One-deterministic-counter automata

https://arxiv.org/abs/2301.13456

Prince Mathew

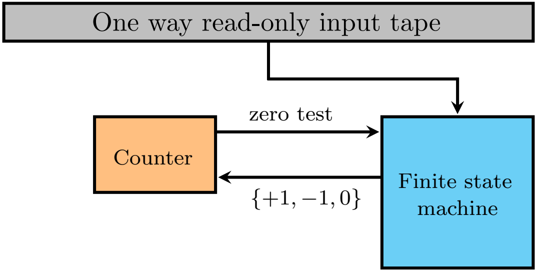
*School of Mathematics & Computer Science
Indian Institute of Technology Goa, India
prince@iitgoa.ac.in[1]

Joint work with Dr. Sreejith A.V., Dr. Vincent Penelle, and Dr. Prakash Saivasan

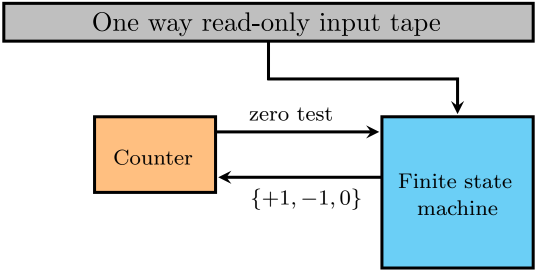
Outline of the talk

1. One-counter Automata
2. One-deterministic-counter automata (ODCA)
   - Our results
3. Weighted One-deterministic-counter automata
   - Reachability problem of ODCA
   - A quick look at other results

One-counter automata (OCA)

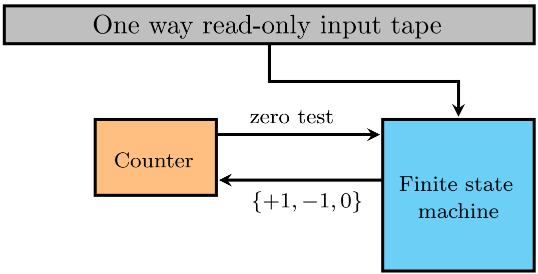


One-counter automata (OCA)



Write to counter: Increment(+1), No change (0), Decrement (-1)

One-counter automata (OCA)



Write to counter: Increment(+1), No change (0), Decrement (-1)
Read from counter: zero (0), or positive (+)

Properties of OCA

- Equivalence of deterministic OCA is in P [2].
- Equivalence of non-deterministic OCA is undecidable [7].
- Equivalence of weighted OCA is open (weights from a field).

One-deterministic-counter automata (ODCA)

One-deterministic-counter automata (ODCA)

Semantic definition

An OCA where all runs of a word lead to the same counter value.

One-deterministic-counter automata (ODCA)

Semantic definition

An OCA where all runs of a word lead to the same counter value.

Syntactic definition

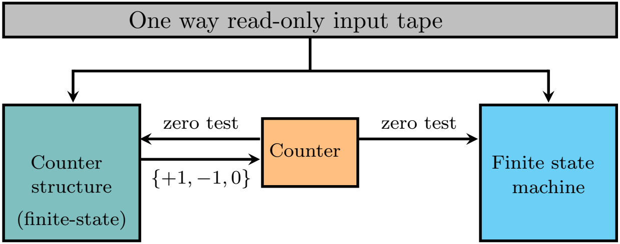


Figure: One-deterministic-counter automata

Examples of ODCA

Example

The following languages/functions are over the alphabet $\Sigma = \{a, b\}$.

(a) The language $\mathcal{L}_1 = \{a^n b a^n \mid n > 0\}$.

(b) The function $f : (a + b)^* \to \mathbb{N}$ defined as follows:
$f(w)$ is the decimal equivalent of w when interpreted as a binary number, if the number of a's $\geq$ number of b's for any prefix of w, and 0 otherwise.

- ODCAs are strict extensions of visibly OCA.

Our results

- Equivalence of deterministic ODCA is in P.
  1. Deterministic OCAs are equivalent to deterministic ODCAs.
  2. Equivalence of deterministic OCA is in P [2].

Our results

- Equivalence of deterministic ODCA is in P.
    1. Deterministic OCAs are equivalent to deterministic ODCAs.
    2. Equivalence of deterministic OCA is in P [2].
- Equivalence of non-deterministic ODCA is in PSPACE.
    1. A non-deterministic ODCA is equivalent to an exponential sized deterministic ODCA.
    2. A PSPACE machine can do this.

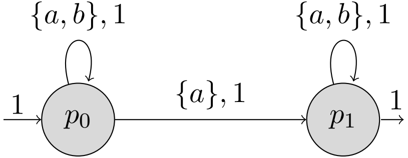
Our results

- Equivalence of deterministic ODCA is in P.
  1. Deterministic OCAs are equivalent to deterministic ODCAs.
  2. Equivalence of deterministic OCA is in P [2].
- Equivalence of non-deterministic ODCA is in PSPACE.
  1. A non-deterministic ODCA is equivalent to an exponential sized deterministic ODCA.
  2. A PSPACE machine can do this.
- Equivalence of weighted ODCA is in P (main result).
  1. Equivalence of weighted finite automata is in P [6] (weights from a field).
  2. We reduce equivalence of weighted ODCA to that of weighted finite automata.

Weighted One-deterministic-counter automata

Weighted Automata

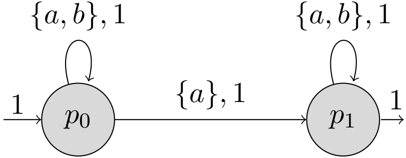
- A weighted automata (WA) can be viewed as an NFA with weights.
- In this talk, we assume that the weights always comes from a field.
- Consider the following WA.



- Function recognised?

Weighted Automata

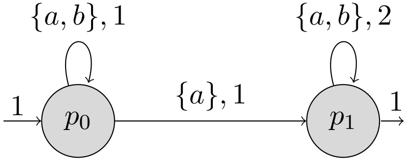
- A weighted automata (WA) can be viewed as an NFA with weights.
- In this talk, we assume that the weights always comes from a field.
- Consider the following WA.



- Function recognised?
    - Number of 'a's in the input.

Another example

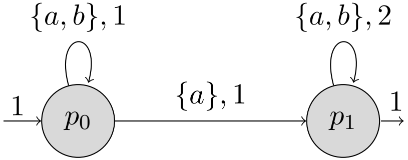
- Consider the following WA.



- Function recognised?

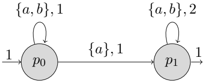
Another example

- Consider the following WA.



- Function recognised?
  - Decimal equivalent of binary number.

Evaluation



Initial distribution

λ

1	0

Transition matrices

$\delta(a)$

1	1
0	2

$\delta(b)$

1	0
0	2

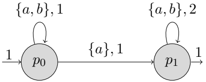
Final distribution

η

0
1

- Let $w = a_1 a_2 \dots a_n$ be a word.

Evaluation



Initial distribution

λ

1	0

Transition matrices

$\delta(a)$

1	1
0	2

$\delta(b)$

1	0
0	2

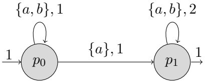
Final distribution

η

0
1

- Let $w = a_1 a_2 \dots a_n$ be a word.
- We define $\delta(w) = \delta(a_1)\delta(a_2)\cdots\delta(a_n)$.

Evaluation



Initial distribution

λ

1	0

Transition matrices

$\delta(a)$

1	1
0	2

$\delta(b)$

1	0
0	2

Final distribution

η

0
1

- Let $w = a_1 a_2 \dots a_n$ be a word.
- We define $\delta(w) = \delta(a_1)\delta(a_2)\cdots\delta(a_n)$.
- Accepting weight of $w = \lambda\delta(w)\eta$.

Weighted visibly ODCA

1. There is an initial distribution on the finite state machine: λ.
2. There is a final distribution on the finite state machine: ρ.
3. For every letter a, there are two matrices: $\delta_0(a)$ and $\delta_+(a)$.
4. For a word $w = a_1 a_2 \ldots a_n$

$$\delta(w) = \delta_{r_1}(a_1)\delta_{r_2}(a_2)\cdots\delta_{r_n}(a_n)$$

where $r_i \in \{0, +\}$.

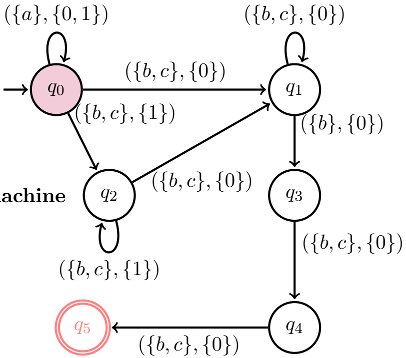
Example - One deterministic-counter Automata

$$L = \{a^m(b+c)^n b(b+c)^2 \mid n, m \in \mathbb{N} \text{ with } n > m\}.$$

a	a	b	c	b	b	c	b

Input tape

Finite state machine



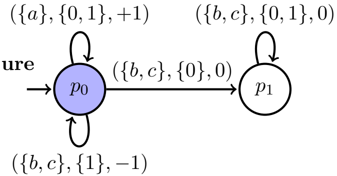
0

Counter

1	0	0	0	0	0

Initial vector

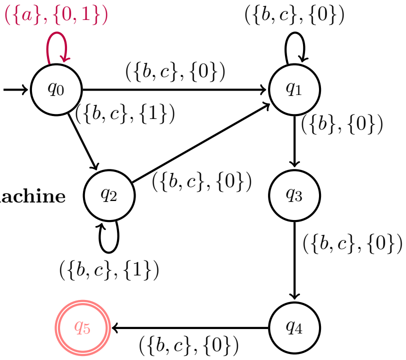
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

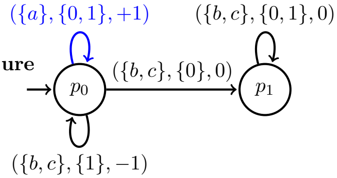


0

Counter

1	0	0	0	0	0

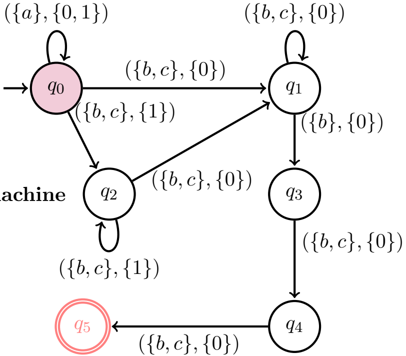
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

