On Complementation of Nondeterministic Finite Automata without Full Determinization

Lukáš Holík, Ondřej Lengál

Brno University of Technology, Czech Republic

Juraj Major, **Adéla Štěpková**, Jan Strejček

Masaryk University, Brno, Czech Republic

Research goal

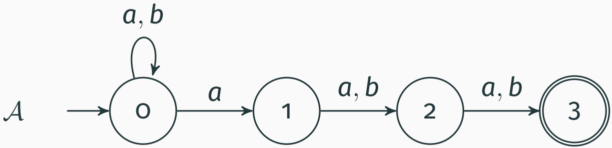
Construct nondeterministic complements of NFAs smaller than standard deterministic complements.

Terminology

- nondeterministic finite automaton (NFA) $\mathcal{A} = (Q, \Sigma, \delta, I, F)$
- **size** of $\mathcal{A}$ is the number of states: $|\mathcal{A}| = |Q|$
- for a language L over the alphabet Σ:
  the **complement** of L is $co(L) = \Sigma^* \setminus L$

Complementation of finite automata

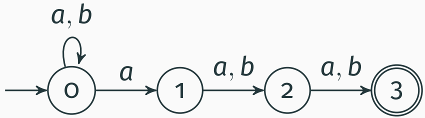
Task: for an NFA $\mathcal{A}$ accepting a language L over the alphabet Σ,
find an NFA $\mathcal{C}$ accepting the language $co(L) = \Sigma^* \setminus L$



accepts $L = \{a, b\}^*.\{a\}.\{a, b\}^2$

Standard complementation approach

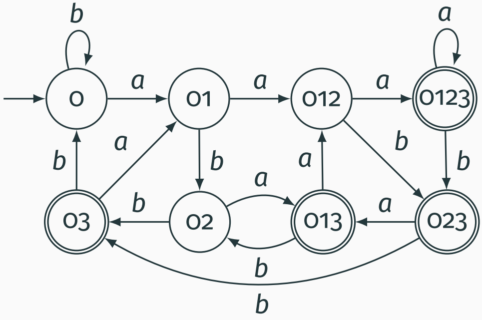
$\mathcal{A}$



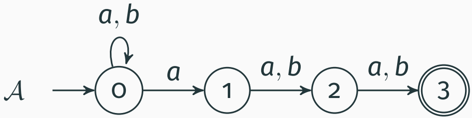
$\rightarrow$ determinize

$\rightarrow$ switch accepting and nonaccepting states

$det(\mathcal{A})$

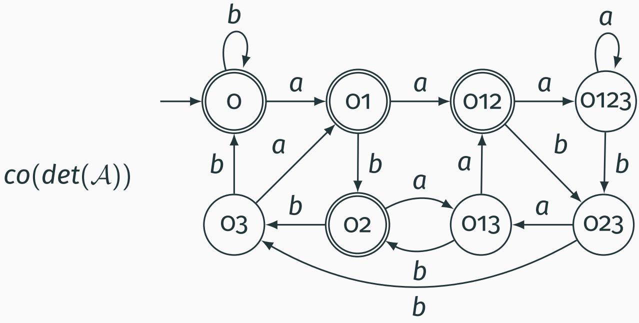


Standard complementation approach



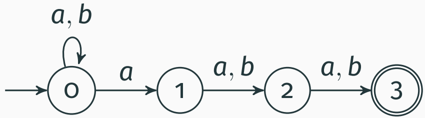
$\rightarrow$ determinize

$\rightarrow$ switch accepting and nonaccepting states



Standard complementation approach

$\mathcal{A}$

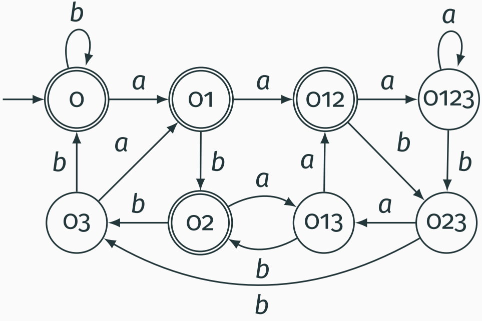


$\rightarrow$ determinize

$\rightarrow$ switch accepting and nonaccepting states

= **forward powerset complementation**

$co(det(\mathcal{A}))$



Problem of the standard approach

exponential upper bound:

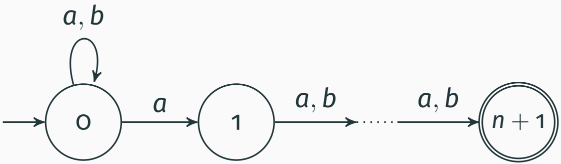
- deterministic complement can have up to $2^{|Q|}$ states
- the bound is optimal

Problem of the standard approach

exponential upper bound:

- deterministic complement can have up to $2^{|Q|}$ states
- the bound is optimal

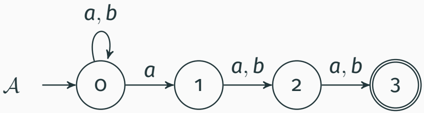
generalization of our example: $\mathcal{A}_n$ accepting $L_n = \{a, b\}^*.\{a\}.\{a, b\}^n$

