Learning Real-Time One-Counter Automata Using Polynomially Many Queries

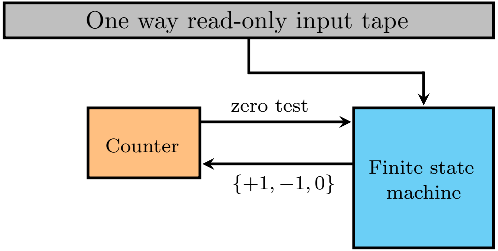
https://doi.org/10.1007/978-3-031-90643-5_14

Prince Mathew
Indian Institute of Technology Goa, India
prince@iitgoa.ac.in

Joint work with: Dr. A.V. Sreejith, Indian Institute of Technology Goa and
Dr. Vincent Penelle, University of Bordeaux

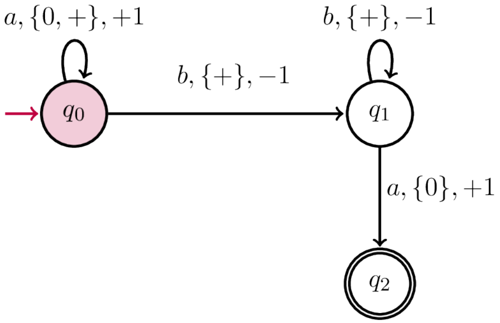
TACAS 2025
Hamilton, Canada

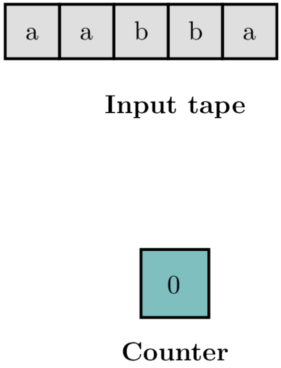
Deterministic real-time one-counter automata (DROCA)



- Write to counter: Increment (+1), No change (0), Decrement (-1).
- Read from counter: zero (0) or positive (+).
- Counter-value is always non-negative.
- Transitions of the finite-state machine are deterministic.
- There are no ε- transitions.

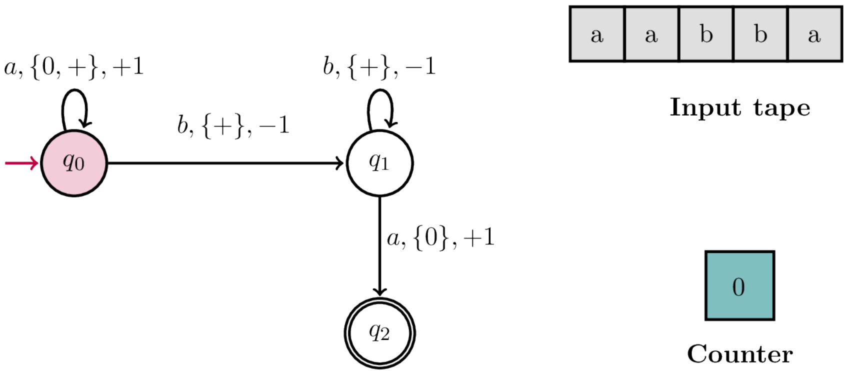
Example: Deterministic real-time one-counter automata



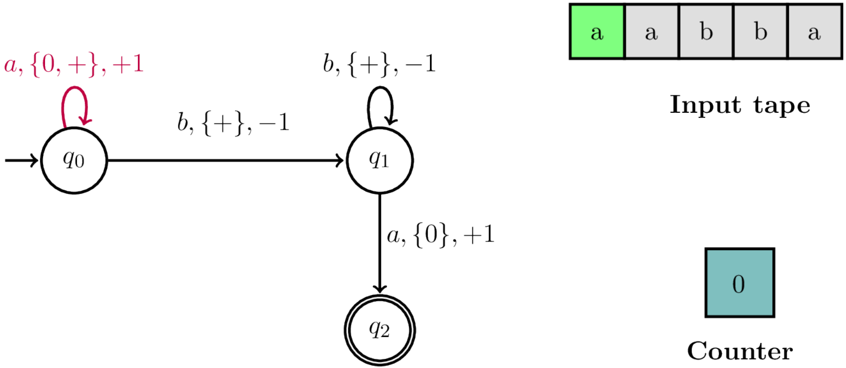


$$\mathscr{L} = \{a^n b^n a \mid n > 0\}$$

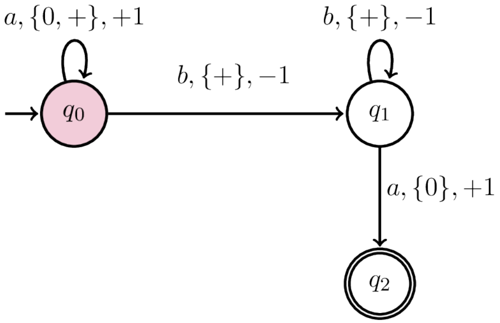
Example: Deterministic real-time one-counter automata

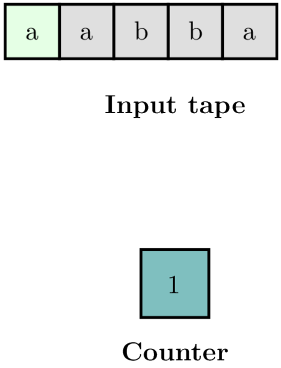


Example: Deterministic real-time one-counter automata

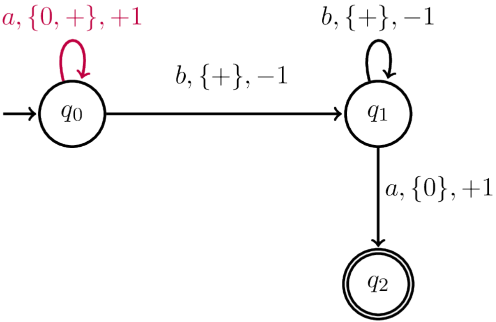


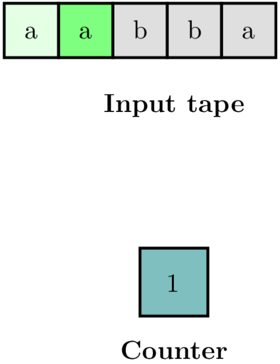
Example: Deterministic real-time one-counter automata