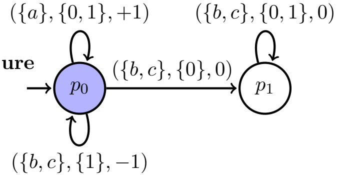


1

Counter

1	0	0	0	0	0

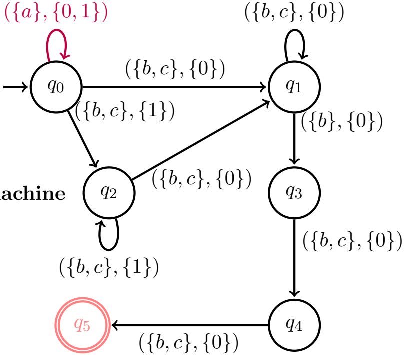
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

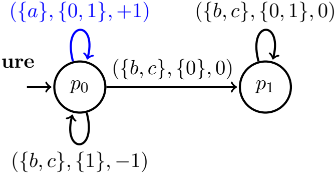


1

Counter

1	0	0	0	0	0

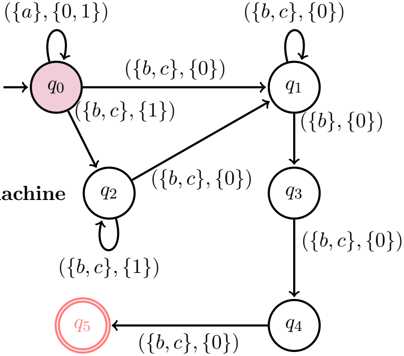
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

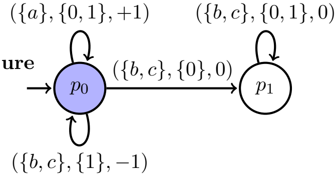


2

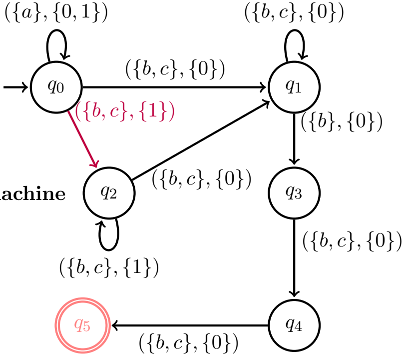
Counter

1	0	0	0	0	0

Counter structure



Finite state machine



a	a	b	c	b	b	c	b

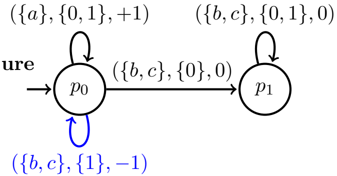
Input tape

2

Counter

1	0	0	0	0	0

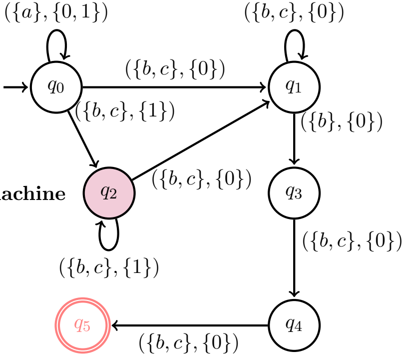
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

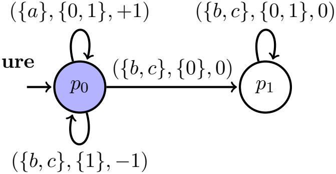


1

Counter

0	0	1	0	0	0

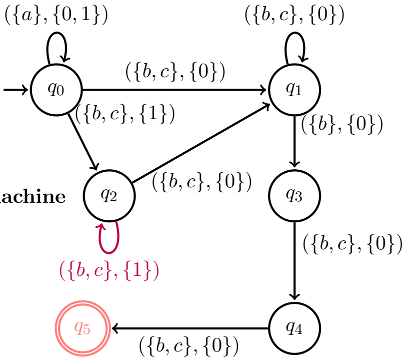
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

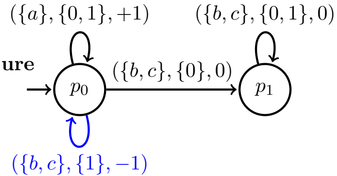


1

Counter

0	0	1	0	0	0

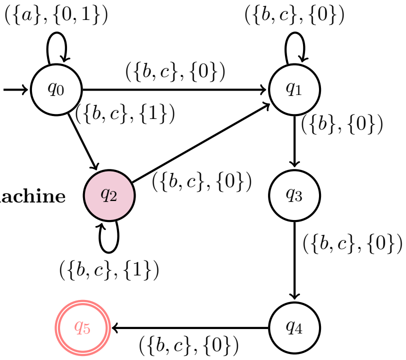
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

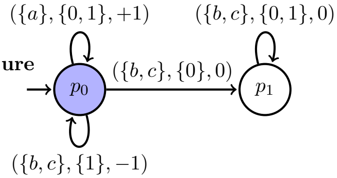


0

Counter

0	0	1	0	0	0

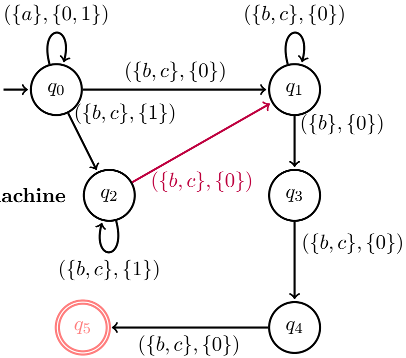
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

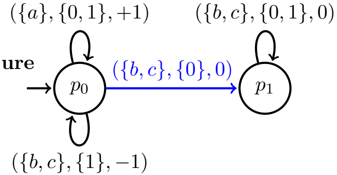


0

Counter

0	0	1	0	0	0

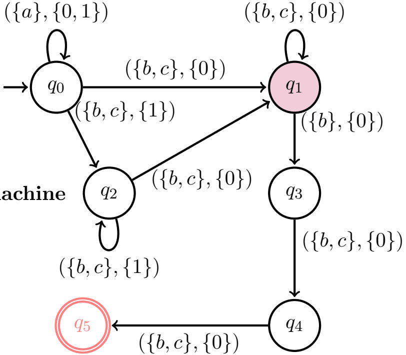
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

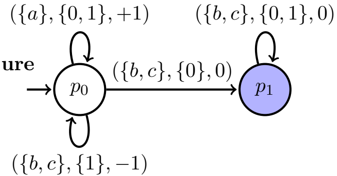


0

Counter

0	1	0	0	0	0

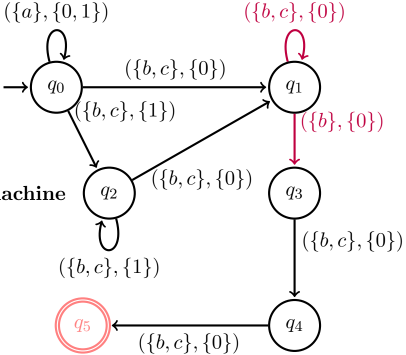
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

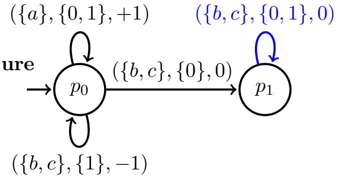


0

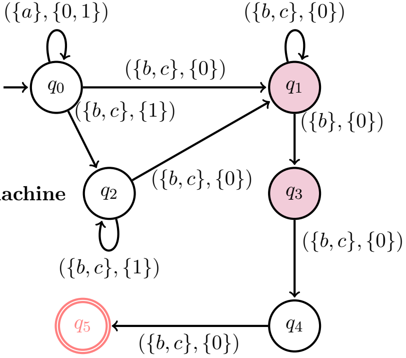
Counter

0	1	0	0	0	0

Counter structure



Finite state machine



a	a	b	c	b	b	c	b

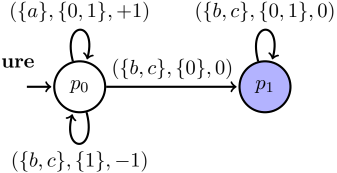
Input tape

0

Counter

0	1	0	1	0	0

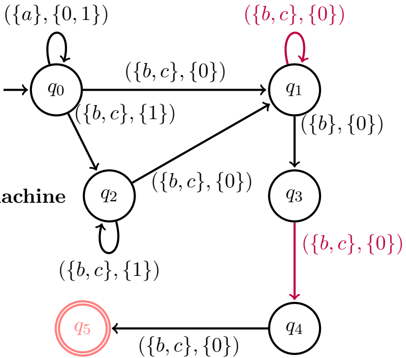
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

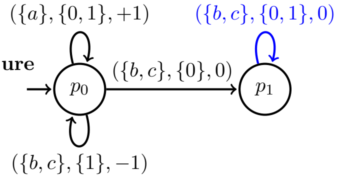


0

Counter

0	1	0	1	0	0

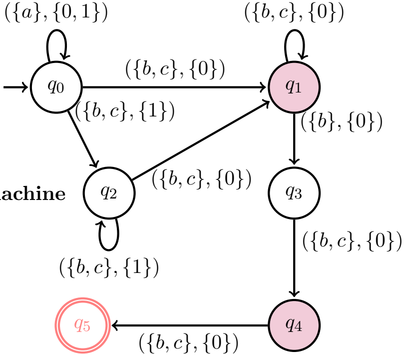
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

