Descriptional Complexity under Parikh Equivalence

Giovanna Lavado[1] Giovanni Pighizzini[1] Shinnosuke Seki[2,3]

1. Dipartimento di Informatica, Università degli Studi di Milano
2. Helsinki Institute for Information Technology (HIIT)
3. Department of Information and Computer Science, Aalto University

January 22, 2015

Research in Descriptional Complexity

How succinctly a formal language can be described by a formalism in comparison with other formalisms.

Take the length of description as complexity measure.

- What is the blow-up when changing from one model to another?
- Are there languages such that a blow-up is achieved?

Investigation of *upper bounds* and *lower bounds*.

Standard equivalence: 1NFAs vs 1DFAs

Subset construction [Rabin&Scott '59]

1NFA
n states
L

$\Longrightarrow$

1DFA
2^n states
L

Moreover, this state bound cannot be reduced in the worst case
[Meyer&Fischer '71, Moore '71]

What happens if we do not care of the order of symbols in the strings?

This problem is related to the concept of *Parikh equivalence*
[Parikh '66]

Standard equivalence: 1NFAs vs 1DFAs

Subset construction [Rabin&Scott '59]

1NFA
n states
L

$\Longrightarrow$

1DFA
2^n states
L

Moreover, this state bound cannot be reduced in the worst case
[Meyer&Fischer '71, Moore '71]

What happens if we do not care of the order of symbols in the strings?

This problem is related to the concept of *Parikh equivalence*
[Parikh '66]

Parikh equivalence: preliminaries

- $\Sigma = \{a_1, \ldots, a_m\}$ alphabet of m symbols
- $|w|_a$ be the number of occurrences of a in $w \in \Sigma^*$

Parikh map

The *Parikh map* $\psi : \Sigma^* \to \mathbb{N}^m$ associates with a word $w \in \Sigma^*$ the m-dimensional nonnegative vector $(|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_m})$.

Parikh image

The *Parikh image of a language* L is $\psi(L) = \{\psi(w) \mid w \in L\}$.

- $w_1 =_\pi w_2$ if $\psi(w_1) = \psi(w_2)$
- $L_1 =_\pi L_2$ if $\psi(L_1) = \psi(L_2)$

Parikh equivalence: Parikh's theorem

Theorem ([Parikh '66])

The Parikh image of a context-free language is a semilinear set. Equivalently, for each context-free language $L \subseteq \Sigma^*$, there exists a Parikh equivalent regular language $R \subseteq \Sigma^*$.

Example ($L =_\pi R$)

$$L = \{a^n b^n \mid n \geq 0\} \quad \text{and} \quad R = (ab)^*$$

have the same Parikh image, namely the set

$$\{(n, n) \mid n \geq 0\}$$

Parikh equivalence

Motivations and current research

Interesting theoretical properties:

- wrt Parikh equivalence regular and context-free languages are indistinguishable [Parikh '66]
- Unary context-free languages are regular [Ginsburg&Rice '62]

Connections with number theory and logic:

- Semilinear sets [Ginsburg&Spanier '64, Ginsburg&Spanier '66]
- Presburger arithmetics [Presburger&Jabcquette '91, Presburger '91]
- Formal verification [Göller&Mayr&To '09]
- Petri nets [Esparza '97]

Our Goal

We investigate the conversion of 1NFAs and CFGs into *Parikh equivalent* 1DFAs and 2DFAs, from a *descriptional complexity* point of view.

Problem (1NFAs to 1DFAs and 2DFAs)

1NFA
n states

$\Longrightarrow_{\pi}$

1DFA, 2DFA
how many states?

Problem (CFGs to 1DFAs and 2DFAs)

CFG
in Chomsky normal form
h variables

$\Longrightarrow_{\pi}$

1DFA, 2DFA
how many states?

From 1NFAs to Parikh equivalent 1DFAs

Our first contribution:

Problem (1NFAs to 1DFAs)

$$\underset{\substack{n \text{ states} \\ L_1}}{1\text{NFA}} \Longrightarrow_{\pi} \underset{\substack{\text{how many states?} \\ L_2}}{1\text{DFA}}$$

- Upper bound: 2^n
  by subset construction [Rabin&Scott '59]
- Lower bound: $e^{\sqrt{n \cdot \ln n}}$
  This bound derives from the *unary case*:
  the state cost of the conversion of unary n-state 1NFAs
  into equivalent 1DFAs is $e^{\Theta(\sqrt{n \cdot \ln n})}$ [Chrobak '86]

Converting 1NFAs accepting only nonunary strings

A preliminary step:

Problem (1NFAs to 1DFAs, restricted)

1NFA *s.t. each accepted string is nonunary*
n states
L_1

$\Longrightarrow_{\pi}$

1DFA
how many states?
L_2

Quite surprisingly, we can obtain a 1DFA with a number of states *polynomial* in n,

i.e., this conversion is less expensive than the conversion in the unary case, which costs $e^{\Theta(\sqrt{n \cdot \ln n})}$

Converting 1NFAs accepting only nonunary strings

A preliminary step:

Problem (1NFAs to 1DFAs, restricted)

1NFA *s.t. each accepted string is nonunary*
n states
L_1

$\Longrightarrow_{\pi}$

1DFA
how many states?
L_2

Quite surprisingly, we can obtain a 1DFA with a number of states *polynomial* in n,

i.e., this conversion is less expensive than the conversion in the unary case, which costs $e^{\Theta(\sqrt{n \cdot \ln n})}$

Converting 1NFAs accepting only nonunary strings

The conversion uses a modification of the following result:

Theorem ([Kopczyński&To '10])

Given $\Sigma = \{a_1, \ldots, a_m\}$, there is a polynomial p s.t. for each n-state 1NFA A over Σ,

$$\psi(L(A)) = \bigcup_{i \in I} Z_i$$

where:

- *I is a set of at most $p(n)$ indices*
- *for $i \in I$, $Z_i \subseteq \mathbb{N}^m$ is a linear set of the form:*

$$Z_i = \{\alpha_0 + n_1\alpha_1 + \cdots + n_k\alpha_k \mid n_1, \ldots, n_k \in \mathbb{N}\}$$

with

  - $0 \leq k \leq m$
  - *the components of α_0 are bounded by $p(n)$*
  - *$\alpha_1, \ldots, \alpha_k$ are linearly independent vectors from $\{0, 1, \ldots, n\}^m$*

Converting 1NFAs accepting only nonunary strings

Outline: linear sets

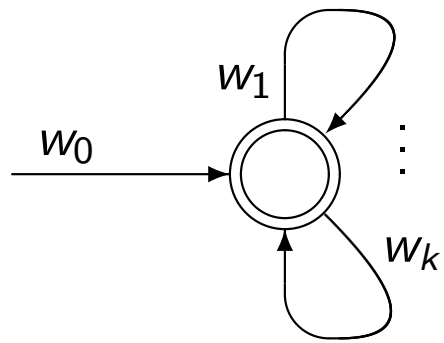
Each above linear set

$$Z_i = \{\alpha_0 + n_1\alpha_1 + \cdots + n_k\alpha_k \mid n_1, \ldots, n_k \in \mathbb{N}\}$$

can be converted into a poly size 1DFA accepting a language

$$R_i = w_0(w_1 + \cdots + w_k)^*$$

s.t. $\psi(w_j) = \alpha_j$, $j = 0, \ldots, k$, and
$w_1, \ldots, w_k$ *begin with different letters*



Converting 1NFAs accepting only nonunary strings

Outline: linear sets

Each above linear set

$$Z_i = \{\alpha_0 + n_1\alpha_1 + \cdots + n_k\alpha_k \mid n_1, \ldots, n_k \in \mathbb{N}\}$$

