FSTTCS 2023

Weighted One-Deterministic-Counter Automata

https://doi.org/10.4230/LIPIcs.FSTTCS.2023.39

https://arxiv.org/abs/2301.13456

Prince Mathew, *prince@iitgoa.ac.in*

Indian Institute of Technology Goa, India

Vincent Penelle, *vincent.penelle@u-bordeaux.fr*

Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400, Talence, France

Prakash Saivasan, *prakashs@imsc.res.in*

The Institute of Mathematical Sciences, HBNI, India; CNRS UMI ReLaX, India

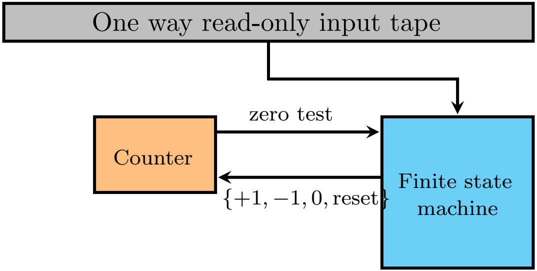
A.V. Sreejith, *sreejithav@iitgoa.ac.in*

Indian Institute of Technology Goa, India

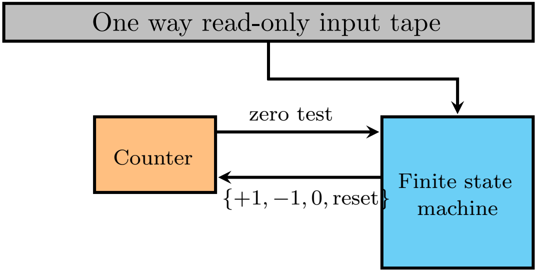
Outline of the talk

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

One-counter automata (OCA)

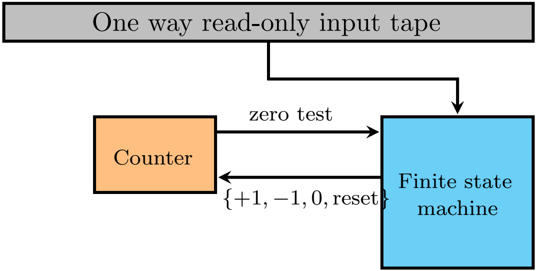


One-counter automata (OCA)



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 (+)

Results - 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

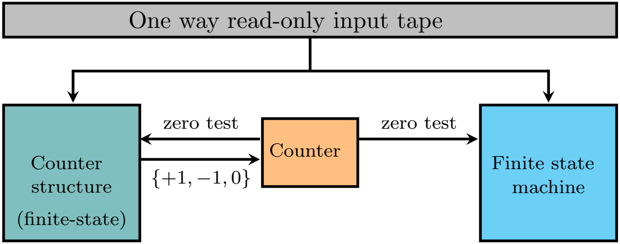


Figure: One-deterministic-counter automata

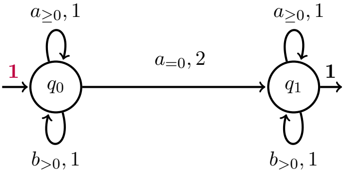
Example - Weighted ODCA (visibly)

Example

$$\texttt{eqPrefix}(w) = \begin{cases} \mathbf{2} \cdot \mathbf{k}, \text{where } k \text{ is the \#proper prefixes of } w \text{ with} \\ \qquad \#a\text{'s} = \ \#b\text{'s, if } \#a\text{'s} \geq \#b\text{'s for all prefixes} \\ \mathbf{0}, \ \text{otherwise} \end{cases}$$

.

- ODCAs are strict extensions of visibly OCA.