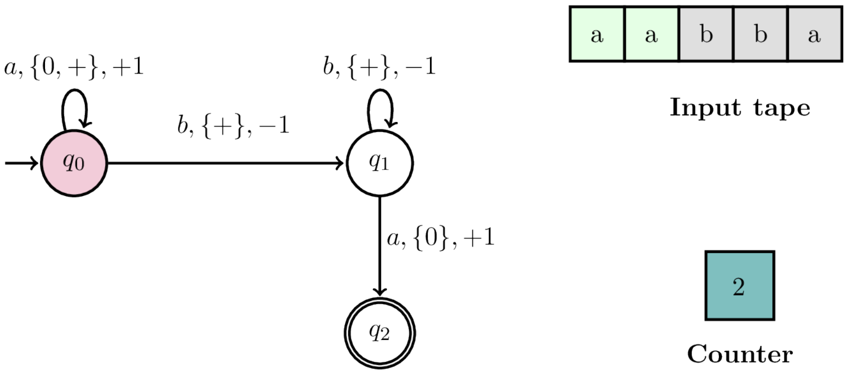


Example: Deterministic real-time one-counter automata

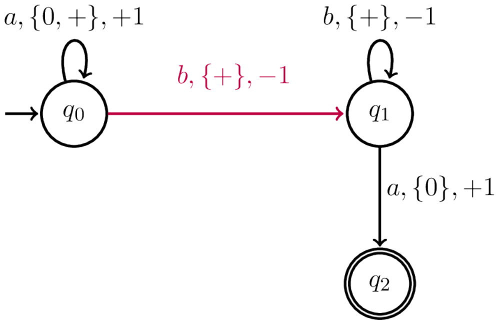


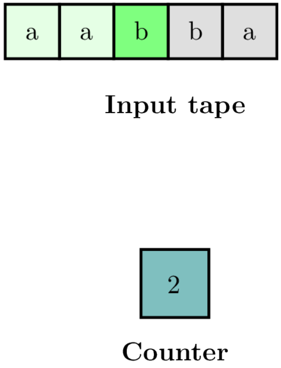


Example: Deterministic real-time one-counter automata

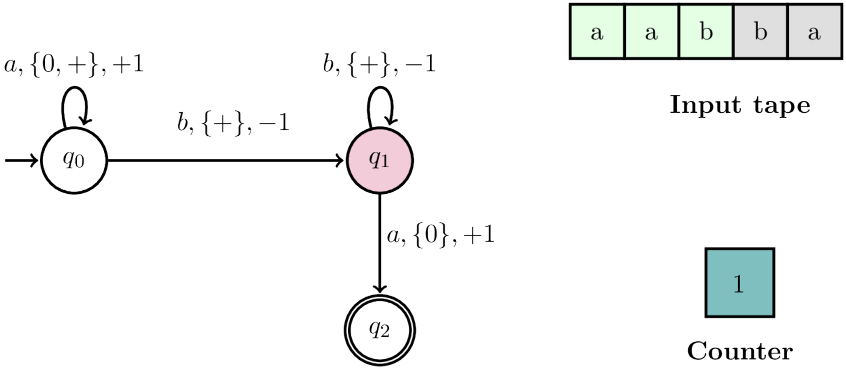


Example: Deterministic real-time one-counter automata

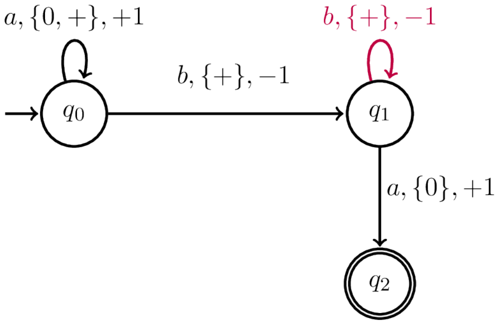


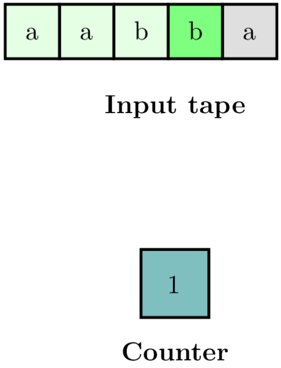


Example: Deterministic real-time one-counter automata

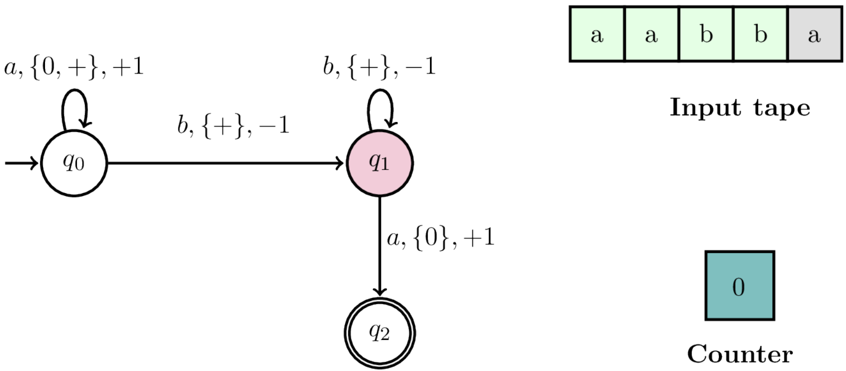


Example: Deterministic real-time one-counter automata

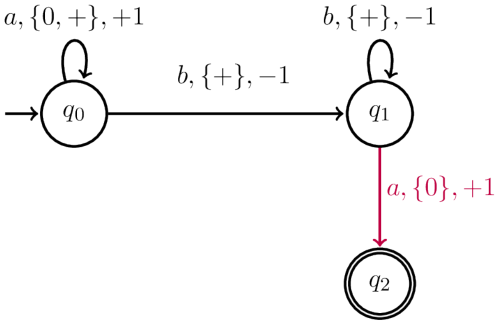


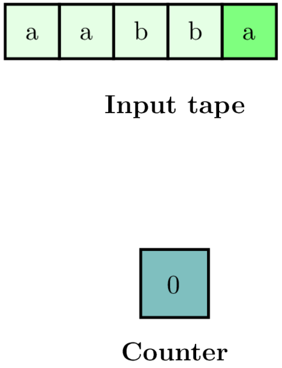


Example: Deterministic real-time one-counter automata

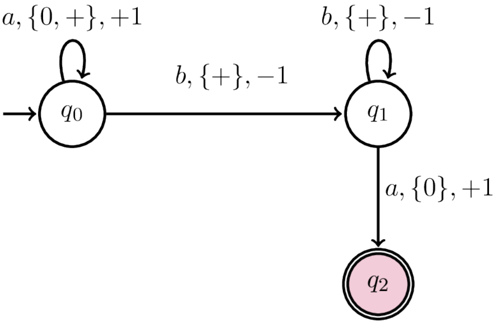


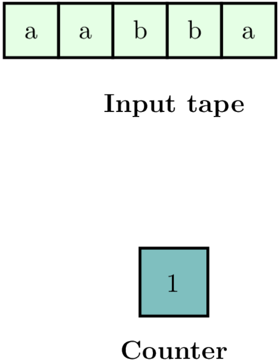
Example: Deterministic real-time one-counter automata



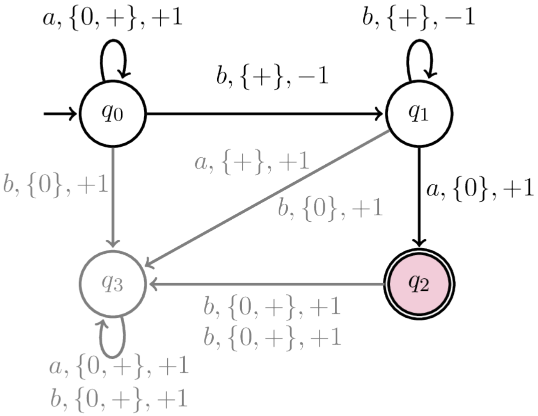


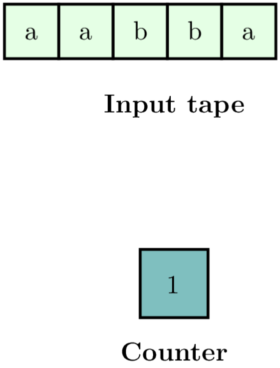
Example: Deterministic real-time one-counter automata





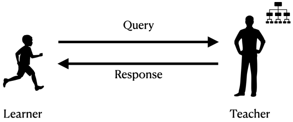
Example: Deterministic real-time one-counter automata





Active Learning DROCAs

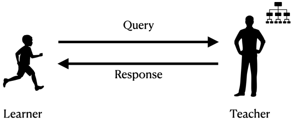
Short history: Learning one-counter automata



(Gold, 1978) → Inferring smallest DFA from a set of labelled samples is NP-complete.

(Angluin, 1987) → Active learning of DFA in polynomial time.

Short history: Learning one-counter automata

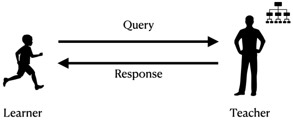


