Polynomial Complementation of Nondeterministic Two-Way Finite Automata by 1-Limited Automata

Bruno Guillon[1] Luca Prigioniero[2] Javad Taheri[1]

[1]LIMOS, CNRS, {Université, INP} Clermont Auvergne

[2]Department of Computer Science, Loughborough University

STACS 2026 — March 11th

The Sakoda & Sipser conjecture

n **states** NFA → 2^n **states** DFA

The Sakoda & Sipser conjecture

The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

***n* states**
2NFA ⟶ 2DFA

The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

n states
2NFA → ??? → 2DFA

The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

n **states**

2NFA —??? → 2DFA

$$L_n = \{u\, x\, \overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$$

$\triangleright$	a	b	b	a	a	a	b	b	a	b	b	b	a	$\triangleleft$

The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

n **states**
2NFA ——???——→ 2DFA

fixed
$n = 3$

$$L_n = \{u\,x\,\overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$$

▷	*a*	*b*	*b*	*a*	*a*	*a*	*b*	*b*	*a*	*b*	*b*	*b*	*a*	◁

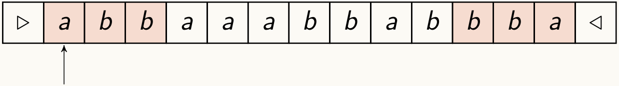
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

n **states**
2NFA ——???——> 2DFA

fixed
$n = 3$

$L_n = \{u\, x\, \overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$



state
$i = 0$
$\sigma =$

The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

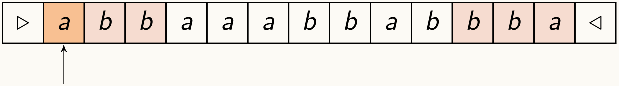
n **states**

2NFA ——— ??? ———→ 2DFA

fixed

$n = 3$

$$L_n = \{u\,x\,\overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$$

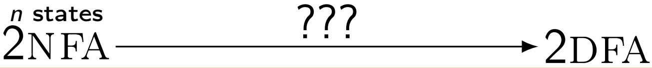


state

$i = 0$

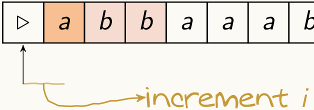
$\sigma = a$

2FA: an automaton that can move its head back and forth



fixed
$n = 3$

$L_n = \{u\, x\, \overleftarrow{u} \mid u \in \{a, b\}^n,\ x \in \{a, b\}^*\}$



state
$i = 1$
$\sigma = a$

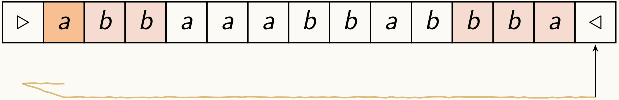
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

n states
2NFA ——???——→ 2DFA

fixed
$n = 3$

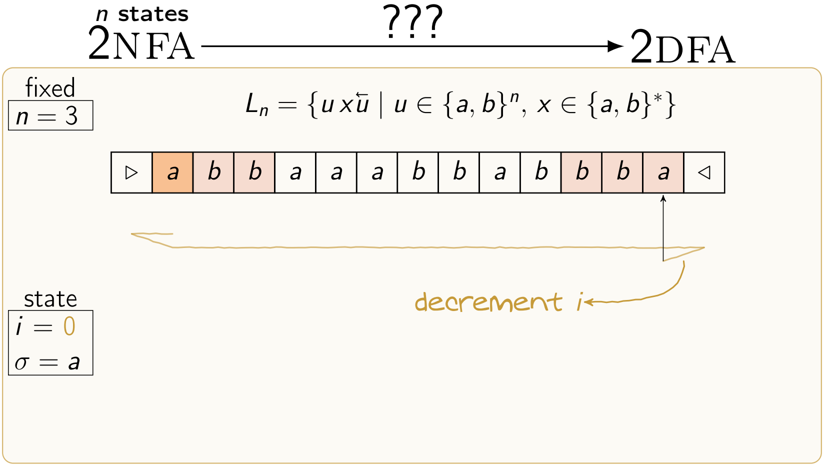
$L_n = \{u\, x\, \overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$



state
$i = 1$
$\sigma = a$

The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth



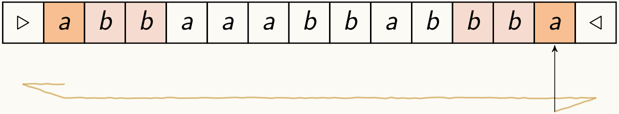
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

n states 2NFA ——???——→ 2DFA

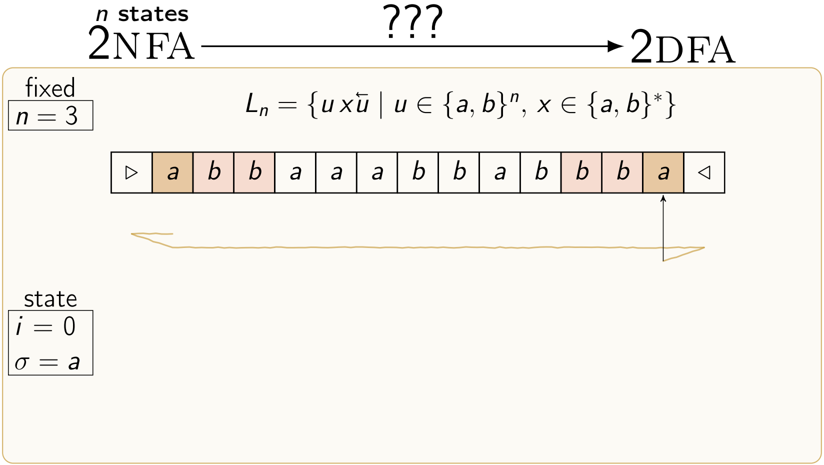
fixed
$n = 3$

$$L_n = \{u\,x\,\overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$$



state
$i = 0$
$\sigma = a$

2FA: an automaton that can move its head back and forth



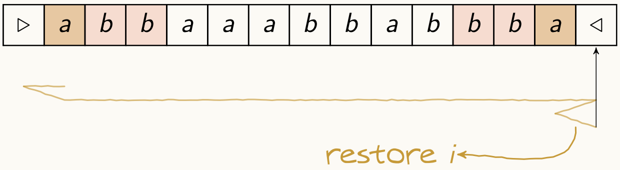
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

n **states**
2NFA ——— ??? ———→ 2DFA

fixed
$n = 3$

$$L_n = \{ u\,x\,\overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^* \}$$



state
$i = 1$
$\sigma = a$

The Sakoda & Sipser conjecture

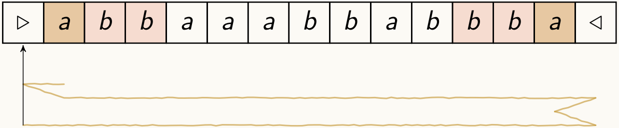
2FA: an automaton that can move its head back and forth

n **states**

2NFA → ??? → 2DFA

fixed
$n = 3$

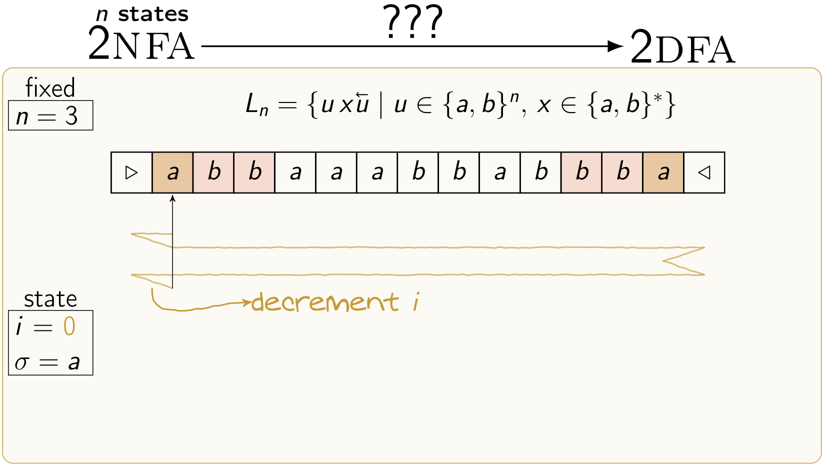
$$L_n = \{u\,x\,\overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$$



state
$i = 1$
$\sigma = a$

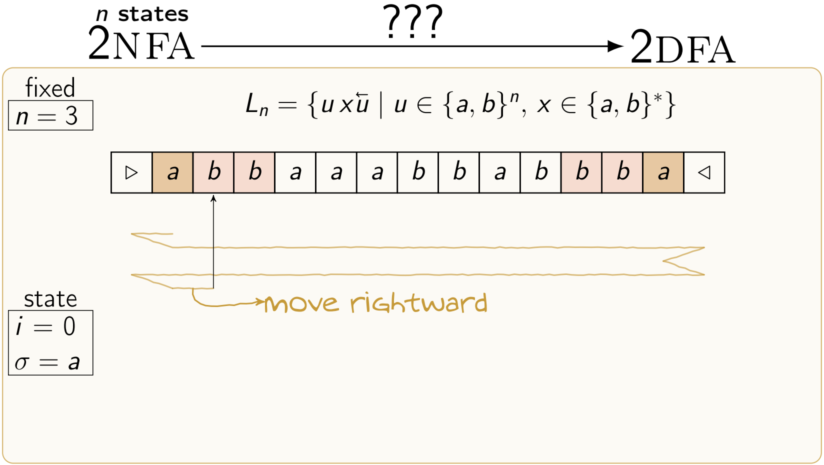
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth



The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth



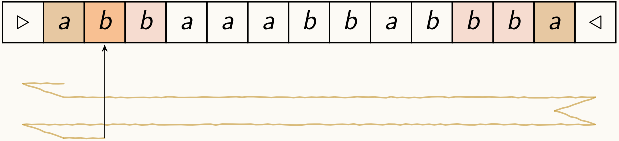
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

n states
2NFA ——???——→ 2DFA

fixed
$n = 3$

$$L_n = \{u\,x\,\overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$$



state
$i = 0$
$\sigma = b$

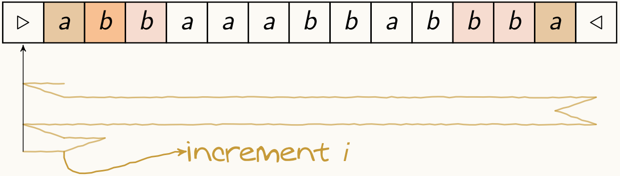
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

***n* states**
2NFA ——— ??? ———→ 2DFA

fixed
$n = 3$

$$L_n = \{u\,x\,\overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$$



state
$i = 2$
$\sigma = b$

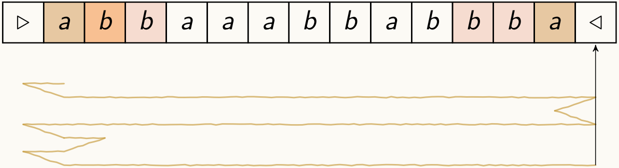
2FA: an automaton that can move its head back and forth

n **states**

2NFA → ??? → 2DFA

fixed
$n = 3$

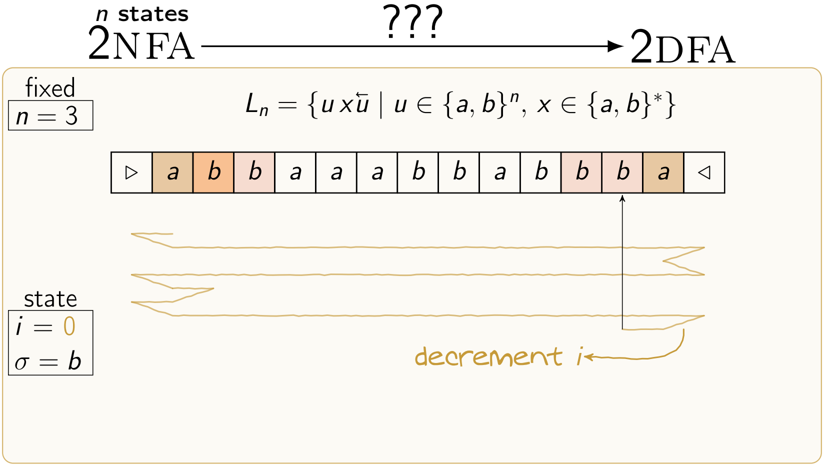
$L_n = \{u\,x\,\overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$



state
$i = 2$
$\sigma = b$

The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth



The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

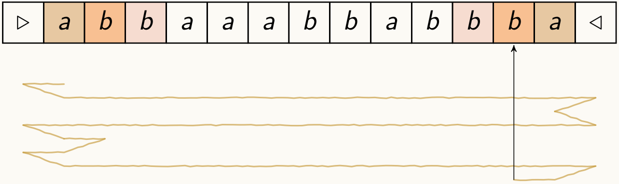
n **states**

2NFA —???→ 2DFA

fixed

$n = 3$

$$L_n = \{u\, x \overleftarrow{u} \mid u \in \{a, b\}^n,\ x \in \{a, b\}^*\}$$



state

$i = 0$

$\sigma = b$

The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

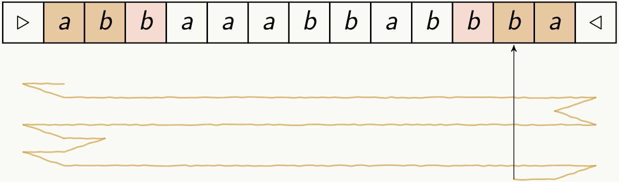
n **states**

2NFA ——???——→ 2DFA

fixed

$n = 3$

$L_n = \{u\, x\, \overleftarrow{u} \mid u \in \{a, b\}^n,\ x \in \{a, b\}^*\}$



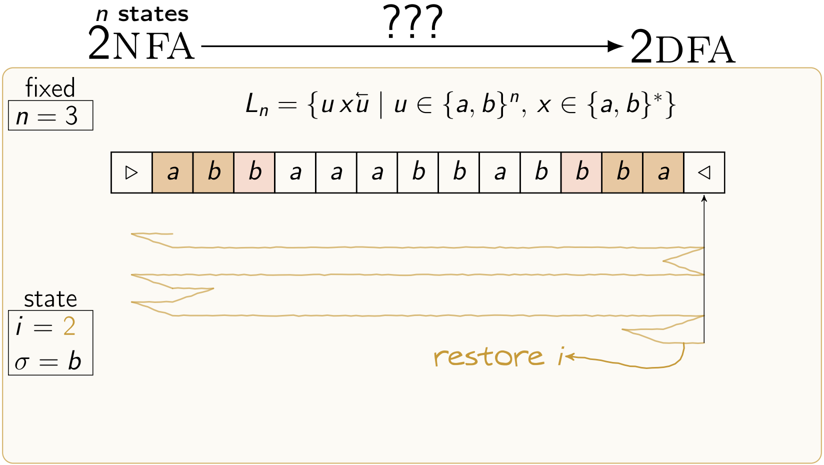
state

$i = 0$

$\sigma = b$

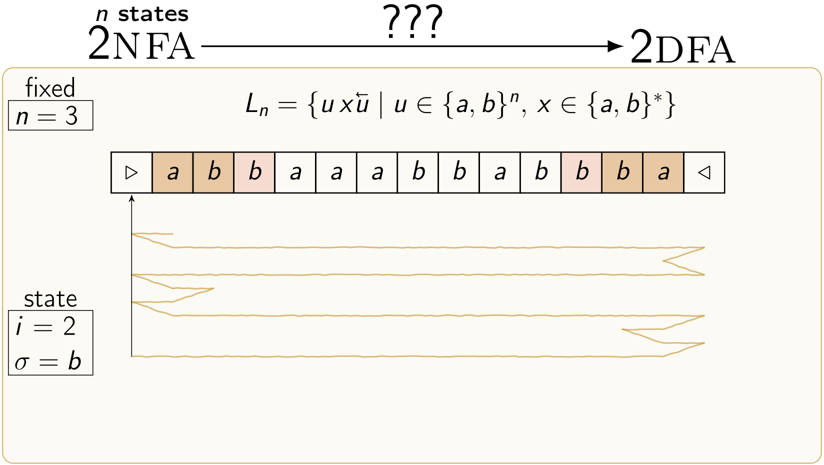
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth



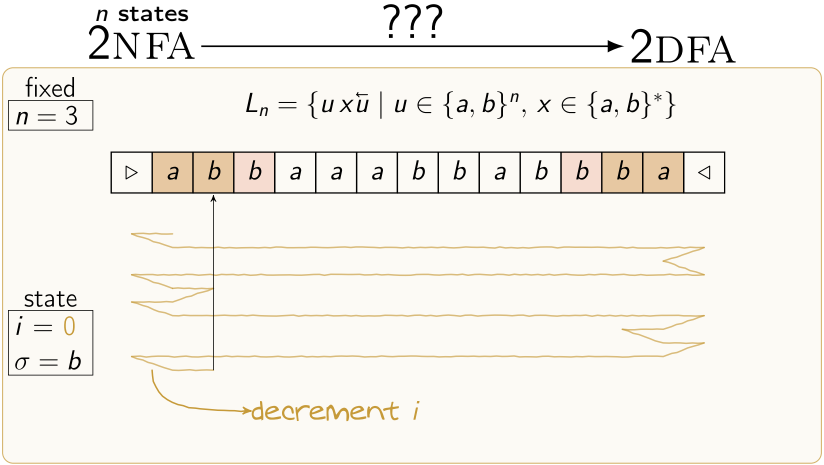
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth



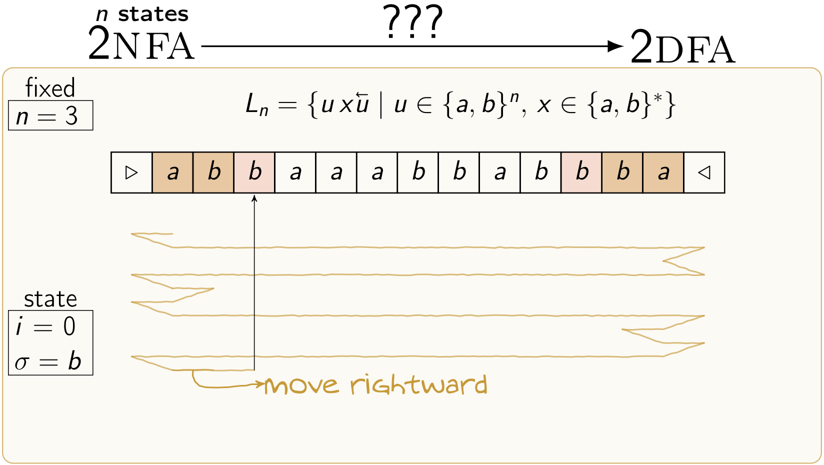
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth



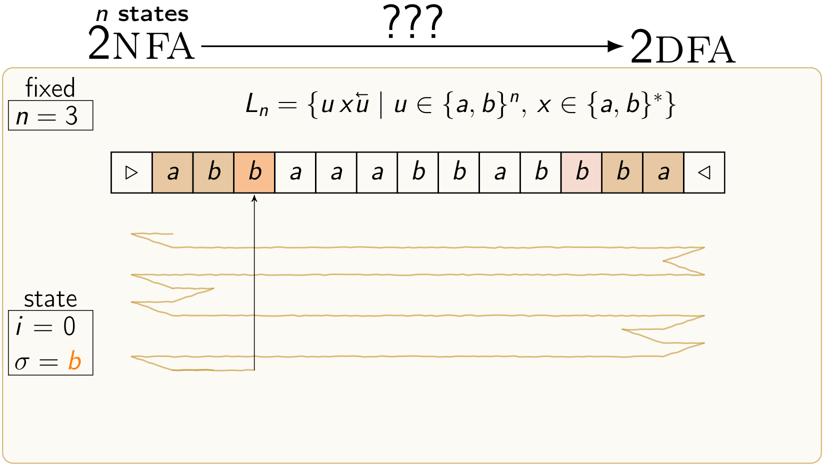
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth



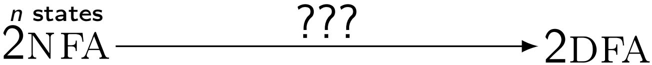
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

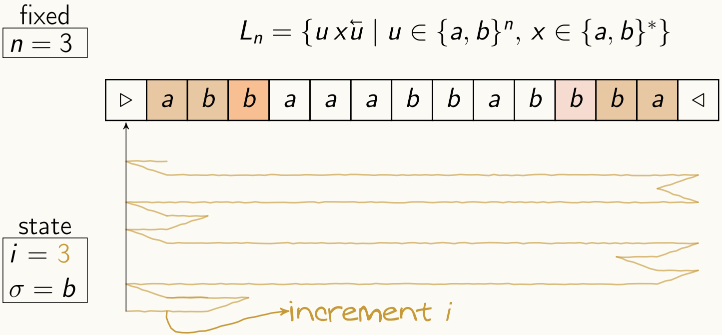


The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth



$$L_n = \{u\,x\,\overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$$



The Sakoda & Sipser conjecture

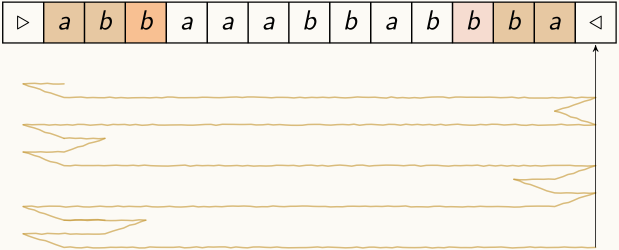
2FA: an automaton that can move its head back and forth

n **states**

2NFA ——???——→ 2DFA

fixed
$n = 3$

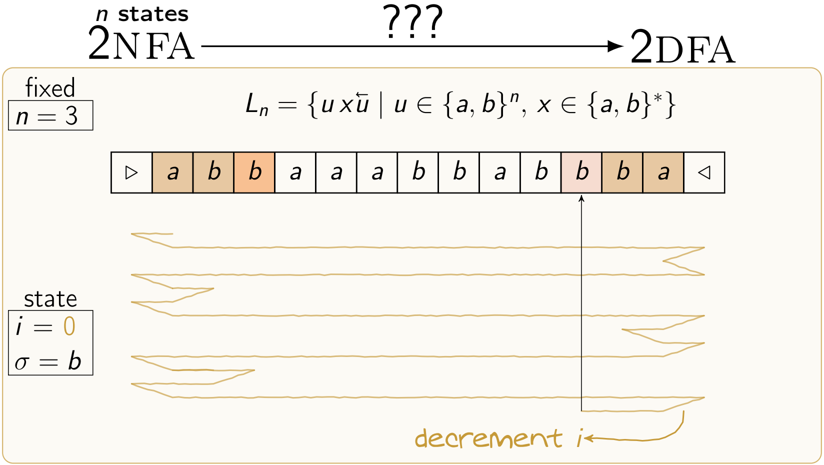
$$L_n = \{u\,x\,\overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$$



state
$i = 3$
$\sigma = b$

The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth



The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

n states

2NFA ——???——> 2DFA

fixed

$n = 3$

$$L_n = \{u\, x\, \overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$$

state

$i = 0$

$\sigma = b$

The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

n states
2NFA ??? → 2DFA

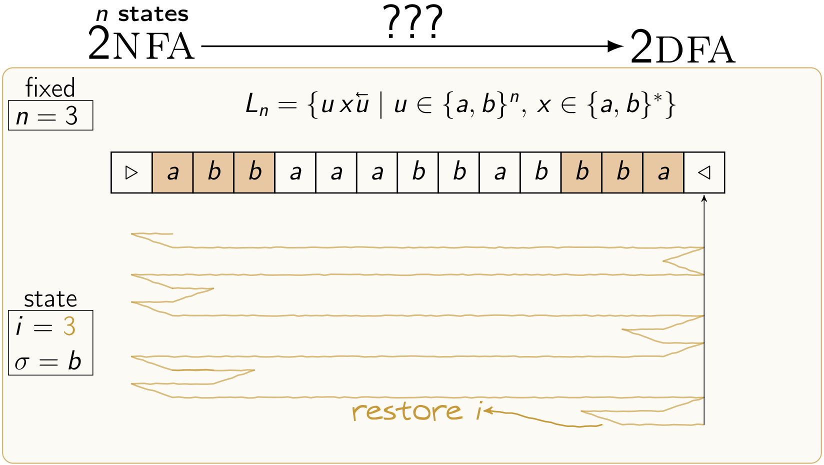
fixed
$n = 3$

$$L_n = \{u\, x\, \overleftarrow{u} \mid u \in \{a,b\}^n,\ x \in \{a,b\}^*\}$$

state
$i = 0$
$\sigma = b$

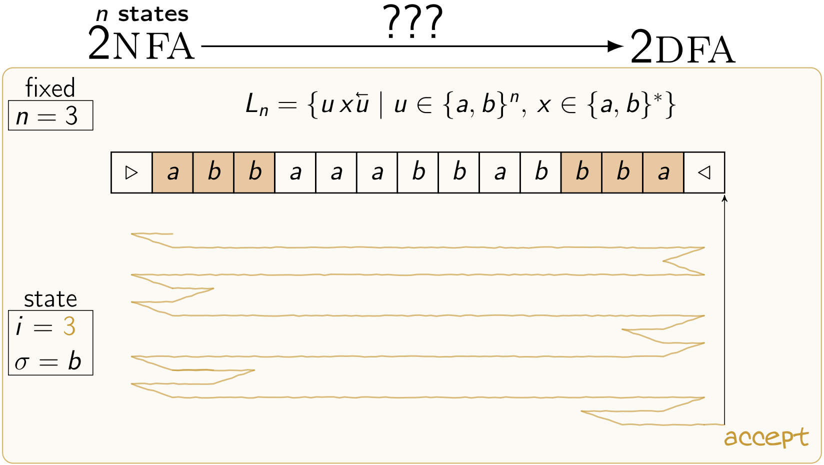
The Sakoda & Sipser conjecture

2FA: an automaton that can move its head back and forth

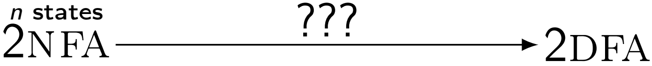


The Sakoda & Sipser conjecture

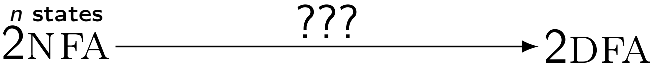
2FA: an automaton that can move its head back and forth



The Sakoda & Sipser conjecture



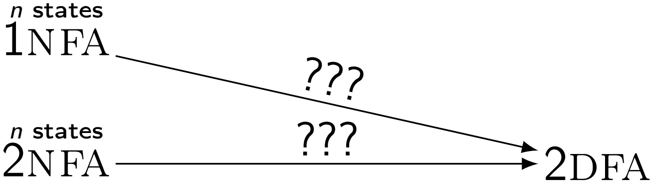
The Sakoda & Sipser conjecture



Conjecture (Sakoda & Sipser 1978)

The conversion of 2NFAs *into* 2DFAs *has super-polynomial cost*

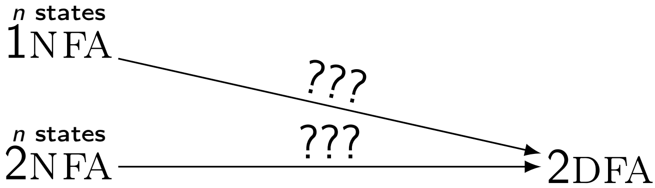
The Sakoda & Sipser conjecture



Conjecture (Sakoda & Sipser 1978)

The conversion of 2NFAs *or* 1NFAs *into* 2DFAs *has super-polynomial cost*

The Sakoda & Sipser conjecture

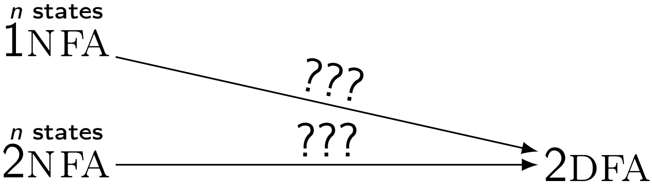


Conjecture (Sakoda & Sipser 1978)

The conversion of 2NFAs *or* 1NFAs *into* 2DFAs *has super-polynomial cost*

- best known upper bound: exp ($\Longleftarrow$ 2NFA $\xrightarrow{\text{exp}}$ 1DFA)