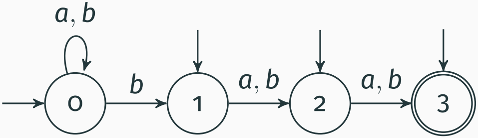


$\mathcal{A}_n$: $n + 2$ states

$det(\mathcal{A}_n)$: 2^{n+1} states
(also after minimization)

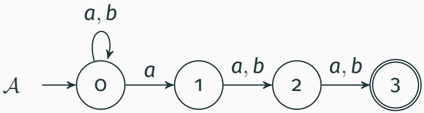
We can do better

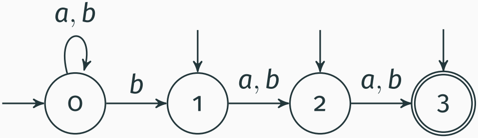




…this is also a complement of $\mathcal{A}$

We can do better



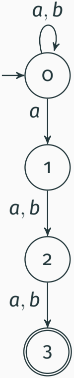


…this is also a complement of $\mathcal{A}$

How to obtain it algorithmically?

Employing the reverse

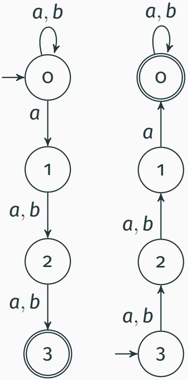
reverse $\to$ determinize $\to$ complement $\to$ reverse



$\mathcal{A}$

Employing the reverse

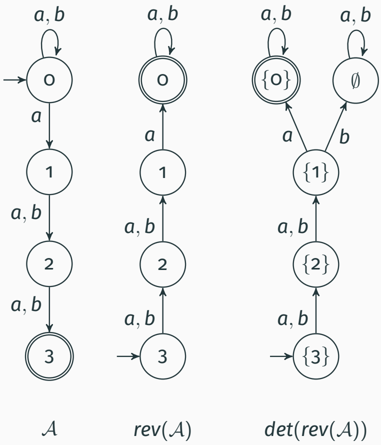
reverse → determinize → complement → reverse



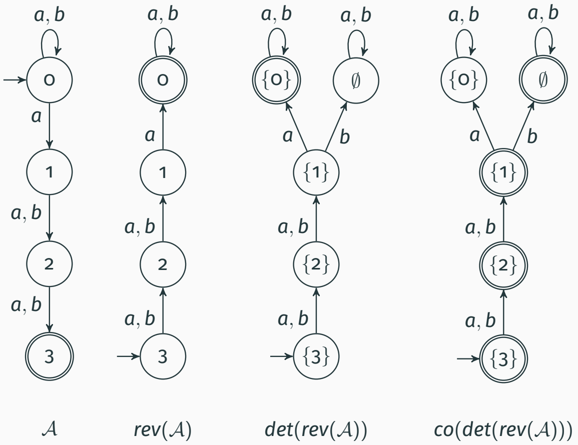
$\mathcal{A}$ $\quad$ $rev(\mathcal{A})$

Employing the reverse

reverse → determinize → complement → reverse

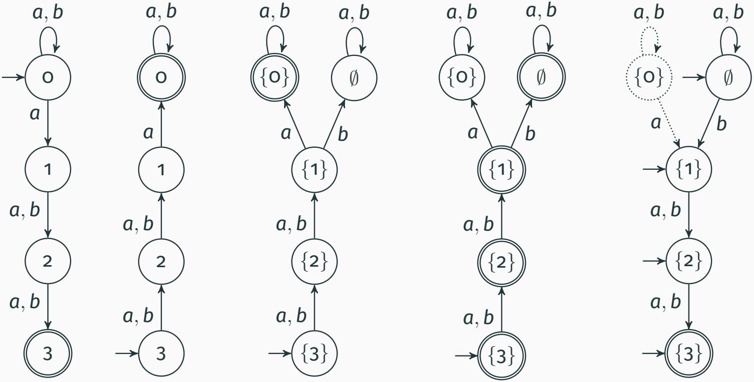


reverse → determinize → complement → reverse



Employing the reverse

reverse → determinize → complement → reverse



$\mathcal{A}$ $rev(\mathcal{A})$ $det(rev(\mathcal{A}))$ $co(det(rev(\mathcal{A})))$ $rev(co(det(rev(\mathcal{A}))))$

Reverse powerset complementation

- can produce nondeterministic complements
  $\rightarrow$ smaller complements for some automata
- for some automata, forward powerset is better

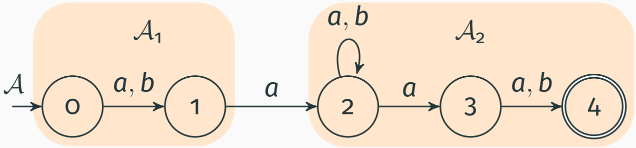
Complementing automata with specific structure

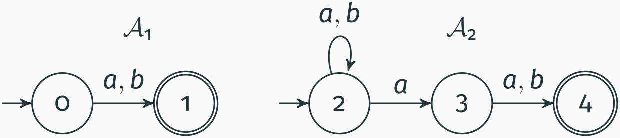
two methods: **sequential** and **gate complementation**

- exploit the specific structure of automata to build smaller complements
- component-based: use complements of parts of an NFA to compose a complement of the whole NFA
- core ideas shown in a simple setting

