Non-sequential finite automata

Szilárd Fazekas **Robert Mercaş** Luca Prigioniero

NCMA 2024

Loughborough University

Automata with non-sequential processing modes

Restarting automata [Jančar et al., 1995]

Revolving input automata [Bordihn et al., 2005]

Automata with translucent letters [Nagy and Otto, 2011]

Jumping automata [Meduna and Zemek, 2012]

(Right) One-way jumping automata [Chigahara et al., 2016]

Jumping finite automata [Meduna and Zemek, 2012]

Presented as $M = (Q, \Sigma, R, s, F)$ (just as a regular DFA)

Process the input in arbitrary order

Jumping finite automata [Meduna and Zemek, 2012]

Presented as $M = (Q, \Sigma, R, s, F)$ (just as a regular DFA)

Process the input in arbitrary order

Can match number of letters

Jumping finite automata [Meduna and Zemek, 2012]

Presented as $M = (Q, \Sigma, R, s, F)$ (just as a regular DFA)

Process the input in arbitrary order

Can match number of letters

Accepts permutation-closed semilinear languages (incomparable to REG)

Right one-way jumping finite automaton [Chigahara et al., 2016]

Right one-way jumping finite automaton ($\circlearrowright_R$DFA)

Presented as $M = (Q, \Sigma, R, s, F)$ (just as a regular DFA)

Right one-way jumping finite automaton [Chigahara et al., 2016]

Right one-way jumping finite automaton ($\circlearrowright_R$DFA)

Presented as $M = (Q, \Sigma, R, s, F)$ (just as a regular DFA)

Elements of R are transition rules $\mathbf{p}a \to \mathbf{q} \in R$

Configurations of M are strings in $Q\Sigma^*$

Working

$x, y \in \Sigma^*$, $a \in \Sigma$, $\mathbf{p}, \mathbf{q} \in Q$ and $\mathbf{p}a \to \mathbf{q} \in R$

Working

$x, y \in \Sigma^*$, $a \in \Sigma$, $\mathbf{p}, \mathbf{q} \in Q$ and $\mathbf{p}a \to \mathbf{q} \in R$

The *right one-way jumping relation* $\circlearrowright_R$ over $Q\Sigma^*$, ($\circlearrowright_R$DFA M jumps from configuration $\mathbf{p}xay$ to $\mathbf{q}yx$):

$$\mathbf{p}xay \circlearrowright \mathbf{q}yx$$

if $x \in \{\Sigma \setminus \Sigma_p\}^*$, where

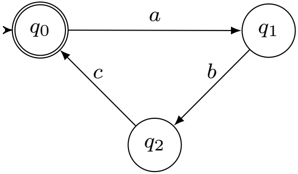
$$\Sigma_p = \{b \in \Sigma \mid (\mathbf{p}, b, \mathbf{p}') \in R \text{ for some } \mathbf{p}' \in Q\}$$

Example

Consider the $\circlearrowright_R$DFA M given in the transition graph below

Accepted language is $\{w \in \{a, b, c\}^* \mid |w|_a = |w|_b = |w|_c\}$

$$\mathbf{q}_0 acbcab \circlearrowright_R \mathbf{q}_1 bcabc \circlearrowright_R \mathbf{q}_2 cabc \circlearrowright_R \mathbf{q}_0 abc \circlearrowright_R \mathbf{q}_1 bc \circlearrowright_R \mathbf{q}_2 c \circlearrowright_R \mathbf{q}_0$$



Accepting power

$\mathbf{REG} \subsetneq \circlearrowright_R\mathbf{DFA}$

$\mathbf{CF}$ and $\circlearrowright_R\mathbf{DFA}$ are incomparable

$\circlearrowright_R\mathbf{DFA} \subsetneq \mathbf{CS}$

$\circlearrowright_R\mathbf{DFA} \subseteq \mathrm{DTIME}(n^2)$

Overview

No unique minimal $\circlearrowright_R$DFA

Different minimal $\circlearrowright_R$DFAs accepting language $\{w \in \{a,b\}^* \mid |w|_b > 0\}$

Overview

No unique minimal $\circlearrowright_R$DFA

Different minimal $\circlearrowright_R$DFAs accepting language $\{w \in \{a,b\}^* \mid |w|_b > 0\}$

$\circlearrowright_R$**DFA** is not closed under "anything" interesting

with End Marker *added to* $\circlearrowright_R$*DFA we have closure on complement*

Overview

No unique minimal $\circlearrowright_R$DFA

Different minimal $\circlearrowright_R$DFAs accepting language $\{w \in \{a,b\}^* \mid |w|_b > 0\}$

$\circlearrowright_R$**DFA** is not closed under "anything" interesting

with End Marker *added to* $\circlearrowright_R$*DFA we have closure on complement*

Some subclasses admit nicer characterisations [Beier and Holzer, 2019]