The Sakoda & Sipser conjecture

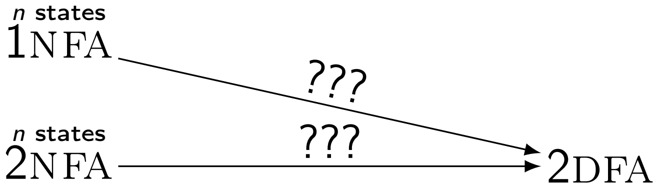


Conjecture (Sakoda & Sipser 1978)

The conversion of 2NFAs *or* 1NFAs *into* 2DFAs *has super-polynomial cost*

- best known upper bound: exp ($\Longleftarrow$ 2NFA $\xrightarrow{\text{exp}}$ 1DFA)
- hardness: related to LogSpace versus NLogSpace question

The Sakoda & Sipser conjecture

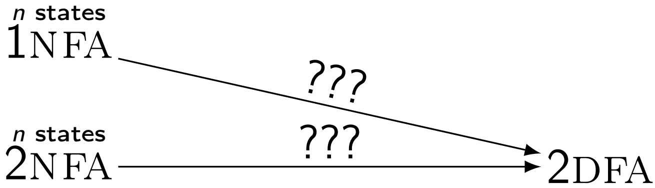


Conjecture (Sakoda & Sipser 1978)

The conversion of 2NFAs *or* 1NFAs *into* 2DFAs *has super-polynomial cost*

- best known upper bound: exp ($\Longleftarrow$ 2NFA $\xrightarrow{\text{exp}}$ 1DFA)
- hardness: related to LogSpace versus NLogSpace question
- some common strategies to tackle the problem:
  - restrict/extend source/target   *e.g.*, 2NFA $\xrightarrow[\text{[Sipser'80]}]{\text{exp}}$ sweeping 2DFA

The Sakoda & Sipser conjecture



Conjecture (Sakoda & Sipser 1978)

The conversion of 2NFAs *or* 1NFAs *into* 2DFAs *has super-polynomial cost*

- best known upper bound: exp ($\Longleftarrow$ 2NFA $\xrightarrow{\text{exp}}$ 1DFA)
- hardness: related to LogSpace versus NLogSpace question
- some common strategies to tackle the problem:
  - restrict/extend source/target *e.g.*, 2NFA $\xrightarrow[\text{[Sipser'80]}]{\text{exp}}$ sweeping 2DFA
  - consider the unary case: 1NFA $\xrightarrow[\text{[Chrobak'86]}]{\text{quadratic}}$ 2DFA $\xleftarrow[\text{[Geffert et al.'03]}]{<\text{subexp}}$ 2NFA

Cost of complementation

$$2\mathrm{NFA} \xrightarrow{\;?\;} \text{co-}2\mathrm{NFA}$$

for $L \subseteq \Sigma^*$ — for $\Sigma^* \setminus L$

Cost of complementation

∃ computation

?

not ∃ computation

2NFA → co-2NFA

for $L \subseteq \Sigma^*$

for $\Sigma^* \setminus L$

Cost of complementation

∃ computation	?	not ∃ computation
2NFA for $L \subseteq \Sigma^*$	$\longrightarrow$	co-2NFA for $\Sigma^* \setminus L$

Usually, complementation is cheap for deterministic devices
and expensive or unknown for nondeterministic ones

1DFA	2DFA	D1-LA	1NFA	2NFA	2NFA+cg $\sim$ 1-LA
trivial	linear	poly	exp	???	exp $\leq$? $\leq$ expexp

Cost of complementation

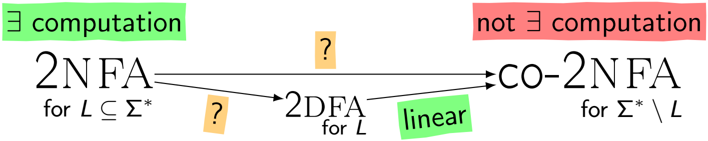
∃ computation — ? → not ∃ computation

2NFA for $L \subseteq \Sigma^*$ ⟶ co-2NFA for $\Sigma^* \setminus L$

Usually, complementation is cheap for deterministic devices
and expensive or unknown for nondeterministic ones

1DFA	2DFA	D1-LA	1NFA	2NFA	2NFA+cg $\sim$ 1-LA
trivial	linear	poly	exp	???	exp $\leq$? $\leq$ expexp

Cost of complementation

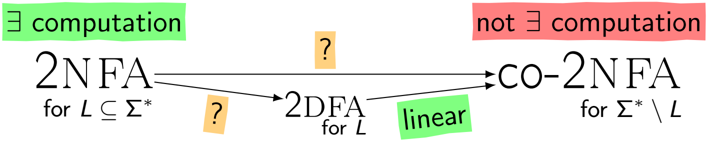


Usually, complementation is cheap for deterministic devices

and expensive or unknown for nondeterministic ones

1DFA	2DFA	D1-LA	1NFA	2NFA	2NFA+cg $\sim$ 1-LA
trivial	linear	poly	exp	???	exp $\leq$? $\leq$ expexp

Cost of complementation



Usually, complementation is cheap for deterministic devices

and expensive or unknown for nondeterministic ones

1DFA	2DFA	D1-LA	1NFA	2NFA	2NFA+cg ∼ 1-LA
trivial	linear	poly	exp	???	exp ≤ ? ≤ expexp

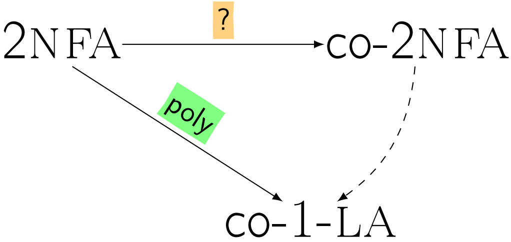
Some known exceptions:

- NLogSpace = co-NLogSpace [Immerman'88, Szelepcsényi'87]
- complementing unary 2NFAs costs polynomial [Geffert *et al.*'07]
- idem for *outer-nondeterminsitic* 2NFAs [Geffert *et al.*'14]

Main result

$$2\mathrm{NFA} \xrightarrow{\ ?\ } \text{co-}2\mathrm{NFA}$$

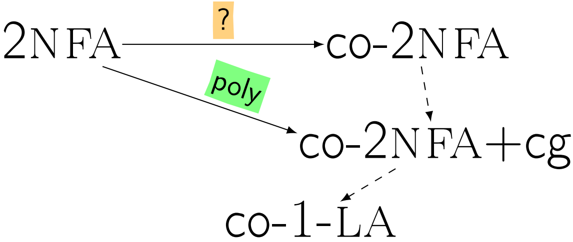
Main result



Theorem

Each 2NFA *can be complemented by a* 1-LA *of polynomial size*

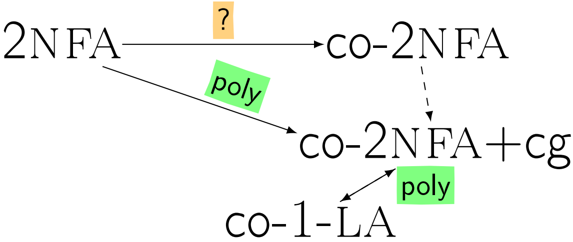
Main result



Theorem

Each 2NFA can be complemented by a ~~1-LA~~ 2NFA+cg of polynomial size

Main result



Theorem

Each 2NFA *can be complemented by a* ~~1-LA~~ 2NFA+cg *of polynomial size*

1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

- 1-LAs: can rewrite each cell when visiting it **for the first time** only

1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

- 1-LAs: can rewrite each cell when visiting it **for the first time** only

Theorem

1-LAs *recognize regular languages* *[Wagner&Wechsung'86]*

1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs

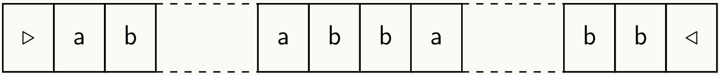
Theorem

1-LAs *recognize regular languages* *[Wagner&Wechsung'86]*

1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

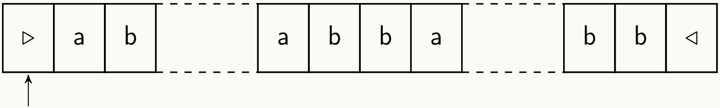
- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs



1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

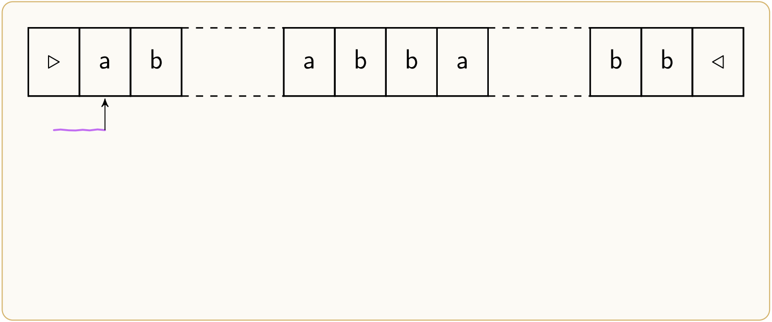
- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs



1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

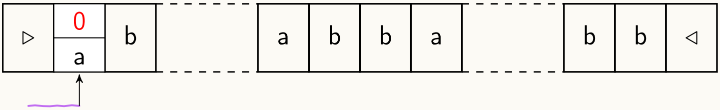
- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs



1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

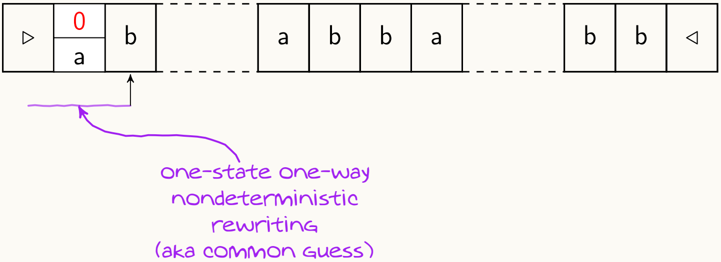
- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs



1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

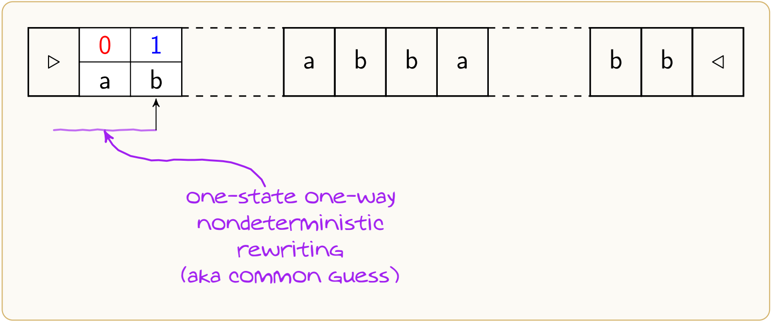
- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs



1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

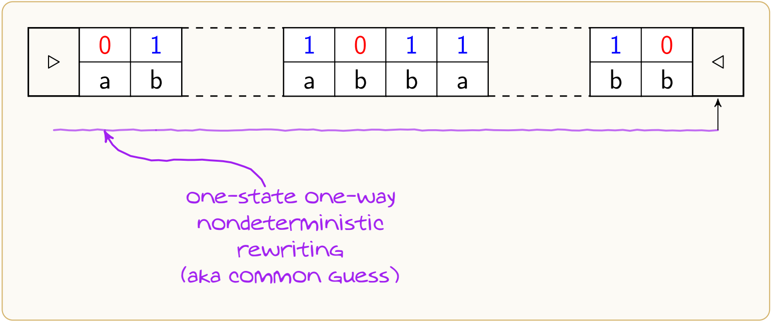
- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs



1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

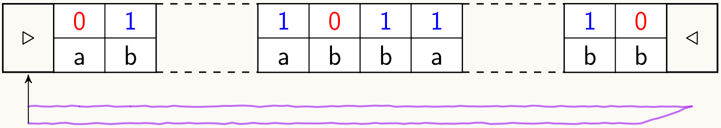
- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs



1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

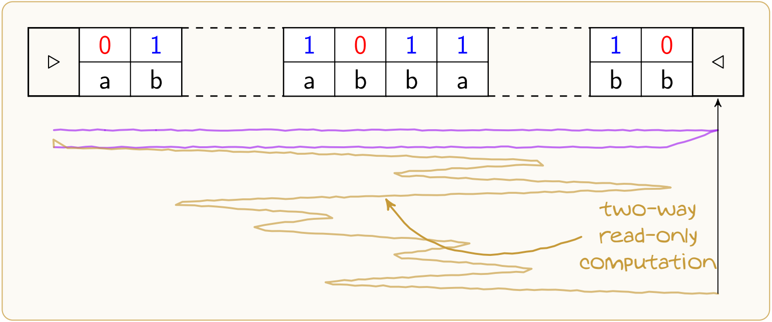
- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs



1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs



1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs : rewrite tape contents **initially** then process it read-only

Theorem

1-LAs *recognize regular languages* *[Wagner&Wechsung'86]*

1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs : rewrite tape contents **initially** then process it read-only
  $\iff$ a 1-LA which annotates memoryless then computes read-only

Theorem

1-LAs *recognize regular languages* *[Wagner&Wechsung'86]*

1-limited automata & automata common-guess

2 extensions of 2NFAs with limited writing capacities

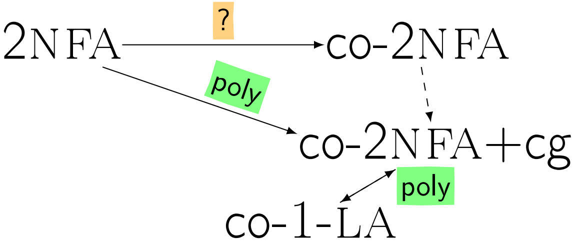
- 1-LAs: can rewrite each cell when visiting it **for the first time** only
- 2NFA+cgs : rewrite tape contents **initially** then process it read-only
  $\iff$ a 1-LA which annotates memoryless then computes read-only

Theorem

1-LAs *recognize regular languages* *[Wagner&Wechsung'86]*

($\implies$ so do 2NFA+cgs*)*

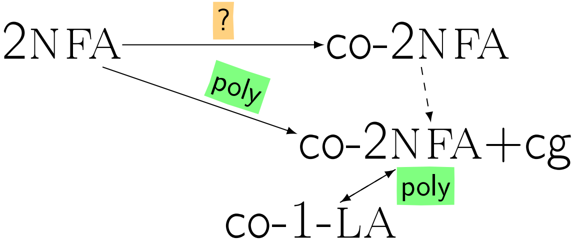
Main result



Theorem

Each 2NFA can be complemented by a ~~1-LA~~ 2NFA+cg of polynomial size

Main result



Theorem

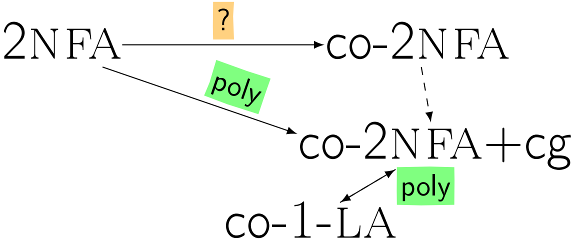
Each 2NFA *can be complemented by a* ~~1-LA~~ 2NFA+cg *of polynomial size*

Proof (sketch), an algorithm

input: an n-state 2NFA recognizing L

output: a poly(n)-state 2NFA+cg with 2 annotation symbols recognizing the complement of L □

Main result



Theorem

Each 1NFA *can be complemented by a* ~~1-LA~~ 2NFA+cg *of polynomial size*

Proof (sketch), an algorithm for this talk

input: an n-state 1NFA recognizing L

output: a poly(n)-state 2NFA+cg with 2 annotation symbols recognizing the complement of L □

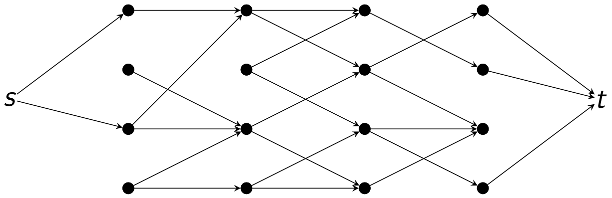
Strategy: inductive counting

STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$



Strategy: inductive counting

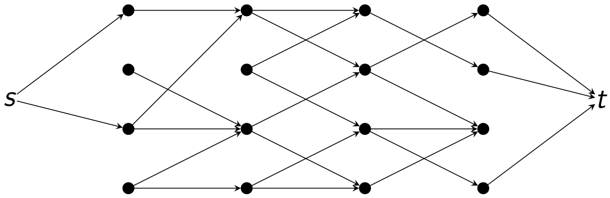
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$

- using n-det and space $\log(n)$, we can find a path from s to t



Strategy: inductive counting

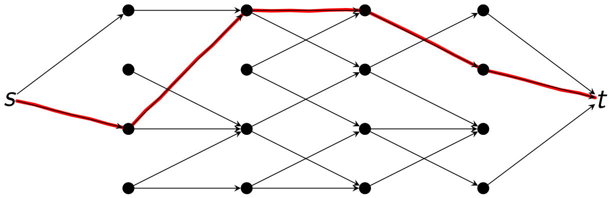
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$

- using n-det and space $\log(n)$, we can find a path from s to t



Strategy: inductive counting

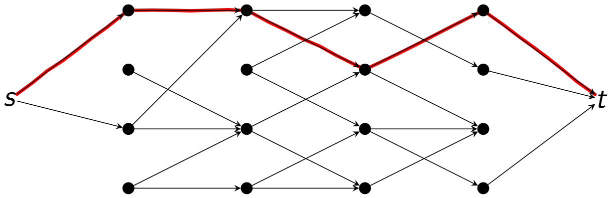
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$

- using n-det and space $\log(n)$, we can find a path from s to t



Strategy: inductive counting

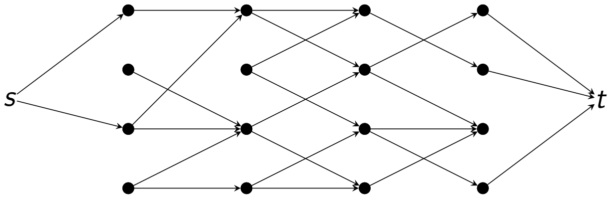
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$

- using n-det and space $\log(n)$, we can find a path from s to t



Strategy: inductive counting

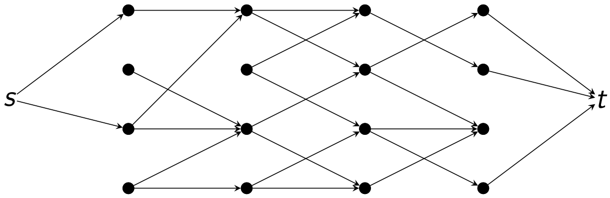
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t



Strategy: inductive counting

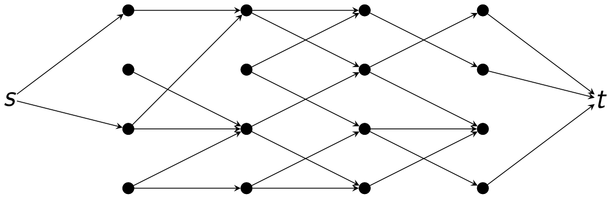
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$