(Gold, 1978) → Inferring smallest DFA from a set of labelled samples is NP-complete.

(Angluin, 1987) → Active learning of DFA in polynomial time.

(Fahmy & Roos, 1995) → Efficient learning of real-time OCA.

Short history: Learning one-counter automata



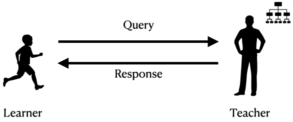
(Gold, 1978) → Inferring smallest DFA from a set of labelled samples is NP-complete.

(Angluin, 1987) → Active learning of DFA in polynomial time.

(Fahmy & Roos, 1995) → Efficient learning of real-time OCA.

(Neider & Löding, 2010) → Learning visibly OCA in EXPTIME.

Short history: Learning one-counter automata



(Gold, 1978) → Inferring smallest DFA from a set of labelled samples is NP-complete.

(Angluin, 1987) → Active learning of DFA in polynomial time.

(Fahmy & Roos, 1995) → Efficient learning of real-time OCA.

(Neider & Löding, 2010) → Learning visibly OCA in EXPTIME.

(Bruyère et. al, 2022) → Learning DROCA with an additional counter-value query in EXPTIME.

Bottlenecks in learning DROCAs

Bottlenecks in learning DROCAs

1. Running time of equivalence query — polynomial, but $\mathcal{O}(n^{26})$.

Bottlenecks in learning DROCAs

1. Running time of equivalence query — polynomial, but $\mathcal{O}(n^{26})$.
2. Number of queries — exponential.

Bottlenecks in learning DROCAs

1. Running time of equivalence query — polynomial, but $\mathcal{O}(n^{26})$.
2. Number of queries — exponential.
3. Time taken to learn — exponential.