Setting

NFA $\mathcal{A}$ composed of two disjoint NFAs $\mathcal{A}_1$, $\mathcal{A}_2$ (components) connected with a single *transfer* transition





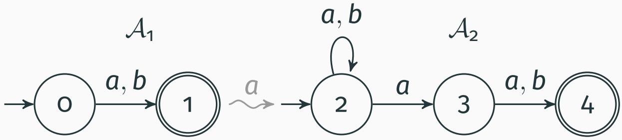
$\mathcal{A}_1$ accepts L_1

$\mathcal{A}_2$ accepts L_2

$\mathcal{A}$ accepts $L_1.\{a\}.L_2$

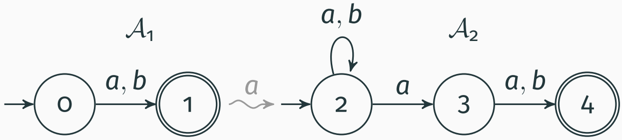
Sequential complementation: idea

Observation: $co(L_1.\{a\}.L_2)$ consists of all words w, such that: for all u, v satisfying $uav = w$, if $u \in L_1$ then $v \in co(L_2)$

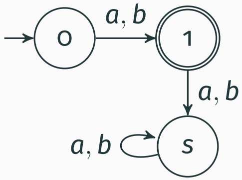


Sequential complementation: idea

Observation: $co(L_1.\{a\}.L_2)$ consists of all words w, such that:
for all u, v satisfying $uav = w$, if $u \in L_1$ then $v \in co(L_2)$

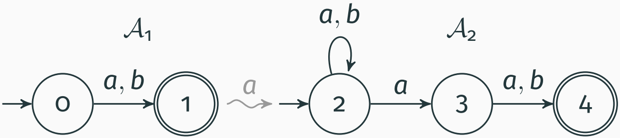


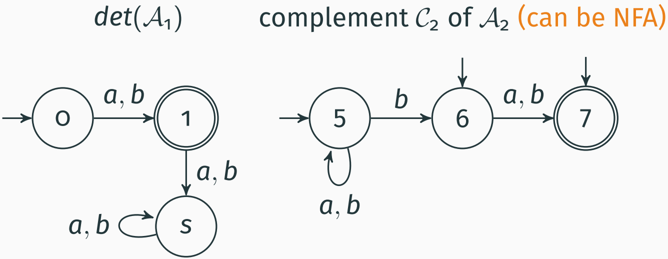
$det(\mathcal{A}_1)$



Sequential complementation: idea

Observation: $co(L_1.\{a\}.L_2)$ consists of all words w, such that: for all u, v satisfying $uav = w$, if $u \in L_1$ then $v \in co(L_2)$

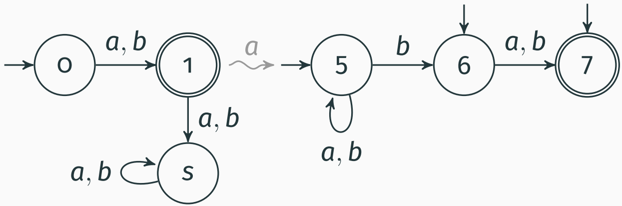




Sequential complementation: idea

Observation: $co(L_1.\{a\}.L_2)$ consists of all words w, such that:
for all u, v satisfying $uav = w$, if $u \in L_1$ then $v \in co(L_2)$

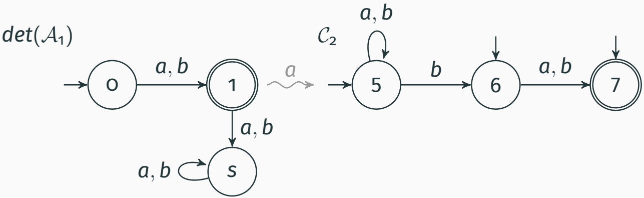
$det(\mathcal{A}_1)$ complement $\mathcal{C}_2$ of $\mathcal{A}_2$ (can be NFA)



→ run $det(\mathcal{A}_1)$ on w

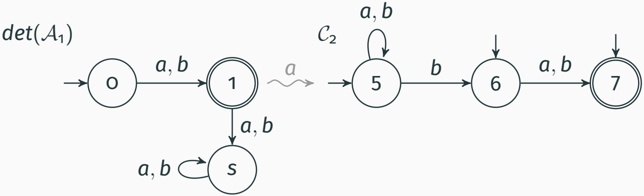
→ when a prefix $u \in L_1$ is read, start an instance of $\mathcal{C}_2$ under a checking $v \in co(L_2)$

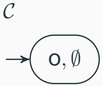
Sequential complementation: construction



state of $\mathcal{C}$**:** state of $det(\mathcal{A}_1)$ + states of current instances of $\mathcal{C}_2$

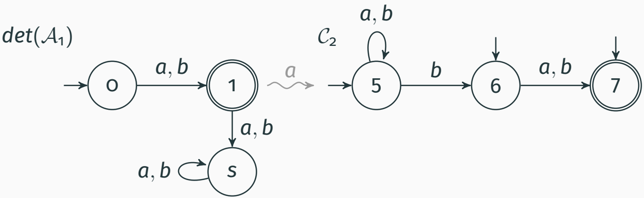
Sequential complementation: construction