Strategy: inductive counting

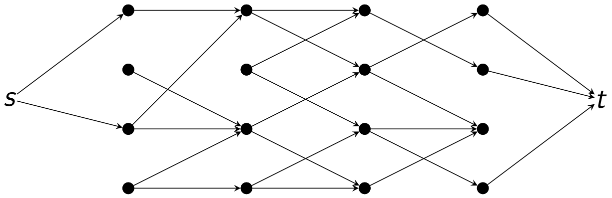
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$k = 3$

Strategy: inductive counting

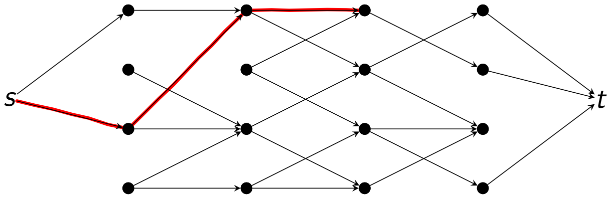
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$k = 3$

Strategy: inductive counting

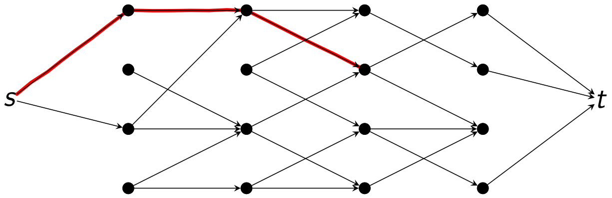
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$k = 3$

Strategy: inductive counting

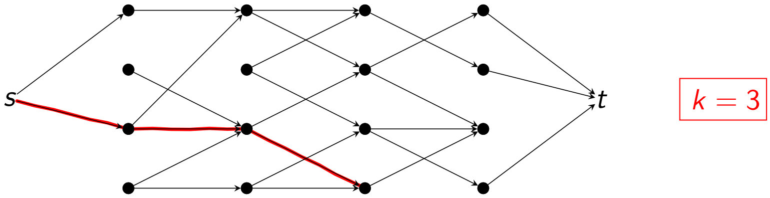
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



Strategy: inductive counting

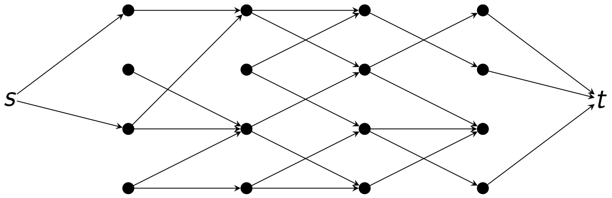
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$k = 3$

Strategy: inductive counting

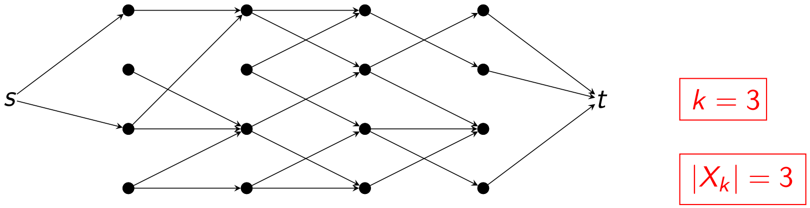
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



knowing $|R_k|$ we can enumerate R_k (assuming order)

Strategy: inductive counting

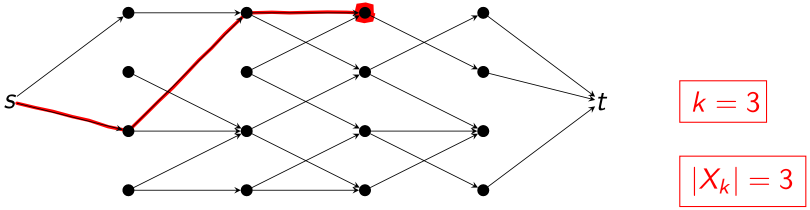
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



knowing $|R_k|$ we can enumerate R_k (assuming order)

Strategy: inductive counting

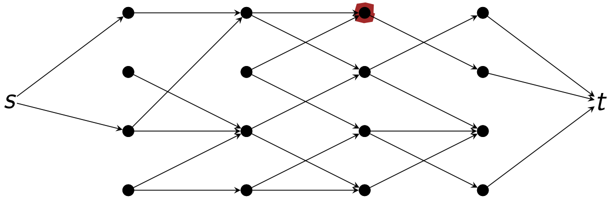
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can enumerate R_k (assuming order)

Strategy: inductive counting

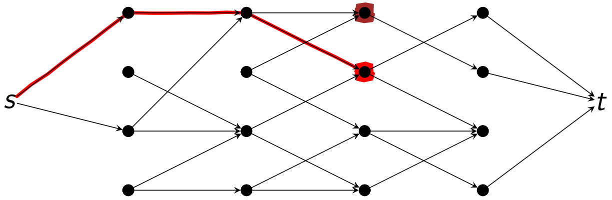
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can enumerate R_k (assuming order)

Strategy: inductive counting

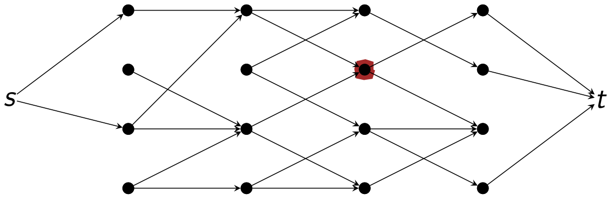
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can enumerate R_k (assuming order)

Strategy: inductive counting

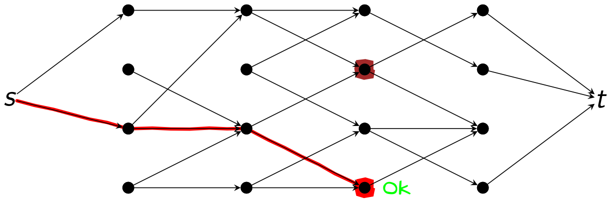
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can enumerate R_k (assuming order)

Strategy: inductive counting

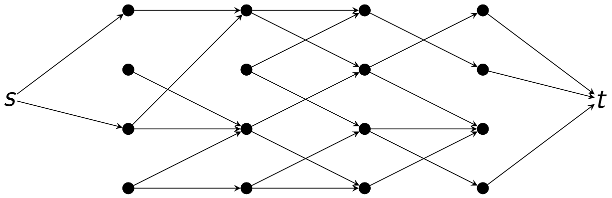
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can enumerate R_k (assuming order)

Strategy: inductive counting

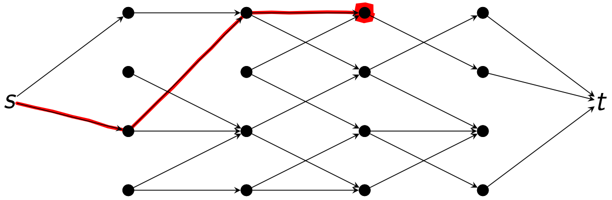
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can enumerate R_k (assuming order)

Strategy: inductive counting

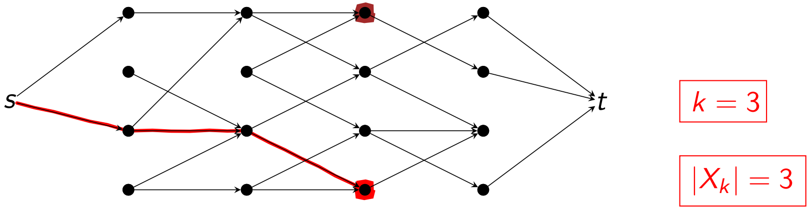
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



knowing $|R_k|$ we can enumerate R_k (assuming order)

Strategy: inductive counting

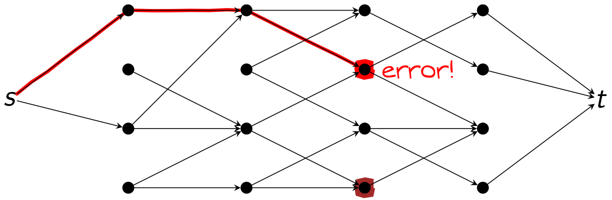
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can enumerate R_k (assuming order)

Strategy: inductive counting

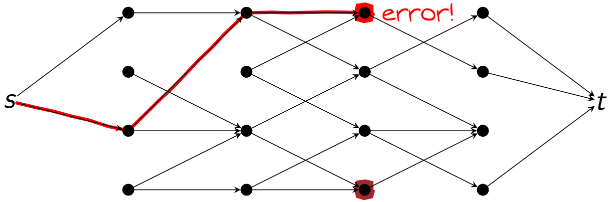
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can enumerate R_k (assuming order)

Strategy: inductive counting

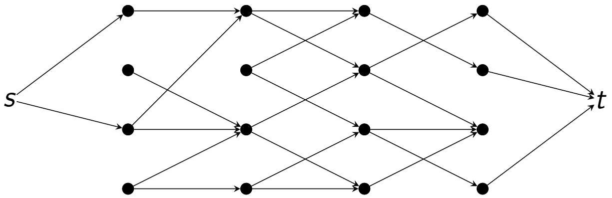
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 0$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

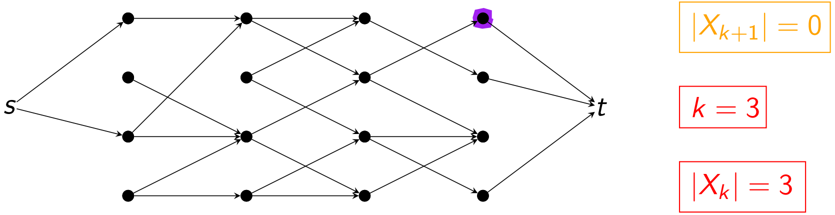
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