Bottlenecks in learning DROCAs

1. Running time of equivalence query — polynomial, but $\mathcal{O}(n^{26})$.
2. Number of queries — exponential.
3. Time taken to learn — exponential.

- In this talk, we address the first two bottlenecks.

Equivalence - A practical approach

Equivalence - A practical approach

- Existing algorithms are not practical — $\mathcal{O}(n^{26})$.

Equivalence - A practical approach

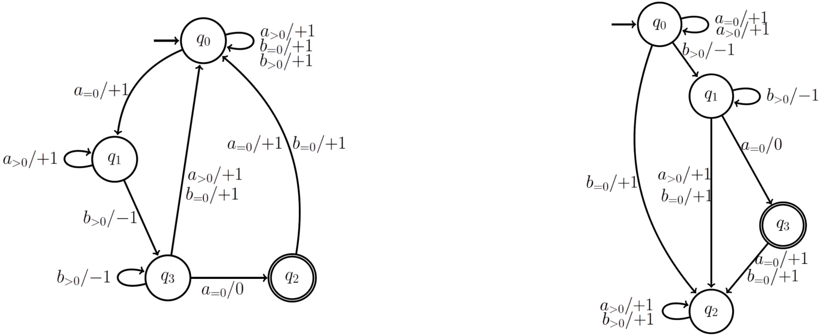
- Existing algorithms are not practical — $\mathcal{O}(n^{26})$.
- We propose *counter-synchronous* equivalence.

Equivalence - A practical approach

- Existing algorithms are not practical — $\mathcal{O}(n^{26})$.
- We propose *counter-synchronous* equivalence.
- Two DROCAs are counter-synchronous if they reach the same counter value on all words.

Equivalence - A practical approach

- Existing algorithms are not practical — $\mathcal{O}(n^{26})$.
- We propose *counter-synchronous* equivalence.
- Two DROCAs are counter-synchronous if they reach the same counter value on all words.



Counter-synchronous DROCAs recognising the language $\{a^n b^n a \mid n > 0\}$.

Counter-synchronous equivalence

Theorem - Counter-synchronous equivalence

Given two DROCAs of size n (not counter-synchronous or not equivalent), there is an $\mathcal{O}(n^6)$ time algorithm to find a word w such that either w is accepted by exactly one DROCA, or counter value of w is different on both DROCAs.

- For visibly OCAs, there is an $\mathcal{O}(n^3)$ time algorithm to check equivalence.

MinOCA

Types of queries

Membership queries:

Learner: “Is w in $\mathscr{L}(\mathscr{A})$?”
Teacher : “yes” or “no”

Counter-value queries:

Learner: “What is the counter-value reached on reading w?”
Teacher: “counter-value reached on w”

Minimal-equivalence queries:

Learner: “Does $\mathscr{B}$ and $\mathscr{A}$ recognise the same language?”
Teacher : “yes” or “no & the smallest counter-example”

Types of queries

Membership queries:

Learner: "Is w in $\mathscr{L}(\mathscr{A})$?"
Teacher : "yes" or "no"

Counter-value queries:

Learner: "What is the counter-value reached on reading w?"
Teacher: "counter-value reached on w"

Minimal-equivalence queries:

Learner: "Does $\mathscr{B}$ and $\mathscr{A}$ recognise the same language?"
Teacher : "yes" or "no & the smallest counter-example"

Membership different

Types of queries

Membership queries:

Learner: "Is w in $\mathscr{L}(\mathscr{A})$?"
Teacher : "yes" or "no"

Counter-value queries:

Learner: "What is the counter-value reached on reading w?"
Teacher: "counter-value reached on w"

Minimal-equivalence queries:

Learner: "Does $\mathscr{B}$ and $\mathscr{A}$ recognise the same language?"
Teacher : "yes" or "no & the smallest counter-example"

Membership different

Counter-value different

MinOCA: active learning algorithm for DROCAs

MinOCA: active learning algorithm for DROCAs

MinOCA: active learning algorithm for DROCAs

Theorem - Polynomially many queries

MinOCA is in P^{NP} and queries the teacher polynomially many times.

MinOCA: active learning algorithm for DROCAs

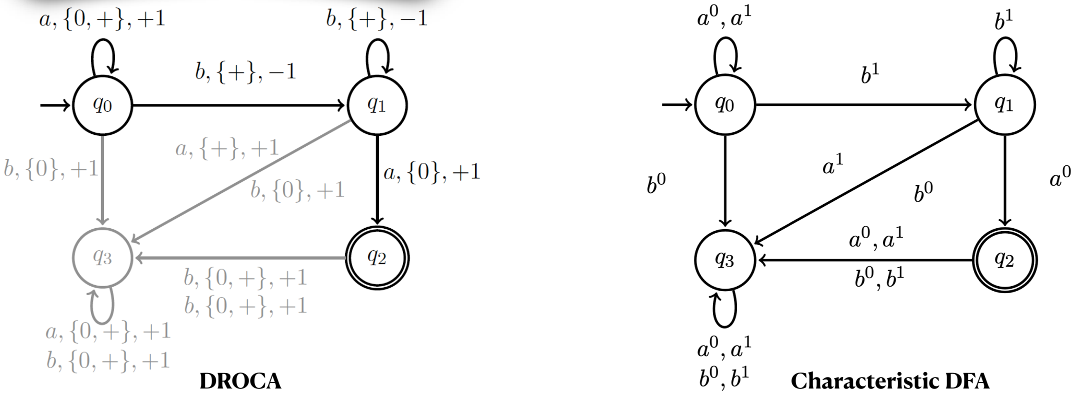
Theorem - Polynomially many queries

MinOCA is in P^{NP} and queries the teacher polynomially many times.

- MinOCA returns a minimal counter-synchronous DROCA.
- Our evaluations show that MinOCA outperforms the existing technique.