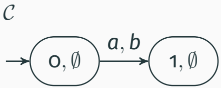




state of $\mathcal{C}$**:** state of $det(\mathcal{A}_1)$ + states of current instances of $\mathcal{C}_2$
initial state of $\mathcal{C}$**:** initial state of $det(\mathcal{A}_1)$ + no running instance of $\mathcal{C}_2$

Sequential complementation: construction





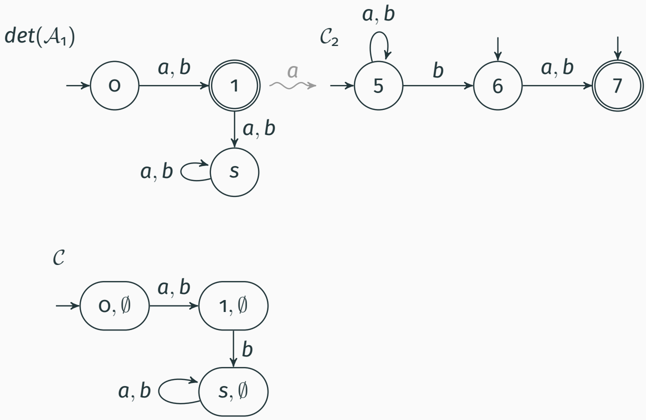
state of $\mathcal{C}$: state of $det(\mathcal{A}_1)$ + states of current instances of $\mathcal{C}_2$

transitions: transition in $det(\mathcal{A})$

+ transitions of running instances of $\mathcal{C}_2$

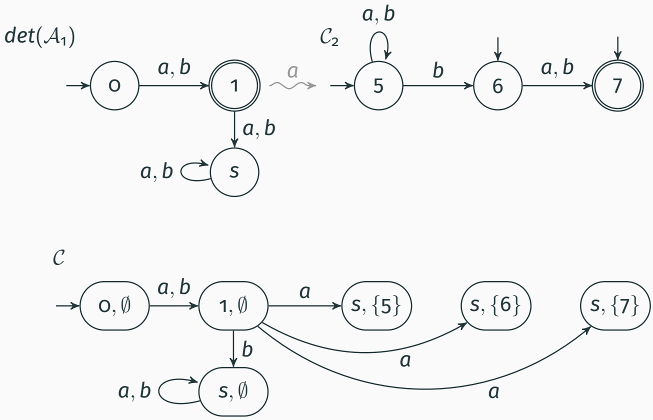
+ possible new instances of $\mathcal{C}_2$

Sequential complementation: construction



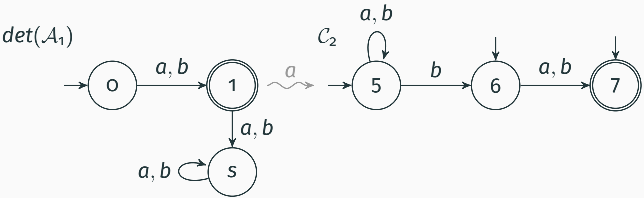
state of $\mathcal{C}$**:** state of $det(\mathcal{A}_1)$ + states of current instances of $\mathcal{C}_2$
transitions: transition in $det(\mathcal{A})$
+ transitions of running instances of $\mathcal{C}_2$
+ possible new instances of $\mathcal{C}_2$

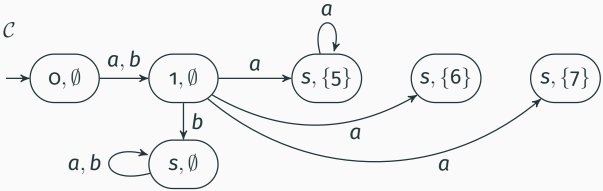
Sequential complementation: construction



state of $\mathcal{C}$**:** state of $det(\mathcal{A}_1)$ + states of current instances of $\mathcal{C}_2$
transitions: transition in $det(\mathcal{A})$
+ transitions of running instances of $\mathcal{C}_2$
+ possible new instances of $\mathcal{C}_2$

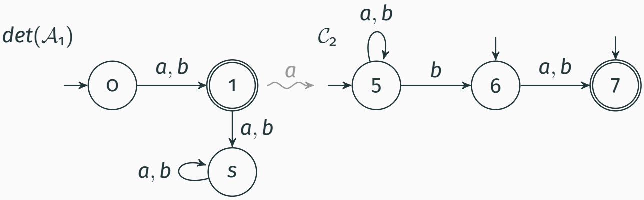
Sequential complementation: construction

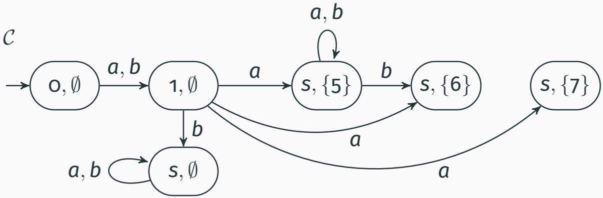




state of $\mathcal{C}$**:** state of $det(\mathcal{A}_1)$ + states of current instances of $\mathcal{C}_2$
transitions: transition in $det(\mathcal{A})$
+ transitions of running instances of $\mathcal{C}_2$
+ possible new instances of $\mathcal{C}_2$