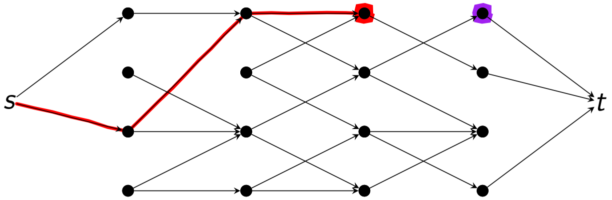
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 0$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

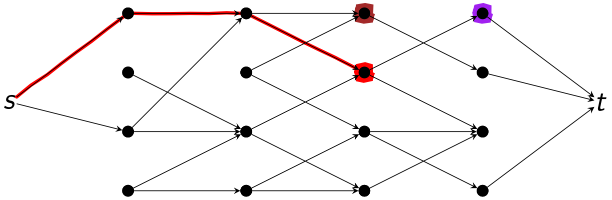
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 0$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

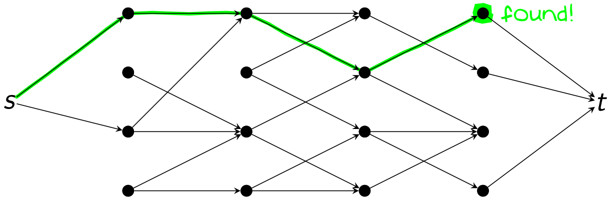
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 1$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

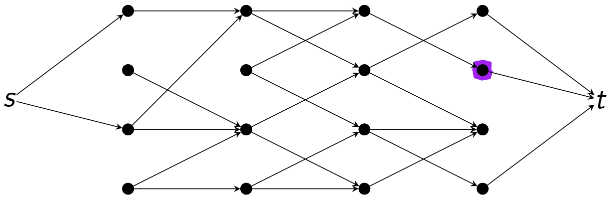
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 1$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

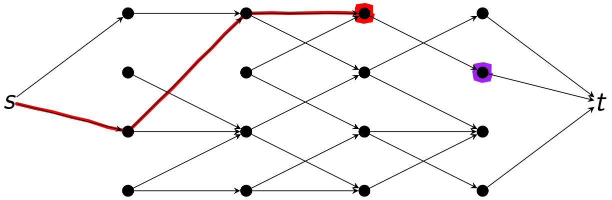
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 1$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

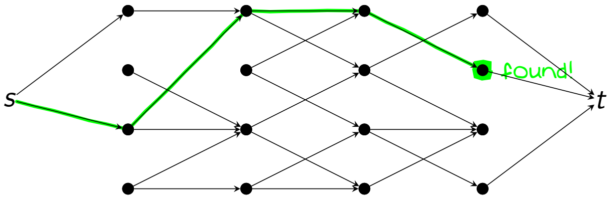
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 2$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

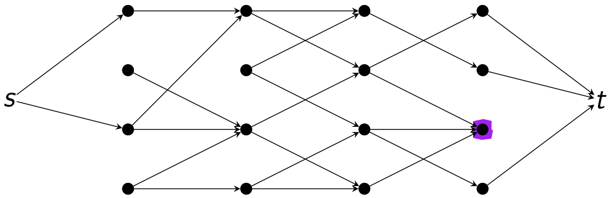
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 2$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

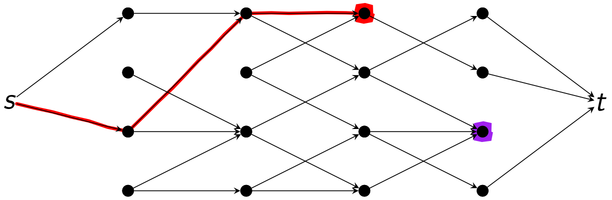
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 2$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

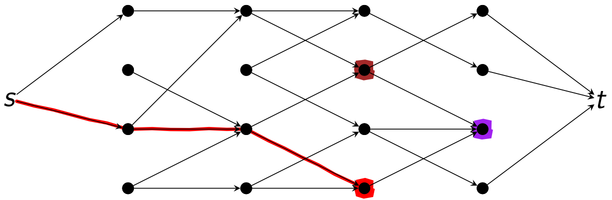
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 2$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

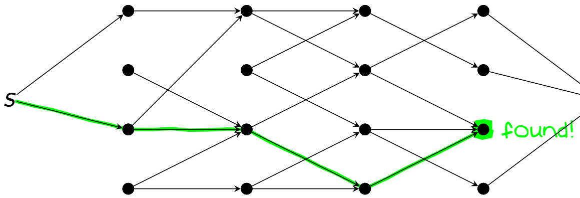
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 3$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

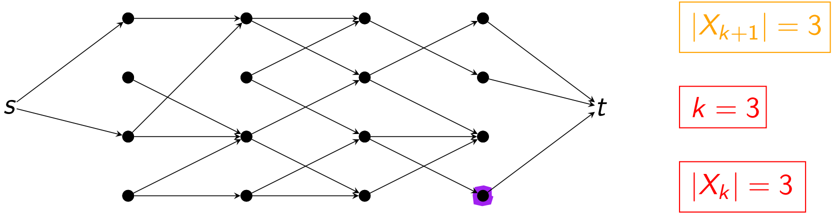
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

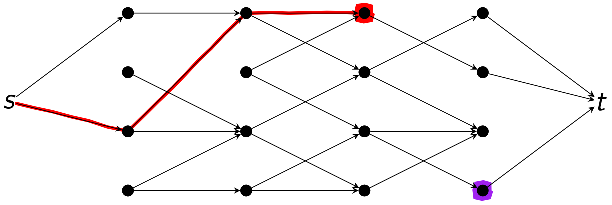
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 3$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

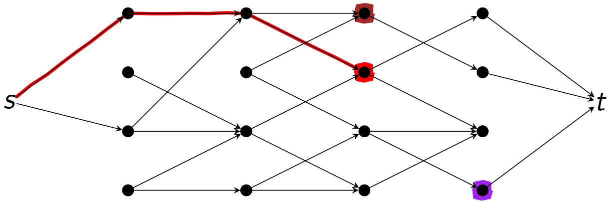
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 3$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

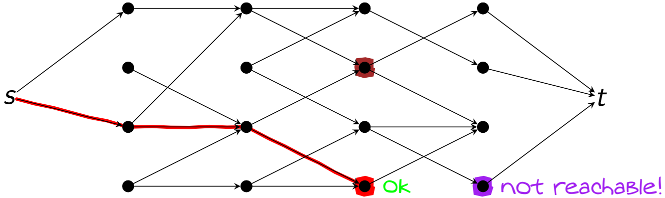
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 3$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

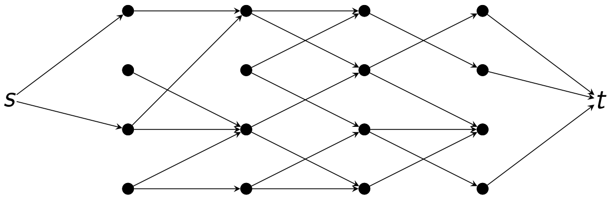
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 3$

$k = 3$

$|X_k| = 3$

knowing $|R_k|$ we can compute $|R_{k+1}|$

Strategy: inductive counting

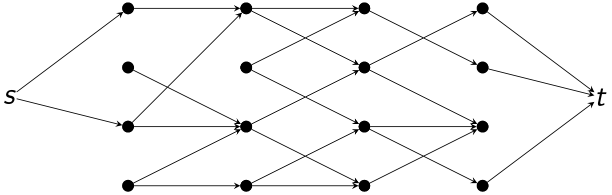
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| = 0$

$k = 4$

$|X_k| = 3$

inductively compute $|R_k|$ for $k < n$;

Strategy: inductive counting

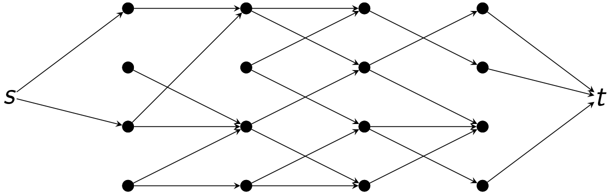
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| =$

$k =$

$|X_k| =$

inductively compute $|R_k|$ for $k < n$; accept if t is reached, reject otherwise

Strategy: inductive counting

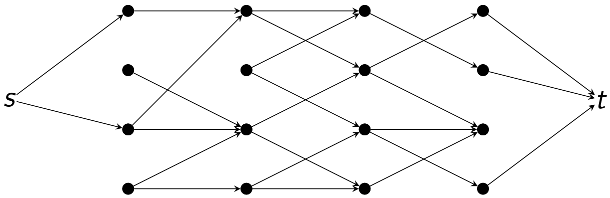
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| =$

$k =$

$|X_k| =$

inductively compute $|R_k|$ for $k < n$; accept if t is reached, reject otherwise

space: store k, $|X_k|$, $|X_{k+1}|$ and 3 vertices

Strategy: inductive counting

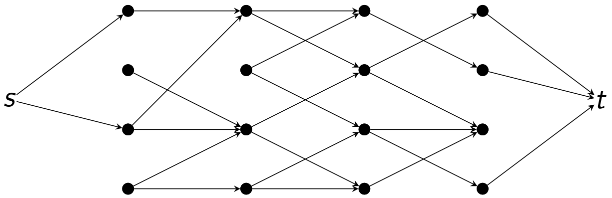
STCON problem

input: a directed graph G, and $s, t \in V(G)$

output: does there exist a path from s to t?