Characteristic DFA

$$\mathscr{L} = \{a^n b^n a \mid n > 0\}$$



- The primary idea is to try and learn the characteristic DFA.

Observation table

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
ab	0	0	(0,+1,+1)
aba	1	1	(1,+1,+1)
b	1	0	(1,+1,+1)
aa	2	0	(1,+1,-1)
abb	1	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)
abab	2	0	(1,+1,+1)

Two rows have the same color if they are equal.

Observation table

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
ab	0	0	(0,+1,+1)
aba	1	1	(1,+1,+1)
b	1	0	(1,+1,+1)
aa	2	0	(1,+1,-1)
abb	1	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)
abab	2	0	(1,+1,+1)

Two rows have the same color if they are equal.

Observation table

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
ab	0	0	(0,+1,+1)
aba	1	1	(1,+1,+1)
b	1	0	(1,+1,+1)
aa	2	0	(1,+1,-1)
abb	1	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)
abab	2	0	(1,+1,+1)

(1, +1, -1)

Two rows have the same color if they are equal.

Observation table

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
ab	0	0	(0,+1,+1)
aba	1	1	(1,+1,+1)
b	1	0	(1,+1,+1)
aa	2	0	(1,+1,-1)
abb	1	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)
abab	2	0	(1,+1,+1)

(1, +1, -1)

Sign of counter value

Two rows have the same color if they are equal.

Observation table

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
ab	0	0	(0,+1,+1)
aba	1	1	(1,+1,+1)
b	1	0	(1,+1,+1)
aa	2	0	(1,+1,-1)
abb	1	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)
abab	2	0	(1,+1,+1)

Counter action on 'a'

(1, +1, -1)

Sign of counter value

Two rows have the same color if they are equal.

Observation table

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
ab	0	0	(0,+1,+1)
aba	1	1	(1,+1,+1)
b	1	0	(1,+1,+1)
aa	2	0	(1,+1,-1)
abb	1	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)
abab	2	0	(1,+1,+1)

Counter action on 'a'

(1, +1, -1)

Sign of counter value

Counter action on 'b'

Two rows have the same color if they are equal.

d-closed table

ε

ε

d-closed table

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
ab	0	0	(0,+1,+1)
aba	1	1	(1,+1,+1)
b	1	0	(1,+1,+1)
aa	2	0	(1,+1,-1)
abb	1	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)
abab	2	0	(1,+1,+1)

Color not present in the top part

Not 1-Closed

- A table is d-**closed** if each row in the bottom part with counter value $\leq$ d, is *equal* to at least one row in the top part.

d-closed table

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
ab	0	0	(0,+1,+1)
aba	1	1	(1,+1,+1)
b	1	0	(1,+1,+1)
aa	2	0	(1,+1,-1)
abb	1	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)
abab	2	0	(1,+1,+1)

Move it to the top part and add it's one letter extensions to the bottom part.

d-closed table

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
ab	0	0	(0,+1,+1)
aba	1	1	(1,+1,+1)
b	1	0	(1,+1,+1)
aa	2	0	(1,+1,-1)
abb	1	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)
abab	2	0	(1,+1,+1)
ba	2	0	(1,+1,+1)

Move it to the top part and add it's one letter extensions to the bottom part.

d-closed table

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
ab	0	0	(0,+1,+1)
aba	1	1	(1,+1,+1)
b	1	0	(1,+1,+1)
aa	2	0	(1,+1,-1)
abb	1	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)
abab	2	0	(1,+1,+1)
ba	2	0	(1,+1,+1)
bb	1	0	(1,+1,+1)

Move it to the top part and add it's one letter extensions to the bottom part.

d-closed table

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
ab	0	0	(0,+1,+1)
aba	1	1	(1,+1,+1)
b	1	0	(1,+1,+1)
aa	2	0	(1,+1,-1)
abb	1	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)
abab	2	0	(1,+1,+1)
ba	2	0	(1,+1,+1)
bb	1	0	(1,+1,+1)

Move it to the top part and add it's one letter extensions to the bottom part.

1-Closed

d-consistent table

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
ab	0	0	(0,+1,+1)
aba	1	1	(1,+1,+1)
b	1	0	(1,+1,+1)
aa	2	0	(1,+1,-1)
abb	1	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)
abab	2	0	(1,+1,+1)
ba	2	0	(1,+1,+1)
bb	2	0	(1,+1,+1)

ε and ab have same color

But ε.a $\neq$ ab.a

Add 'a' to the column

- A table is d-**consistent** if equal rows with counter value $\leq$ d in the top part has equal extensions.

Not 0-Consistent

d-consistent table

	Counter Value	ε		a	
		Memb	Actions	Memb	Actions
ε	0	0	(0, +1, +1)	0	(1,+1,-1)
a	1	0	(1,+1,-1)	0	(1,+1,-1)
ab	0	0	(0,+1,+1)	1	(1,+1,+1)
aba	1	1	(1,+1,+1)	0	(1,+1,+1)
b	1	0	(1,+1,+1)	0	(1,+1,+1)
aa	2	0	(1,+1,-1)	0	(1,+1,-1)
abb	1	0	(1,+1,+1)	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)	0	(1,+1,+1)
abab	2	0	(1,+1,+1)	0	(1,+1,+1)
ba	2	0	(1,+1,+1)	0	(1,+1,+1)
bb	2	0	(1,+1,+1)	0	(1,+1,+1)

d-consistent table

	Counter Value	ε		a	
		Memb	Actions	Memb	Actions
ε	0	0	(0, +1, +1)	0	(1,+1,-1)
a	1	0	(1,+1,-1)	0	(1,+1,-1)
ab	0	0	(0,+1,+1)	1	(0,+1,+1)
aba	1	1	(1,+1,+1)	0	(1,+1,+1)
b	1	0	(1,+1,+1)	0	(1,+1,+1)
aa	2	0	(1,+1,-1)	0	(1,+1,-1)
abb	1	0	(1,+1,+1)	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)	0	(1,+1,+1)
abab	2	0	(1,+1,+1)	0	(1,+1,+1)
ba	2	0	(1,+1,+1)	0	(1,+1,+1)
bb	2	0	(1,+1,+1)	0	(1,+1,+1)