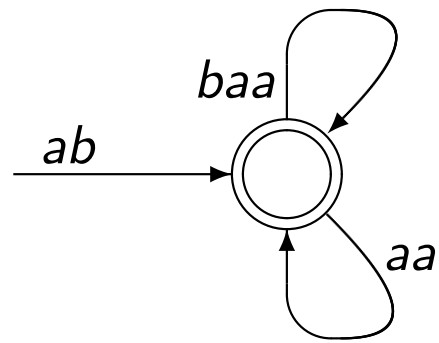
can be converted into a poly size 1DFA accepting a language

$$R_i = w_0(w_1 + \cdots + w_k)^*$$

s.t. $\psi(w_j) = \alpha_j$, $j = 0, \ldots, k$, and
$w_1, \ldots, w_k$ *begin with different letters*

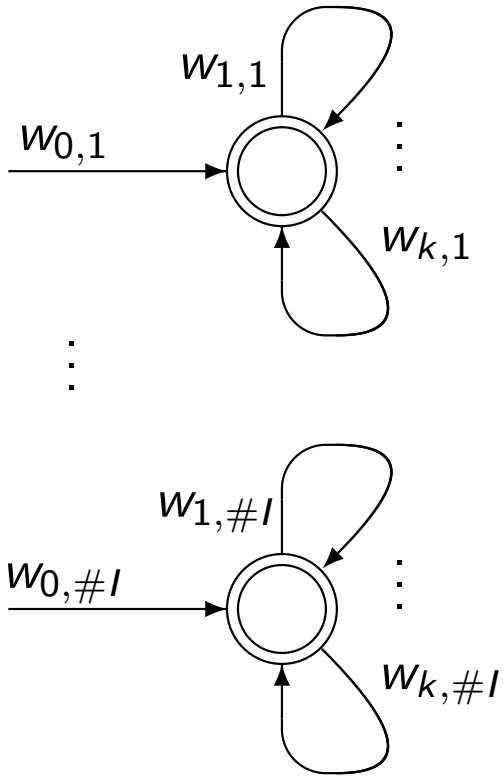
Example:

- $\{(1,1) + n_1(2,1) + n_2(2,0) \mid n_1, n_2 \geq 0\}$
- $ab(baa + aa)^*$



Converting 1NFAs accepting only nonunary strings

Outline: from linear to semilinear



- Standard construction for union of 1DFAs: number of states = *product*

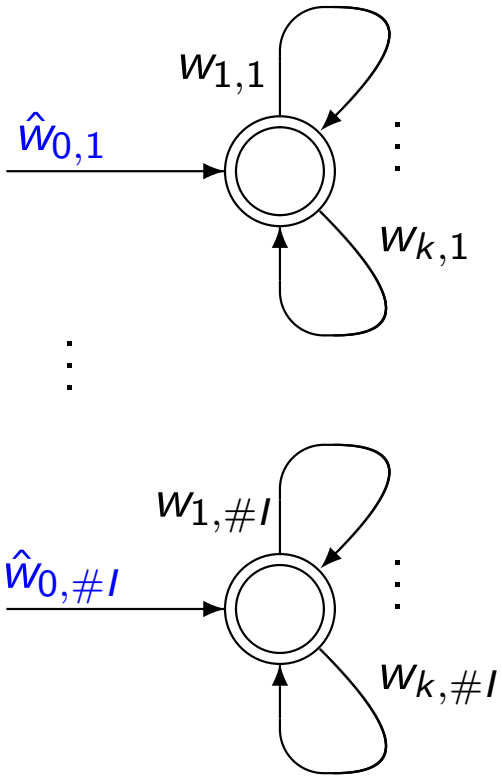
  $\#I \leq p(n) \Rightarrow$ Too large!!!

- Strings $w_{0,i}$ can be replaced by Parikh equivalent strings $\hat{w}_{0,i}$ in such a way that $W_0 = \{\hat{w}_{0,i} \mid i \in I\}$ is a *prefix code*

- After this change:
  number of states $\leq$ *sum* Polynomial!!!

Converting 1NFAs accepting only nonunary strings

Outline: from linear to semilinear



- Standard construction for union of 1DFAs: number of states = *product*

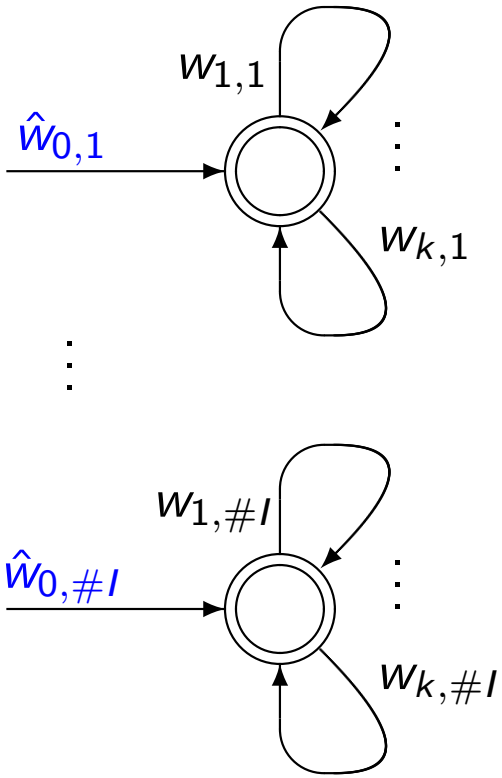
  $\#I \leq p(n) \Rightarrow$ Too large!!!

- Strings $w_{0,i}$ can be replaced by Parikh equivalent strings $\hat{w}_{0,i}$ in such a way that $W_0 = \{\hat{w}_{0,i} \mid i \in I\}$ is a *prefix code*

- After this change:
  number of states $\leq$ *sum* Polynomial!!!

Converting 1NFAs accepting only nonunary strings

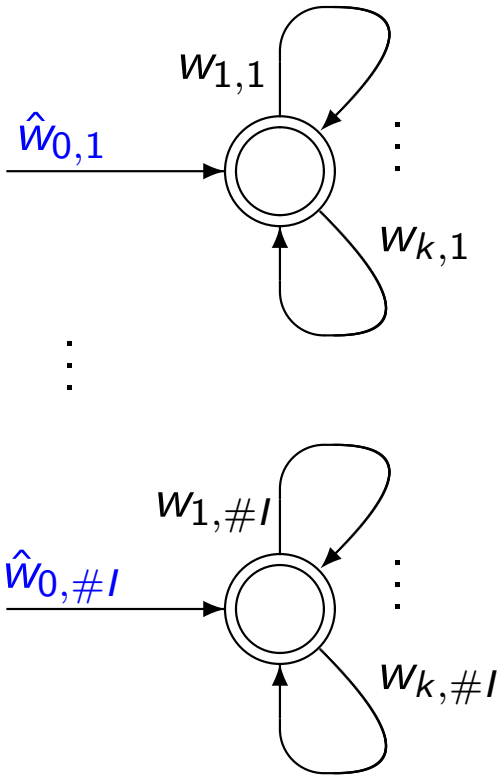
Outline: from linear to semilinear



- Standard construction for union of 1DFAs:
  number of states = *product*
  $\#I \leq p(n) \Rightarrow$ Too large!!!
- Strings $w_{0,i}$ can be replaced by Parikh equivalent strings $\hat{w}_{0,i}$ in such a way that $W_0 = \{\hat{w}_{0,i} \mid i \in I\}$ is a *prefix code*
- After this change:
  number of states $\leq$ *sum* Polynomial!!!

Converting 1NFAs accepting only nonunary strings

Outline: from linear to semilinear



- Standard construction for union of 1DFAs: number of states = *product*
  $\#I \leq p(n) \Rightarrow$ Too large!!!
- Strings $w_{0,i}$ can be replaced by Parikh equivalent strings $\hat{w}_{0,i}$ in such a way that $W_0 = \{\hat{w}_{0,i} \mid i \in I\}$ is a *prefix code*
- After this change:
  number of states $\leq$ *sum* Polynomial!!!