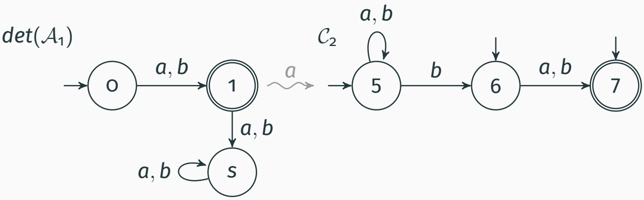
Sequential complementation: construction

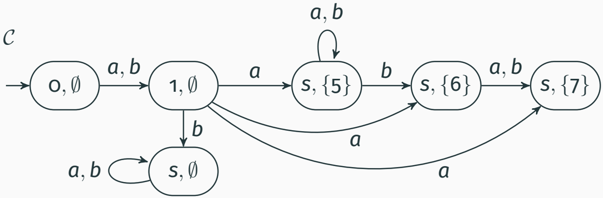




state of $\mathcal{C}$: state of $det(\mathcal{A}_1)$ + states of current instances of $\mathcal{C}_2$
transitions: transition in $det(\mathcal{A})$
+ transitions of running instances of $\mathcal{C}_2$
+ possible new instances of $\mathcal{C}_2$

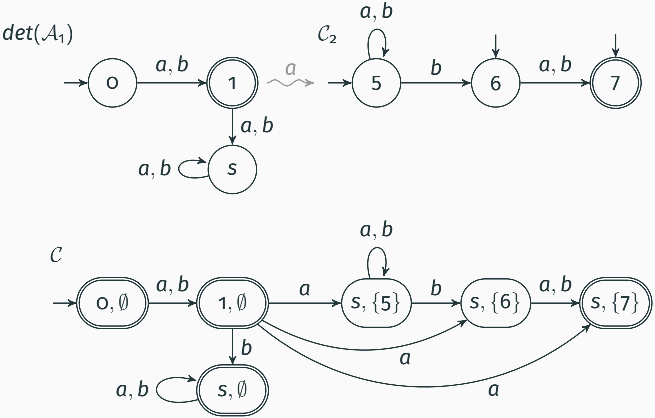
Sequential complementation: construction





state of $\mathcal{C}$**:** state of $det(\mathcal{A}_1)$ + states of current instances of $\mathcal{C}_2$
transitions: transition in $det(\mathcal{A})$
+ transitions of running instances of $\mathcal{C}_2$
+ possible new instances of $\mathcal{C}_2$

Sequential complementation: construction



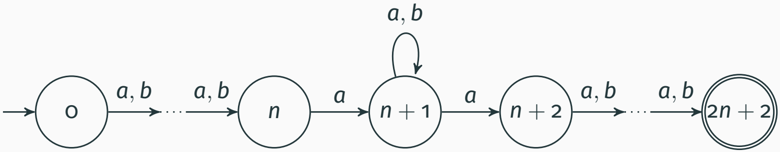
state of $\mathcal{C}$: state of $det(\mathcal{A}_1)$ + states of current instances of $\mathcal{C}_2$
final states: $\mathcal{C}$ accepts when all current instances of $\mathcal{C}_2$ accept

Sequential complementation: properties

- preserves nondeterminism
- size of $\mathcal{C}$ depends on the size of $det(\mathcal{A}_1)$ and $\mathcal{C}_2$

Sequential complementation: properties

- preserves nondeterminism
- size of $\mathcal{C}$ depends on the size of $det(\mathcal{A}_1)$ and $\mathcal{C}_2$

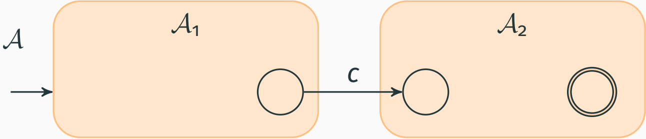


NFA accepting $L_n = \{a, b\}^n.\{a\}.\{a, b\}^*.\{a\}.\{a, b\}^n$

- sequential complement: $2n + 3$ states
- forward and reverse powerset complements: $2^{n+1} + n + 1$ states

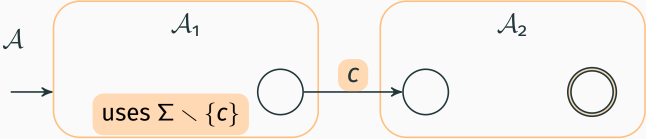
Gate complementation: setting

NFA $\mathcal{A}$ composed of two disjoint NFAs $\mathcal{A}_1$, $\mathcal{A}_2$
connected with a single transition



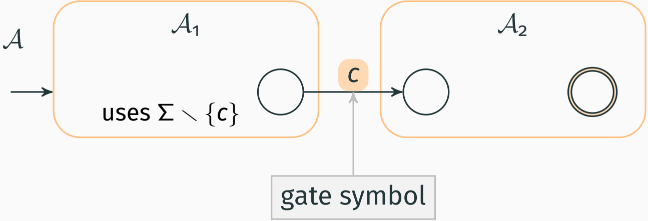
Gate complementation: setting

NFA $\mathcal{A}$ composed of two disjoint NFAs $\mathcal{A}_1$, $\mathcal{A}_2$
connected with a single transition under a symbol not in $\mathcal{A}_1$