1-Consistent

d-consistent table

	Counter Value	ε		a	
		Memb	Actions	Memb	Actions
ε	0	0	(0, +1, +1)	0	(1,+1,-1)
a	1	0	(1,+1,-1)	0	(1,+1,-1)
ab	0	0	(0,+1,+1)	1	(0,+1,+1)
aba	1	1	(1,+1,+1)	0	(1,+1,+1)
b	1	0	(1,+1,+1)	0	(1,+1,+1)
aa	2	0	(1,+1,-1)	0	(1,+1,-1)
abb	1	0	(1,+1,+1)	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)	0	(1,+1,+1)
abab	2	0	(1,+1,+1)	0	(1,+1,+1)
ba	2	0	(1,+1,+1)	0	(1,+1,+1)
bb	2	0	(1,+1,+1)	0	(1,+1,+1)

1-Consistent

d-consistent table

	Counter Value	ε		a	
		Memb	**Actions**	**Memb**	**Actions**
ε	0	0	(0, +1, +1)	0	(1,+1,-1)
a	1	0	(1,+1,-1)	0	(1,+1,-1)
ab	0	0	(0,+1,+1)	1	(0,+1,+1)
aba	1	1	(1,+1,+1)	0	(1,+1,+1)
b	1	0	(1,+1,+1)	0	(1,+1,+1)
aa	2	0	(1,+1,-1)	0	(1,+1,-1)
abb	1	0	(1,+1,+1)	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)	0	(1,+1,+1)
abab	2	0	(1,+1,+1)	0	(1,+1,+1)
ba	2	0	(1,+1,+1)	0	(1,+1,+1)
bb	2	0	(1,+1,+1)	0	(1,+1,+1)

1-Consistent

$\{a,b\}^* \rightarrow \{a^0, a^1, b^0, b^1\}^*$

d-consistent table

	Counter Value	ε		a	
		Memb	Actions	Memb	Actions
ε	0	0	(0, +1, +1)	0	(1,+1,-1)
a	1	0	(1,+1,-1)	0	(1,+1,-1)
ab	0	0	(0,+1,+1)	1	(0,+1,+1)
aba	1	1	(1,+1,+1)	0	(1,+1,+1)
b	1	0	(1,+1,+1)	0	(1,+1,+1)
aa	2	0	(1,+1,-1)	0	(1,+1,-1)
abb	1	0	(1,+1,+1)	0	(1,+1,+1)
abaa	2	0	(1,+1,+1)	0	(1,+1,+1)
abab	2	0	(1,+1,+1)	0	(1,+1,+1)
ba	2	0	(1,+1,+1)	0	(1,+1,+1)
bb	2	0	(1,+1,+1)	0	(1,+1,+1)

1-Consistent

$\{a,b\}^* \rightarrow \{a^0, a^1, b^0, b^1\}^*$

$a\ b\ a \rightarrow a^0\ b^1\ a^0$

Sketch: MinOCA

1. Initialise the observation table with $Rows = \{\varepsilon\}$, $Columns = \{\varepsilon\}$, and $d = 0$.

	Counter Value	ε	
		Memb	Actions
ε	0		

Sketch: MinOCA

1. Initialise the observation table with *Rows*= $\{\varepsilon\}$, *Columns*= $\{\varepsilon\}$, and $d = 0$.
2. Construct a d-closed and d-consistent observation table **H** using membership & counter-value queries.

	Counter Value	ε	
		Memb	**Actions**
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
b	1	0	(1,+1,+1)

Sketch: MinOCA

1. Initialise the observation table with *Rows*$= \{\varepsilon\}$, *Columns*$= \{\varepsilon\}$, and $d = 0$.
2. Construct a d-closed and d-consistent observation table **H** using membership & counter-value queries.
3. Convert entries of **H** into the modified alphabet and create sets **POS** and **NEG**.

	Counter Value	ε	
		Memb	**Actions**
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
b	1	0	(1,+1,+1)

POS	NEG
	ε
	$\mathbf{a^o}$
	$\mathbf{b^o}$

Sketch: MinOCA

1. Initialise the observation table with *Rows*= $\{\varepsilon\}$, *Columns*= $\{\varepsilon\}$, and $d = 0$.
2. Construct a d-closed and d-consistent observation table **H** using membership & counter-value queries.
3. Convert entries of **H** into the modified alphabet and create sets **POS** and **NEG**.

	Counter Value	ε		
		Memb	Actions	
ε	0	0	(0, +1, +1)	→ x
a	1	0	(1,+1,-1)	→ y
b	1	0	(1,+1,+1)	→ z

POS	NEG
	ε
	a^{o}
	b^{o}

Sketch: MinOCA

1. Initialise the observation table with $Rows = \{\varepsilon\}$, $Columns = \{\varepsilon\}$, and $d = 0$.
2. Construct a d-closed and d-consistent observation table **H** using membership & counter-value queries.
3. Convert entries of **H** into the modified alphabet and create sets **POS** and **NEG**.