Finite state machine

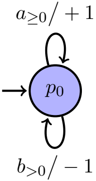
a	a	b	b	a	b	a

Input tape

0

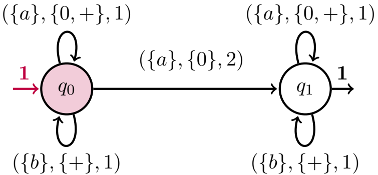
Counter

Counter structure



1	0

Initial vector



Finite state machine

a	a	b	b	a	b	a

Input tape

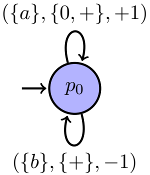
0

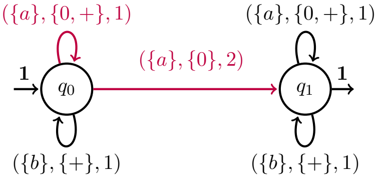
Counter

1	0

Initial vector

Counter structure





Finite state machine

a	a	b	b	a	b	a

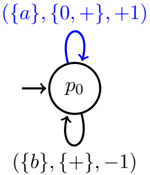
Input tape

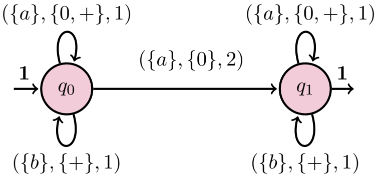
0

Counter

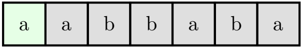
1	0

Counter structure



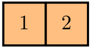


Finite state machine

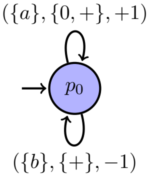


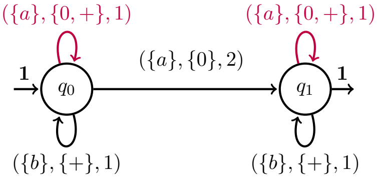
Input tape

Counter



Counter structure





Finite state machine

a	a	b	b	a	b	a

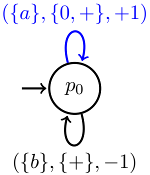
Input tape

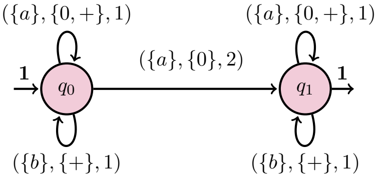
1

Counter

1	2

Counter structure





Finite state machine

a	a	b	b	a	b	a

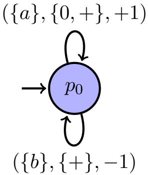
Input tape

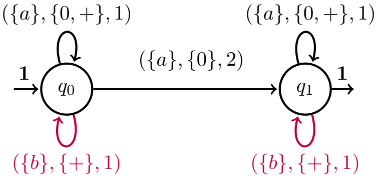
2

Counter

1	2

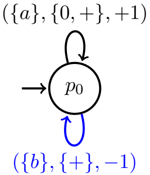
Counter structure





Finite state machine

Counter structure



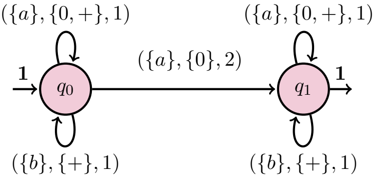
a	a	b	b	a	b	a

Input tape

2

Counter

1	2



Finite state machine

a	a	b	b	a	b	a

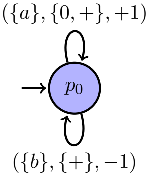
Input tape

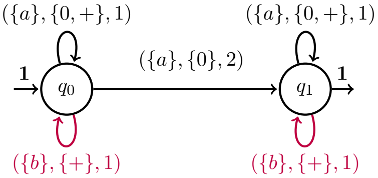
1

Counter

1	2

Counter structure





Finite state machine

a	a	b	b	a	b	a

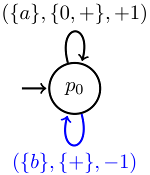
Input tape

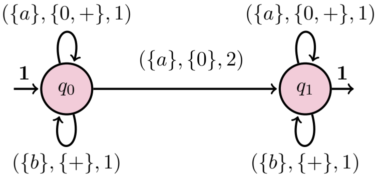
1

Counter

1	2

Counter structure





Finite state machine

a	a	b	b	a	b	a

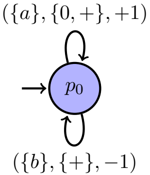
Input tape

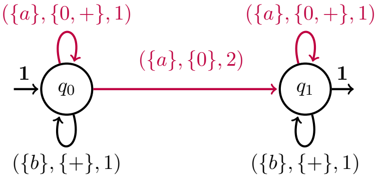
0

Counter

1	2

Counter structure





Finite state machine

a	a	b	b	a	b	a

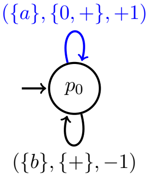
Input tape

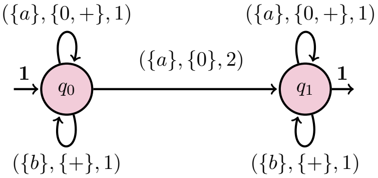
0

Counter

1	2

Counter structure





Finite state machine

a	a	b	b	a	b	a

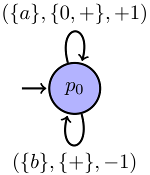
Input tape

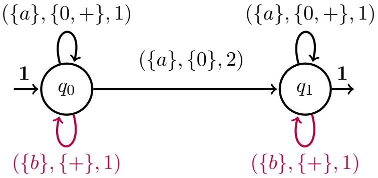
1

Counter

1	4

Counter structure





Finite state machine

a	a	b	b	a	b	a

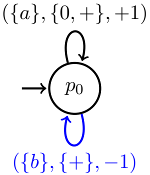
Input tape

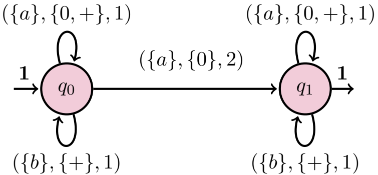
1

Counter

1	4

Counter structure





Finite state machine

a	a	b	b	a	b	a

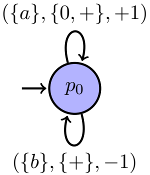
Input tape

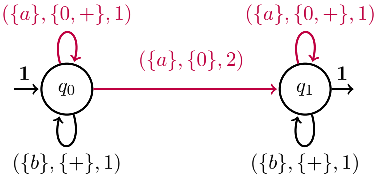
0

Counter

1	4

Counter structure





Finite state machine

a	a	b	b	a	b	a

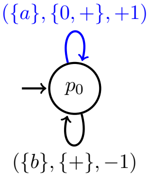
Input tape

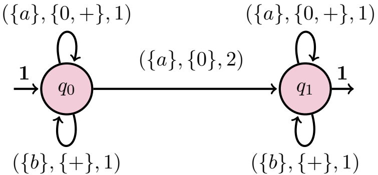
0

Counter

1	4

Counter structure





Finite state machine

a	a	b	b	a	b	a

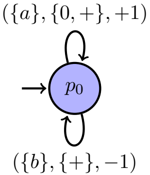
Input tape

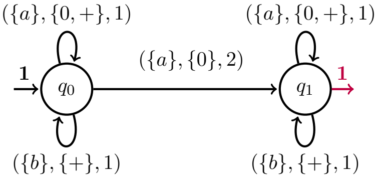
1

Counter

1	6

Counter structure





Finite state machine

a	a	b	b	a	b	a

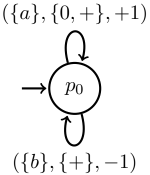
Input tape

1

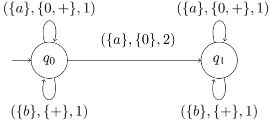
Counter

1	6

Counter structure



Weighted visibly ODCA- Evaluation



Initial distribution

λ

1	0

Transition matrices

$\delta_0(a)$

1	2
0	1

$\delta_+(a)$

1	0
0	1

$\delta_+(b)$

1	0
0	1

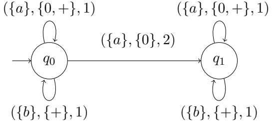
Final distribution

ρ

0
1

- Let $w = aababba$.

Weighted visibly ODCA- Evaluation



Initial distribution | Transition matrices | Final distribution

λ

1	0

$\delta_0(a)$

1	2
0	1

$\delta_+(a)$

1	0
0	1

$\delta_+(b)$

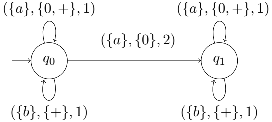
1	0
0	1

ρ

0
1

- Let $w = aababba$.
- $\delta(w) = \delta_0(a)\ \delta_+(a)\ \delta_+(b)\ \delta_+(a)\ \delta_+(b)\ \delta_+(b)\ \delta_0(a)$.

Weighted visibly ODCA- Evaluation



Initial distribution

λ

1	0

Transition matrices

$\delta_0(a)$

1	2
0	1

$\delta_+(a)$

1	0
0	1

$\delta_+(b)$

1	0
0	1

Final distribution

ρ

0
1

- Let $w = aababba$.
- $\delta(w) = \delta_0(a)\ \delta_+(a)\ \delta_+(b)\ \delta_+(a)\ \delta_+(b)\ \delta_+(b)\ \delta_0(a)$.
- Accepting weight of $w = \lambda\delta(w)\rho$.

Motivation

- Decidability of equivalence of pPDA is an open problem [3].
- Equivalence of probabilistic OCA is also not known.
- Probabilistic ODCA is a class of probabilistic OCA for which equivalence is decidable.
- This is a strict super class of visibly probabilistic OCA.

Our equivalence 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 equivalence 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 equivalence 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.

Reachability problem

Reachability

co-VS Reachability problem

INPUT:

- A weighted visibly OCA $\mathcal{A}$ over a field,
- an initial configuration $\mathtt{c}$,
- 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 $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times \{m\}$.

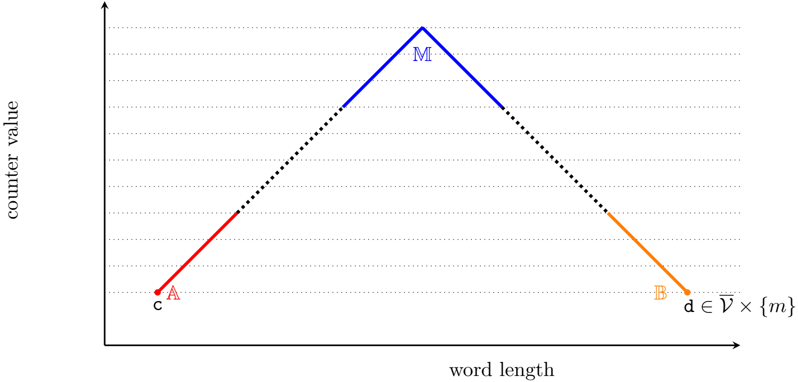
co-VS reachability

Theorem - co-VS reachability

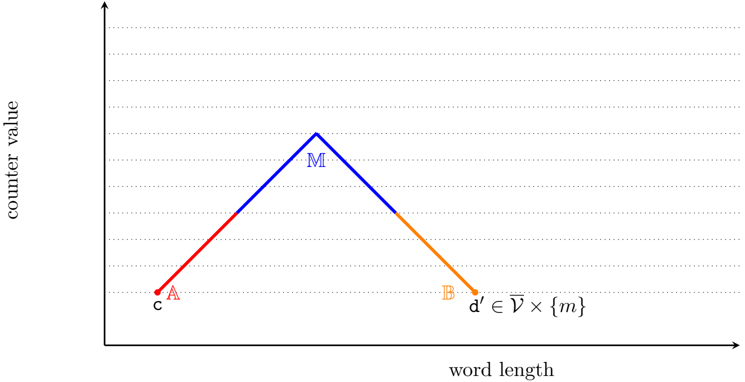
co-VS reachability is decidable in polynomial time.

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

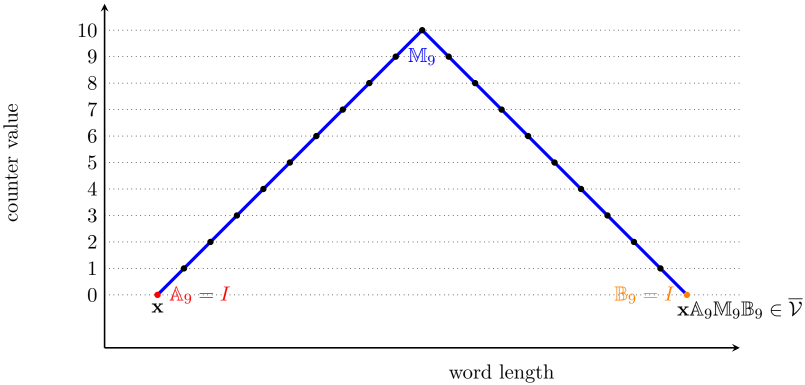
Pseudo-pumping lemma (pumping down)



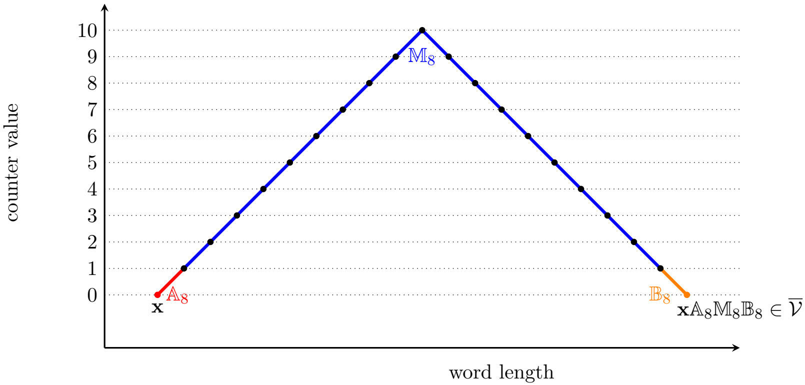
Pseudo-pumping lemma (pumping down)