Overview

No unique minimal $\circlearrowright_R$DFA

Different minimal $\circlearrowright_R$DFAs accepting language $\{w \in \{a,b\}^* \mid |w|_b > 0\}$

$\circlearrowright_R$**DFA** is not closed under "anything" interesting

with End Marker *added to* $\circlearrowright_R$*DFA we have closure on complement*

Some subclasses admit nicer characterisations [Beier and Holzer, 2019]

Good decidability properties [Beier and Holzer, 2020]

Overview

No unique minimal $\circlearrowright_R$DFA

Different minimal $\circlearrowright_R$DFAs accepting language $\{w \in \{a,b\}^* \mid |w|_b > 0\}$

$\circlearrowright_R$**DFA** is not closed under "anything" interesting

with End Marker *added to* $\circlearrowright_R$*DFA we have closure on complement*

Some subclasses admit nicer characterisations [Beier and Holzer, 2019]

Good decidability properties [Beier and Holzer, 2020]

$\circlearrowright_R$ mode studied with more general machine models: NFA, 2DFA, PDA, LBA [Fazekas et al., 2019, Beier and Holzer, 2022, Fazekas et al., 2021]

Jumps and sweeps

A $\circlearrowright_R$DFA transition from $\mathbf{p}ax$ to $\mathbf{q}y$, denoted $\mathbf{p}ax \vdash \mathbf{q}y$:

(i) $\mathbf{p}ax \Rightarrow \mathbf{q}y$, where $x = y$ and $\mathbf{p}a \to \mathbf{q} \in R$ (*sequential trans.*)

(ii) $\mathbf{p}ax \circlearrowright \mathbf{p}xa$, when $a \in \Sigma \setminus \Sigma_p$, $\mathbf{p} = \mathbf{q}$ and $xa = y$ (*a jump*)

w accepted by M if $\mathbf{s}w \models^* \mathbf{f}$, and $L(M) = \{x \in \Sigma^* \mid \exists \mathbf{f} \in F : \mathbf{s}x \models^* \mathbf{f}\}$

Jumps and sweeps

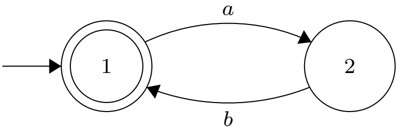
A $\circlearrowright_R$DFA transition from $\mathbf{p}ax$ to $\mathbf{q}y$, denoted $\mathbf{p}ax \vdash \mathbf{q}y$:

(i) $\mathbf{p}ax \Rightarrow \mathbf{q}y$, where $x = y$ and $\mathbf{p}a \to \mathbf{q} \in R$ (*sequential trans.*)

(ii) $\mathbf{p}ax \circlearrowright \mathbf{p}xa$, when $a \in \Sigma \setminus \Sigma_p$, $\mathbf{p} = \mathbf{q}$ and $xa = y$ (*a jump*)

w accepted by M if $\mathbf{s}w \models^* \mathbf{f}$, and $L(M) = \{x \in \Sigma^* \mid \exists \mathbf{f} \in F : \mathbf{s}x \models^* \mathbf{f}\}$

Deficient states: $\mathbf{p} \in Q$ is S-deficient if for every $a \in S \subset \Sigma$ and any $\mathbf{q} \in Q$, we have $\mathbf{p}a \to \mathbf{q} \notin R$



$\circlearrowright_R$DFA accepting $\{w \mid |w|_a = |w|_b\}$

Sweeps:

position :	0	1	2	3	4	5	6	7
input	**a**	**a**	**a**	**a**	**b**	**b**	**b**	**b**
after sweep 1	ε	**a**	**a**	**a**	ε	**b**	**b**	**b**
after sweep 2	ε	ε	**a**	**a**	ε	ε	**b**	**b**
after sweep 3	ε	ε	ε	**a**	ε	ε	ε	**b**
after sweep 4	ε	ε	ε	ε	ε	ε	ε	ε

Computation table for a^4b^4

Sweep complexity [Fazekas et al., 2022]

The *sweep complexity* of an automaton M is $sc_M(n)$ is the maximum number of sweeps M makes on processing inputs $w \in L(M)$ of length n

SWEEP$(f(n))$ class of languages accepted by $\circlearrowright_R$DFA with

$$sc_M(n) = \mathcal{O}(f(n))$$

Starting point

When is the accepted language regular?

Starting point

When is the accepted language regular?

Theorem ([Fazekas and Yamamura, 2016])

For any $\circlearrowright_R$*DFA* $\mathcal{A} = (Q, \Sigma, R, s, F)$ *and any constant* k*, the set of words accepted by* $\mathcal{A}$ *in at most* k *sweeps is regular.*

Starting point

When is the accepted language regular?