	Counter Value	ε		
		Memb	Actions	
ε	0	0	(0, +1, +1)	→ x
a	1	0	(1,+1,-1)	→ y
b	1	0	(1,+1,+1)	→ z

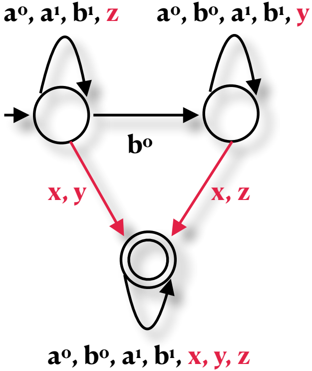
POS	NEG
x	ε
$a^{o}y$	a^{o}
$b^{o}z$	b^{o}
	$a^{o}z$
	$b^{o}y$

Sketch: MinOCA

1. Initialise the observation table with *Rows*$= \{\varepsilon\}$, *Columns*$= \{\varepsilon\}$, and $d = 0$.
2. Construct a d-closed and d-consistent observation table **H** using membership & counter-value queries.
3. Convert entries of **H** into the modified alphabet and create sets **POS** and **NEG**.
4. Use SAT solver to find a DFA with minimal size that separates **POS** and **NEG**.

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
b	1	0	(1,+1,+1)

POS	NEG
x	ε
a^0y	a^0
b^0z	b^0
	a^0z
	b^0y

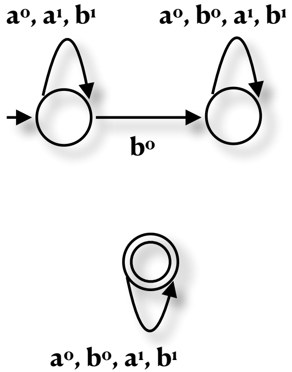


Sketch: MinOCA

1. Initialise the observation table with *Rows*$= \{\varepsilon\}$, *Columns*$= \{\varepsilon\}$, and $d = 0$.
2. Construct a d-closed and d-consistent observation table **H** using membership & counter-value queries.
3. Convert entries of **H** into the modified alphabet and create sets **POS** and **NEG**.
4. Use SAT solver to find a DFA with minimal size that separates **POS** and **NEG**.

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
b	1	0	(1,+1,+1)

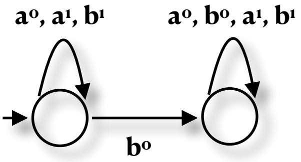
POS	NEG
x	ε
a^0y	a^0
b^0z	b^0
	a^0z
	b^0y



Sketch: MinOCA

1. Initialise the observation table with *Rows*= $\{\varepsilon\}$, *Columns*= $\{\varepsilon\}$, and $d = 0$.
2. Construct a d-closed and d-consistent observation table **H** using membership & counter-value queries.
3. Convert entries of **H** into the modified alphabet and create sets **POS** and **NEG**.
4. Use SAT solver to find a DFA with minimal size that separates **POS** and **NEG**.
5. Convert this characteristic DFA to an OCA **A** over Σ and ask equivalence query.

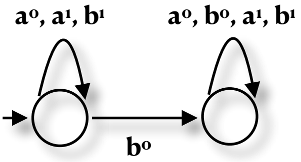
	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
b	1	0	(1,+1,+1)

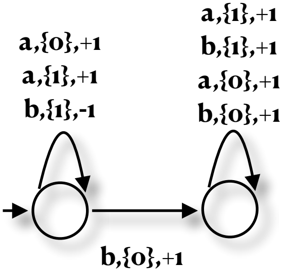


Sketch: MinOCA

1. Initialise the observation table with *Rows*$= \{\varepsilon\}$, *Columns*$= \{\varepsilon\}$, and $d = 0$.
2. Construct a d-closed and d-consistent observation table **H** using membership & counter-value queries.
3. Convert entries of **H** into the modified alphabet and create sets **POS** and **NEG**.
4. Use SAT solver to find a DFA with minimal size that separates **POS** and **NEG**.
5. Convert this characteristic DFA to an OCA **A** over Σ and ask equivalence query.

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
b	1	0	(1,+1,+1)

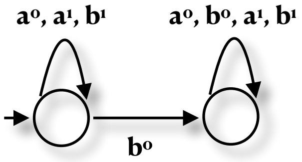


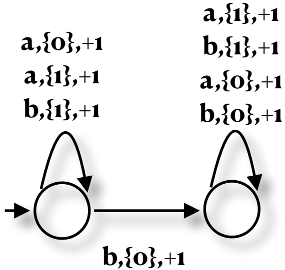


Sketch: MinOCA

1. Initialise the observation table with *Rows*= $\{\varepsilon\}$, *Columns*= $\{\varepsilon\}$, and $d = 0$.
2. Construct a d-closed and d-consistent observation table **H** using membership & counter-value queries.
3. Convert entries of **H** into the modified alphabet and create sets **POS** and **NEG**.
4. Use SAT solver to find a DFA with minimal size that separates **POS** and **NEG**.
5. Convert this characteristic DFA to an OCA **A** over Σ and ask equivalence query.
6. If the teacher returns a counter example, then add all its prefixes to *Rows*, increment d & repeat steps 2-5.

	Counter Value	ε	
		Memb	Actions
ε	0	0	(0, +1, +1)
a	1	0	(1,+1,-1)
b	1	0	(1,+1,+1)





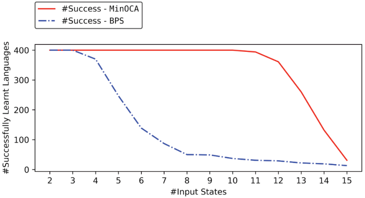
Counter example: aba

Sketch: MinOCA

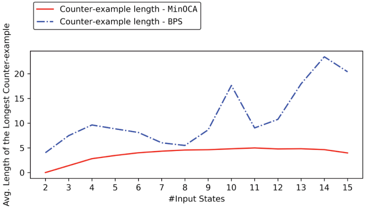
1. Initialise the observation table with *Rows*= $\{\varepsilon\}$, *Columns*= $\{\varepsilon\}$, and $d = 0$.
2. Construct a d-closed and d-consistent observation table **H** using membership & counter-value queries.
3. Convert entries of **H** into the modified alphabet and create sets **POS** and **NEG**.
4. Use SAT solver to find a DFA with minimal size that separates **POS** and **NEG**.
5. Convert this characteristic DFA to an OCA **A** over Σ and ask equivalence query.
6. If the teacher returns a counter example, then add all its prefixes to *Rows*, increment d & repeat steps 2-5.
7. Else stop and output **A**.