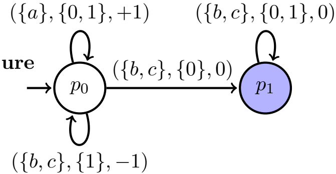


0

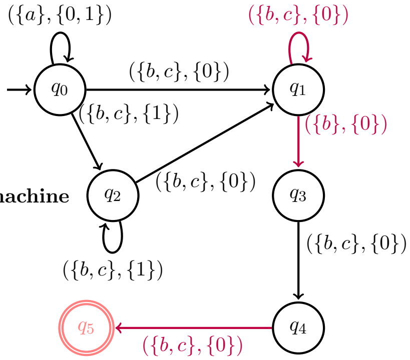
Counter

0	1	0	0	1	0

Counter structure



Finite state machine



a	a	b	c	b	b	c	b

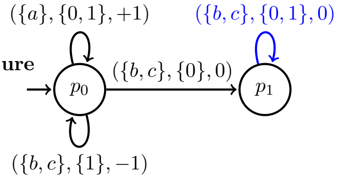
Input tape

0

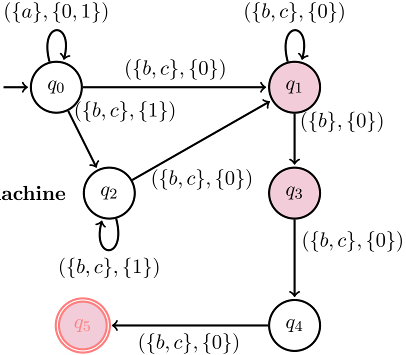
Counter

0	1	0	0	1	0

Counter structure



Finite state machine



a	a	b	c	b	b	c	b

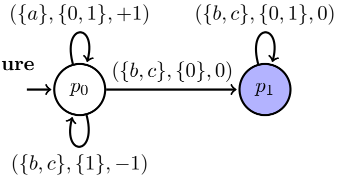
Input tape

0

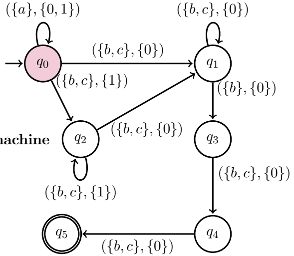
Counter

0	1	0	1	0	1

Counter structure

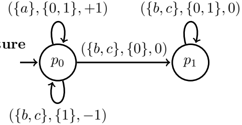


Finite state machine



x

Counter structure



0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\}, \{1\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

$(\{b\}, \{0\})$

$(\{a\}, \{0,1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b,c\}, \{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\}, \{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

$(\{c\}, \{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

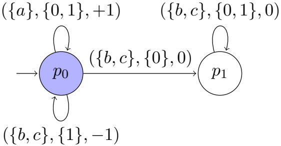
Transition matrices

a	c	c	b	c	b

Input tape

1	0	0	0	0	0

Initial vector



0

Counter structure **Counter**

$(\{a\}, \{0,1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b, c\}, \{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\}, \{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

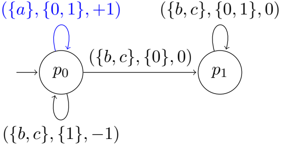
$(\{c\}, \{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

a	c	c	b	c	b

1	0	0	0	0	0



0

Counter

$(\{a\},\{0,1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b,c\},\{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\},\{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

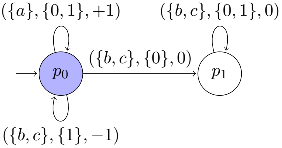
$(\{c\},\{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

a	c	c	b	c	b

1	0	0	0	0	0



1

Counter

$(\{a\}, \{0,1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b,c\}, \{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\}, \{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

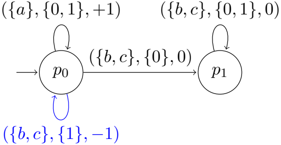
$(\{c\}, \{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

a	c	c	b	c	b

1	0	0	0	0	0



1

Counter

$(\{a\}, \{0, 1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b, c\}, \{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\}, \{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

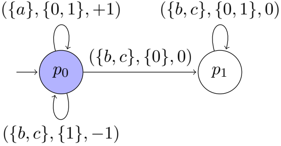
$(\{c\}, \{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

a	c	c	b	c	b

0	0	1	0	0	0



0

Counter

$(\{a\},\{0,1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b,c\},\{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\},\{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

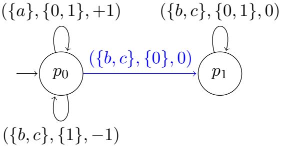
$(\{c\},\{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

a	c	c	b	c	b

0	0	1	0	0	0



0

Counter

$(\{a\}, \{0, 1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b, c\}, \{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\}, \{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

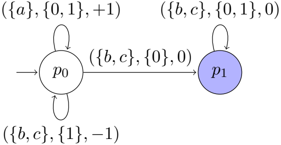
$(\{c\}, \{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

a	c	c	b	c	b

0	1	0	0	0	0



0

Counter

$(\{a\},\{0,1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b,c\},\{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\},\{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

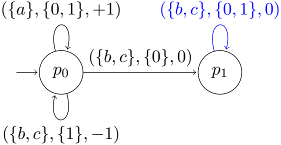
$(\{c\},\{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

a	c	c	b	c	b

0	1	0	0	0	0



$\boxed{0}$

Counter

$(\{a\}, \{0,1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b,c\}, \{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\}, \{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

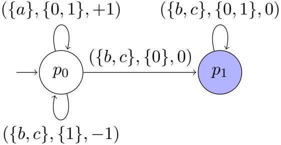
$(\{c\}, \{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

a	c	c	b	c	b

0	1	0	1	0	0



0

Counter

$(\{a\}, \{0,1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b,c\}, \{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\}, \{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

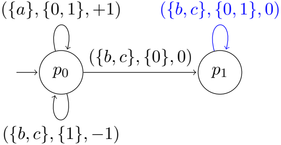
$(\{c\}, \{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

a	c	c	b	c	b

0	1	0	1	0	0



0

Counter

$(\{a\}, \{0,1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b, c\}, \{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\}, \{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

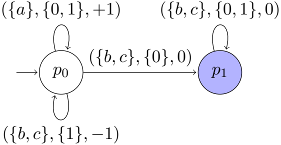
$(\{c\}, \{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

a	c	c	b	c	b

0	1	0	0	1	0



0

Counter

$(\{a\},\{0,1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b,c\},\{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\},\{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

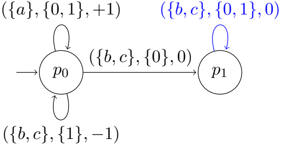
$(\{c\},\{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

a	c	c	b	c	b

0	1	0	0	1	0



0

Counter

$(\{a\}, \{0,1\})$

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b,c\}, \{1\})$

0	0	1	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$(\{b\}, \{0\})$

0	1	0	0	0	0
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

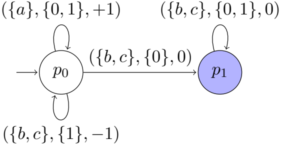
$(\{c\}, \{0\})$

0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

.

a	c	c	b	c	b

0	1	0	1	0	1



0

Counter

Reachability problem

Reachability

co-VS Reachability problem

INPUT:

- A weighted visibly OCA $\mathcal{A}$ over a field,
- an initial configuration c,
- a vector space $\mathcal{V}$, and
- counter value m.

Reachability

co-VS Reachability **problem**

INPUT:

- A weighted visibly OCA $\mathcal{A}$ over a field,
- an initial configuration c,
- a vector space $\mathcal{V}$, and
- counter value m.

OUTPUT:

- *Yes*, if there exists a run $\mathsf{c} \xrightarrow{*} \overline{\mathcal{V}} \times \{m\}$ in $\mathcal{A}$, and
- *No*, otherwise.

Reachability

co-VS Reachability problem

INPUT:

- A weighted visibly OCA $\mathcal{A}$ over a field,
- an initial configuration c,
- a vector space $\mathcal{V}$, and
- counter value m.

OUTPUT:

- *Yes*, if there exists a run $\mathsf{c} \xrightarrow{*} \overline{\mathcal{V}} \times \{m\}$ in $\mathcal{A}$, and
- *No*, otherwise.

- $z \in \Sigma^*$ is a reachability witness for $(\mathsf{c}, \overline{\mathcal{V}}, m)$ if $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times \{m\}$.

co-VS reachability

Theorem - co-VS reachability

co-VS reachability is decidable in polynomial time.

- We prove this by showing a pseudo-pumping lemma.

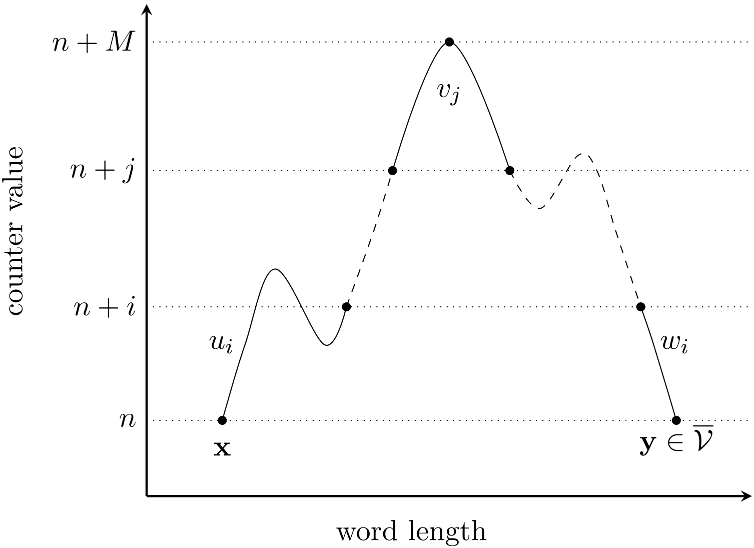
Pseudo-pumping lemma (pumping down)

Pseudo-pumping lemma

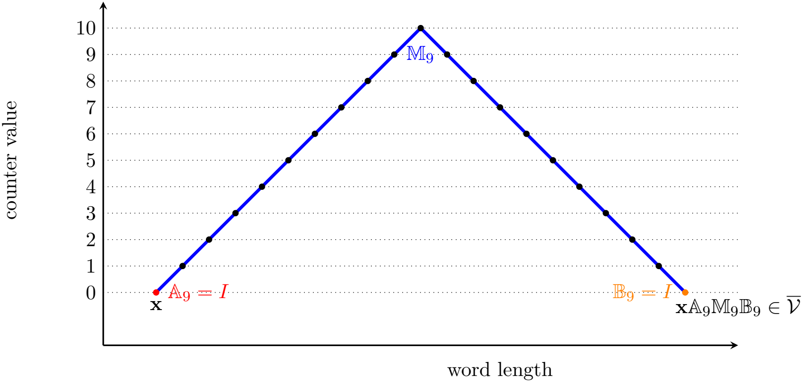
If z is a reachability witness for $(\mathsf{c}, \overline{\mathcal{V}}, m)$ and the maximum counter value encountered during the run of z is not polynomially bounded in the input size then,

1. there exists a subword z_{sub} of z, such that $\mathsf{c} \xrightarrow{z_{sub}} \overline{\mathcal{V}} \times \{m\}$, and the maximum counter value encountered during this run is less than the maximum counter value encountered during the run of z.