Theorem

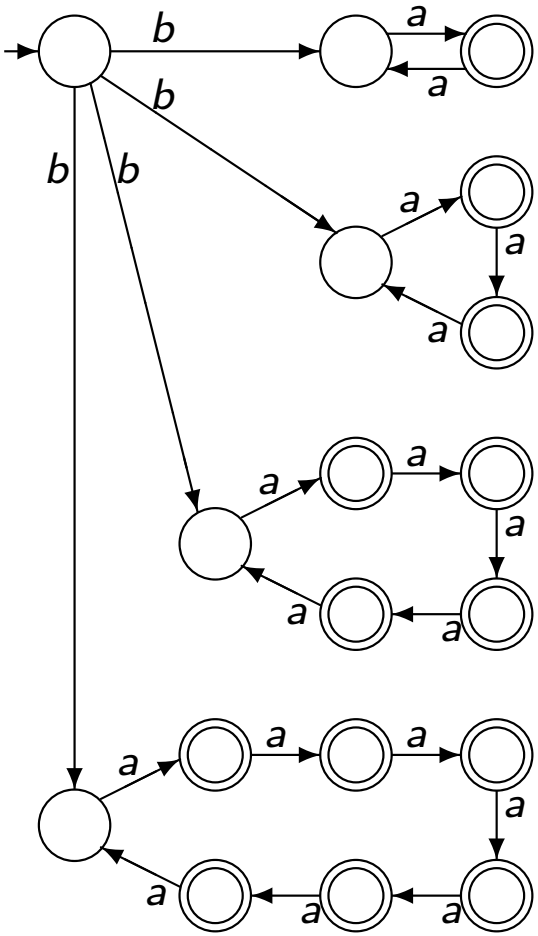
For each n-state 1NFA *accepting a language none of whose words are unary, there exists a Parikh equivalent* 1DFA *with a number of states polynomial in n.*

Converting 1NFAs accepting only nonunary strings

Example

Let us consider the following language

$$L = \{ba^n \mid n \bmod 210 \neq 0\}$$



- L does not contain any unary word.
- $L_1 = \{ba^n \mid n \bmod 2 \neq 0\}$,
- $L_2 = \{ba^n \mid n \bmod 3 \neq 0\}$,
- $L_3 = \{ba^n \mid n \bmod 5 \neq 0\}$,
- $L_4 = \{ba^n \mid n \bmod 7 \neq 0\}$,
- $L = L_1 \cup L_2 \cup L_3 \cup L_4$.

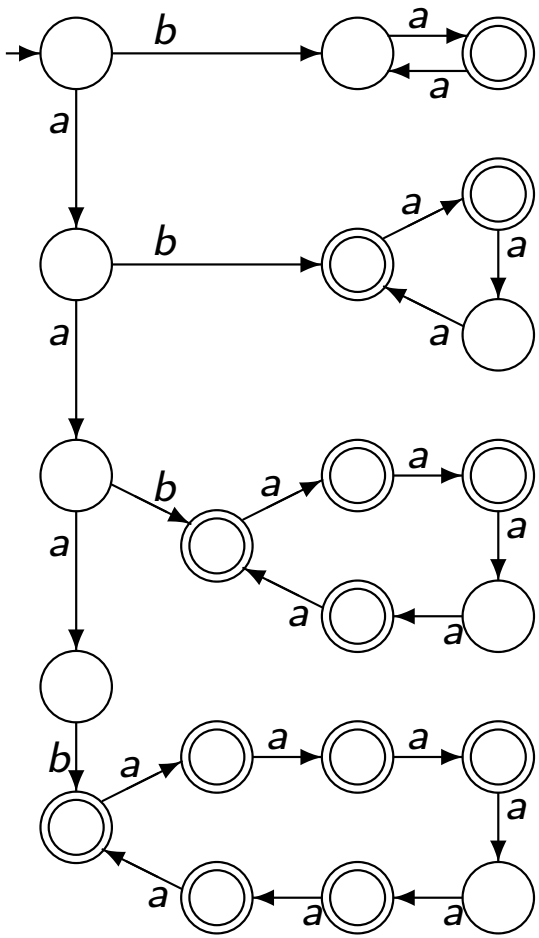
- L is accepted by the 18-state 1NFA A.
- The smallest 1DFA for L uses 211 states.

Converting 1NFAs accepting only nonunary strings

Example

We can build a complete 1DFA A' with only 22 states, accepting a language L' Parikh equivalent to L.

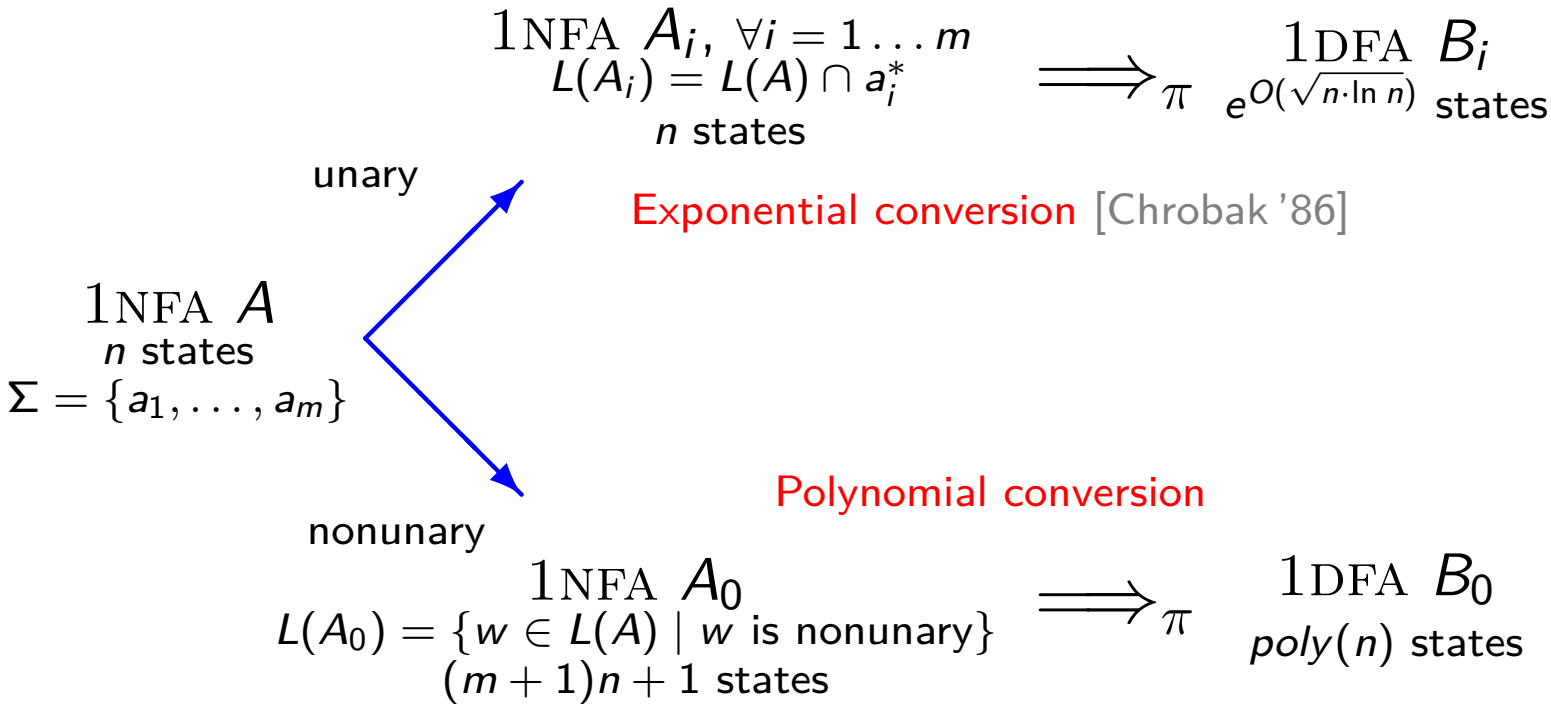
- $L_i =_\pi L'_i$, $i = 1, \ldots, 4$
- Words in L'_i begin with the prefix $a^{i-1}b$
- $L'_1 = \{ba^n \mid n \bmod 2 \neq 0\} = L_1$,
- $L'_2 = \{aba^{n-1} \mid n \bmod 3 \neq 0\}$,
- $L'_3 = \{a^2ba^{n-2} \mid n \bmod 5 \neq 0\}$,
- $L'_4 = \{a^3ba^{n-3} \mid n \bmod 7 \neq 0\}$,
- $L' = L'_1 \cup L'_2 \cup L'_3 \cup L'_4$.