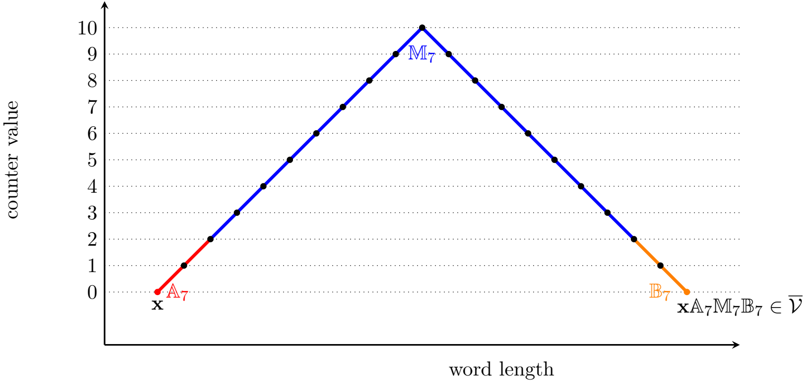
Pseudo-pumping lemma



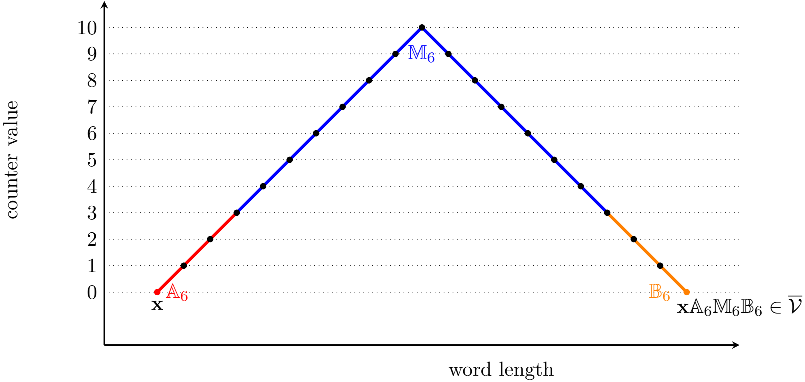
Pseudo-pumping lemma



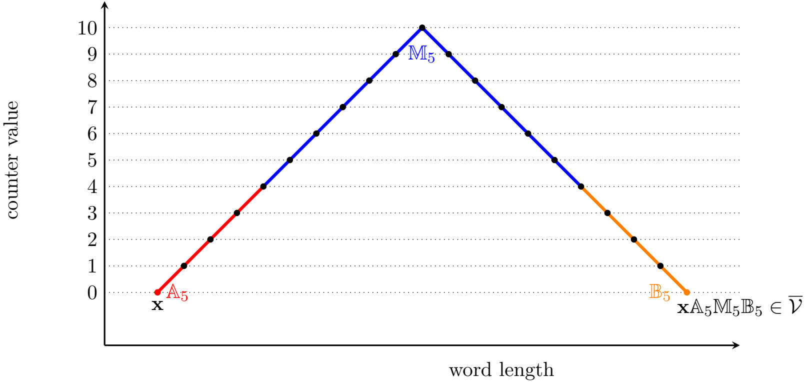
Pseudo-pumping lemma



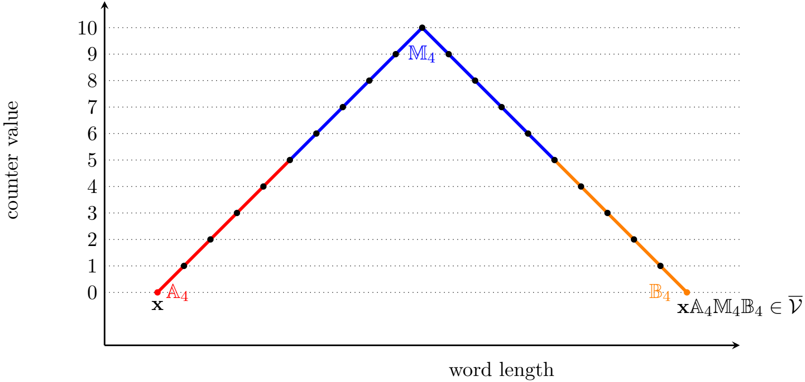
Pseudo-pumping lemma



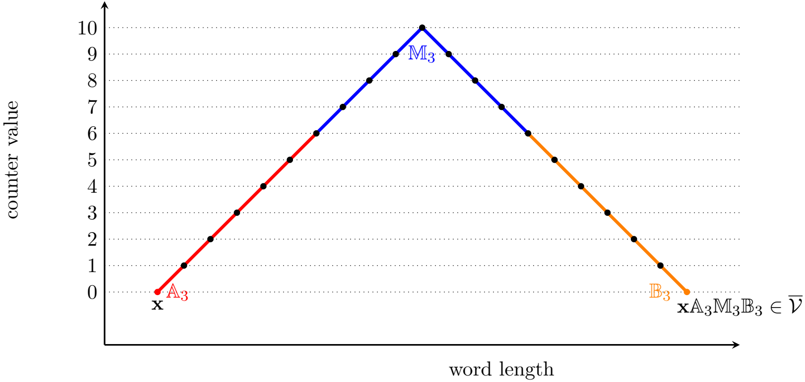
Pseudo-pumping lemma



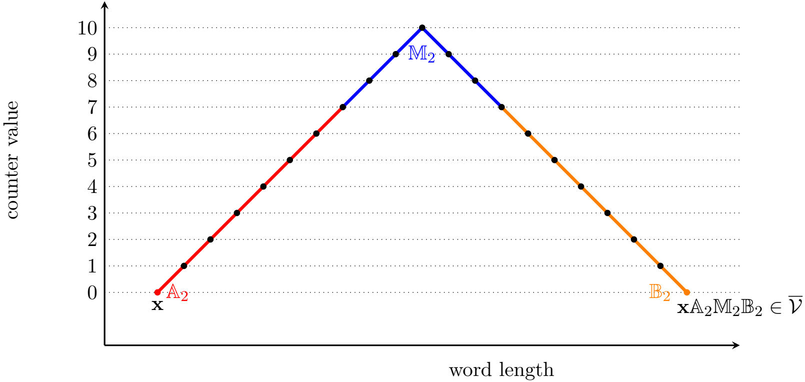
Pseudo-pumping lemma



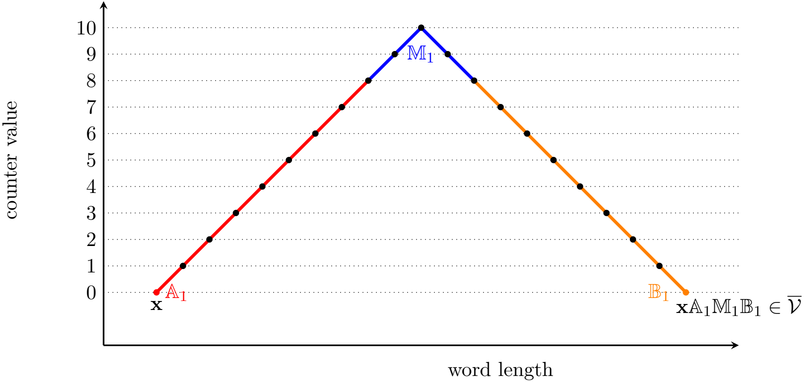
Pseudo-pumping lemma



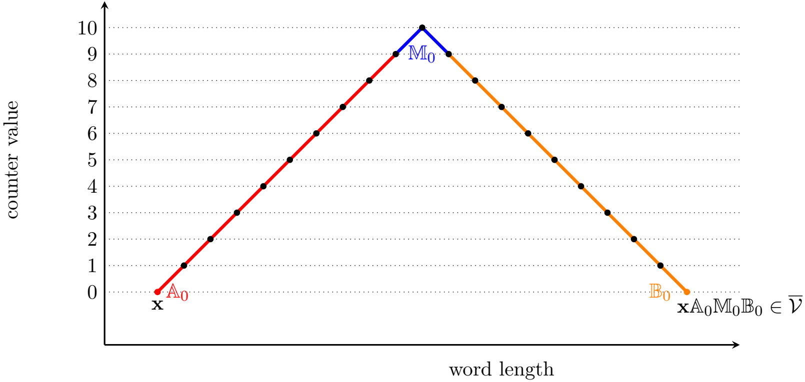
Pseudo-pumping lemma



Pseudo-pumping lemma

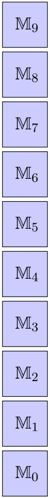


Pseudo-pumping lemma



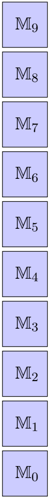
Pseudo-pumping lemma

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



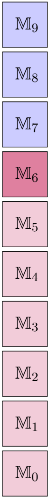
Pseudo-pumping lemma

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



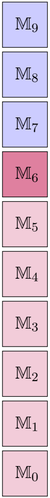
Pseudo-pumping lemma

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



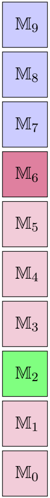
Pseudo-pumping lemma

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

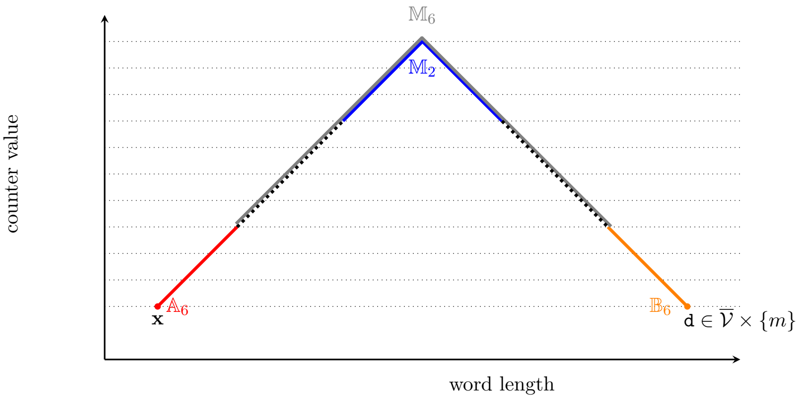


Pseudo-pumping lemma

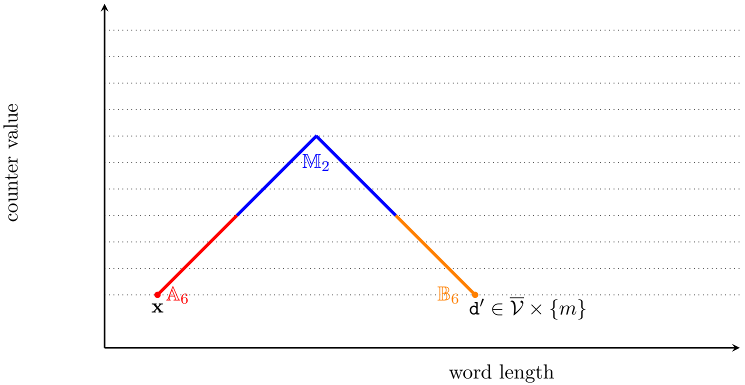
- Assume for all $i \in [0,9]$, $\mathbb{M}_i$ is a 3×3 matrix.
- For all $i \in [0,9]$, $\mathbf{x}\mathbb{A}_i\mathbb{M}_i\mathbb{B}_i \in \overline{\mathcal{V}}$.
- $\exists j \in [0,9]$, $\mathbb{M}_j = \alpha_0\mathbb{M}_0 + \alpha_1\mathbb{M}_1 + \cdots + \alpha_{j-1}\mathbb{M}_{j-1}$.
- $\mathbf{x}\mathbb{A}_j(\alpha_0\mathbb{M}_0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