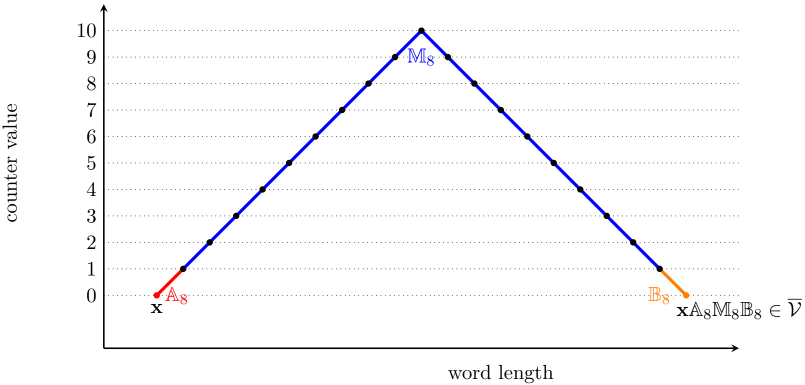
Pseudo-pumping lemma - Figure



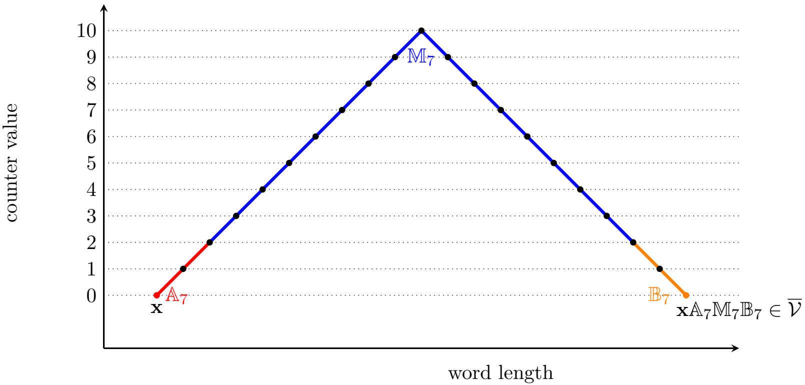
Pseudo-pumping lemma



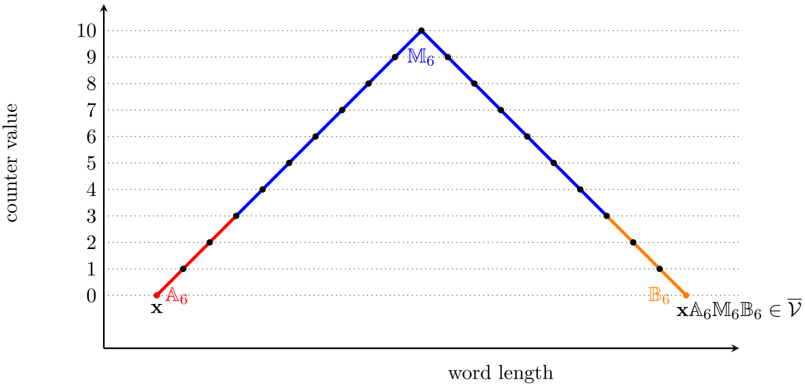
Pseudo-pumping lemma



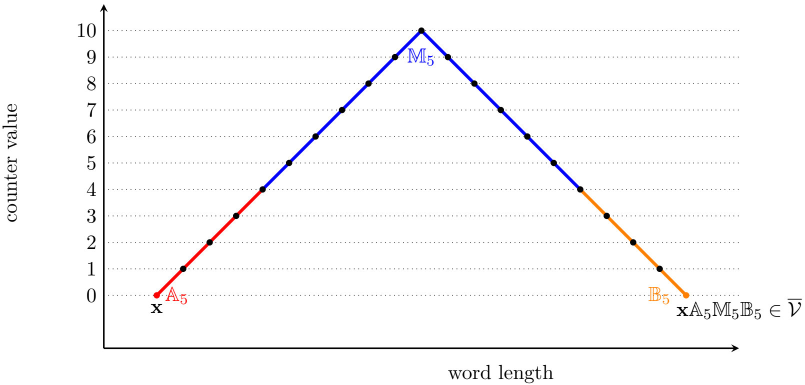
Pseudo-pumping lemma



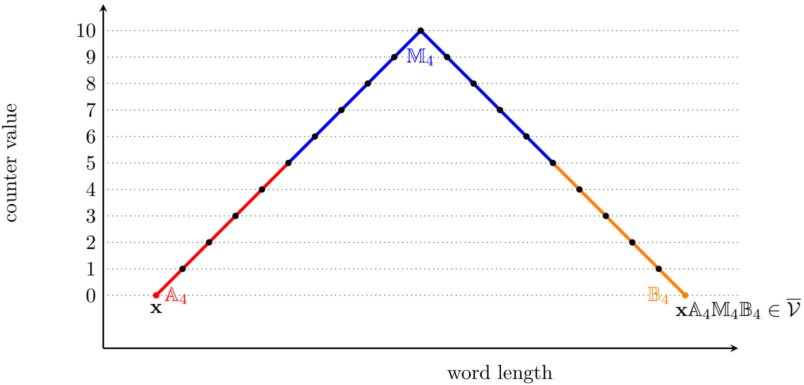
Pseudo-pumping lemma



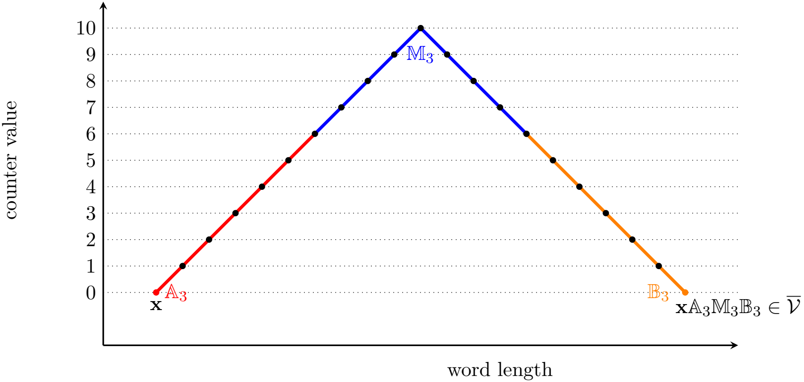
Pseudo-pumping lemma



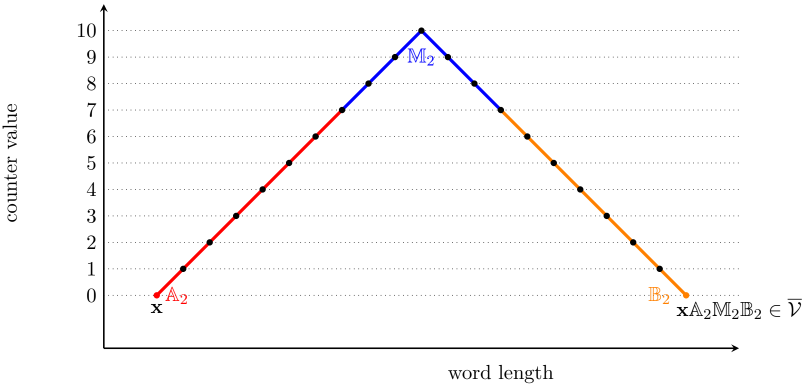
Pseudo-pumping lemma



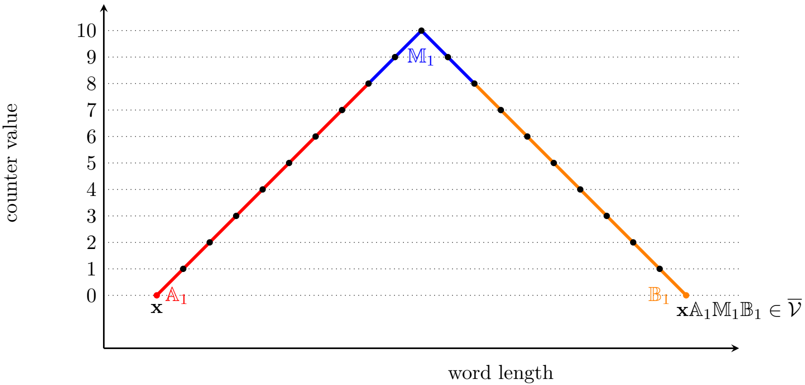
Pseudo-pumping lemma



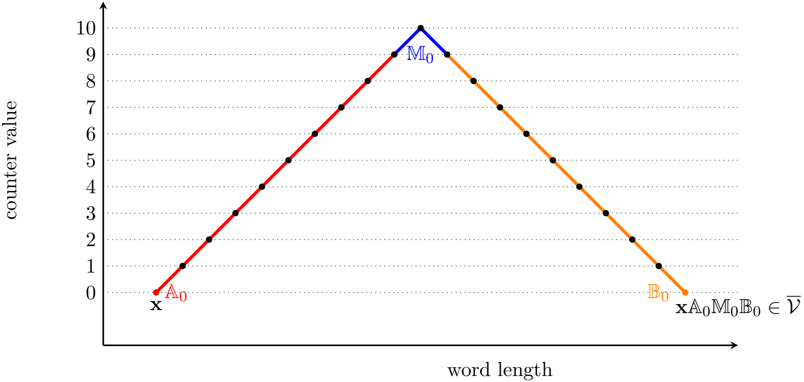
Pseudo-pumping lemma



Pseudo-pumping lemma

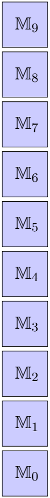


Pseudo-pumping lemma



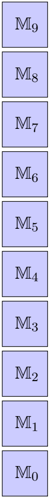
Pseudo-pumping lemma

- Assume for all $i \in [0, 9]$, $\mathbb{M}_i$ is a 3×3 matrices.