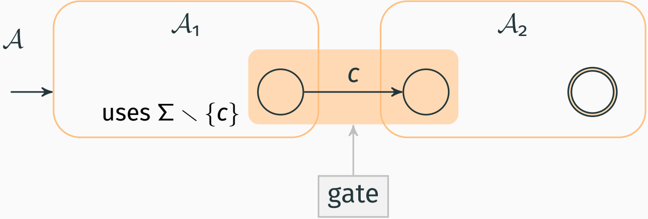
Gate complementation: setting

NFA $\mathcal{A}$ composed of two disjoint NFAs $\mathcal{A}_1$, $\mathcal{A}_2$
connected with a single transition under a symbol not in $\mathcal{A}_1$



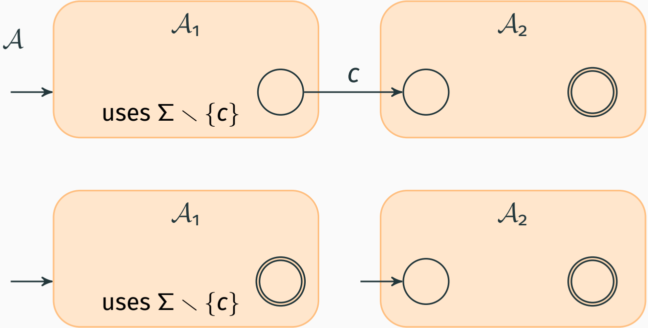
Gate complementation: setting

NFA $\mathcal{A}$ composed of two disjoint NFAs $\mathcal{A}_1$, $\mathcal{A}_2$
connected with a single transition under a symbol not in $\mathcal{A}_1$



Gate complementation: setting

NFA $\mathcal{A}$ composed of two disjoint NFAs $\mathcal{A}_1$, $\mathcal{A}_2$
connected with a single transition under a symbol not in $\mathcal{A}_1$



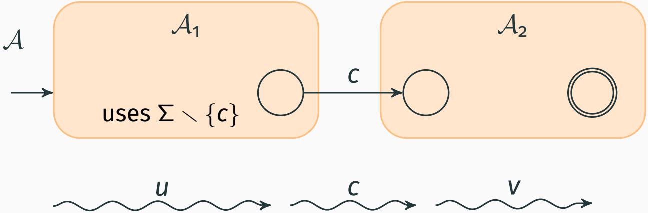
$\mathcal{A}_1$ accepts L_1 $\mathcal{A}_2$ accepts L_2

$\mathcal{A}$ accepts $L_1.\{c\}.L_2$

Gate complementation: idea

Observation: the first c in a word w must be read on the gate

w belongs to $co(L_1.\{c\}.L_2)$ if:

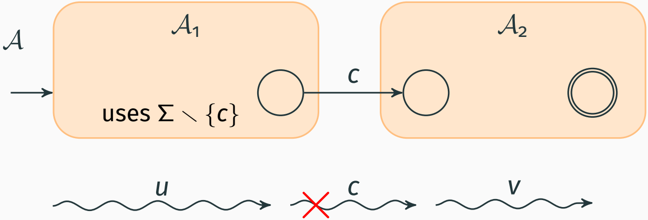


Gate complementation: idea

Observation: the first c in a word w must be read on the gate

w belongs to $co(L_1.\{c\}.L_2)$ if:

1. w does not contain any c, or

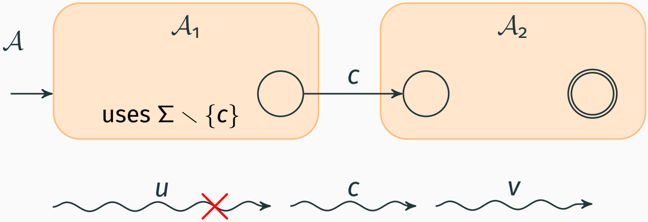


Gate complementation: idea

Observation: the first c in a word w must be read on the gate

w belongs to $co(L_1.\{c\}.L_2)$ if:

1. w does not contain any c, or
2. $w = ucv$ where $u \in (\Sigma \setminus \{c\})^*$ and $u \notin L_1$, or

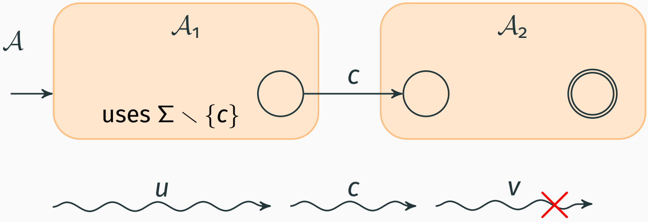


Gate complementation: idea

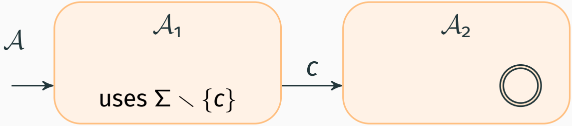
Observation: the first c in a word w must be read on the gate

w belongs to $co(L_1.\{c\}.L_2)$ if:

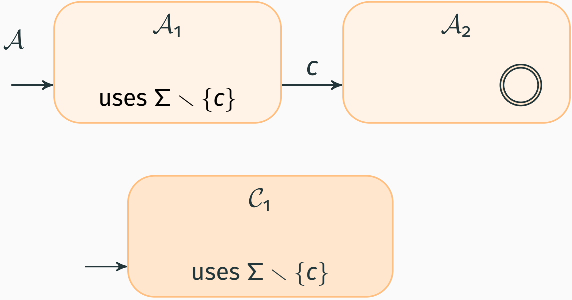
1. w does not contain any c, or
2. $w = ucv$ where $u \in (\Sigma \setminus \{c\})^*$ and $u \notin L_1$, or
3. $w = ucv$ where $u \in (\Sigma \setminus \{c\})^*$ and $v \notin L_2$



Gate complementation: construction

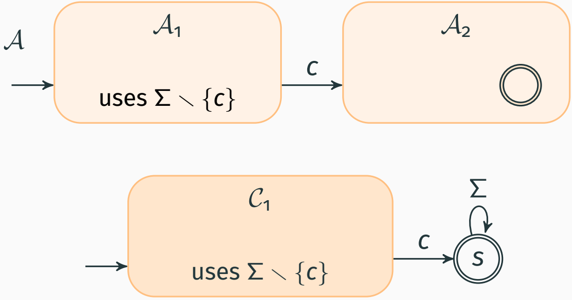


Gate complementation: construction



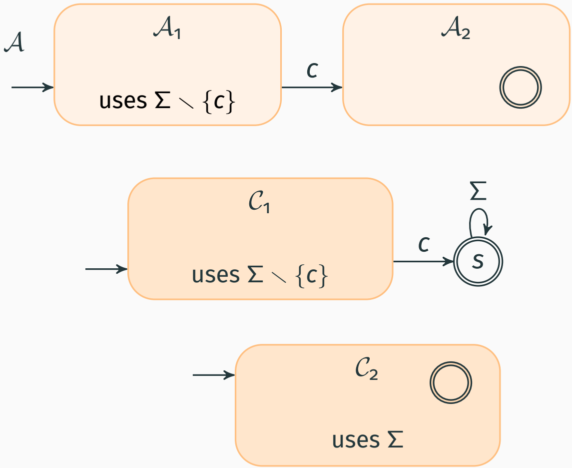
2. $w = ucv$ where $u \in (\Sigma \setminus \{c\})^*$ and $u \notin L_1$

Gate complementation: construction



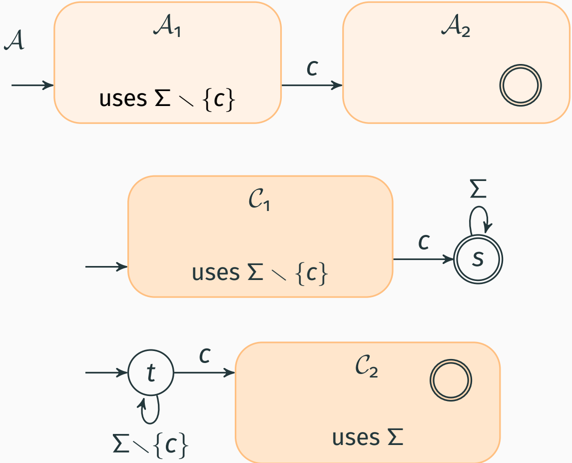
2. $w = ucv$ where $u \in (\Sigma \setminus \{c\})^*$ and $u \notin L_1$

Gate complementation: construction



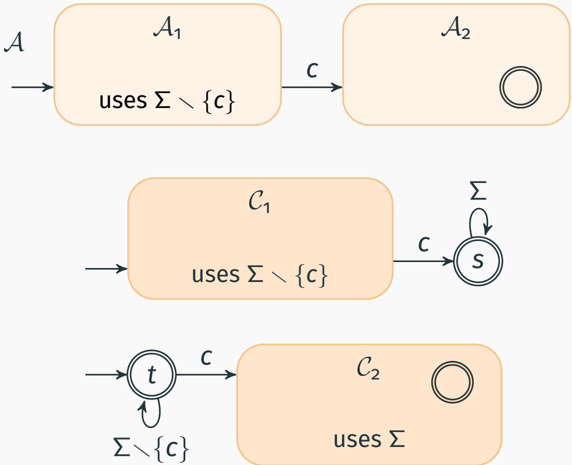
3. $w = ucv$ where $u \in (\Sigma \setminus \{c\})^*$ and $v \notin L_2$

Gate complementation: construction



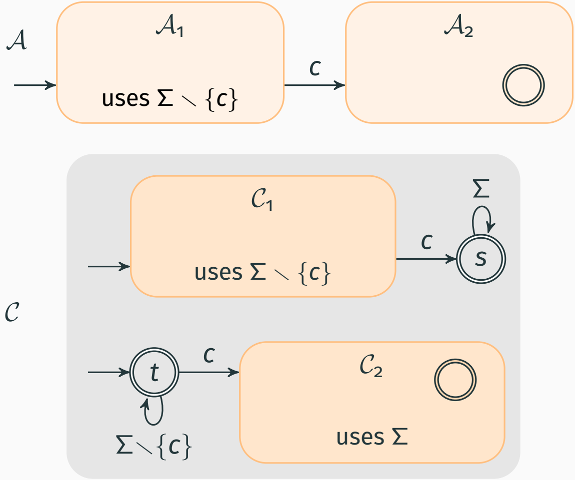
3. $w = ucv$ where $u \in (\Sigma \setminus \{c\})^*$ and $v \notin L_2$

Gate complementation: construction



1. w does not contain any c

Gate complementation: construction



$$|\mathcal{C}| = |\mathcal{C}_1| + |\mathcal{C}_2| + 2$$

More in the paper!