Experimental Results

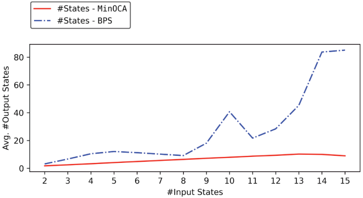
Comparison with existing method BPS (Bruyère et al., 2022)



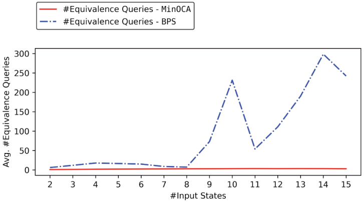
Number of successfully learnt languages (out of 400)



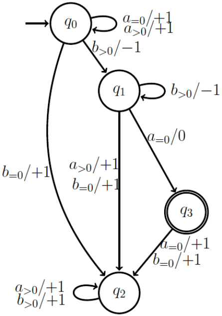
Average length of the longest counter-example



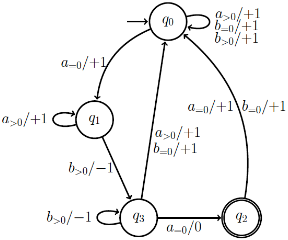
Average number of states in the learnt DROCA



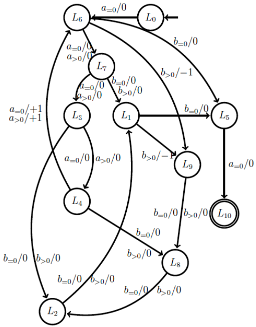
Average number of equivalence queries used for learning



(a) Input DROCA



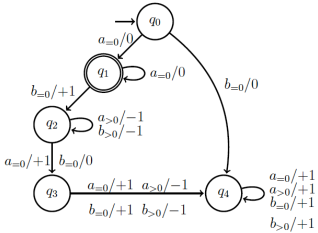
(b) Learnt by MinOCA



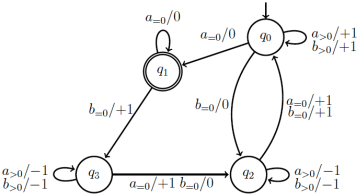
(c) Learnt by BPS

The input DROCA recognises the language $\{a^n b^n a \mid n > 0\}$.

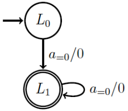
Example-2: DROCAs learnt by minOCA and BPS



(a) Input DROCA



(b) Learnt by MinOCA



(c) Learnt by BPS

The input DROCA recognises the language $\{w \in \{a,b\}^* \mid w \text{ does not contain a } b\}$.

Summary

- Active learning of DROCAs.
    - Three types of queries used: membership, counter value query, and minimal-equivalence query.
    - Existing algorithms for active learning of DROCAs need exponentially many queries.
- We proposed an active learning algorithm (MinOCA).
    - MinOCA is in P^{NP} — queries the teacher polynomial number of times.
    - Learns a minimal counter-synchronous DROCA.
    - Learns DROCAs of size up to size 10 in 5 minutes — better than existing technique.

Future Work

- Major bottleneck for practical applications is the equivalence check of DROCAs.
    - Can we improve this?
- Bottleneck for using MinOCA is finding the minimal separating DFA.
    - We can have a faster algorithm, if there is a better way to find the separating DFA.
- Does there exists a polynomial time algorithm for learning DROCAs theoretically?

Future Work

- Major bottleneck for practical applications is the equivalence check of DROCAs.
    - Can we improve this?
- Bottleneck for using MinOCA is finding the minimal separating DFA.
    - We can have a faster algorithm, if there is a better way to find the separating DFA.
- Does there exists a polynomial time algorithm for learning DROCAs theoretically? Yes.

Future Work

- Major bottleneck for practical applications is the equivalence check of DROCAs.
    - Can we improve this?
- Bottleneck for using MinOCA is finding the minimal separating DFA.
    - We can have a faster algorithm, if there is a better way to find the separating DFA.
- Does there exists a polynomial time algorithm for learning DROCAs theoretically? Yes.

Learning Deterministic One-Counter Automata in Polynomial Time
https://arxiv.org/abs/2503.04525
LICS 2025 (to appear)

References

Fahmy and Roos.
"Efficient learning of real time one-counter automata"
In ALT: International Workshop on Algorithmic Learning Theory, 1995.

Véronique Bruyère, Guillermo A. Pérez, and Gaëtan Staquet.
"Learning Realtime One-Counter Automata"
Tools and Algorithms for the Construction and Analysis of Systems, pages 244–262, 2022.

Daniel Neider and Christof Loding.
"Learning visibly one-counter automata in polynomial time"
Technical Report, RWTH Aachen, AIB-2010-02, 2010.

Dana Angluin.
"Learning regular sets from queries and counter examples"
Information and Computation, pages 87-106, 1987.

E Mark Gold.
"Complexity of automaton identification from given data"
Information and Control, pages 302-320, 1978.

Thank You !

Learning Real-Time One-Counter Automata Using Polynomially Many Queries

https://doi.org/10.1007/978-3-031-90643-5_14

Prince Mathew

Indian Institute of Technology Goa, India

prince@iitgoa.ac.in

Joint work with: Dr. A.V. Sreejith, Indian Institute of Technology Goa and Dr. Vincent Penelle, University of Bordeaux

TACAS 2025

Hamilton, Canada