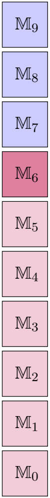
Pseudo-pumping lemma

- Assume for all $i \in [0,9]$, $\mathbb{M}_i$ is a 3×3 matrices.
- For all $i \in [0,9]$, $\mathbf{x}\mathbb{A}_i\mathbb{M}_i\mathbb{B}_i \in \overline{\mathcal{V}}$.



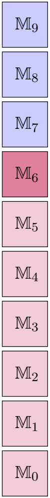
Pseudo-pumping lemma

- Assume for all $i \in [0, 9]$, $\mathbb{M}_i$ is a 3×3 matrix.
- For all $i \in [0, 9]$, $\mathbf{x}\mathbb{A}_i\mathbb{M}_i\mathbb{B}_i \in \overline{\mathcal{V}}$.
- $\exists j \in [0, 9]$, $\mathbb{M}_j = \alpha_0\mathbb{M}_0 + \alpha_1\mathbb{M}_1 + \cdots + \alpha_{j-1}\mathbb{M}_{j-1}$.



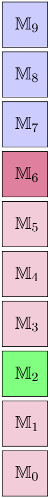
Pseudo-pumping lemma

- Assume for all $i \in [0, 9]$, $\mathbb{M}_i$ is a 3×3 matrix.
- For all $i \in [0, 9]$, $\mathbf{x}\mathbb{A}_i\mathbb{M}_i\mathbb{B}_i \in \overline{\mathcal{V}}$.
- $\exists j \in [0, 9]$, $\mathbb{M}_j = \alpha_0\mathbb{M}_0 + \alpha_1\mathbb{M}_1 + \cdots + \alpha_{j-1}\mathbb{M}_{j-1}$.
- $\mathbf{x}\mathbb{A}_j(\alpha_0\mathbb{M}_0 + \alpha_1\mathbb{M}_1 + \cdots + \alpha_{j-1}\mathbb{M}_{j-1})\mathbb{B}_j \in \overline{\mathcal{V}}$.

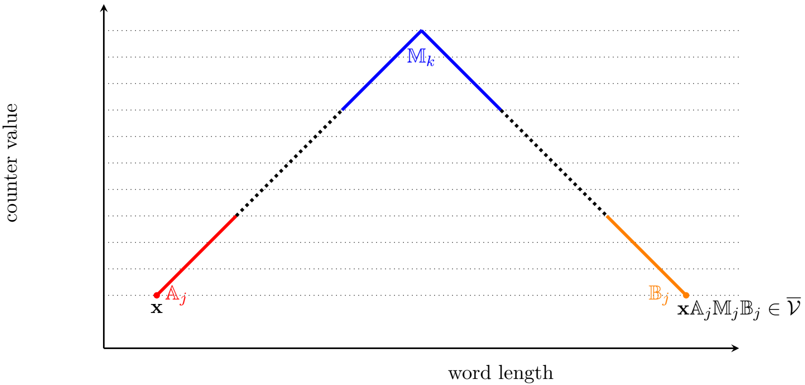


Pseudo-pumping lemma

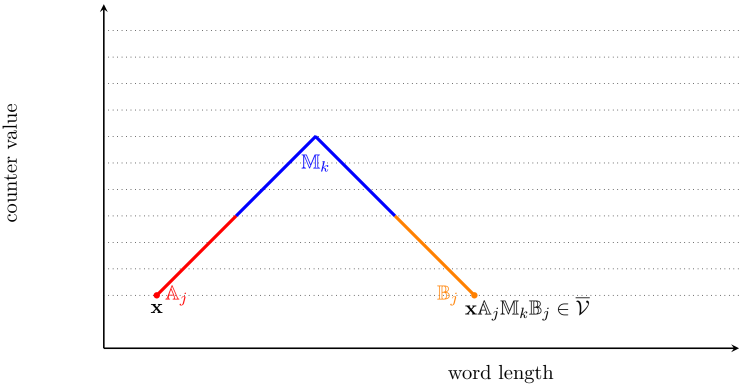
- Assume for all $i \in [0, 9]$, $\mathbb{M}_i$ is a 3×3 matrix.
- For all $i \in [0, 9]$, $\mathbf{x}\mathbb{A}_i\mathbb{M}_i\mathbb{B}_i \in \overline{\mathcal{V}}$.
- $\exists j \in [0, 9]$, $\mathbb{M}_j = \alpha_0\mathbb{M}_0 + \alpha_1\mathbb{M}_1 + \cdots + \alpha_{j-1}\mathbb{M}_{j-1}$.
- $\mathbf{x}\mathbb{A}_j(\alpha_0\mathbb{M}_0 + \alpha_1\mathbb{M}_1 + \cdots + \alpha_{j-1}\mathbb{M}_{j-1})\mathbb{B}_j \in \overline{\mathcal{V}}$.
- There exists $k \in [0, j-1]$, $\mathbf{x}\mathbb{A}_j\mathbb{M}_k\mathbb{B}_j \in \overline{\mathcal{V}}$.



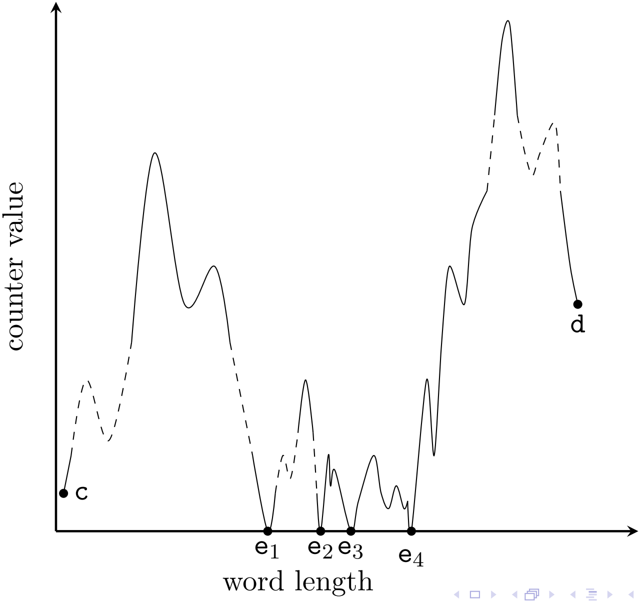
Pseudo-pumping lemma



Pseudo-pumping lemma



Multiple cuts



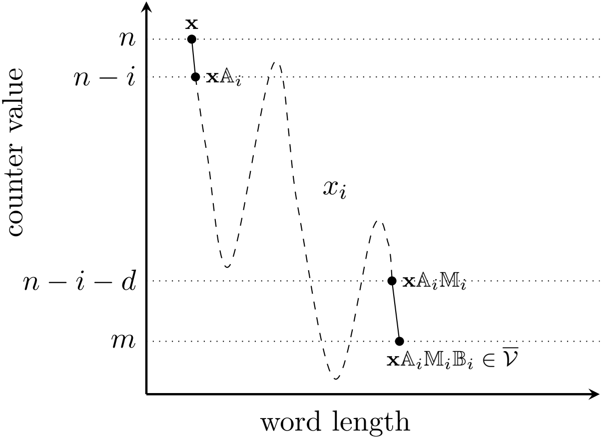
Special word Lemma

Lemma- Special word

The lexicographically minimal reachability witness z, if it exists, satisfies the following conditions :

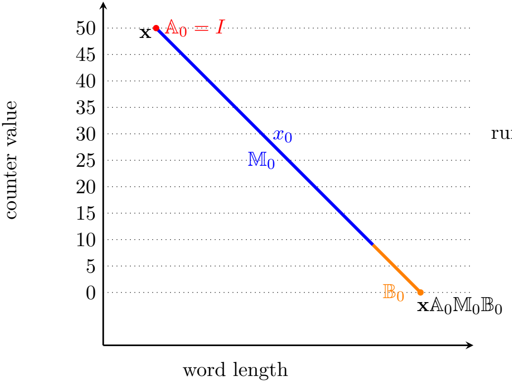
1. $z = uy_1^{r_1}w_1y_2^{r_2}w_2$ such that $|uy_1w_1y_2w_2|$ is polynomially bounded in input size, and
2. r_1, r_2 are polynomially bounded in input size.

Special word- Claim 1



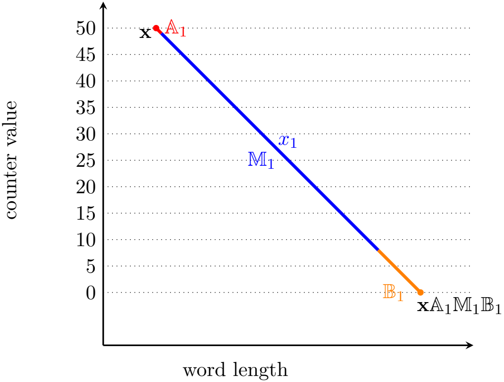
- We show that a factor repeats in the lexicographically minimal word if d is large enough.