Theorem ([Fazekas and Yamamura, 2016])

For any $\circlearrowright_R$*DFA* $\mathcal{A} = (Q, \Sigma, R, s, F)$ *and any constant* k*, the set of words accepted by* $\mathcal{A}$ *in at most* k *sweeps is regular.*

Conjecture

For any $\circlearrowright_R$*DFA* $\mathcal{A} = (Q, \Sigma, R, s, F)$*, the language accepted by* $\mathcal{A}$ *is regular if and only if it has constant sweep complexity.*

Starting point

When is the accepted language regular?

Theorem ([Fazekas and Yamamura, 2016])

For any $\circlearrowright_R$DFA $\mathcal{A} = (Q, \Sigma, R, s, F)$ and any constant k, the set of words accepted by $\mathcal{A}$ in at most k sweeps is regular.

Conjecture

For any $\circlearrowright_R$DFA $\mathcal{A} = (Q, \Sigma, R, s, F)$, the language accepted by $\mathcal{A}$ is regular if and only if it has constant sweep complexity.

Previously known languages accepted by $\circlearrowright_R$DFA had sweep complexity either constant or linear (which is also the upper bound)

Questions regarding sweep complexity [Fazekas et al., 2022]

1. Is the language of each machine with $\omega(1)$ complexity non-regular? 'YES'
2. Is there a machine with sweep complexity between constant and linear, that is, $\omega(1)$ and $o(n)$? 'NO'
3. Is there a *language* with sweep complexity between constant and linear, that is, all machines accepting it have superconstant complexity and at least one has sublinear? 'NO'
4. Is there an upper bound in terms of sweep complexity on machines accepting regular languages? 'YES'
5. Are machines less complex in the case of binary alphabets? 'YES'

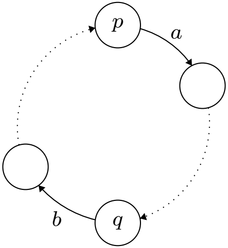
Questions regarding sweep complexity [Fazekas and Mercaş, 2023]

1. Is the language of each machine with $\omega(1)$ complexity non-regular? NO
2. Is there a machine with sweep complexity between constant and linear, that is, $\omega(1)$ and $o(n)$? YES
3. Is there a *language* with sweep complexity between constant and linear, that is, all machines accepting it have superconstant complexity and at least one has sublinear? YES
4. Is there an upper bound in terms of sweep complexity on machines accepting regular languages? NO
5. Are machines less complex in the case of binary alphabets? NO

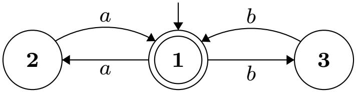
Necessary condition for $\omega(1)$ sweep complexity

Lemma ([Fazekas et al., 2022])

If a $\circlearrowright_R$DFA has superconstant sweep complexity, then it has two reachable and co-reachable states **p** *and* **q** *such that* **p** *is a-deficient,* **q** *is b-deficient, for some* $a, b \in \Sigma$ *with* $a \neq b$*, and* $\mathbf{p}buav \vdash^* \mathbf{q}av \vdash^* \mathbf{p}$*, for some* $u, v \in \Sigma^*$*.*

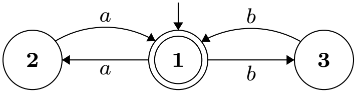


Logarithmic complexity



$$L(\mathcal{A}) = \{w \in \{a,b\}^* \mid |w|_a \equiv 0 \bmod 2, |w|_b \equiv 0 \bmod 2\}$$

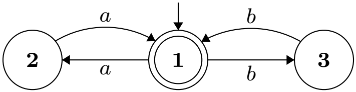
Logarithmic complexity



$$L(\mathcal{A}) = \{w \in \{a,b\}^* \mid |w|_a \equiv 0 \bmod 2, |w|_b \equiv 0 \bmod 2\}$$

$L(\mathcal{A})$ is regular

Logarithmic complexity



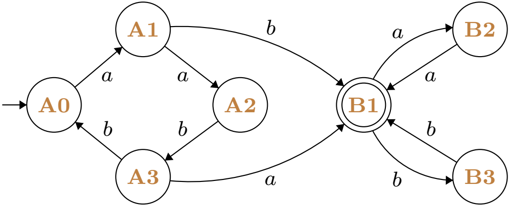
$$L(\mathcal{A}) = \{w \in \{a,b\}^* \mid |w|_a \equiv 0 \bmod 2, |w|_b \equiv 0 \bmod 2\}$$

$L(\mathcal{A})$ is regular

Proposition

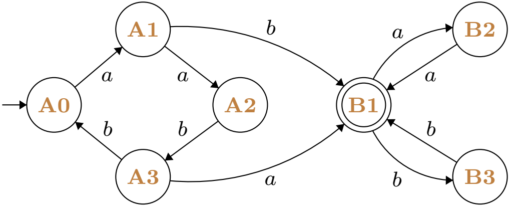
The sweep complexity of $\mathcal{A}$ *is* $\Theta(\log n)$.