- generalizations of sequential and gate complementation
- more complexity results
- heuristic to choose between forward and reverse powerset

More in the paper!

- generalizations of sequential and gate complementation
- more complexity results
- heuristic to choose between forward and reverse powerset

generalized complementation problem:
port NFA = an NFA with multiple sets of initial and final states

Implementation

- `AliGater`: a Python tool
- all algorithms in their general versions
- backend: `mata` library [1]
- NFA reduction applied on results (`RABIT/Reduce`) [2]

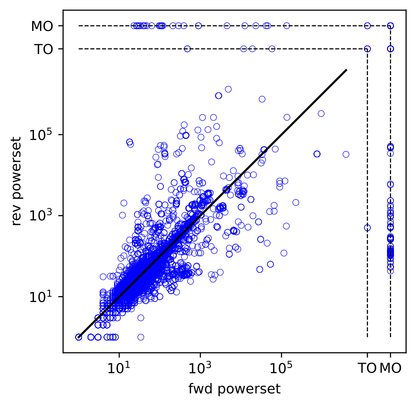
[1] D. Chocholatý et al. **Mata: A Fast and Simple Finite Automata Library.** TACAS'24.
[2] R. Mayr and L. Clemente. **Advanced automata minimization.** POPL'13.

Experimental settings

- 9,450 benchmarks from diverse applications [1]
- all plots compare the complement sizes (number of states)
- `TO` = timeout (5 min), `MO` = out of memory (8 GiB)

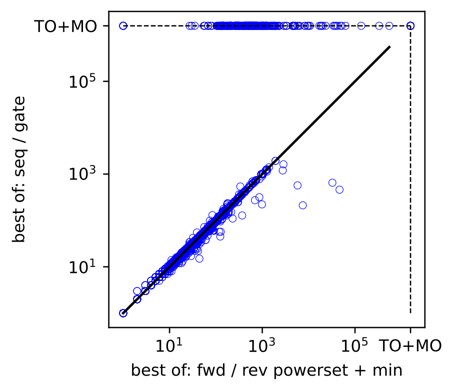
[1] VeriFIT. `nfa-bench:` **Extensive benchmark for reasoning about regular properties.** `https://github.com/VeriFIT/nfa-bench`.

Experimental results: forward vs. reverse powerset



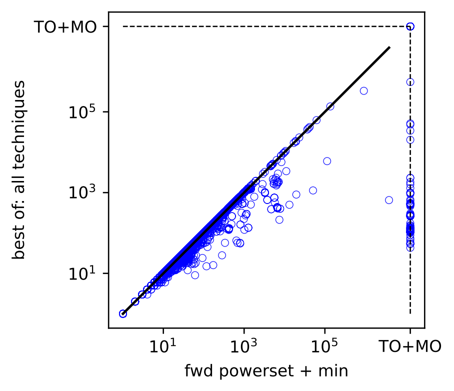
- each method very effective for some automata
- out of resources while the other finishes

Experimental results: sequential + gate



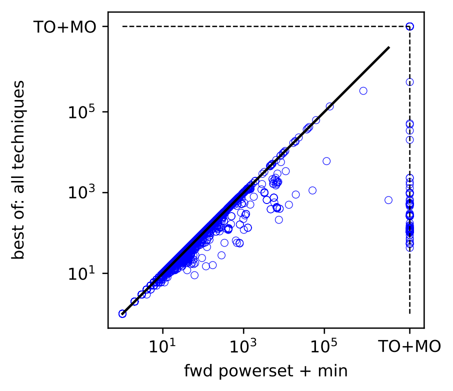
- can give significantly smaller results than powerset
- resource demanding, many timeouts

Conclusion



TO+MO
10^5
10^3
10^1
best of: all techniques
10^1
10^3
10^5
TO+MO
fwd powerset + min

Conclusion

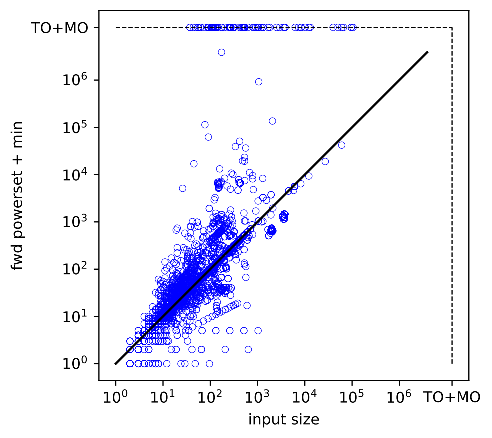


Technical report

Implementation

NFA complementation still has room for improvement:
let's explore it further!

How hard are the benchmarks to determinize?



Comparison of forward and reverse powerset