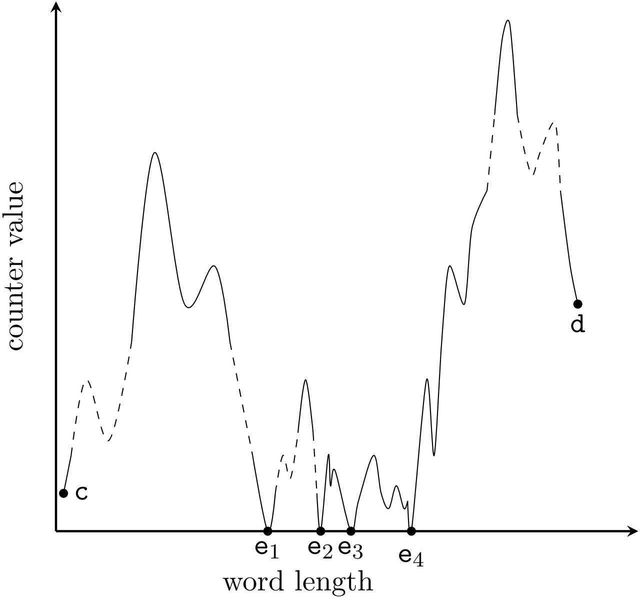
Pseudo-pumping lemma



Pseudo-pumping lemma



Multiple cuts



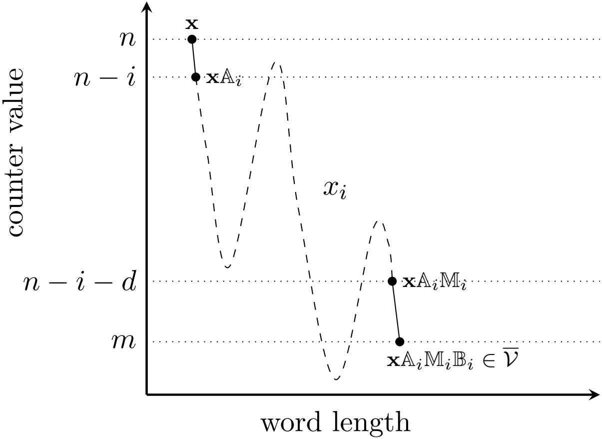
Special word Lemma

Lemma- Special word

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

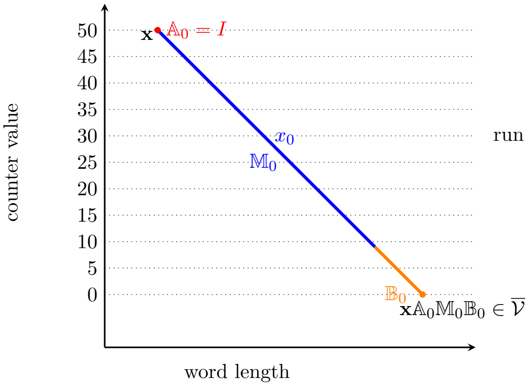
1. $z = uy_1^{r_1}w_1y_2^{r_2}w_2$ such that $|uy_1w_1y_2w_2|$ is polynomially bounded in input size, and
2. r_1, r_2 are polynomially bounded in input size.

Special word- Claim 1



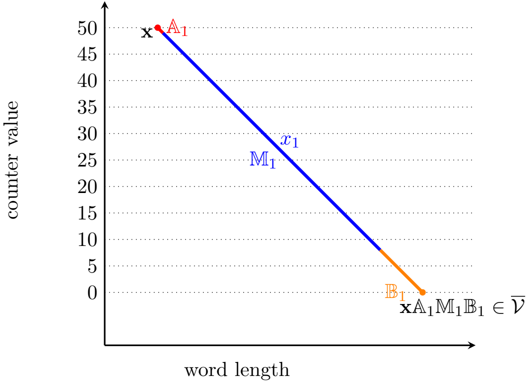
- We show that a factor repeats in the lexicographically minimal word if d is large enough.

Repeating subword

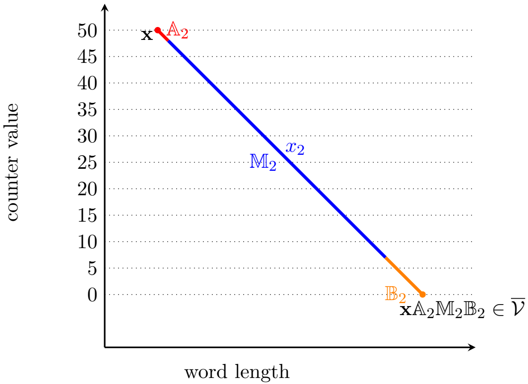


run of the lexicographically minimal witness z

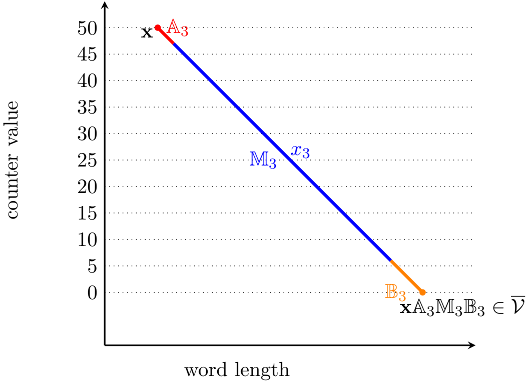
Repeating subword



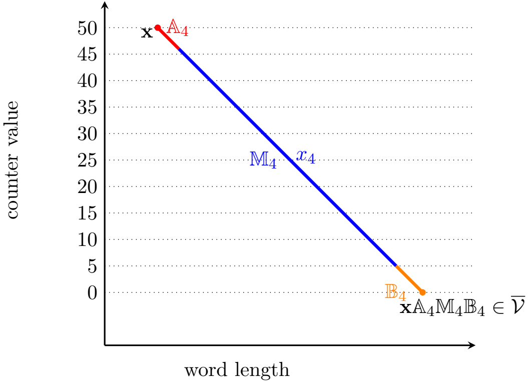
Repeating subword



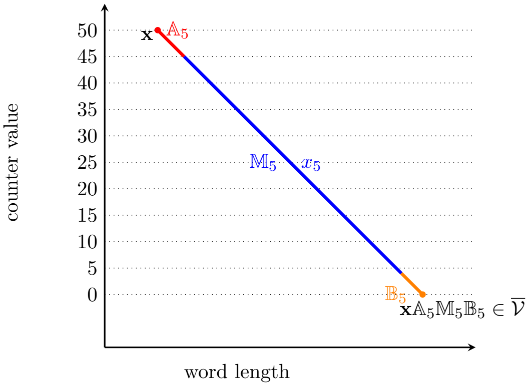
Repeating subword



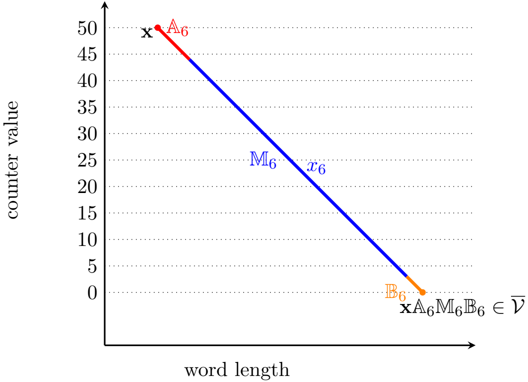
Repeating subword



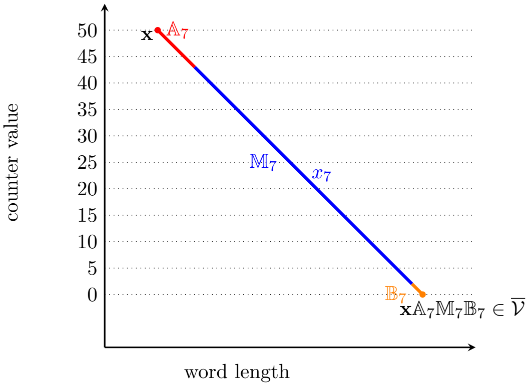
Repeating subword



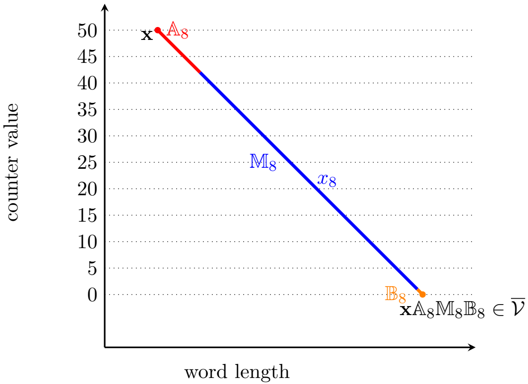
Repeating subword



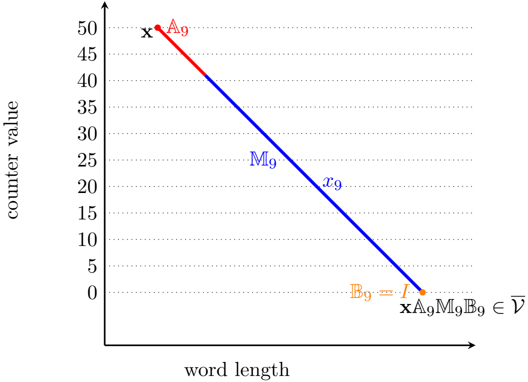
Repeating subword



Repeating subword

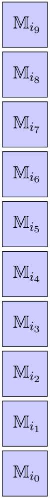


Repeating subword



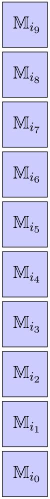
Special word- Claim 1

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



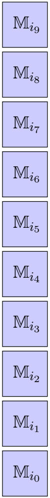
Special word- Claim 1

- 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 length lex. ordering.