Linear complexity



$$L(\mathcal{B}) = \{w \in \{a,b\}^* \mid |w|_a \equiv 1 \mod 2, |w|_b \equiv 1 \mod 2\}$$

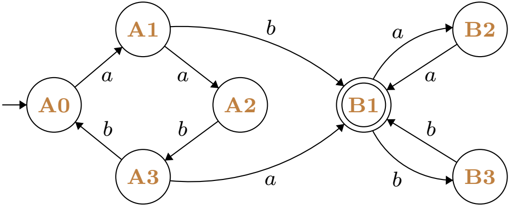
Linear complexity



$$L(\mathcal{B}) = \{w \in \{a,b\}^* \mid |w|_a \equiv 1 \mod 2, |w|_b \equiv 1 \mod 2\}$$

$L(\mathcal{B})$ is regular

Linear complexity



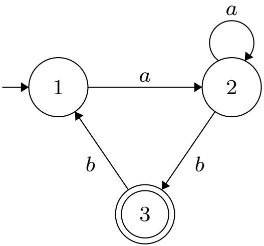
$$L(\mathcal{B}) = \{w \in \{a,b\}^* \mid |w|_a \equiv 1 \mod 2, |w|_b \equiv 1 \mod 2\}$$

$L(\mathcal{B})$ is regular

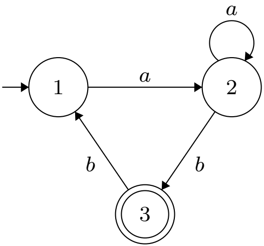
Proposition

The sweep complexity of $\mathcal{B}$ is $\Theta(n)$.

Non-REG language with sublinear sweep complexity



Non-REG language with sublinear sweep complexity



Lemma

The $\circlearrowleft_R$DFA $\mathcal{C}$ accepts a non-regular language.

Lemma

The sweep complexity of $\mathcal{C}$ is $\Theta(\log n)$.

$L(\mathcal{C})$ is not regular

Consider the morphism $\varphi : \{a,b\}^* \to \{a,b\}^*$ with $\varphi(a) = abab$, $\varphi(b) = b$

$L(\mathcal{C})$ is not regular

Consider the morphism $\varphi : \{a, b\}^* \to \{a, b\}^*$ with $\varphi(a) = abab$, $\varphi(b) = b$

$$\varphi(ab) = ababb, \varphi^2(ab) = ababbababbb, \ldots$$

$L(\mathcal{C})$ is not regular

Consider the morphism $\varphi : \{a,b\}^* \to \{a,b\}^*$ with $\varphi(a) = abab$, $\varphi(b) = b$

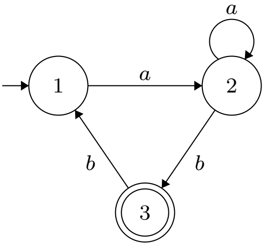
$$\varphi(ab) = ababb, \varphi^2(ab) = ababbababbb, \ldots$$

(By induction) the last block of b's in $\varphi^n(ab)$ has length $n+1$, and is preceded by 2^n blocks of a's separated by blocks of b's

$L(\mathcal{C})$ is not regular

$\varphi : \{a, b\}^* \to \{a, b\}^*$ with $\varphi(a) = abab$, $\varphi(b) = b$

$$\varphi(ab) = ababb, \varphi^2(ab) = ababbababbb, \ldots$$



$$\mathbf{1}\varphi(ab) = \mathbf{1}ababb \Rightarrow^2 \mathbf{3}abb \circlearrowright a\mathbf{3}bb \Rightarrow a\mathbf{1}b \circlearrowright \mathbf{1}ab = \mathbf{1}\varphi^0(ab)$$

Complexity of $\mathcal{C}$ is $\Theta(\log n)$

$\mathcal{C}$ accepts $\varphi^k(ab)$ in $k+1$ sweeps and $|\varphi^k(ab)| \in \mathcal{O}(2^k)$
so sweep complexity is $\Omega(\log n)$

Complexity of $\mathcal{C}$ is $\Theta(\log n)$

$\mathcal{C}$ accepts $\varphi^k(ab)$ in $k+1$ sweeps and $|\varphi^k(ab)| \in \mathcal{O}(2^k)$

so sweep complexity is $\Omega(\log n)$

Within a sweep

- each block of a's is fully processed if a letter is processed from them
- no two consecutive blocks of a can be jumped over
- the number of blocks of a's is reduced by at least half

so at most $\mathcal{O}(\log n)$ sweeps possible

Separating complexity classes

Theorem

$SWEEP(1) \subsetneq SWEEP(\log n)$.

Separating complexity classes

Theorem

$SWEEP(1) \subsetneq SWEEP(\log n)$.

