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

https://arxiv.org/abs/2411.08815

Prince Mathew

PhD Scholar

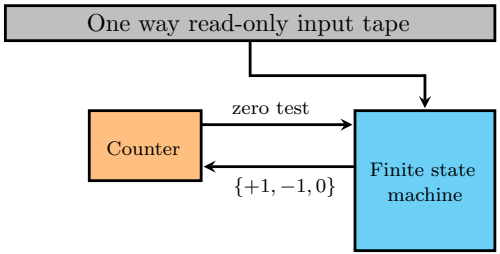
Indian Institute of Technology Goa, India

prince@iitgoa.ac.in

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

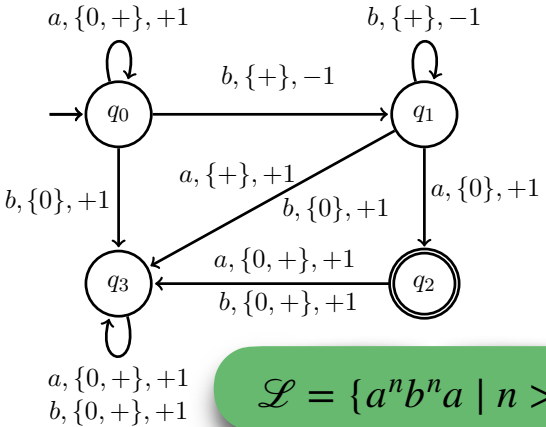
RHPL 2024

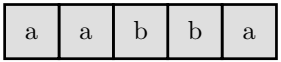
Deterministic real-time one-counter automata (DROCA)