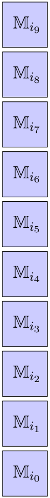
Special word- Claim 1

- 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 length lex. ordering.
- Let $x_{i_0} < x_{i_1} < \ldots < x_{i_9}$.



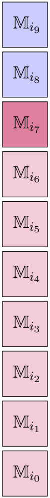
Special word- Claim 1

- 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 length lex. ordering.
- Let $x_{i_0} < x_{i_1} < \ldots < x_{i_9}$.
- For all $j \in [0, 9]$, $\mathbf{x}\mathbb{A}_{i_j}\mathbb{M}_{i_j}\mathbb{B}_{i_j} \in \overline{\mathcal{V}}$.



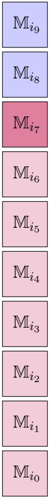
Special word- Claim 1

- 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 length lex. 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 $j \in [0,9]$, $\mathbb{M}_{i_j} = \alpha_0\mathbb{M}_{i_0} + \alpha_1\mathbb{M}_{i_1} + \cdots + \alpha_{j-1}\mathbb{M}_{i_{j-1}}$.



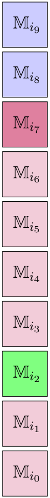
Special word- Claim 1

- 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 length lex. 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 $j \in [0,9]$, $\mathbb{M}_{i_j} = \alpha_0\mathbb{M}_{i_0} + \alpha_1\mathbb{M}_{i_1} + \cdots + \alpha_{j-1}\mathbb{M}_{i_{j-1}}$.
- $\mathbf{x}\mathbb{A}_{i_j}(\alpha_0\mathbb{M}_{i_0} + \alpha_1\mathbb{M}_{i_1} + \cdots + \alpha_{j-1}\mathbb{M}_{i_{j-1}})\mathbb{B}_{i_j} \in \overline{\mathcal{V}}$.

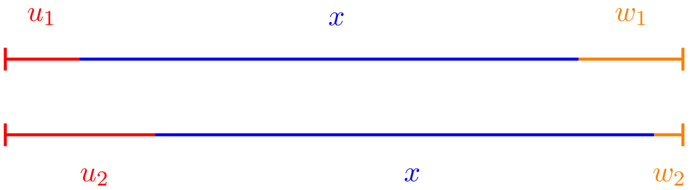


Special word- Claim 1

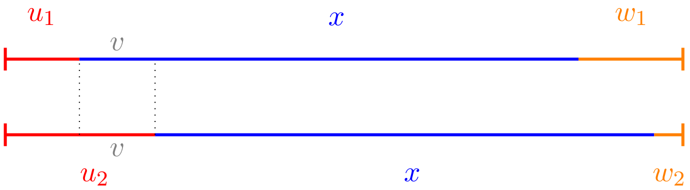
- Assume for all $i \in [0,9]$, $\mathbb{M}_i$ is a 3×3 matrix.
- Arrange the words $x_0, x_1, \ldots, x_9$ in the length lex. ordering.
- Let $x_{i_0} < x_{i_1} < \ldots < x_{i_9}$.
- For all $j \in [0,9]$, $\mathbf{x}\mathbb{A}_{i_j}\mathbb{M}_{i_j}\mathbb{B}_{i_j} \in \overline{\mathcal{V}}$.
- There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



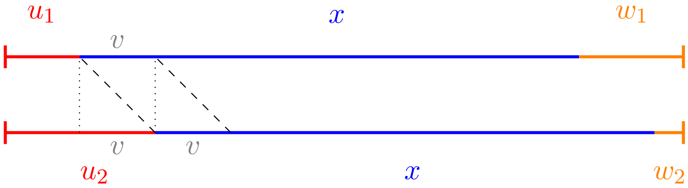
Special word- Claim 2



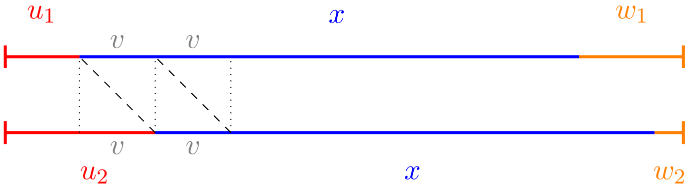
Special word- Claim 2



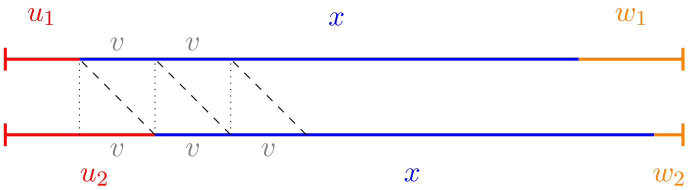
Special word- Claim 2



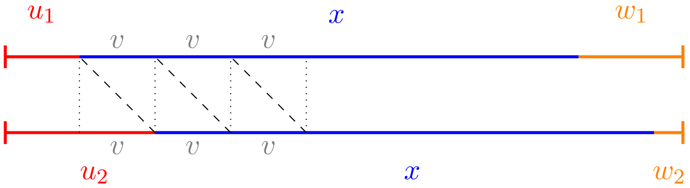
Special word- Claim 2



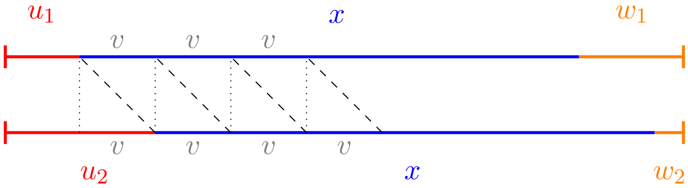
Special word- Claim 2



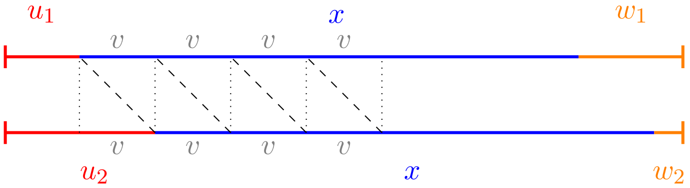
Special word- Claim 2



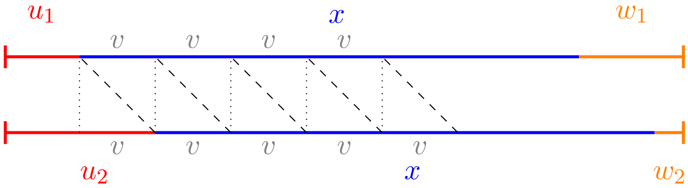
Special word- Claim 2



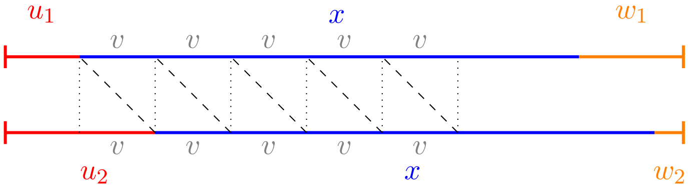
Special word- Claim 2



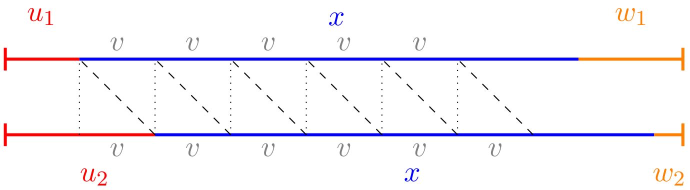
Special word- Claim 2



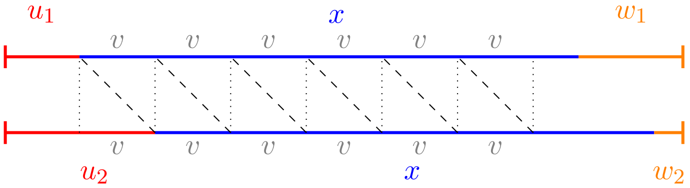
Special word- Claim 2



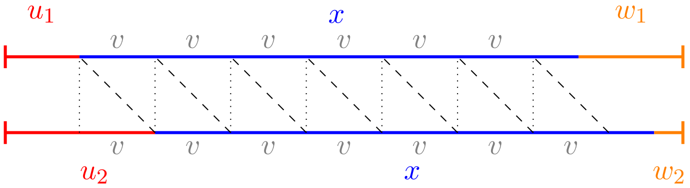
Special word- Claim 2



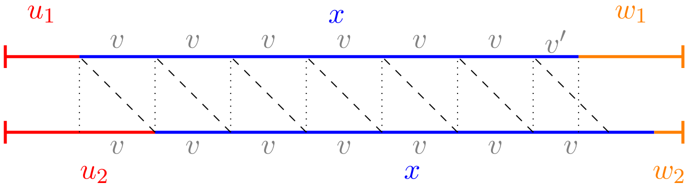
Special word- Claim 2



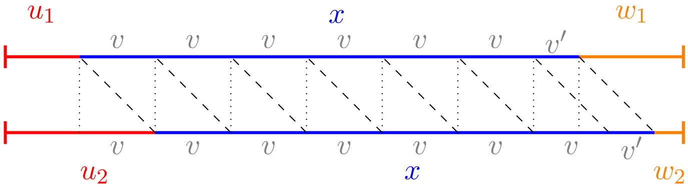
Special word- Claim 2



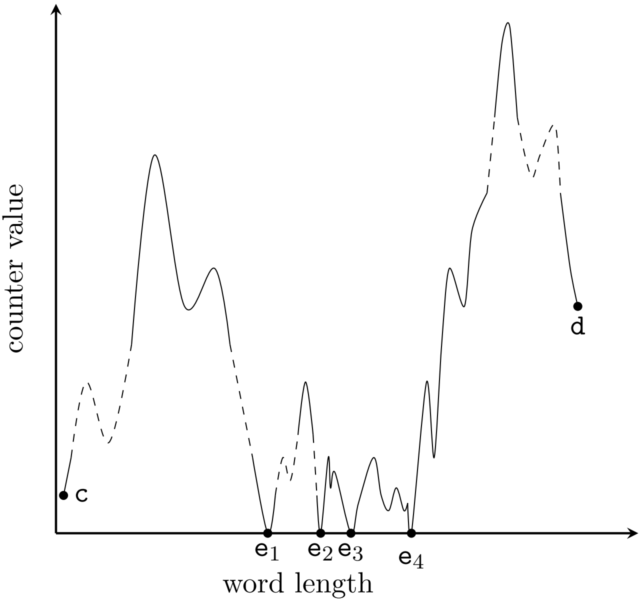
Special word- Claim 2



Special word- Claim 2



Multiple cuts



Equivalence

Equivalence

Lemma - Witness bound

If two weighted ODCAs $\mathcal{A}_1$ 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 ODCAs $\mathcal{A}_1$ 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