Lemma

Every automaton which accepts $L_{ab} = \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$ *has sweep complexity* $\Theta(n)$.

Separating complexity classes

Theorem

$SWEEP(1) \subsetneq SWEEP(\log n)$.

Lemma

Every automaton which accepts $L_{ab} = \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$ *has sweep complexity* $\Theta(n)$.

Theorem

For any $f : \mathbb{N} \Rightarrow \mathbb{N}$ *with* $f(n) \in o(n)$ *we have* $SWEEP(f(n)) \subsetneq SWEEP(n)$.

Why state complexity? [Fazekas, M., Prigioniero]

- one of the main features of these machines would be their size, as compared to regular DFA

Why state complexity? [Fazekas, M., Prigioniero]

- one of the main features of these machines would be their size, as compared to regular DFA
- we could identify a minimum equivalent DFA when the expressed language is regular

$\circlearrowright_R$DFA to DFA

Theorem

There is an exponential gap in between the representation of an $\circlearrowright_R$DFA and that of a DFA accepting the same regular language.

$\circlearrowright_R$DFA to DFA

Theorem

There is an exponential gap in between the representation of an $\circlearrowright_R$DFA and that of a DFA accepting the same regular language.

$$L_n = \{x_1 x_2 \cdots x_k \$ x \mid x_i, x \in \{a,b\}^n \text{ and } \exists j \text{ such that } x_j = x, j \in [k]\}$$

$\circlearrowright_R$DFA to DFA

Theorem

There is an exponential gap in between the representation of an $\circlearrowright_R$DFA and that of a DFA accepting the same regular language.

$$L_n = \{x_1 x_2 \cdots x_k \$ x \mid x_i, x \in \{a,b\}^n \text{ and } \exists j \text{ such that } x_j = x, j \in [k]\}$$

A $\circlearrowright_R$DFA accepts L_n by jumping the prefix of the input until reaching \$. Then stores x, and then, computing the jumped prefix, it compares each factor of length n with the string x stored in the finite control.

approach requires 2^n states (to store x)

$\circlearrowright_R$DFA to DFA

Theorem

There is an exponential gap in between the representation of an $\circlearrowright_R$DFA and that of a DFA accepting the same regular language.

$$L_n = \{x_1 x_2 \cdots x_k \$ x \mid x_i, x \in \{a,b\}^n \text{ and } \exists j \text{ such that } x_j = x, j \in [k]\}$$

A $\circlearrowright_R$DFA accepts L_n by jumping the prefix of the input until reaching \$. Then stores x, and then, computing the jumped prefix, it compares each factor of length n with the string x stored in the finite control.

approach requires 2^n states (to store x)

A DFA accepts L_n by storing in its finite control the set S of n-length factors $x_1 x_2 \cdots x_k$ and, after reading \$, verifying that x is contained in S.

DFA requires double exponentially, in n, many states (to store S)

NFA to $\circlearrowright_R$DFA still exponential

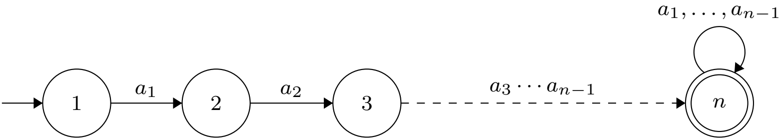
The $\Theta(2^n)$ bound is straightforward

NFA to $\circlearrowright_R$DFA still exponential

The $\Theta(2^n)$ bound is straightforward

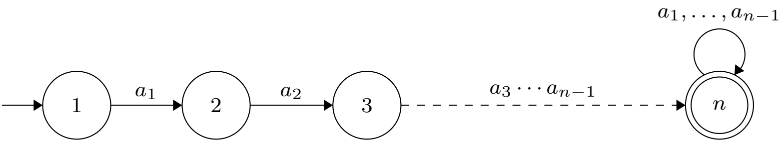
- Lower bound: classical example of n-th letter from the end being a
- Upper bound: follows from the fact that each DFA is an $\circlearrowright_R$DFA

$\circlearrowright_R$DFA to NFA ($\Sigma \in \mathcal{O}(n)$) still exponential



$\circlearrowright_R$DFA $\mathcal{T}$ from [Fazekas and Yamamura, 2016]

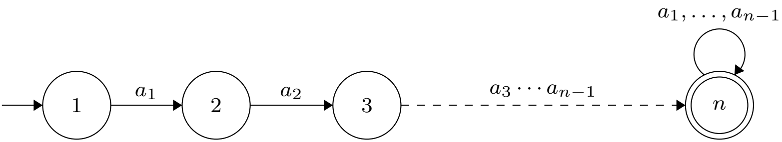
$\circlearrowright_R$DFA to NFA ($\Sigma \in \mathcal{O}(n)$) still exponential



$\circlearrowright_R$DFA $\mathcal{T}$ from [Fazekas and Yamamura, 2016]