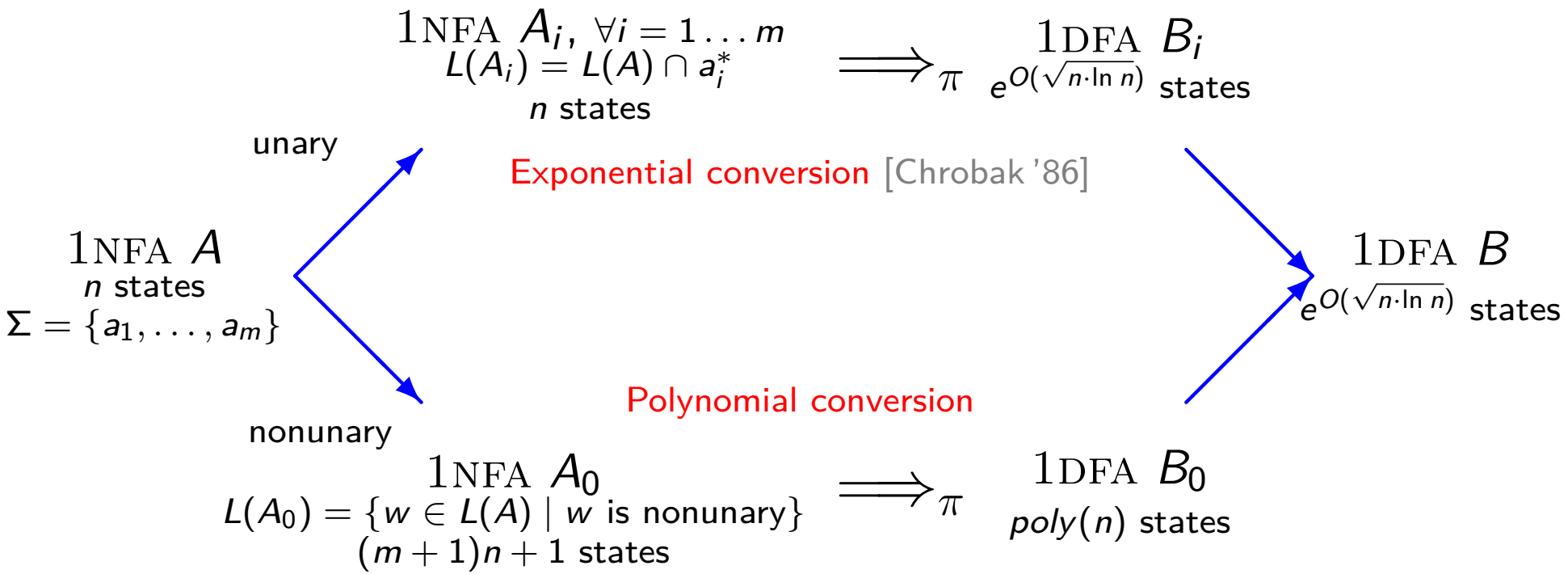
From 1NFAs to Parikh equivalent 1DFAs: general case

1NFA A
n states
$\Sigma = \{a_1, \ldots, a_m\}$

From 1NFAs to Parikh equivalent 1DFAs: general case



From 1NFAs to Parikh equivalent 1DFAs: general case



Theorem

For each n-state 1NFA *over* Σ, *there exists a Parikh equivalent* 1DFA *with* $e^{O(\sqrt{n \cdot \ln n})}$ *states. Furthermore, this cost is optimal.*

From CFGs to Parikh equivalent 1DFAs

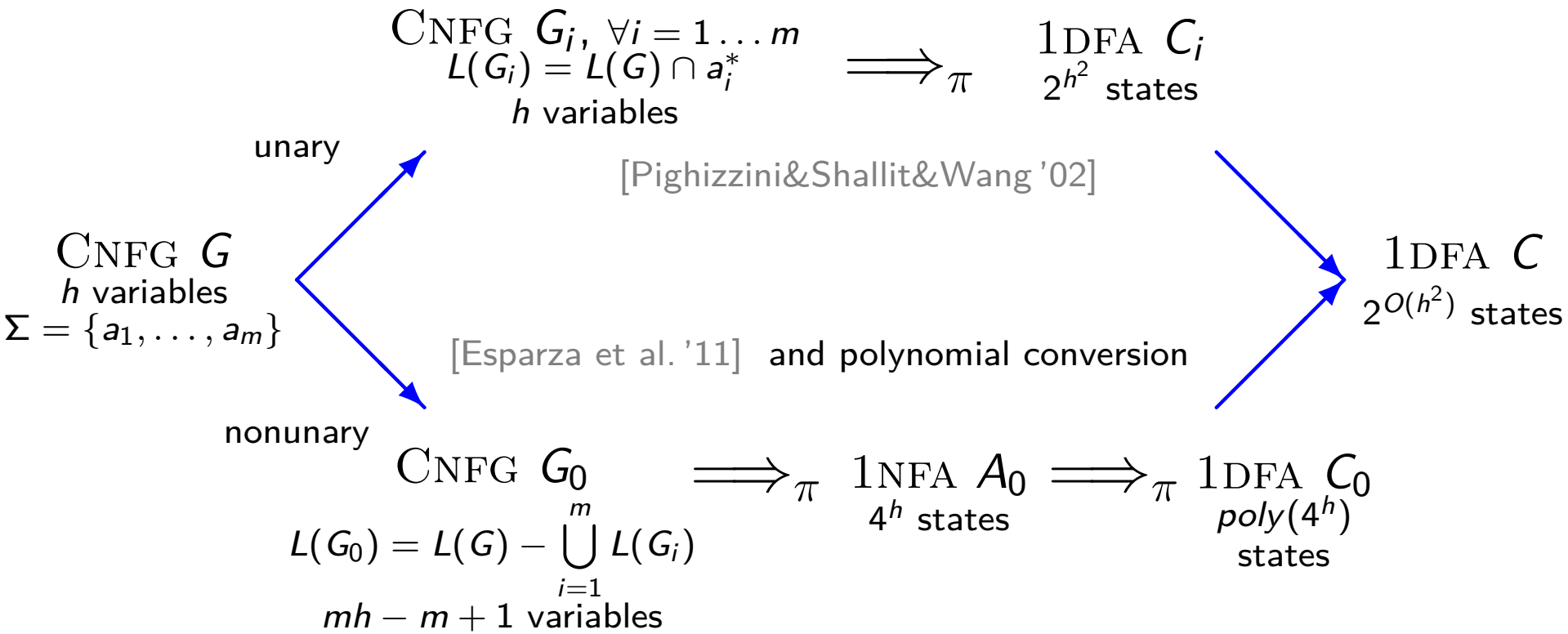
Our second contribution:

Problem (CFGs to 1DFAs)

$$\underset{\substack{\text{CFG } \textit{in Chomsky normal form}\\ h \textit{ variables}\\ L_1}}{\text{CNFG}} \quad \Longrightarrow_{\pi} \quad \underset{\substack{\textit{how many states?}\\ L_2}}{\text{1DFA}}$$

- Upper bound: $2^{O(4^h)}$
  by subset construction
  and [Esparza&Ganty&Kiefer&Luttenberger '11]
- Lower bound: 2^{ch^2}
  This bound derives from the *unary case*: the state cost of the conversion of unary h-variable CNFGs into equivalent 1DFAs is less than 2^{h^2} [Pighizzini&Shallit&Wang '02]

From CFGs to Parikh equivalent 1DFAs: proof idea



Theorem

For each h-variable CNFG *over Σ, there exists a Parikh equivalent* 1DFA *with at most $2^{O(h^2)}$ states. Furthermore, this cost is optimal.*

Conversions into Parikh equivalent 2DFAs

Summary table

Summarizing, we have the following optimal bounds:

	1DFA	2DFA
1NFA n states	$e^{O(\sqrt{n \cdot \ln n})}$	
CNFG h variables	$2^{O(h^2)}$	

Conversions into Parikh equivalent 2DFAs

Summary table

Our third contribution

	1DFA	2DFA
1NFA n states	$e^{O(\sqrt{n \cdot \ln n})}$	$poly(n)$
CNFG h variables	$2^{O(h^2)}$	$2^{O(h)}$

The idea is to split the accepted language into its unary and nonunary parts:

- The state cost of the conversion of unary n-state 1NFAs into equivalent 2DFAs is $O(n^2)$ [Chrobak '86]
- The state cost of the conversion of unary h-variable CNFGs into equivalent 1NFAs is at most $2^{2h-1} + 1$ [Pighizzini&Shallit&Wang '02]

Operational state complexity under Parikh equivalence

Our Goal

We investigate, under Parikh equivalence, the state complexity of some language operations which preserve regularity $(\cup, \cap, ^c, \cdot, ^*, \sqcup, ^R, P_{\Sigma_0})$.