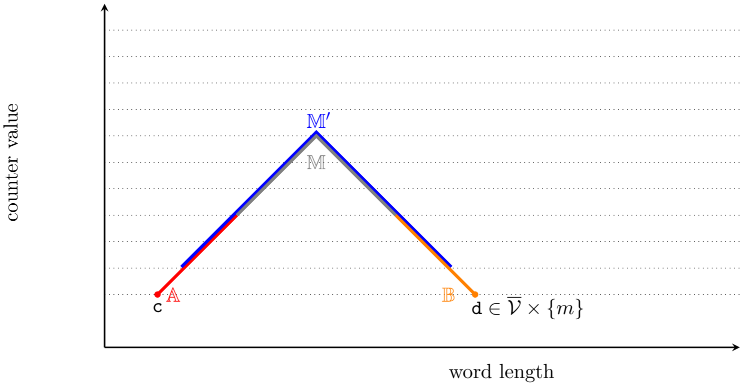
There is a polynomial time algorithm to check the equivalence of two weighted ODCAs (weights from a field).

Other results

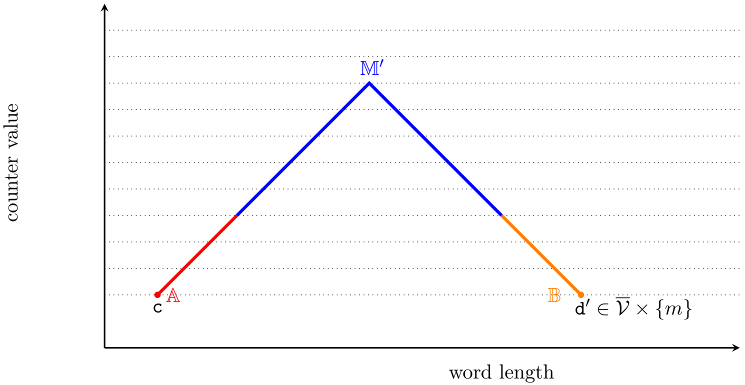
Theorem - Regularity

Given a weighted ODCA (weights from a field), determining whether there exists a weighted automata recognising the same function is in P.

Regularity - pumping up



Regularity - pumping up



Summary

- 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	[7]
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

Stanislav Böhm and Stefan Göller.
Language equivalence of deterministic real-time one-counter automata is NL-Complete.
In *MFCS*, Lecture Notes in Computer Science, pages 194–205. Springer, 2011.

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.

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

John E. Hopcroft and Jeffrey D. Ullman.
Introduction to Automata Theory, Languages and Computation.
Addison Wesley, 1979.

References II

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

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

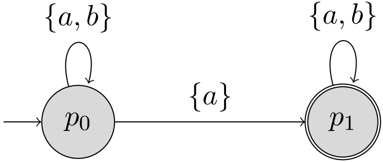
Valiant and Paterson.
Deterministic one-counter automata.
JCSS: Journal of Computer and System Sciences, 10, 1975.

Thank You!

Appendix

Non-deterministic Finite Automata

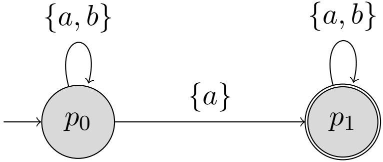
- Consider the following NFA.



- Language recognised?

Non-deterministic Finite Automata

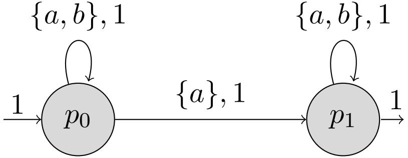
- Consider the following NFA.



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

Weighted Automata

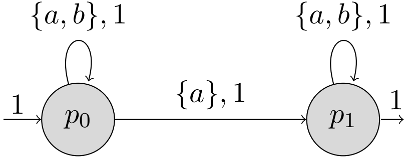
- A weighted automata (WA) can be viewed as an NFA with weights.
- In this talk, we assume that the weights always comes from a field.
- Consider the following WA.



- Function recognised?

Weighted Automata

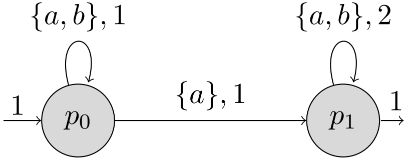
- A weighted automata (WA) can be viewed as an NFA with weights.
- In this talk, we assume that the weights always comes from a field.
- Consider the following WA.



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

Another example

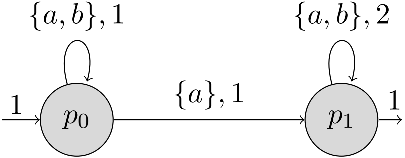
- Consider the following WA.



- Function recognised?

Another example

- Consider the following WA.



- Function recognised?
    - Decimal equivalent of binary number.

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

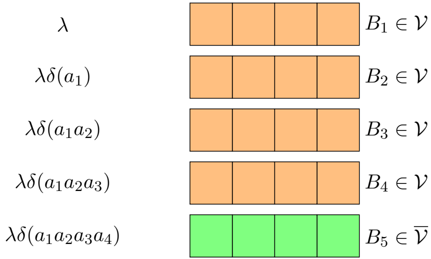
The co-VS reachability problem for weighted automata is decidable in polynomial time.

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 $\mathcal{C}$.

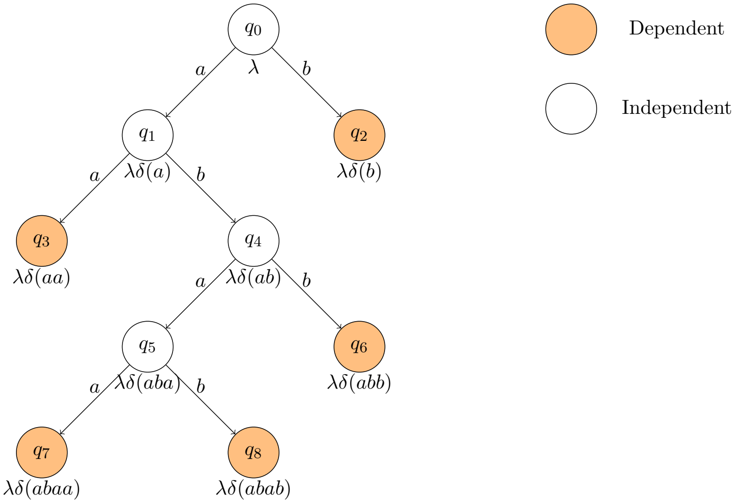
Minimal word

- Let the number of states of $\mathcal{C}$ be 4.
- Assume $w = a_1a_2a_3a_4$ is a minimal reachability witness.



- $B_5 = c_1B_1 + c_2B_2 + c_3B_3 + c_4B_4$ 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 ODCA (a)

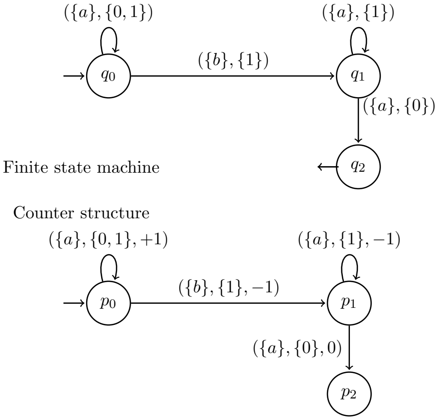
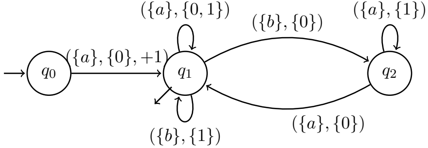


Figure: $\mathcal{L}_1 = \{a^n b a^n \mid n > 0\}$

Example ODCA (b)



Finite state machine

Counter structure

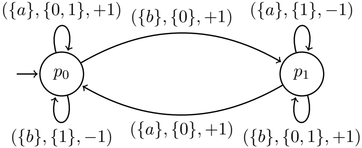


Figure: $\mathcal{L}_2 = \{(a+b)^* \mid \#\text{a's} > \#\text{b's}\}$

Example ODCA (c)

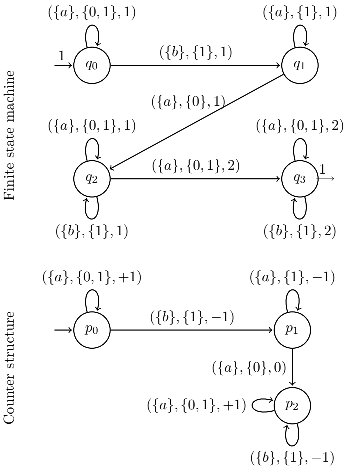
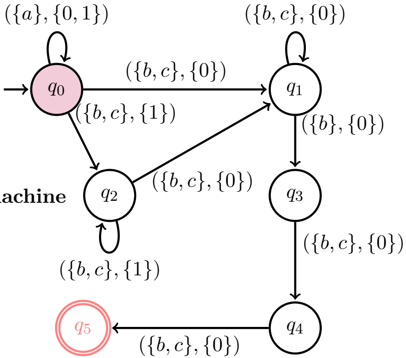


Figure: $f(w_1w_2)$ = decimal value of w_2's binary interpretation, if $w_1 \in \mathcal{L}_1$ and $\#a$'s $\geq \#b$'s for any prefix of w_2; 0 otherwise.