- 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



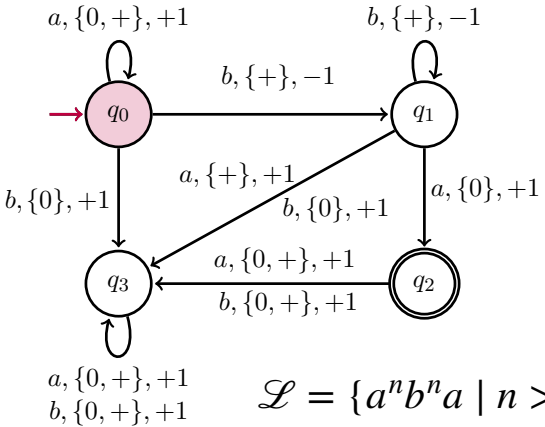


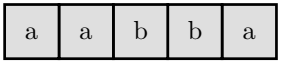
Input tape

Counter

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

Example: Deterministic real-time one-counter automata



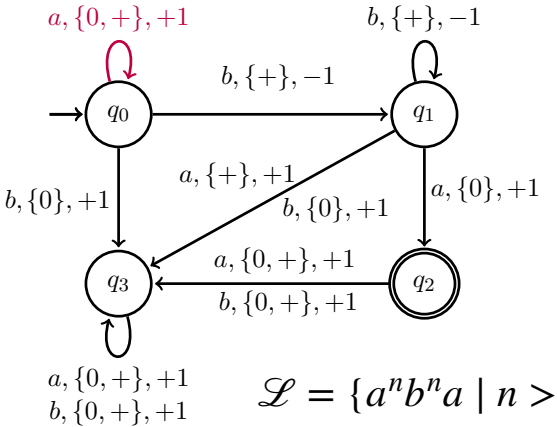


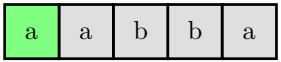
Input tape

Counter

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

Example: Deterministic real-time one-counter automata



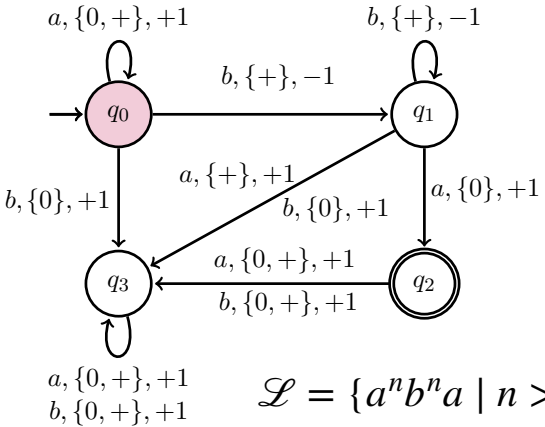


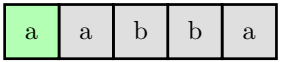
Input tape

Counter

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

Example: Deterministic real-time one-counter automata



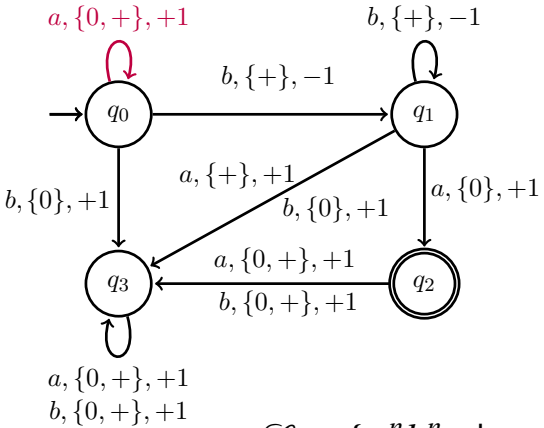


Input tape

Counter

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

Example: Deterministic real-time one-counter automata

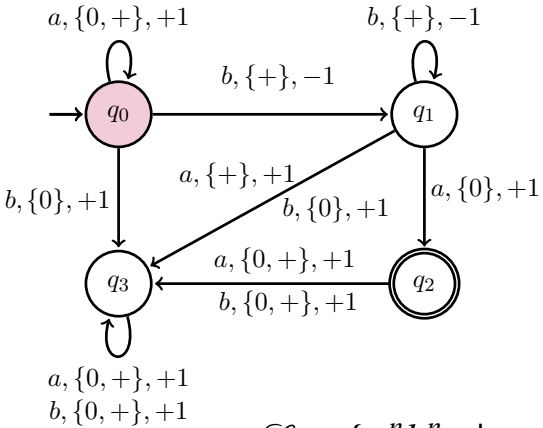


Input tape

Counter

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

Example: Deterministic real-time one-counter automata

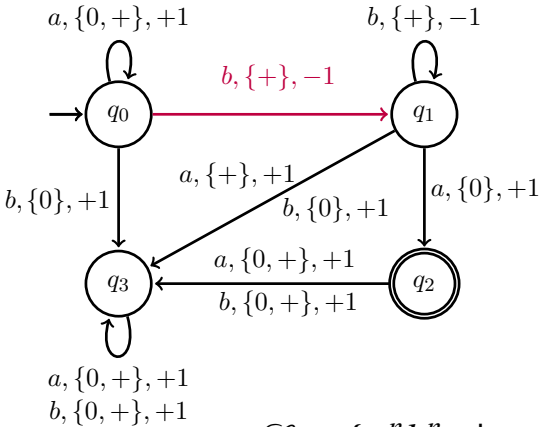


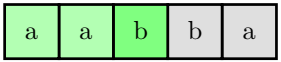
Input tape

Counter

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

Example: Deterministic real-time one-counter automata



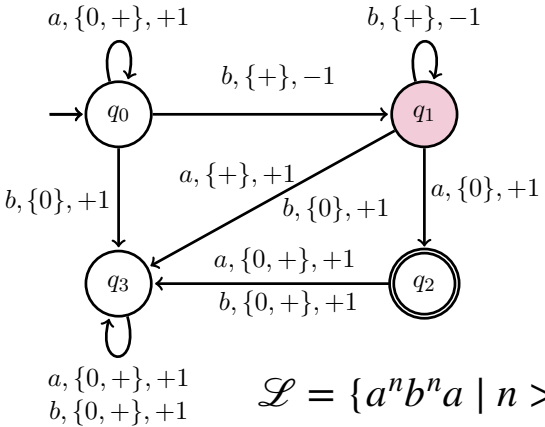