Problem (1DFAs to 1DFA)

$$\begin{array}{c} A,\ B\ \text{1DFAs} \\ n_1,\ n_2\ \text{states} \\ L(A), L(B) \end{array} \quad \Longrightarrow_{\pi} \quad \begin{array}{c} C\ \text{1DFA} \\ L(C) =_{\pi} L \\ \textit{how many states?} \end{array}$$

where:

- $L = L(A) \cup L(B)$
- $L = L(A) \cap L(B)$
- $L = L(A)L(B)$
- ...

Concatenation under Parikh equivalence

Problem setting

Problem (1DFAs to 1DFA)

A, B 1DFAs		C 1DFA
n_1, n_2 states	$\Longrightarrow_\pi$	$L(C) =_\pi L$
$L = L(A)L(B)$		*how many states?*

- Upper bound: $e^{\sqrt{n \cdot \ln n}}$, where $n = n_1 + n_2$
  by Parikh equivalent conversion
- Lower bound: $n_1 n_2$ states
  by unary case

[Yu '00]

Concatenation under Parikh equivalence: proof idea

1DFAs A, B
n_1, n_2 states
$L = L(A)L(B)$
$\Sigma = \{a_1, \ldots, a_m\}$

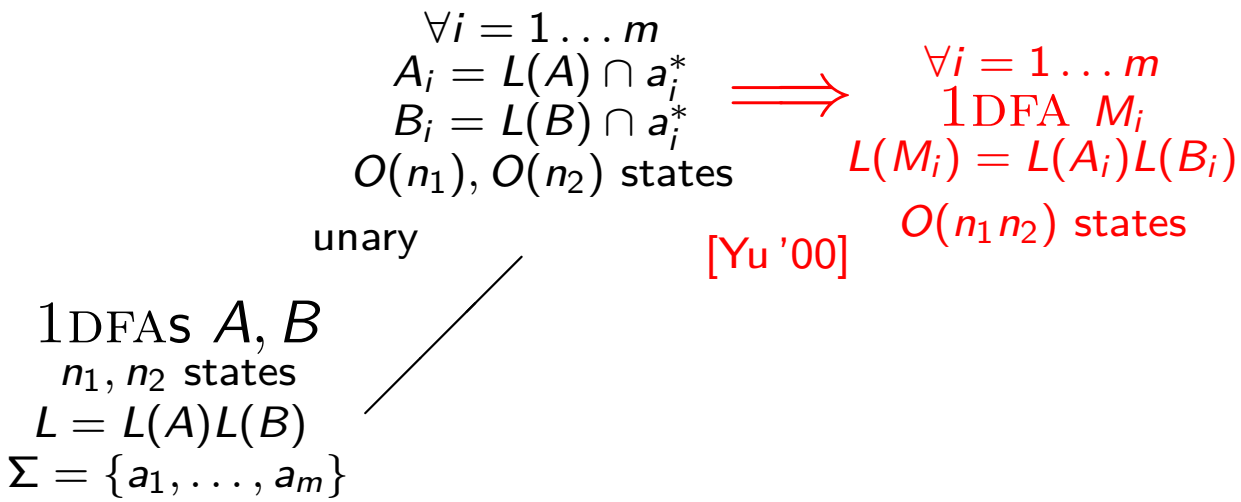
Concatenation under Parikh equivalence: proof idea

$\forall i = 1 \ldots m$
$A_i = L(A) \cap a_i^*$
$B_i = L(B) \cap a_i^*$
$O(n_1), O(n_2)$ states