Repeating subword

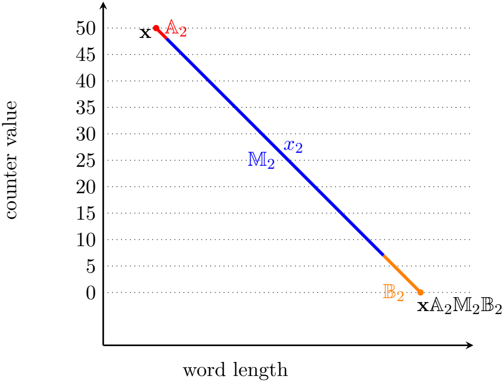


run of the lexicographically minimal witness z

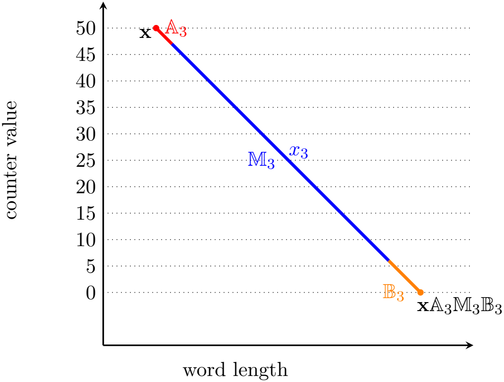
Repeating subword



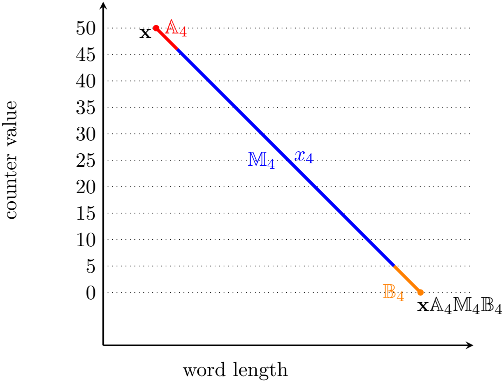
Repeating subword



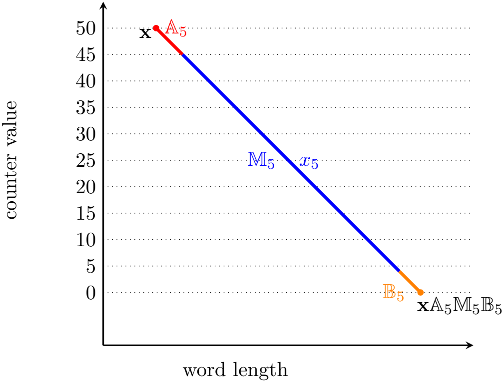
Repeating subword



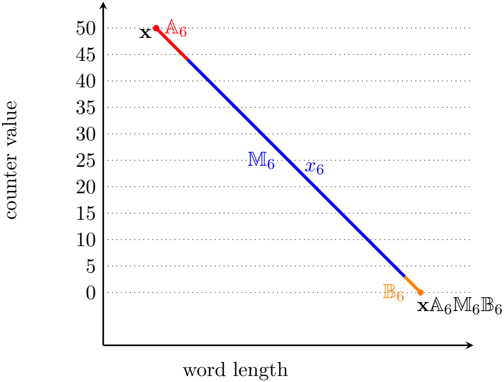
Repeating subword



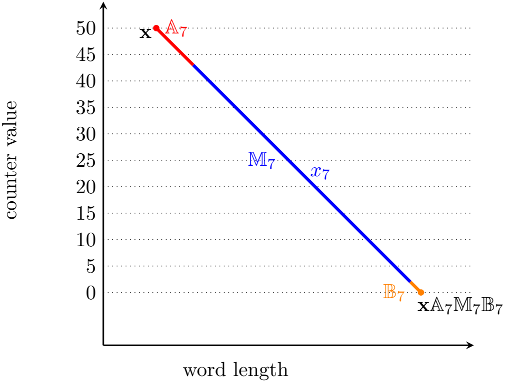
Repeating subword



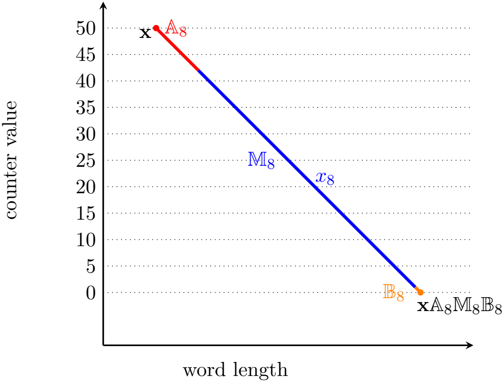
Repeating subword



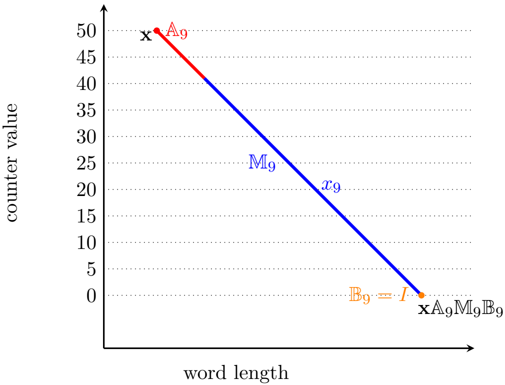
Repeating subword



Repeating subword

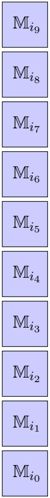


Repeating subword



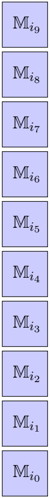
Special word- Claim 1

- Assume for all $i \in [0, 9]$, $\mathbb{M}_i$ is a 3×3 matrix.



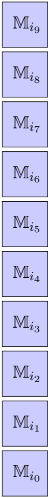
Special word- Claim 1

- Assume for all $i \in [0, 9]$, $\mathbb{M}_i$ is a 3×3 matrix.
- Arrange the words $x_0, x_1, \ldots, x_9$ in the lexicographical ordering.



Special word- Claim 1

- Assume for all $i \in [0, 9]$, $\mathbb{M}_i$ is a 3×3 matrix.
- Arrange the words $x_0, x_1, \ldots, x_9$ in the lexicographical ordering.
- Let $x_{i_0} < x_{i_1} < \ldots < x_{i_9}$.



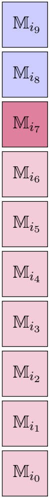
Special word- Claim 1

- Assume for all $i \in [0, 9]$, $\mathbb{M}_i$ is a 3×3 matrix.
- Arrange the words $x_0, x_1, \ldots, x_9$ in the lexicographical ordering.
- Let $x_{i_0} < x_{i_1} < \ldots < x_{i_9}$.
- For all $j \in [0, 9]$, $\mathbf{x}\mathbb{A}_{i_j}\mathbb{M}_{i_j}\mathbb{B}_{i_j} \in \overline{\mathcal{V}}$.

$\mathbb{M}_{i_9}$
$\mathbb{M}_{i_8}$
$\mathbb{M}_{i_7}$
$\mathbb{M}_{i_6}$
$\mathbb{M}_{i_5}$
$\mathbb{M}_{i_4}$
$\mathbb{M}_{i_3}$
$\mathbb{M}_{i_2}$
$\mathbb{M}_{i_1}$
$\mathbb{M}_{i_0}$

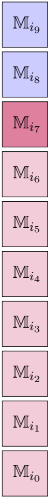
Special word- Claim 1

- Assume for all $i \in [0, 9]$, $\mathbb{M}_i$ is a 3×3 matrix.
- Arrange the words $x_0, x_1, \ldots, x_9$ in the lexicographical ordering.
- Let $x_{i_0} < x_{i_1} < \ldots < x_{i_9}$.
- For all $j \in [0, 9]$, $\mathbf{x}\mathbb{A}_{i_j}\mathbb{M}_{i_j}\mathbb{B}_{i_j} \in \overline{\mathcal{V}}$.
- There exists $j \in [0, 9]$, $\mathbb{M}_{i_j} = \alpha_0\mathbb{M}_{i_0} + \alpha_1\mathbb{M}_{i_1} + \cdots + \alpha_{j-1}\mathbb{M}_{i_{j-1}}$.



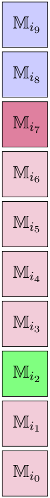
Special word- Claim 1

- Assume for all $i \in [0, 9]$, $\mathbb{M}_i$ is a 3×3 matrix.
- Arrange the words $x_0, x_1, \ldots, x_9$ in the lexicographical ordering.
- Let $x_{i_0} < x_{i_1} < \ldots < x_{i_9}$.
- For all $j \in [0, 9]$, $\mathbf{x}\mathbb{A}_{i_j}\mathbb{M}_{i_j}\mathbb{B}_{i_j} \in \overline{\mathcal{V}}$.
- There exists $j \in [0, 9]$, $\mathbb{M}_{i_j} = \alpha_0\mathbb{M}_{i_0} + \alpha_1\mathbb{M}_{i_1} + \cdots + \alpha_{j-1}\mathbb{M}_{i_{j-1}}$.
- $\mathbf{x}\mathbb{A}_{i_j}(\alpha_0\mathbb{M}_{i_0} + \alpha_1\mathbb{M}_{i_1} + \cdots + \alpha_{j-1}\mathbb{M}_{i_{j-1}})\mathbb{B}_{i_j} \in \overline{\mathcal{V}}$.

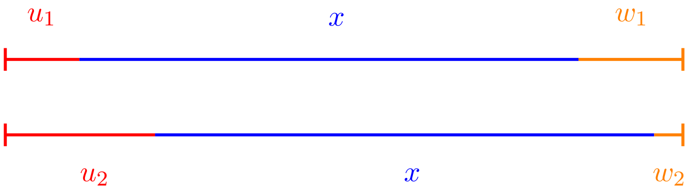


Special word- Claim 1

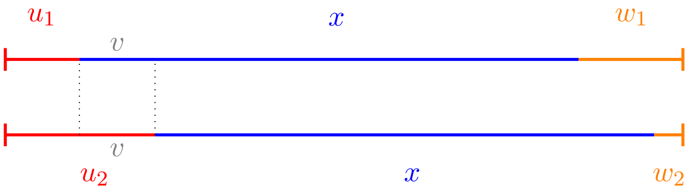
- Assume for all $i \in [0, 9]$, $\mathbb{M}_i$ is a 3×3 matrix.
- Arrange the words $x_0, x_1, \ldots, x_9$ in the lexicographical ordering.
- Let $x_{i_0} < x_{i_1} < \ldots < x_{i_9}$.
- For all $j \in [0, 9]$, $\mathbf{x}\mathbb{A}_{i_j}\mathbb{M}_{i_j}\mathbb{B}_{i_j} \in \overline{\mathcal{V}}$.
- There exists $j \in [0, 9]$, $\mathbb{M}_{i_j} = \alpha_0\mathbb{M}_{i_0} + \alpha_1\mathbb{M}_{i_1} + \cdots + \alpha_{j-1}\mathbb{M}_{i_{j-1}}$.
- $\mathbf{x}\mathbb{A}_{i_j}(\alpha_0\mathbb{M}_{i_0} + \alpha_1\mathbb{M}_{i_1} + \cdots + \alpha_{j-1}\mathbb{M}_{i_{j-1}})\mathbb{B}_{i_j} \in \overline{\mathcal{V}}$.
- There exists $k \in [0, j-1]$, $\mathbf{x}\mathbb{A}_{i_j}\mathbb{M}_{i_k}\mathbb{B}_{i_j} \in \overline{\mathcal{V}}$.



