
- $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$ accepted by a 2-head automaton making $\approx \log_2 n$ reversals

Problem

Does there exist a unary nonregular language accepted by a multi-head automaton making $o(\log n)$ head reversals?

Multi-Head Finite Automata

- More powerful than one-head finite automata,
  even if the heads are *one-way*, e.g., $\{a^n b^n \mid n \geq 0\}$

- *Unary case:*
  with a *constant number* of head reversals
  they accept only regular languages [Sudborough '74]

- $\mathcal{L}_p = \{a^{2^m} \mid m \geq 0\}$
  accepted by a 2-head automaton making $\approx \log_2 n$ reversals

Problem

Does there exist a unary nonregular language accepted by a multi-head automaton making $o(\log n)$ head reversals?

- Unary multi-head 2PDAs making $O(1)$ input head reversals accept only regular languages [Ibarra '74]

Conclusion

Final Remarks

Unary Automata and Languages

- Interesting properties and differences with respect to the general case
- Special methods (e.g., from number theory)
- Important relationships with the general case
- Several open problems

Thank you for your attention!