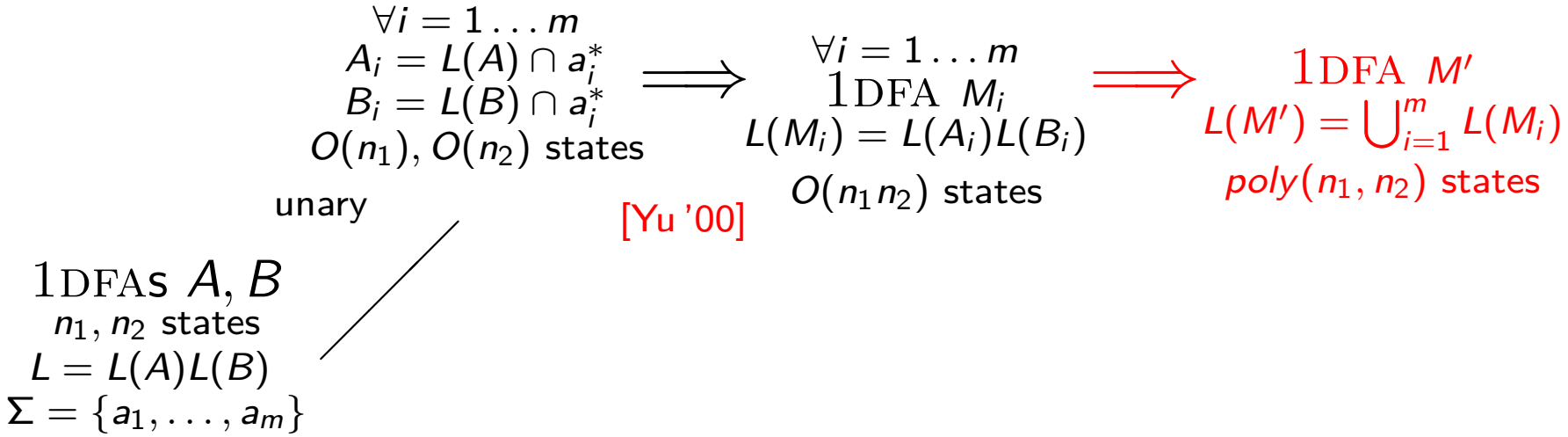
unary

1DFAs A, B
n_1, n_2 states
$L = L(A)L(B)$
$\Sigma = \{a_1, \ldots, a_m\}$

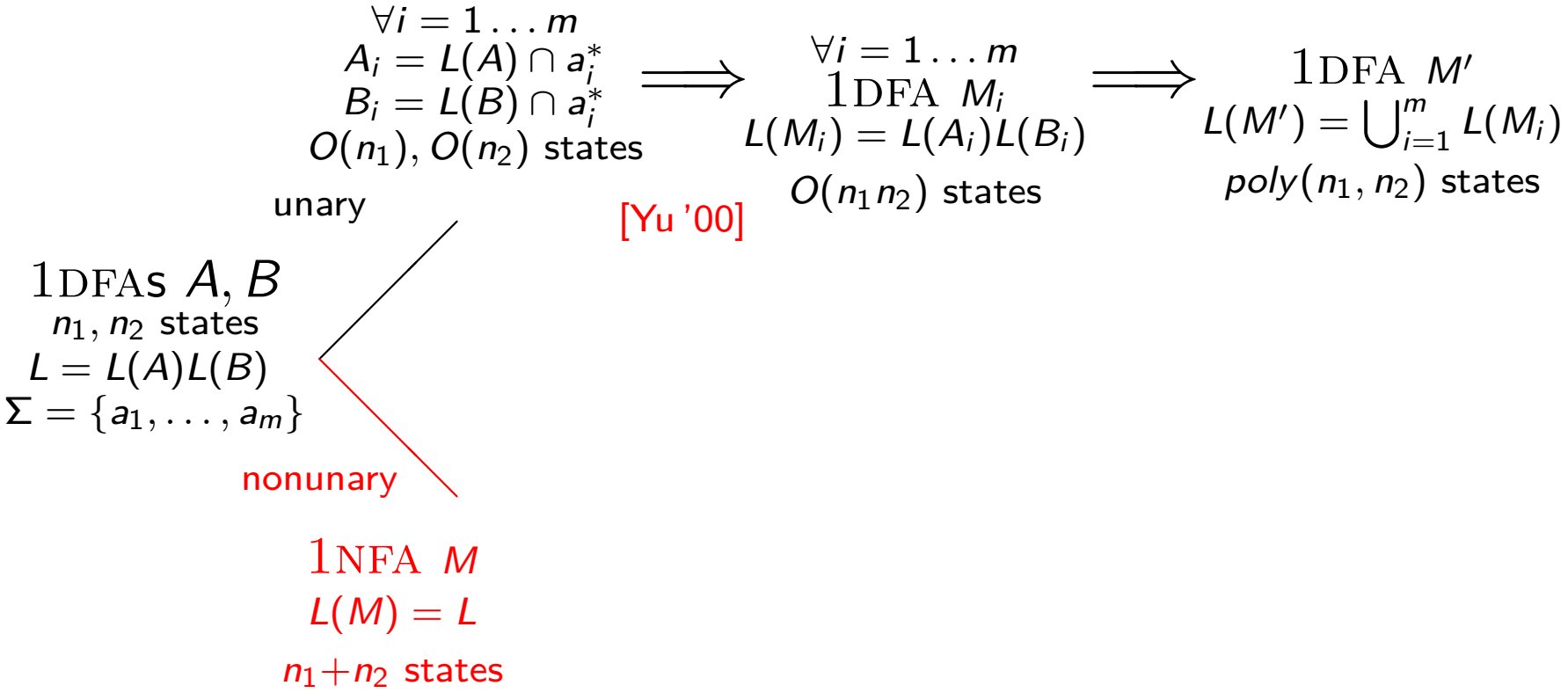
Concatenation under Parikh equivalence: proof idea



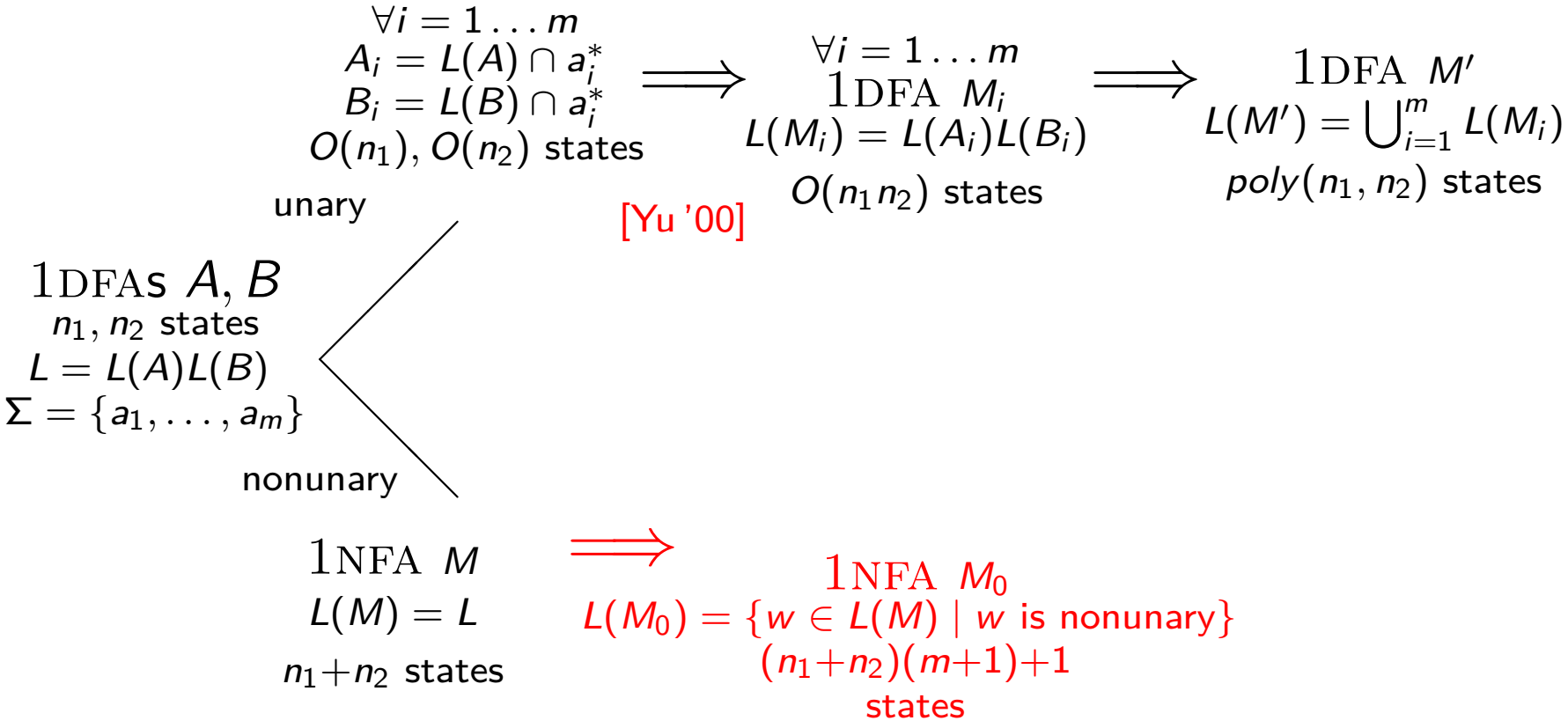
Concatenation under Parikh equivalence: proof idea



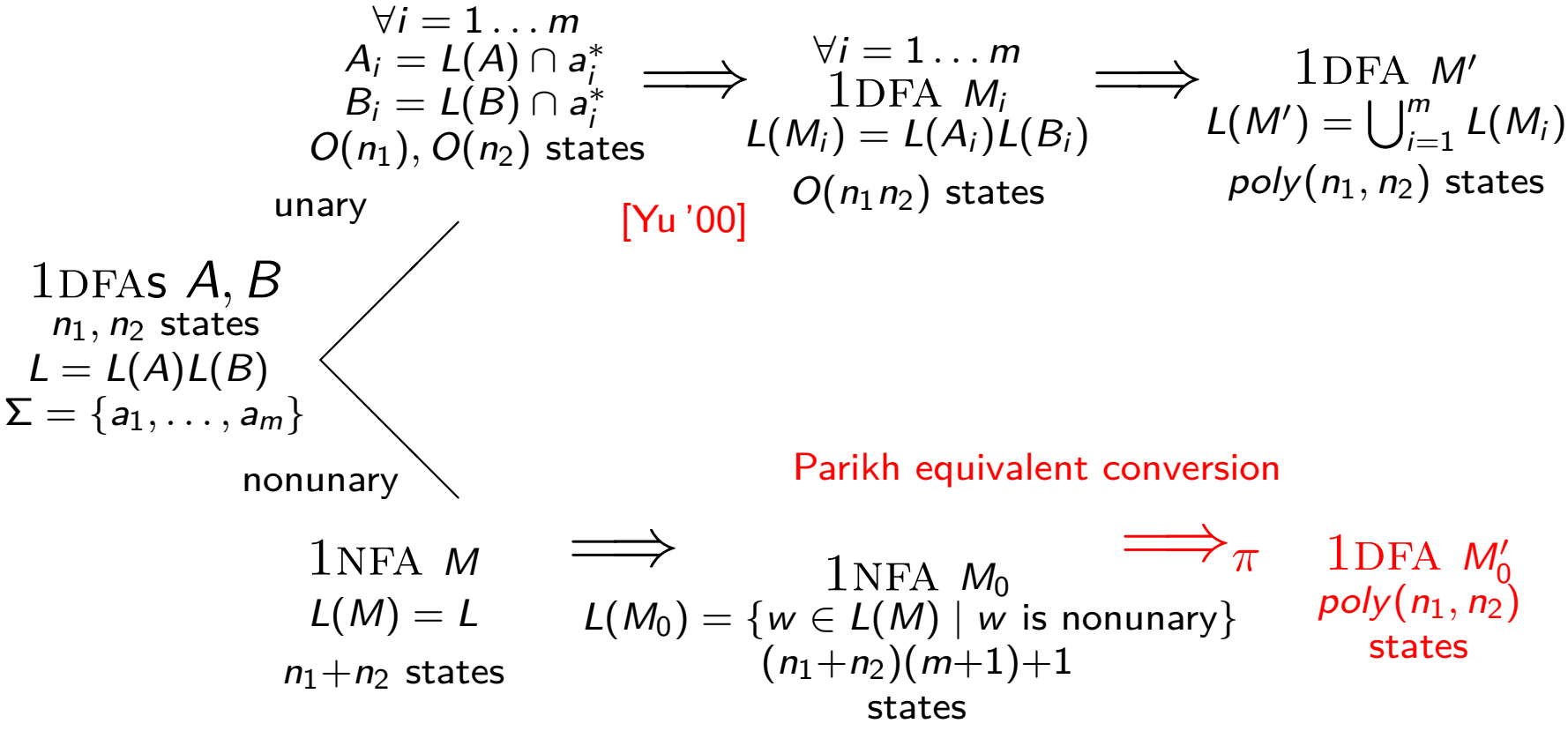
Concatenation under Parikh equivalence: proof idea