Example - One deterministic-counter Automata

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

Finite state machine



a	a	b	c	b	b	c	b

Input tape

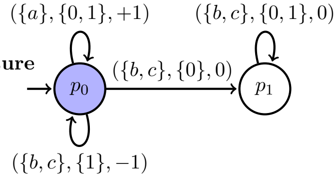
0

Counter

1	0	0	0	0	0

Initial vector

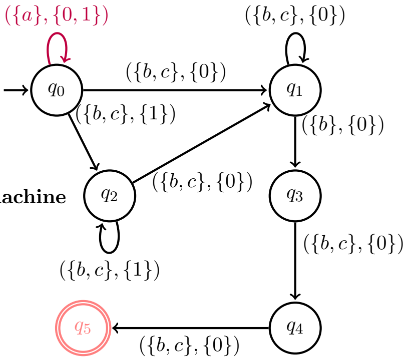
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

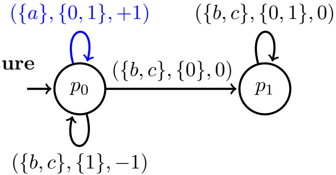


0

Counter

1	0	0	0	0	0

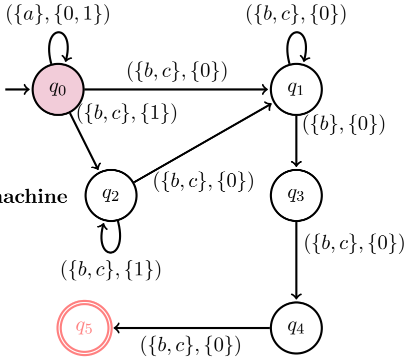
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

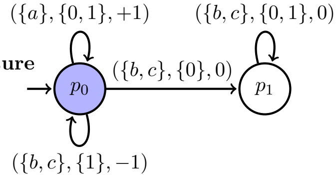


1

Counter

1	0	0	0	0	0

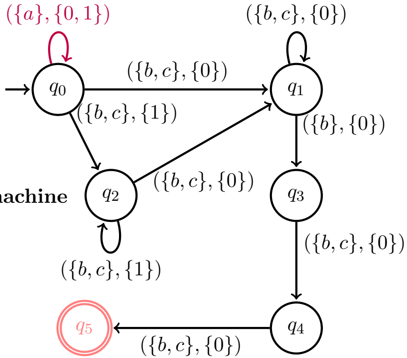
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

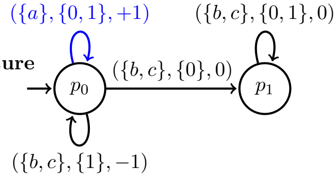


1

Counter

1	0	0	0	0	0

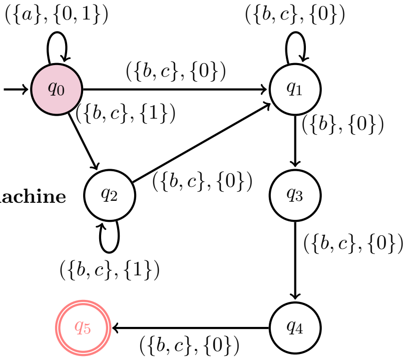
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

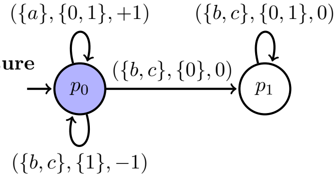


2

Counter

1	0	0	0	0	0

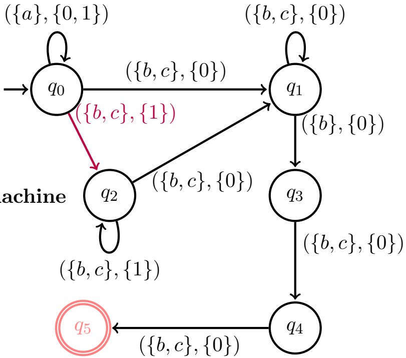
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

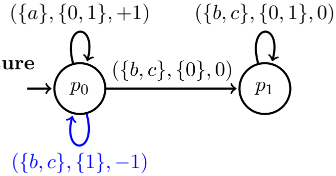


2

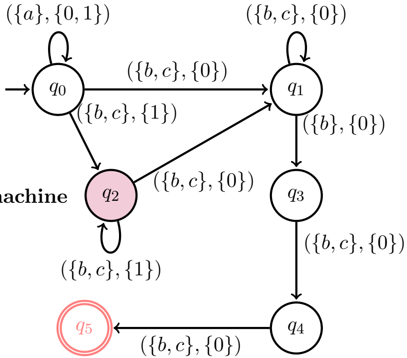
Counter

1	0	0	0	0	0

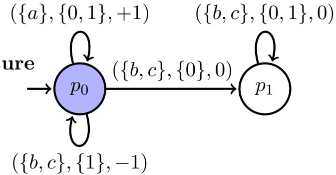
Counter structure



Finite state machine



Counter structure



a	a	b	c	b	b	c	b

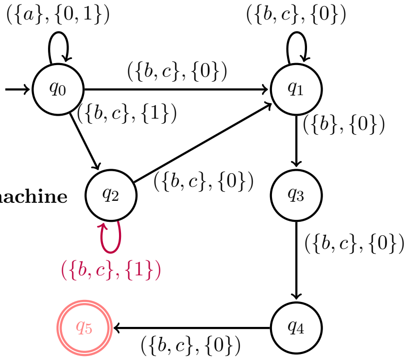
Input tape

1

Counter

0	0	1	0	0	0

Finite state machine



a	a	b	c	b	b	c	b

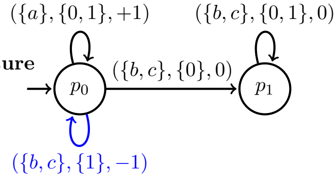
Input tape

1

Counter

0	0	1	0	0	0

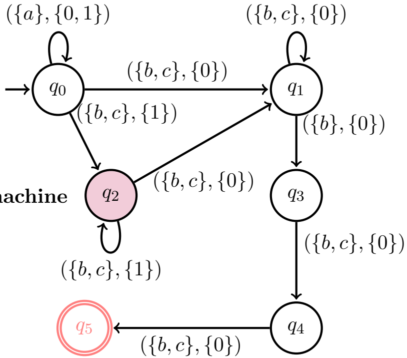
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

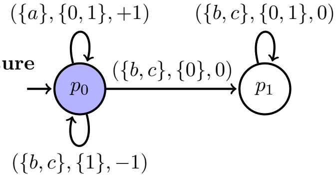


0

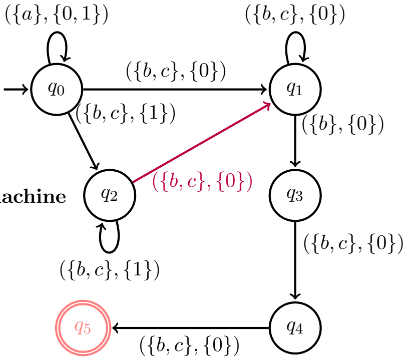
Counter

0	0	1	0	0	0

Counter structure



Finite state machine



a	a	b	c	b	b	c	b

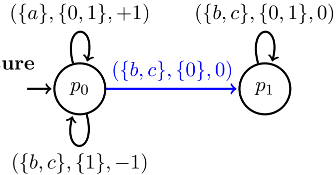
Input tape

0

Counter

0	0	1	0	0	0

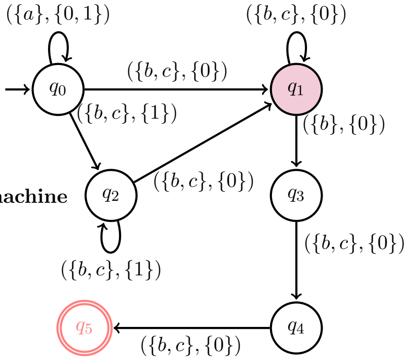
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