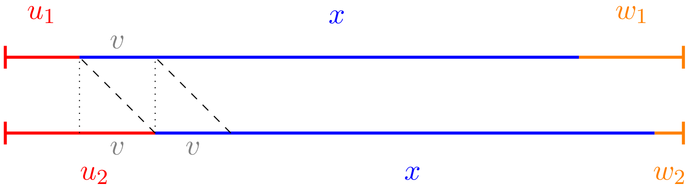
Special word- Claim 2



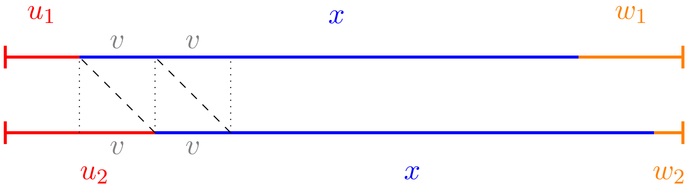
Special word- Claim 2



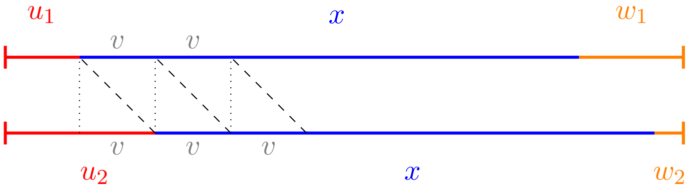
Special word- Claim 2



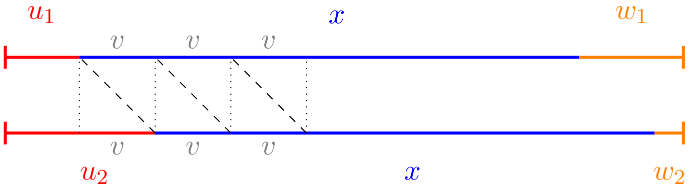
Special word- Claim 2



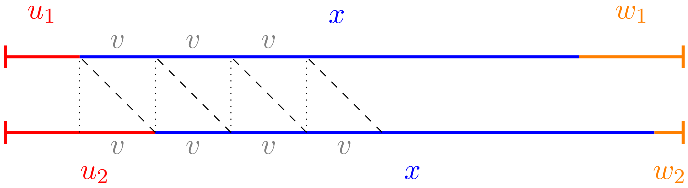
Special word- Claim 2



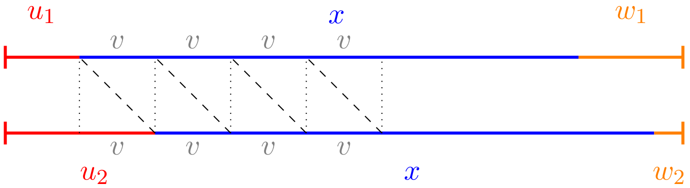
Special word- Claim 2



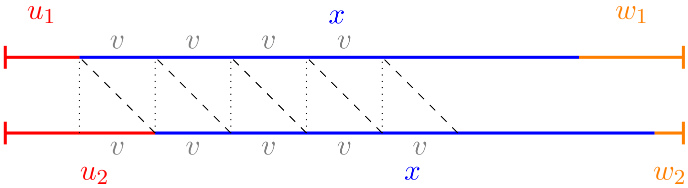
Special word- Claim 2



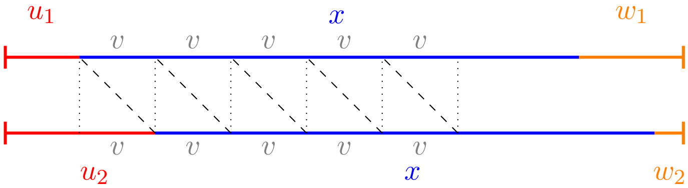
Special word- Claim 2



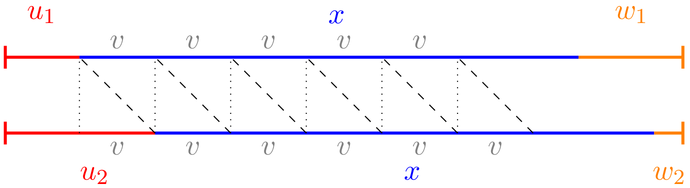
Special word- Claim 2



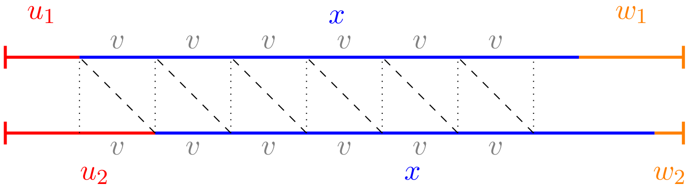
Special word- Claim 2



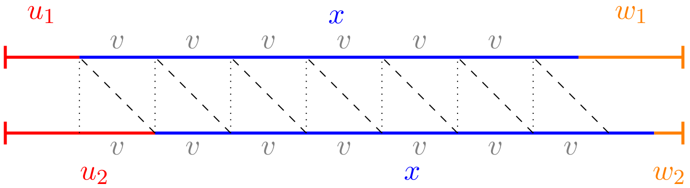
Special word- Claim 2



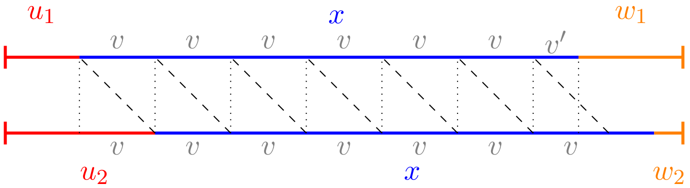
Special word- Claim 2



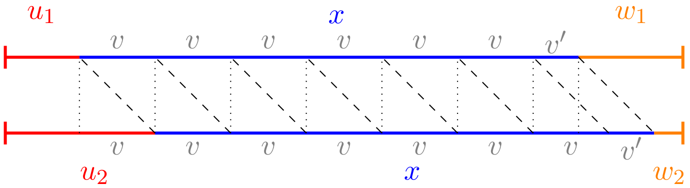
Special word- Claim 2



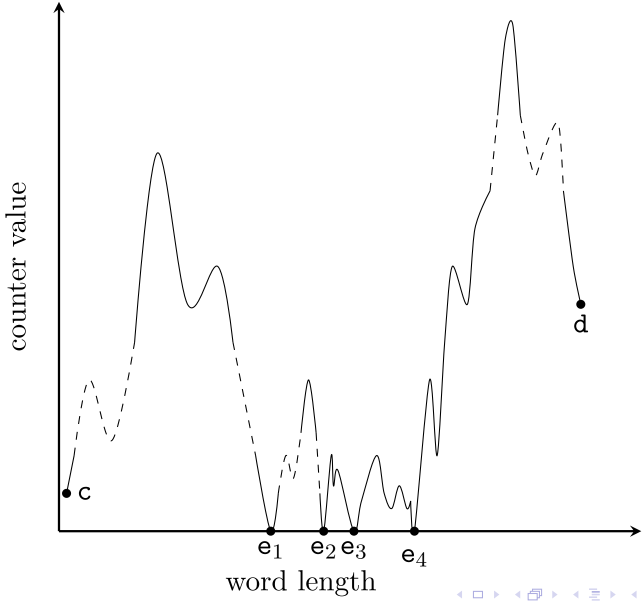
Special word- Claim 2



Special word- Claim 2



Multiple cuts



Equivalence

Equivalence

Lemma - Witness bound

If two weighted ODCA*s* $\mathcal{A}_1$ and $\mathcal{A}_2$ are not equivalent, then there exists a witness z such that the counter values encountered during the run of z are less than a polynomial in the input size.

Equivalence

Lemma - Witness bound

If two weighted ODCA*s* $\mathcal{A}_1$ and $\mathcal{A}_2$ are not equivalent, then there exists a witness z such that the counter values encountered during the run of z are less than a polynomial in the input size.

Theorem - Equivalence

There is a polynomial time algorithm to check the equivalence of two weighted ODCAs.

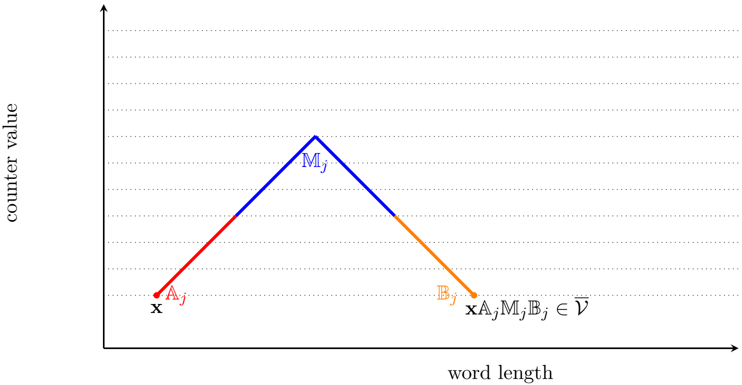
Other results

Other results

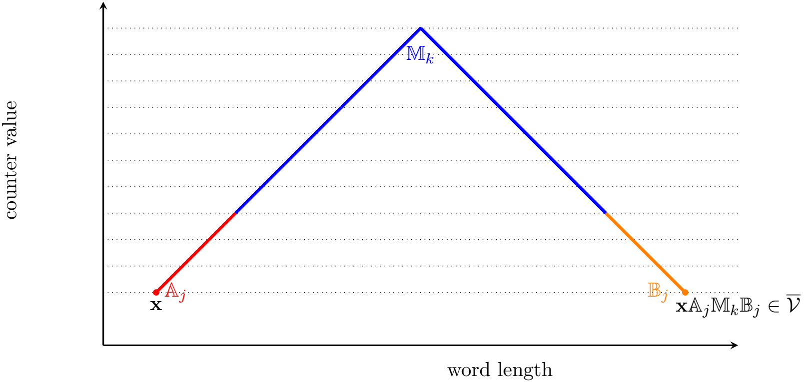
Theorem - Regularity

Given a ODCA, determining whether there exists a weighted automata recognising the same function is in P.