let $n = |V(G)|$ and $R_k = \{v \in V(G) \mid v \text{ reachable from } s \text{ in } k \text{ steps}\}$

- using n-det and space $\log(n)$, we can find a path from s to t
- $\exists$ path from s to t iff $t \in R_k$ for some $k < n$
- given k, using n-det and space $\log(n)$, we can find any $v \in R_k$



$|X_{k+1}| =$

$k =$

$|X_k| =$

inductively compute $|R_k|$ for $k < n$; accept if t is reached, reject otherwise

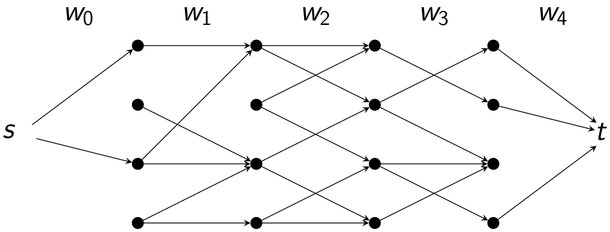
space: store k, $|X_k|$, $|X_{k+1}|$ and 3 vertices $\implies \mathcal{O}(\log(n))$

Complementing 1NFAs

Fix a 1NFA A with n states, and an input word w:

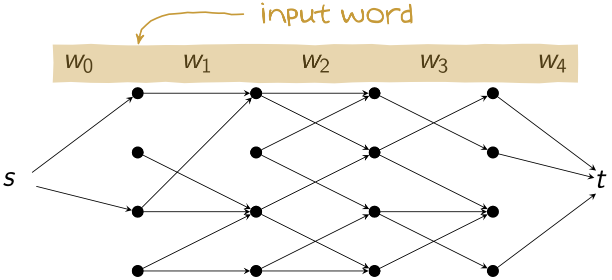
Complementing 1NFAs

Fix a 1NFA A with n states, and an input word w:



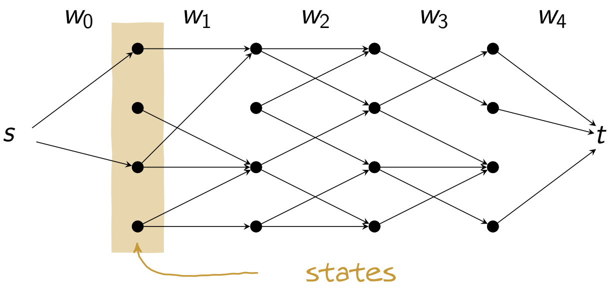
Complementing 1NFAs

Fix a 1NFA A with n states, and an input word w:



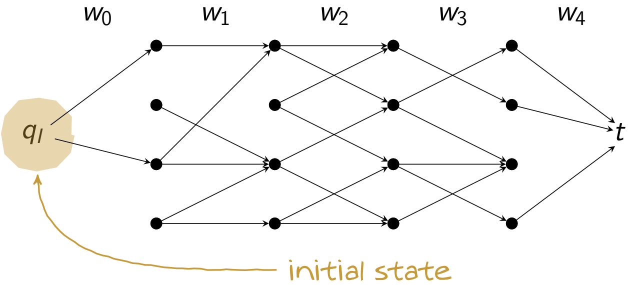
Complementing 1NFAs

Fix a 1NFA A with n states, and an input word w:



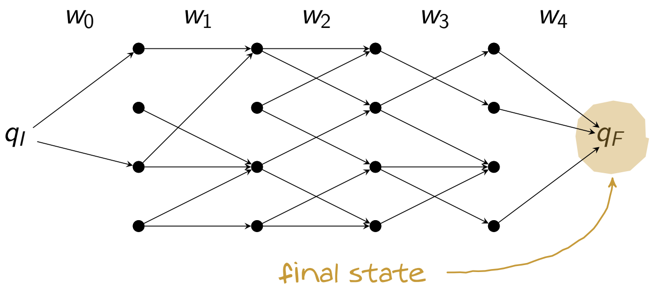
Complementing 1NFAs

Fix a 1NFA A with n states, and an input word w:



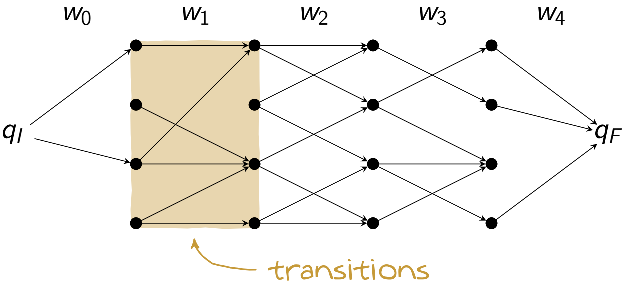
Complementing 1NFAs

Fix a 1NFA A with n states, and an input word w:



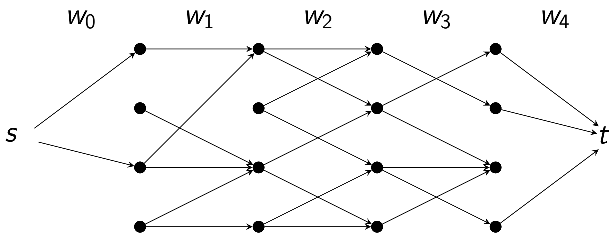
Complementing 1NFAs

Fix a 1NFA A with n states, and an input word w:



Complementing 1NFAs

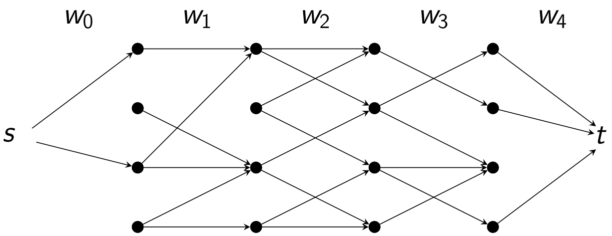
Fix a 1NFA A with n states, and an input word w:



- find a path of length k from q_I:
  rerun k steps of A from initial config

Complementing 1NFAs

Fix a 1NFA A with n states, and an input word w:

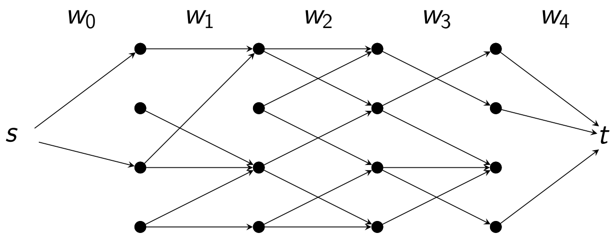


- find a path of length k from q_I:
  rerun k steps of A from initial config

main issue: k ranges over $|w|+1$ (not bounded!)

Complementing 1NFAs

Fix a 1NFA A with n states, and an input word w:

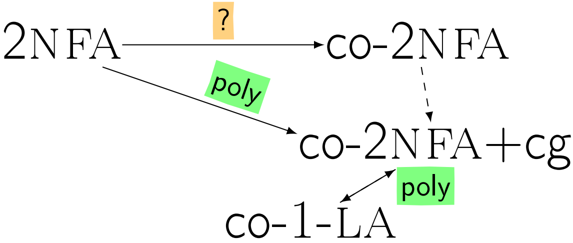


- find a path of length k from q_I:
  rerun k steps of A from initial config

main issue: k ranges over $|w| + 1$ (not bounded!)

solution: annotate the tape so that it is sufficient
to rerun $\leq k$ steps of A from a config that is not far

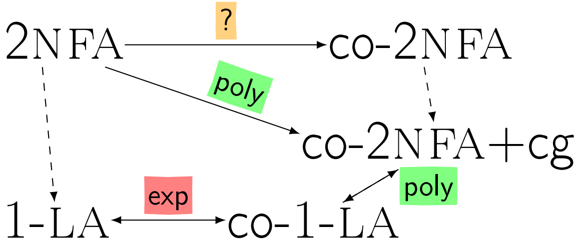
Main result



Theorem

Each 2NFA *can be complemented by a* 1-LA *of polynomial size*

Main result



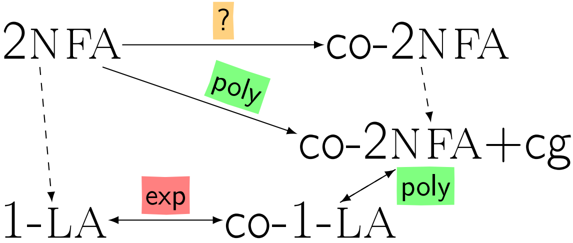
Theorem

*Each 2*NFA *can be complemented by a 1-*LA *of polynomial size*

Corollary

*Each 1-*LA *can be complemented by a 1-*LA *of exponential size (tight)*

Main result



Theorem

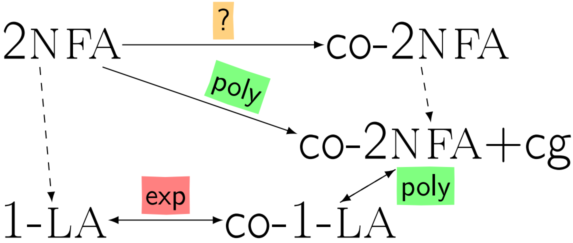
Each 2NFA *can be complemented by a* 1-LA *of polynomial size*

Corollary

Each 1-LA *can be complemented by a* 1-LA *of exponential size (tight)*

Future: can we convert/complement 2NFA into 2DFA+cg of poly size?

Main result



Theorem

Each 2NFA *can be complemented by a* 1-LA *of polynomial size*

Corollary

Each 1-LA *can be complemented by a* 1-LA *of exponential size (tight)*

Future: can we convert/complement 2NFA into 2DFA+cg of poly size?

thank you!