- $\circlearrowright_R$DFA $\mathcal{T}$ accepts the regular language of words in which each letter of the alphabet $\{a_1, \ldots, a_{n-1}\}$ occurs at least once

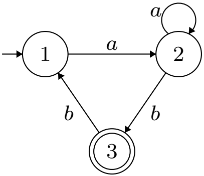
$\circlearrowright_R$DFA to NFA ($\Sigma \in \mathcal{O}(n)$) still exponential



$\circlearrowright_R$DFA $\mathcal{T}$ from [Fazekas and Yamamura, 2016]

- $\circlearrowright_R$DFA $\mathcal{T}$ accepts the regular language of words in which each letter of the alphabet $\{a_1, \ldots, a_{n-1}\}$ occurs at least once
- $\circlearrowright_R$DFA $\mathcal{T}$ has n states, but any NFA needs 2^{n-1} states to keep track of which letters have already occurred in the input

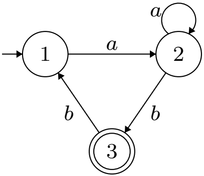
$\circlearrowright_R$DFA to NFA ($|\Sigma| = 3$) still exponential



$\circlearrowright_R$DFA $\mathcal{C} \notin REG$ of sweep complexity $\Theta(\log n)$ [Fazekas and Mercaş, 2023]

- $L(\mathcal{C}) \cap \Sigma^* b^+ = \{wb^n \mid$ number of blocks of a's in w is $\Omega(2^n)\}$

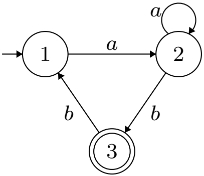
$\circlearrowright_R$DFA to NFA ($|\Sigma| = 3$) still exponential



$\circlearrowright_R$DFA $\mathcal{C} \notin REG$ of sweep complexity $\Theta(\log n)$ [Fazekas and Mercaş, 2023]

- $L(\mathcal{C}) \cap \Sigma^* b^+ = \{wb^n \mid \text{number of blocks of } a\text{'s in } w \text{ is } \Omega(2^n)\}$
- Words in $L(\mathcal{C})$ accepted in $\mathcal{O}(\log n)$ sweeps (some $\Omega(\log n)$ sweeps)

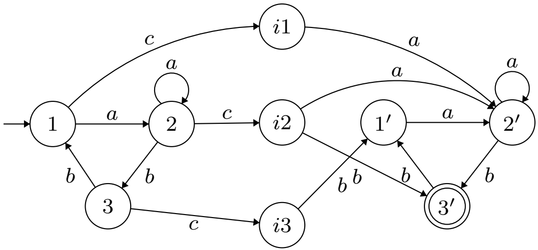
$\circlearrowright_R$DFA to NFA ($|\Sigma| = 3$) still exponential



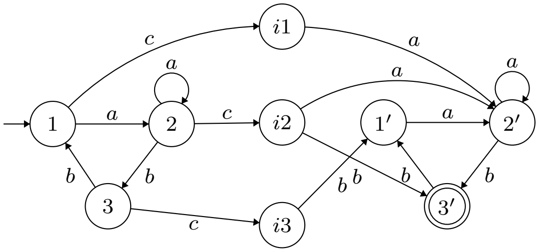
$\circlearrowright_R$DFA $\mathcal{C} \notin REG$ of sweep complexity $\Theta(\log n)$ [Fazekas and Mercaş, 2023]

- $L(\mathcal{C}) \cap \Sigma^* b^+ = \{wb^n \mid$ number of blocks of a's in w is $\Omega(2^n)\}$
- Words in $L(\mathcal{C})$ accepted in $\mathcal{O}(\log n)$ sweeps (some $\Omega(\log n)$ sweeps)
- Concatenate $n+1$ copies of $\mathcal{C}$ using a new symbol c to label the transitions between copies

$\circlearrowright_R$DFA to NFA ($|\Sigma| = 3$) still exponential

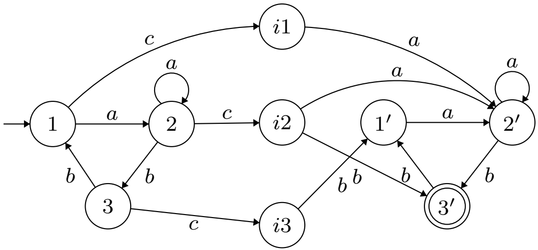


$\circlearrowright_R$DFA to NFA ($|\Sigma| = 3$) still exponential



The resulting machine accepts $K \in REG$ and
$K \cap \{a, b\}^* b^{2n} c^n = \{w b^{2n} c^n \mid \text{number of blocks of } a\text{'s in } w \text{ is } \Omega(2^n)\}$