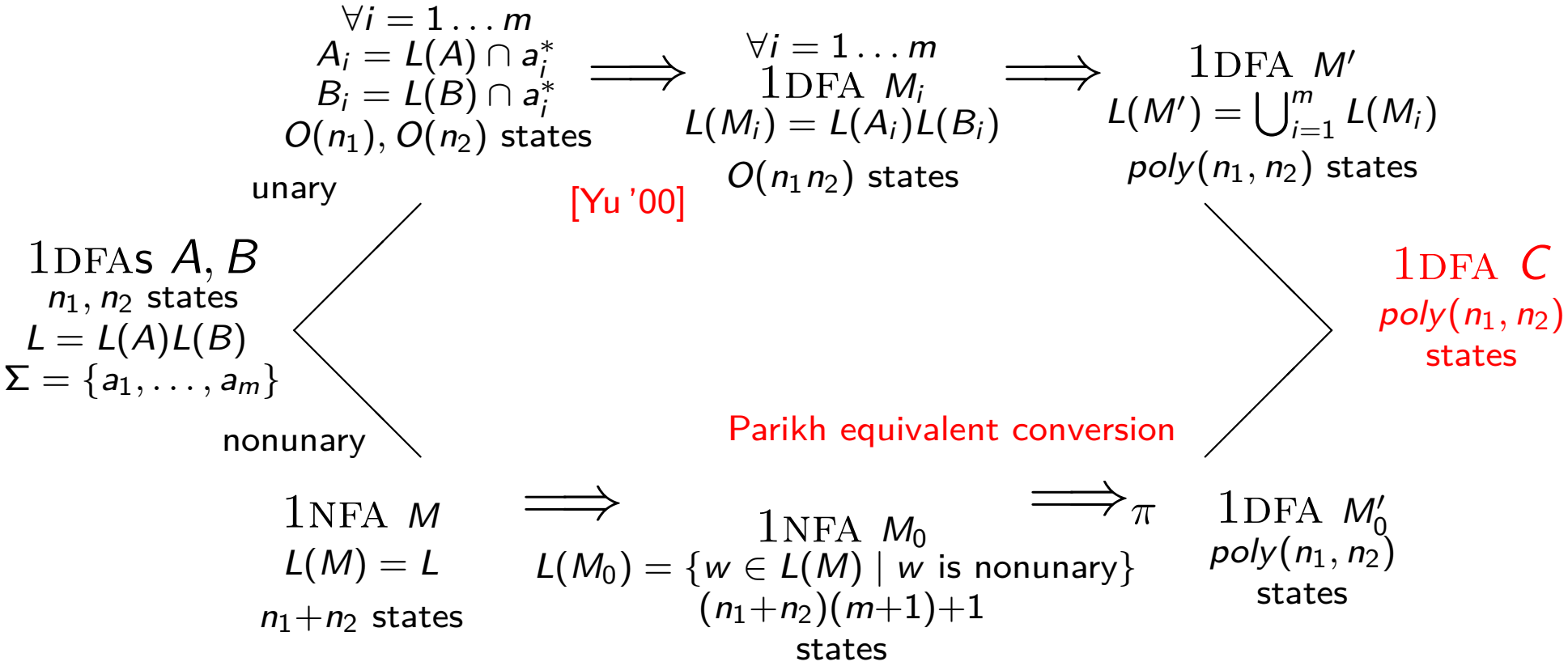
Concatenation under Parikh equivalence: proof idea



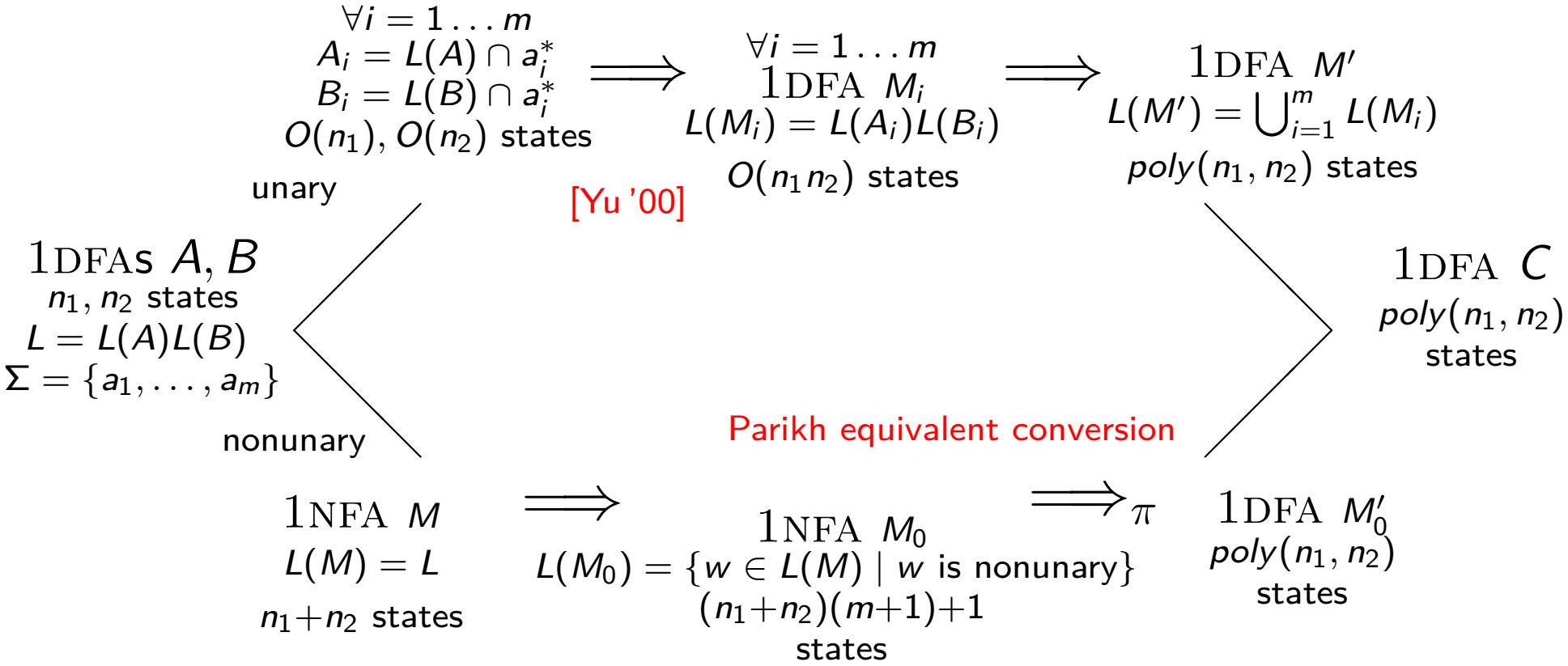
Concatenation under Parikh equivalence: proof idea