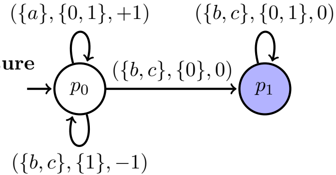


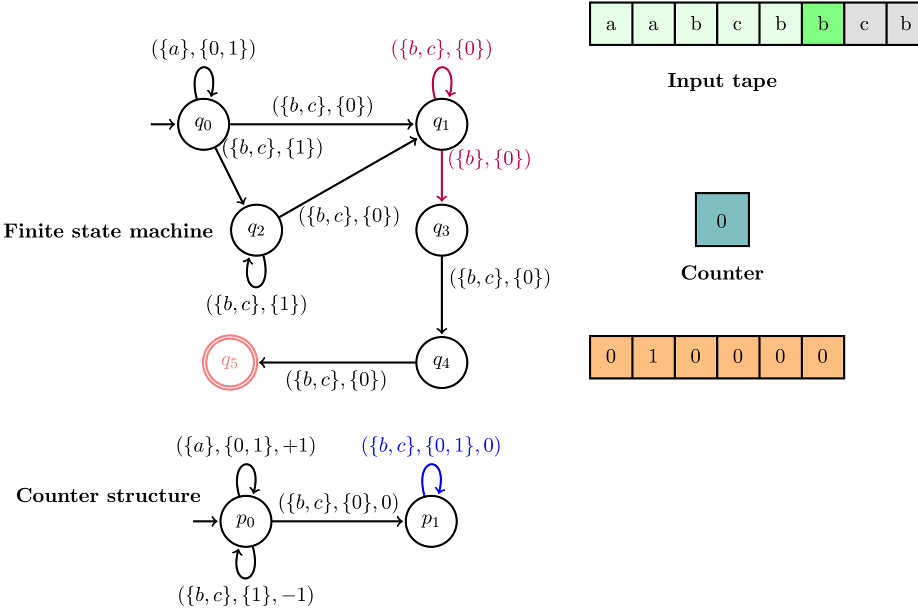
0

Counter

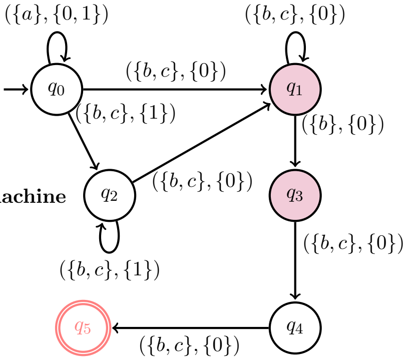
0	1	0	0	0	0

Counter structure




a a b c b b c b
Input tape
({a}, {0, 1})
({b, c}, {0})
q0
q1
({b, c}, {1})
({b}, {0})
Finite state machine
q2
({b, c}, {0})
q3
({b, c}, {0})
({b, c}, {1})
q5
q4
({b, c}, {0})
0
Counter
0 1 0 0 0 0
({a}, {0, 1}, +1)
({b, c}, {0, 1}, 0)
Counter structure
p0
({b, c}, {0}, 0)
p1
({b, c}, {1}, −1)

Finite state machine



a	a	b	c	b	b	c	b

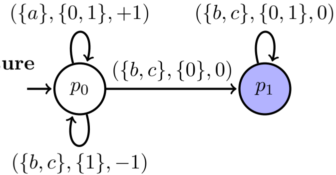
Input tape

0

Counter

0	1	0	1	0	0

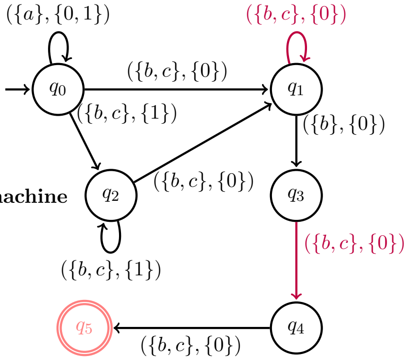
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

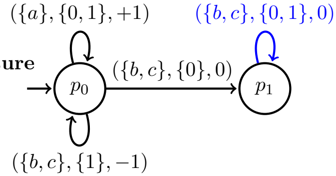


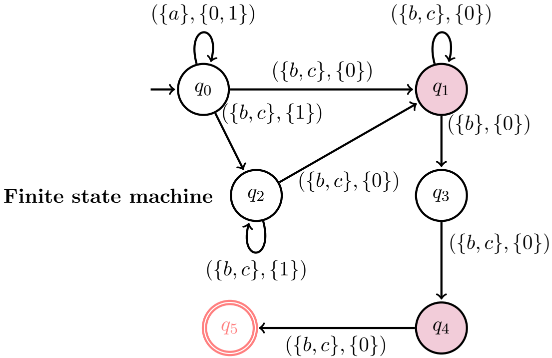
0

Counter

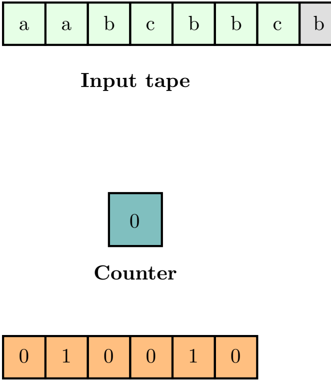
0	1	0	1	0	0

Counter structure

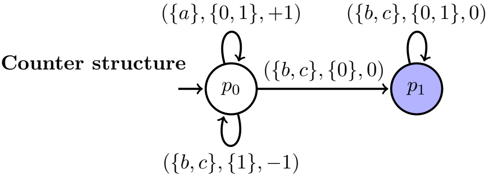




Finite state machine
q_0
q_1
q_2
q_3
q_4
q_5
$(\{a\},\{0,1\})$
$(\{b,c\},\{0\})$
$(\{b,c\},\{0\})$
$(\{b,c\},\{1\})$
$(\{b\},\{0\})$
$(\{b,c\},\{0\})$
$(\{b,c\},\{1\})$
$(\{b,c\},\{0\})$
$(\{b,c\},\{0\})$

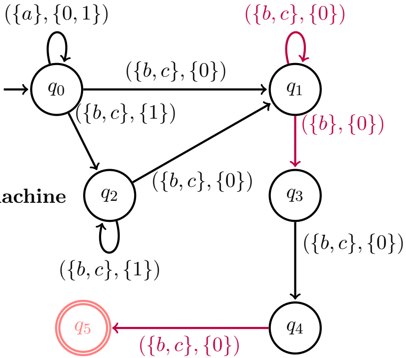


a a b c b b c b
Input tape
0
Counter
0 1 0 0 1 0



Counter structure
p_0
p_1
$(\{a\},\{0,1\},+1)$
$(\{b,c\},\{0,1\},0)$
$(\{b,c\},\{0\},0)$
$(\{b,c\},\{1\},-1)$

Finite state machine



a	a	b	c	b	b	c	b

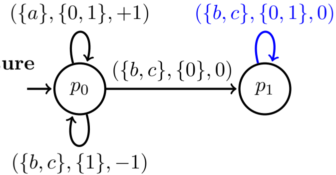
Input tape

0

Counter

0	1	0	0	1	0

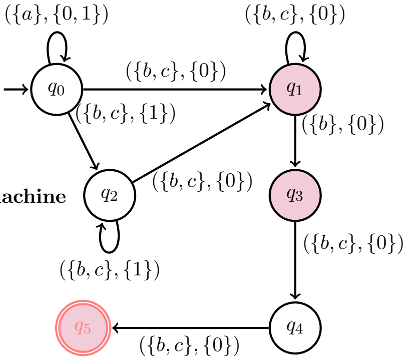
Counter structure



a	a	b	c	b	b	c	b

Input tape

Finite state machine

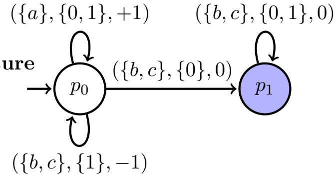


0

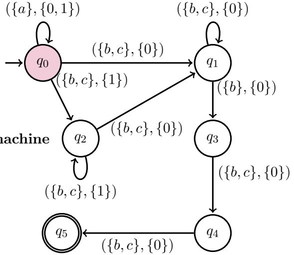
Counter

0	1	0	1	0	1

Counter structure

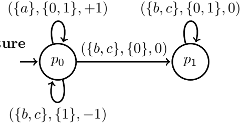


Finite state machine



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

.

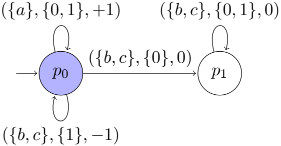
Transition matrices

a	c	c	b	c	b

Input tape

1	0	0	0	0	0

Initial vector



Counter structure

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

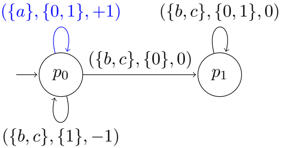
$(\{c\}, \{0\})$

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



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

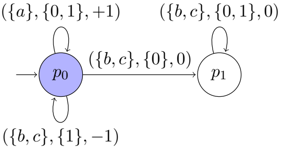
$(\{c\},\{0\})$

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

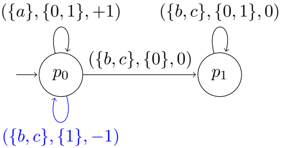
$(\{c\},\{0\})$

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

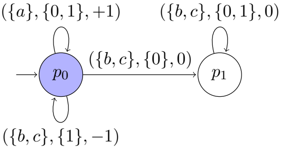
$(\{c\},\{0\})$

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

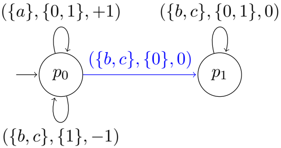
$(\{c\}, \{0\})$

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

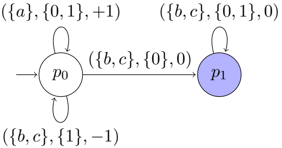
$(\{c\},\{0\})$

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

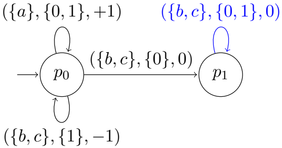
$(\{c\},\{0\})$

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



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

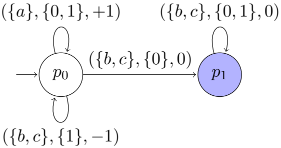
$(\{c\}, \{0\})$

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

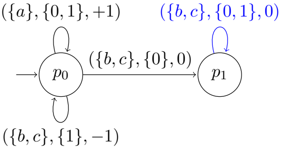
$(\{c\},\{0\})$

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

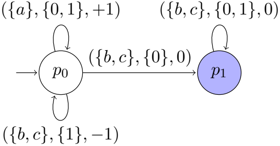
$(\{c\},\{0\})$

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

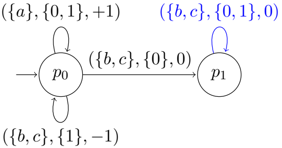
$(\{c\},\{0\})$

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

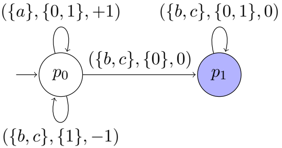
$(\{c\}, \{0\})$

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



$\boxed{0}$

Counter

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