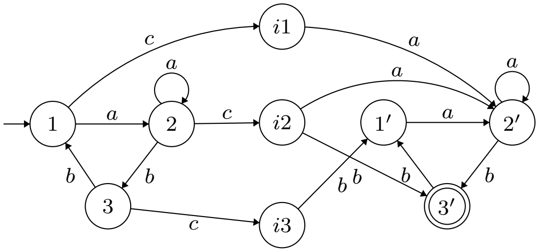
$\circlearrowright_R$DFA to NFA ($|\Sigma| = 3$) still exponential



The resulting machine accepts $K \in REG$ and
$K \cap \{a, b\}^* b^{2n} c^n = \{w b^{2n} c^n \mid \text{number of blocks of } a\text{'s in } w \text{ is } \Omega(2^n)\}$

An NFA accepting the language counts the a's and compares to count of b's

$\circlearrowright_R$DFA to NFA ($|\Sigma| = 3$) still exponential



The resulting machine accepts $K \in REG$ and
$K \cap \{a,b\}^* b^{2n} c^n = \{w b^{2n} c^n \mid \text{number of blocks of } a\text{'s in } w \text{ is } \Omega(2^n)\}$

An NFA accepting the language counts the a's and compares to count of b's, so it needs $\Omega(2^n)$ states (versus the $3(n+1)$ states of $\circlearrowright_R$DFA)

What is next?

- Are there machines with arbitrary (constructible) sublinear complexity ($\Theta(\log^k n)$ and $\Theta(n^\epsilon)$)?
- Is it decidable, given a machine or language and a function $f(n)$, whether the machine/language has $\Theta(f(n))$ sweep complexity?
- Solve the open problems regarding equivalence and regularity
- Nondeterminism...
- Smaller alphabet size for the descriptional complexity trade-offs

What is next?

- Are there machines with arbitrary (constructible) sublinear complexity ($\Theta(\log^k n)$ and $\Theta(n^\epsilon)$)?
- Is it decidable, given a machine or language and a function $f(n)$, whether the machine/language has $\Theta(f(n))$ sweep complexity?
- Solve the open problems regarding equivalence and regularity
- Nondeterminism...
- Smaller alphabet size for the descriptional complexity trade-offs

Thank you!

References I

Beier, S. and Holzer, M. (2019).
Properties of right one-way jumping finite automata.
Theoretical Computer Science, 798:78 – 94.

Beier, S. and Holzer, M. (2020).
Decidability of right one-way jumping finite automata.
International Journal of Foundations of Computer Science, 31(06):805–825.

Beier, S. and Holzer, M. (2022).
Nondeterministic right one-way jumping finite automata.
Information and Computation, 284:104687.

Bordihn, H., Holzer, M., and Kutrib, M. (2005).
Revolving-input finite automata.
In de Felice, C. and Restivo, A., editors, *DLT 2005*, volume 3572 of *LNCS*, pages 168–179.

Chigahara, H., Fazekas, S. Z., and Yamamura, A. (2016).
One-way jumping finite automata.
International Journal of Foundations of Computer Science, 27(3):391–405.

Fazekas, S. Z., Hoshi, K., and Yamamura, A. (2019).
Enhancement of automata with jumping modes.
In Castillo-Ramirez, A. and de Oliveira, P. P. B., editors, *AUTOMATA 2019*, pages 62–76. Springer.

Fazekas, S. Z., Hoshi, K., and Yamamura, A. (2021).
Two-way deterministic automata with jumping mode.
Theoretical Computer Science, 864:92–102.

References II

Fazekas, S. Z. and Mercaş, R. (2023).
Sweep complexity revisited.
In Nagy, B., editor, *CIAA*, volume 14151 of *LNCS*, pages 116–127. Springer.

Fazekas, S. Z., Mercaş, R., and Wu, O. (2022).
Complexities for jumps and sweeps.
Journal of Automata, Languages and Combinatorics, 27(1-3):131–149.

Fazekas, S. Z. and Yamamura, A. (2016).
On regular languages accepted by one-way jumping finite automata.
In *NCMA, short papers*, pages 7–14.

Jančar, P., Mráz, F., Plátek, M., and Vogel, J. (1995).
Restarting automata.
In Reichel, H., editor, *Fundamentals of Computation Theory*, pages 283–292.

Meduna, A. and Zemek, P. (2012).
Jumping finite automata.
International Journal of Foundations of Computer Science, 23(7):1555–1578.

Nagy, B. and Otto, F. (2011).
Finite-state acceptors with translucent letters.
In *ICAART 2011*, pages 3–13.