Concatenation under Parikh equivalence: proof idea



Concatenation under Parikh equivalence: proof idea



Theorem

Given two 1DFAs *A and B of* n_1 *and* n_2 *states, respectively, there exists a* 1DFA *of polynomial number of states in* n_1 *and* n_2 *that is Parikh equivalent to* $L(A)L(B)$*. Moreover, this cost is tight.*

Operational state complexity under Parikh equivalence

Summary table

Our fourth contribution:

Operation	Standard equivalence	Parikh equivalence
$L_1 \cup L_2$	$n_1 n_2$	$n_1 n_2$
$L_1 \cap L_2$	$n_1 n_2$	$n_1 n_2$
L_1^c	n_1	n_1
$L_1 L_2$	$(2n_1 - 1)2^{n_2 - 1}$	$poly(n_1, n_2)$
L_1^*	$2^{n_1 - 1} + 2^{n_1 - 2}$	$poly(n_1)$
$L_1 \sqcup\!\sqcup L_2$	$2^{n_1 n_2} - 1$	$poly(n_1, n_2)$
L_1^R	2^{n_1}	n_1
$P_{\Sigma_0}(L_1)$	$3 \cdot 2^{n_1 - 2} - 1$	$e^{O(\sqrt{n_1 \cdot \ln n_1})}$

[Yu '00, Campeanu&Salomaa&Yu '02, Yu&Zhuang&Salomaa '94, Jiraskova&Masopust '12]

Intersection: revisited

Non-commutativity with Parikh mapping

Intersection does not commute with Parikh mapping

$\psi(a^+b^+ \cap b^+a^+) \neq \psi(a^+b^+) \cap \psi(b^+a^+)$ holds; in fact,

$$\begin{aligned} \psi(a^+b^+ \cap b^+a^+) &= \emptyset \\ \psi(a^+b^+) \cap \psi(b^+a^+) &= \{(i,j) \mid i,j \geq 1\}. \end{aligned}$$

Problem: intersection

A, B 1DFAs
n_1, n_2 states

$\Longrightarrow$

M 1DFA
$\psi(L(M)) = \psi(L(A)) \cap \psi(L(B))$
How many states needed?

Intersection: revisited

Non-commutativity with Parikh mapping

Intersection does not commute with Parikh mapping

$\psi(a^+b^+ \cap b^+a^+) \neq \psi(a^+b^+) \cap \psi(b^+a^+)$ holds; in fact,

$$\begin{aligned} \psi(a^+b^+ \cap b^+a^+) &= \emptyset \\ \psi(a^+b^+) \cap \psi(b^+a^+) &= \{(i,j) \mid i,j \geq 1\}. \end{aligned}$$

Problem: intersection

A, B 1DFAs
n_1, n_2 states

$\Longrightarrow$

M 1DFA
$\psi(L(M)) = \psi(L(A)) \cap \psi(L(B))$
How many states needed?

Intersection: revisited

Theorem

Let A, B be 1DFA*s with respectively n_1, n_2 states over $\Sigma = \{a_1, \ldots, a_m\}$. There exists a* 1DFA *$M$ whose Parikh map is equal to $\psi(L(A)) \cap \psi(L(B))$ and which contains*

$$O(n^{(2m-1)(3m^3+6m^2)+2} p(n)^{2(3m^3+6m^2)+m})$$

states, where:

- $n = \max\{n_1, n_2\}(m+1)+1$
- $p(n) = O(n^{3m^2} m^{m^2/2+2})$

Proof.

Revisiting the Ginsburg and Spanier's proof [Ginsburg&Spanier '64] of the closure property of semilinear sets under intersection. □

Conclusion and future works

- Quite surprisingly, the costs into Parikh equivalent 1DFA is due to the unary parts of the languages.
- Conversion of the parts consisting of nonunary strings is less expensive.
- Conversions into Parikh equivalent 2DFA are less expensive than the corresponding ones into 1DFA.
- State complexity of some operations on regular languages have been studied under Parikh equivalence.
- For each two 1DFAs A and B, it is possible to obtain a 1DFA with a polynomial number of states, whose accepted language has as Parikh image $\psi(L(A)) \cap \psi(L(B))$.

Conclusion and future works

- Quite surprisingly, the costs into Parikh equivalent 1DFA is due to the unary parts of the languages.
- Conversion of the parts consisting of nonunary strings is less expensive.
- Conversions into Parikh equivalent 2DFA are less expensive than the corresponding ones into 1DFA.
- State complexity of some operations on regular languages have been studied under Parikh equivalence.
- For each two 1DFAs A and B, it is possible to obtain a 1DFA with a polynomial number of states, whose accepted language has as Parikh image $\psi(L(A)) \cap \psi(L(B))$.

Conclusion and future works

- Quite surprisingly, the costs into Parikh equivalent 1DFA is due to the unary parts of the languages.
- Conversion of the parts consisting of nonunary strings is less expensive.
- Conversions into Parikh equivalent 2DFA are less expensive than the corresponding ones into 1DFA.
- State complexity of some operations on regular languages have been studied under Parikh equivalence.
- For each two 1DFAs A and B, it is possible to obtain a 1DFA with a polynomial number of states, whose accepted language has as Parikh image $\psi(L(A)) \cap \psi(L(B))$.

Conclusion and future works

- Quite surprisingly, the costs into Parikh equivalent 1DFA is due to the unary parts of the languages.
- Conversion of the parts consisting of nonunary strings is less expensive.
- Conversions into Parikh equivalent 2DFA are less expensive than the corresponding ones into 1DFA.
- State complexity of some operations on regular languages have been studied under Parikh equivalence.
- For each two 1DFAs A and B, it is possible to obtain a 1DFA with a polynomial number of states, whose accepted language has as Parikh image $\psi(L(A)) \cap \psi(L(B))$.

Conclusion and future works

Complement does not commute with Parikh mapping

$\psi((a^*b^*)^c) \neq (\psi(a^*b^*))^c$ holds; in fact,

$$\begin{aligned} \psi((a^*b^*)^c) &= \{(i,j) \mid i,j \geq 1\} \\ (\psi(a^*b^*))^c &= \emptyset. \end{aligned}$$

Problem: complement (left open!)

A 1DFA
n states

$\Longrightarrow$

M 1DFA
$\psi(L(M)) = (\psi(L(A)))^c$
How many states needed?

Conclusion and future works

Complement does not commute with Parikh mapping

$\psi((a^*b^*)^c) \neq (\psi(a^*b^*))^c$ holds; in fact,

$$\begin{aligned} \psi((a^*b^*)^c) &= \{(i,j) \mid i,j \geq 1\} \\ (\psi(a^*b^*))^c &= \emptyset. \end{aligned}$$

Problem: complement (left open!)

A 1DFA
n states

$\Longrightarrow$

M 1DFA
$\psi(L(M)) = (\psi(L(A)))^c$
How many states needed?

Conclusion and future works

Given a deterministic automaton, finding the smallest Parikh equivalent deterministic automaton is an open question.

Problem: minimization problem

$$\begin{array}{c} A \text{ 1DFA} \\ n_1 \text{ states} \end{array} \quad \Longrightarrow_{\pi} \quad \begin{array}{c} B \text{ 1DFA} \\ L(B) =_{\pi} L(A) \\ n_2 \text{ states s.t. } n_2 \leq n_1 \end{array}$$

Grazie!

Conclusion and future works

Given a deterministic automaton, finding the smallest Parikh equivalent deterministic automaton is an open question.

Problem: minimization problem

$$\begin{array}{c} A \text{ 1DFA} \\ n_1 \text{ states} \end{array} \quad \Longrightarrow_{\pi} \quad \begin{array}{c} B \text{ 1DFA} \\ L(B) =_{\pi} L(A) \\ n_2 \text{ states s.t. } n_2 \leq n_1 \end{array}$$

Grazie!