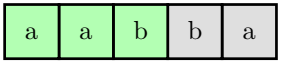
Input tape

Counter

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

Example: Deterministic real-time one-counter automata



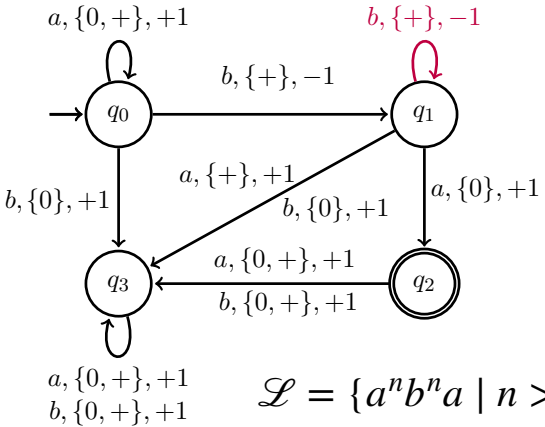


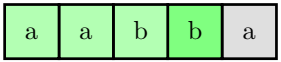
Input tape

Counter

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

Example: Deterministic real-time one-counter automata



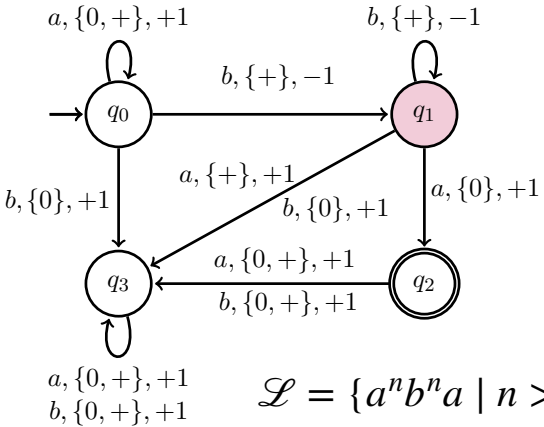


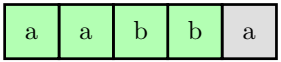
Input tape

Counter

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

Example: Deterministic real-time one-counter automata



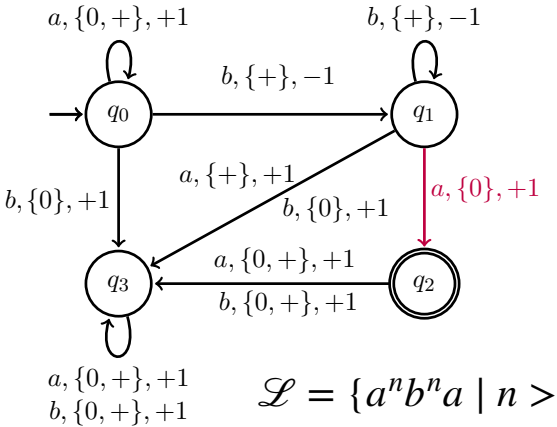


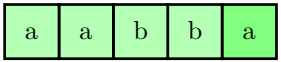
Input tape

Counter

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

Example: Deterministic real-time one-counter automata



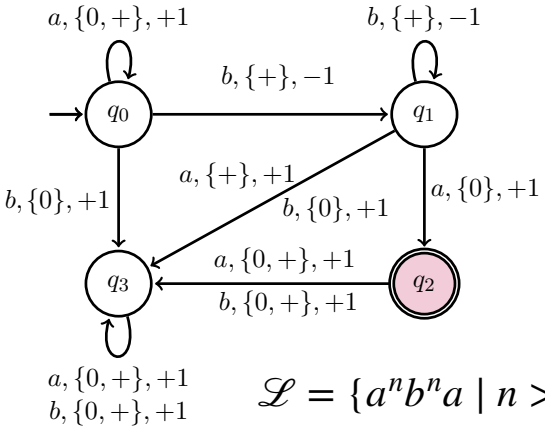


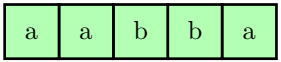
Input tape

Counter

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

Example: Deterministic real-time one-counter automata





Input tape

Counter

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

Equivalence - A practical algorithm

Equivalence - A practical algorithm

- The existing algorithms for checking equivalence of DROCAs cannot be used in practice.