Extra I

Problem

Given an $\circlearrowright_R$*DFA* A *and word* w*, does there exist a* $u \in L(A)$ *such that* $w \leq_s u$*?*

Extra I

Problem

Given an $\circlearrowright_R$*DFA* A *and word* w*, does there exist a* $u \in L(A)$ *such that* $w \leq_s u$*?*

Theorem (Fazekas, Koss, Manea, M., Specht)

Given a $\circlearrowright_R$*DFA (or DFAwtl)* A *and a word* w *it is decidable (in NP time) whether there exists a word* $u \in L(A)$ *such that* $w \leq_s v$*?*

Exponential upper bound on the length of factors between letters of the subsequence

Done by iteratively inserting factors of bounded length (wrt number of states) between letters in each sweep, bottom up

Extra I

Problem

Given an $\circlearrowright_R$DFA A and word w, does there exist a $u \in L(A)$ such that $w \leq_s u$?

Theorem (Fazekas, Koss, Manea, M., Specht)

Given a $\circlearrowright_R$DFA (or DFAwtl) A and a word w it is decidable (in NP time) whether there exists a word $u \in L(A)$ such that $w \leq_s v$?

Exponential upper bound on the length of factors between letters of the subsequence
Done by iteratively inserting factors of bounded length (wrt number of states) between letters in each sweep, bottom up

Theorem ([Beier and Holzer, 2020, Theorem 6])

Given a $\circlearrowright_R$DFA A and a word w, it is decidable in PSPACE whether there exists a $v \in L(A)$ such that (1) the word w is a prefix of v, (2) the word w is a sufix of v, (3) the word w is a factor of v, and (4) the word w is a subsequence of v.

Extra II

Theorem (Fazekas, Koss, Manea, M., Specht)

Given a $\circlearrowright_R$DFA A and a word w, both defined over a binary alphabet, it is decidable in NP time whether there exists a word $u \in L(A)$ such that $w \leq_s v$?

Assume that there exists $v \in L(A)$ with $w \leq_s v$

Extra II

Theorem (Fazekas, Koss, Manea, M., Specht)

Given a $\circlearrowright_R$DFA A and a word w, both defined over a binary alphabet, it is decidable in NP time whether there exists a word $u \in L(A)$ such that $w \leq_s v$?

Assume that there exists $v \in L(A)$ with $w \leq_s v$

For every $w \in \{a, b\}^k$, for every $v \in \{a, b\}^*$ with at least $2k$ blocks $w \leq_s v$

Extra II

Theorem (Fazekas, Koss, Manea, M., Specht)

Given a $\circlearrowright_R$DFA A and a word w, both defined over a binary alphabet, it is decidable in NP time whether there exists a word $u \in L(A)$ such that $w \leq_s v$?

Assume that there exists $v \in L(A)$ with $w \leq_s v$

For every $w \in \{a,b\}^k$, for every $v \in \{a,b\}^*$ with at least $2k$ blocks $w \leq_s v$

If DFA accepts words with arbitrarily many blocks, then answer YES

Extra II

Theorem (Fazekas, Koss, Manea, M., Specht)

Given a $\circlearrowright_R$DFA A and a word w, both defined over a binary alphabet, it is decidable in NP time whether there exists a word $u \in L(A)$ such that $w \leq_s v$?

Assume that there exists $v \in L(A)$ with $w \leq_s v$

For every $w \in \{a, b\}^k$, for every $v \in \{a, b\}^*$ with at least $2k$ blocks $w \leq_s v$

If DFA accepts words with arbitrarily many blocks, then answer YES

If not, then A has no loop with binary label
Straighten each loop with label b^ℓ to a path of length $k\ell$ and analyse the resulting machine which accepts a finite language

(reduction to finite language case)

Extra II

Theorem (Fazekas, Koss, Manea, M., Specht)

Given a $\circlearrowright_R$DFA A and a word w, both defined over a binary alphabet, it is decidable in NP time whether there exists a word $u \in L(A)$ such that $w \leq_s v$?

Assume that there exists $v \in L(A)$ with $w \leq_s v$

For every $w \in \{a, b\}^k$, for every $v \in \{a, b\}^*$ with at least $2k$ blocks $w \leq_s v$

If DFA accepts words with arbitrarily many blocks, then answer YES

If not, then A has no loop with binary label
Straighten each loop with label b^ℓ to a path of length $k\ell$ and analyse the resulting machine which accepts a finite language
(reduction to finite language case)

When language accepted by A is finite, then the longest word accepted has at most as many letters as the number of states of A,
the length of the witness v is also upper bounded by $|A|$

Extra III

Problem

Given a OWJFA A and a word w, do we have for all $u \in L(A)$ that $w \leq_s v$?

Extra III

Problem

Given a OWJFA A and a word w, do we have for all $u \in L(A)$ that $w \leq_s v$?

Lemma ([Beier and Holzer, 2020, Lemma 14])

Let A be an $\circlearrowright_R$DFA and $L \subset \Sigma^*$ be a finite language. Then it is decidable whether (1) $L(A) \cap L = \emptyset$, (2) $L \subseteq L(A)$, (3) $L(A) \subseteq L$; and (4) $L = L(A)$.