Regularity - pumping up



Regularity - pumping up



Other results

Theorem - Regularity

Given a ODCA, determining whether there exists a weighted automata recognising the same function is in P.

Other results

Theorem - Regularity

Given a ODCA, determining whether there exists a weighted automata recognising the same function is in P.

Theorem - Covering

Given two ODCAs $\mathcal{A}_1$, $\mathcal{A}_2$ without initial configurations, determining whether for all initial distributions of $\mathcal{A}_1$ does $\mathcal{A}_2$ have an initial distribution which makes them equivalent is in P.

Summary

- co-VS reachability problem of ODCAs is in P.
- Equivalence of weighted ODCAs is in P.
- Regularity of weighted ODCAs is in P.

Existing Results

Machine	**Equivalence**	**Reference**
DFA	P (NL)	[4]
DOCA	P (NL-Complete)	[1]
NOCA	Undecidable	[1]
DPDA	Decidable	[5]
NPDA	Undecidable	[1]
PA	P	[6]
DWROCA	P	*Our result*
Weighted ODCA	P	*Our result*
Deterministic-stack PDA	*open*	-
pPDA	*open*	[3]

References I

[1] Stanislav Böhm and Stefan Göller.
Language equivalence of deterministic real-time one-counter automata is nl-complete.
In Filip Murlak and Piotr Sankowski, editors, *MFCS*, volume 6907 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 2011.

[2] Stanislav Böhm, Stefan Göller, and Petr Jancar.
Equivalence of deterministic one-counter automata is nl-complete.
In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 131–140. ACM, 2013.

[3] Vojtech Forejt, Petr Jancar, Stefan Kiefer, and James Worrell 0001.
Language equivalence of probabilistic pushdown automata.
Inf. Comput, 237:1–11, 2014.

[4] John E. Hopcroft and Jeffrey D. Ullman.
Introduction to Automata Theory, Languages and Computation.
Addison Wesley, Reading, Mass., 1979.

References II

[5] Géraud Sénizergues.
Decidability of the equivalence problem for deterministic pushdown automata.
In *Proceedings of INFINITY'97*, 1997.

[6] Wen-Guey Tzeng.
A polynomial-time algorithm for the equivalence of probabilistic automata.
SIAM J. Comput, 21(2):216–227, 1992.

[7] Valiant and Paterson.
Deterministic one-counter automata.
JCSS: Journal of Computer and System Sciences, 10, 1975.

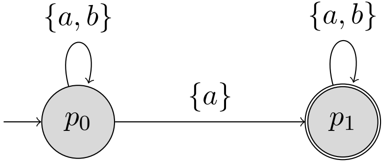
[8] https://latex-beamer.com/beamer-template/

Thank You!

Appendix

Non-deterministic Finite Automata

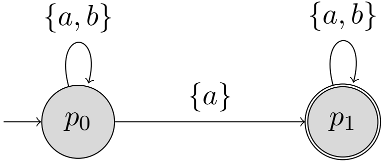
- Consider the following NFA.



- Language recognised?

Non-deterministic Finite Automata

- Consider the following NFA.



- Language recognised?
  - Set of all words containing at least one ‘a’.

Equivalence WA(1)

- Given two weighted automata $\mathcal{A}$ and $\mathcal{B}$, the equivalence problem asks whether for all $w \in \Sigma^*$, $f_{\mathcal{A}}(w) = f_{\mathcal{B}}(w)$.

Theorem- Equivalence WA

The equivalence problem for weighted automata over a field is decidable in polynomial time.

- Given two weighted automata $\mathcal{A}$ and $\mathcal{B}$ over a field, we can construct a weighted automata $\mathcal{C}$ recognising the function $f_{\mathcal{A}} - f_{\mathcal{B}}$.
- To check equivalence of $\mathcal{A}$ and $\mathcal{B}$, it suffices to check the existence of a word which is accepted with non-zero weight by $\mathcal{C}$.

Equivalence(2)

- Let K be the number of states of $\mathcal{C}$ and η its final distribution.
- $\mathcal{V} = \{\mathbf{x} \in \mathcal{F}^{\mathsf{K}} \mid \mathbf{x}\eta = 0\}$ is a vector space.
- Given a vector space $\mathcal{U} \subseteq \mathcal{F}^{\mathsf{K}}$, we define $\overline{\mathcal{U}} = \mathcal{F}^{\mathsf{K}} \setminus \mathcal{U}$.
- The equivalence problem is to check the existence of a word w such that $\lambda\delta(w) \in \overline{\mathcal{V}}$.

co-VS reachability of weighted automata

co-VS reachability problem of WA

Input: Vector space $\mathcal{V}$, Weighted automaton $\mathcal{C}$.
Output: Yes, If there is a run of $\mathcal{C}$ that reaches a configuration in $\overline{\mathcal{V}}$.
No, otherwise.

Theorem- co-VS reachability

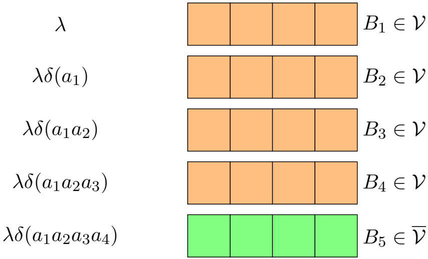
The co-VS reachability problem for weighted automata is decidable in polynomial time.

Lemma- minimal word

The length of the minimal run, if it exists, that reaches a configuration in $\overline{\mathcal{V}}$ is less than the number of states of $\mathcal{C}$.

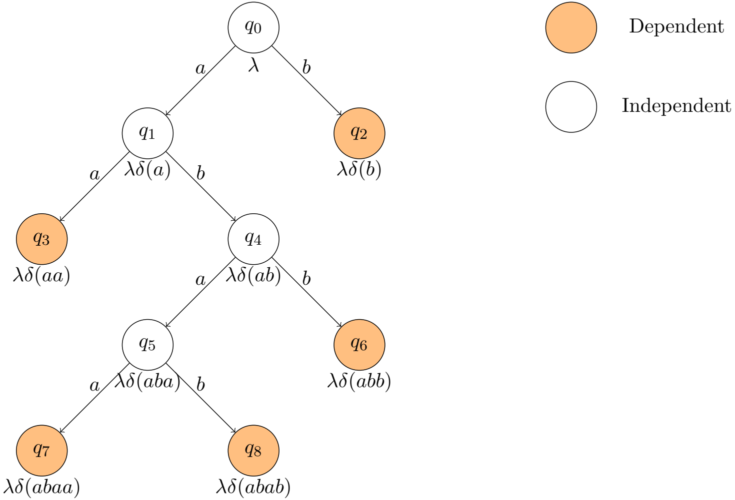
Minimal word

- Let the number of states of $\mathcal{C}$ be 4.
- Assume $w = a_1 a_2 a_3 a_4$ is a minimal reachability witness.



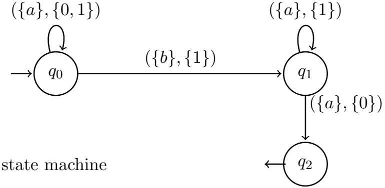
- $B_5 = c_1 B_1 + c_2 B_2 + c_3 B_3 + c_4 B_4$ for some $c_1, c_2, c_3, c_4 \in \mathcal{F}$.
- There exists $i < 5$ such that $B_i \in \overline{\mathcal{V}}$. This is a contradiction.

Polynomial time algorithm



- Starting from q_0 construct a basis set by traversing the tree in a breadth-first manner and adding independent vectors to it.

Example (a)



Finite state machine

Counter structure

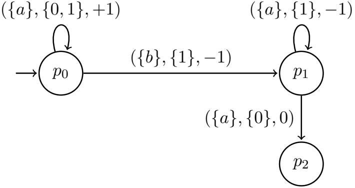
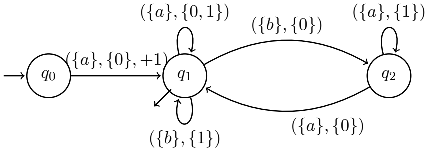


Figure: $\mathcal{L}_1 = \{a^n b a^n \mid n > 0\}$

Example (b)



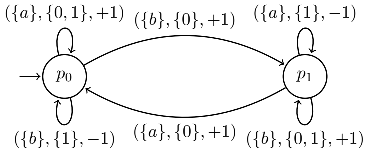


Figure: $\mathcal{L}_2 = \{(a+b)^* \mid \#\text{a's} > \#\text{b's}\}$

Example (c)

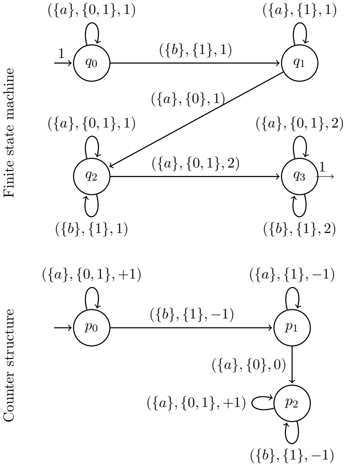


Figure: $f(w_1w_2)$ = decimal value of w_2's binary interpretation, if $w_1 \in \mathcal{L}_1$ and $\#a$'s $\geq \#b$'s for any prefix of w_2; 0 otherwise.