Equivalence - A practical algorithm

- The existing algorithms for checking equivalence of DROCAs cannot be used in practice.
- We show that there exists a faster algorithm to check their equivalence of counter-synchronous DROCAs.

Equivalence - A practical algorithm

- The existing algorithms for checking equivalence of DROCAs cannot be used in practice.
- We show that there exists a faster algorithm to check their equivalence of counter-synchronous DROCAs.
- Two DROCAs are said to be counter-synchronous if they reach the same counter value on all words.

Equivalence - A practical algorithm

- The existing algorithms for checking equivalence of DROCAs cannot be used in practice.
- We show that there exists a faster algorithm to check their equivalence of counter-synchronous DROCAs.
- Two DROCAs are said to be counter-synchronous if they reach the same counter value on all words.

Theorem - Check counter-synchronous

Given two DROCAs of size n, there is an algorithm that runs in $O(n^5)$ time to check whether they are counter-synchronous.

Equivalence - A practical algorithm

- The existing algorithms for checking equivalence of DROCAs cannot be used in practice.
- We show that there exists a faster algorithm to check their equivalence of counter-synchronous DROCAs.
- Two DROCAs are said to be counter-synchronous if they reach the same counter value on all words.

Theorem - Check counter-synchronous

Given two DROCAs of size n, there is an algorithm that runs in $O(n^5)$ time to check whether they are counter-synchronous.

Theorem - Equivalence of counter-synchronous DROCAs

Given two counter-synchronous DROCAs of size n, there is an algorithm that runs in $O(n^5)$ time to check their equivalence.

Equivalence - A practical algorithm

- The existing algorithms for checking equivalence of DROCAs cannot be used in practice.
- We show that there exists a faster algorithm to check their equivalence of counter-synchronous DROCAs.
- Two DROCAs are said to be counter-synchronous if they reach the same counter value on all words.

Theorem - Check counter-synchronous

Given two DROCAs of size n, there is an algorithm that runs in $O(n^5)$ time to check whether they are counter-synchronous.

Theorem - Equivalence of counter-synchronous DROCAs

Given two counter-synchronous DROCAs of size n, there is an algorithm that runs in $O(n^5)$ time to check their equivalence.

- For the class of visibly OCAs there is a $O(n^3)$ time algorithm for checking equivalence.

Learning DROCAs

Our work on learning DROCA

- Existing results for learning DROCA need exponential time & number of queries.
- We have the following result:
  - Learning DROCA can be done using **polynomially** many queries using a SAT solver.
- Implemented a learning algorithm for DROCA in Python.
- Our evaluations show that the proposed method significantly outperforms the existing technique.
- We use the idea of finding a minimal separating DFA from a given set of positive & negative samples.

Minimal separating DFA

- Given two sets POS and NEG.
- Construct a DFA $\mathscr{A}$ with the minimal number of states such that
  - $\mathscr{A}$ accepts all words in POS, and
  - $\mathscr{A}$ rejects all words in NEG.

Minimal separating DFA

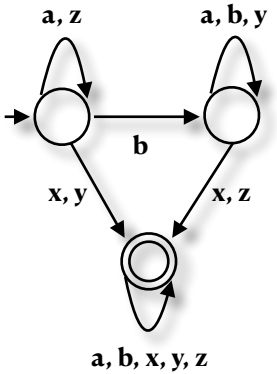
- Given two sets POS and NEG.
- Construct a DFA $\mathscr{A}$ with the minimal number of states such that
  - $\mathscr{A}$ accepts all words in POS, and
  - $\mathscr{A}$ rejects all words in NEG.

POS	NEG
x	ϵ
ay	**a**
bz	**b**
	az
	by

Minimal separating DFA

- Given two sets POS and NEG.
- Construct a DFA $\mathscr{A}$ with the minimal number of states such that
  - $\mathscr{A}$ accepts all words in POS, and
  - $\mathscr{A}$ rejects all words in NEG.

POS	NEG
x	ϵ
ay	a
bz	b
	az
	by



Our learning framework

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"

Our learning framework

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

Our learning framework

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"

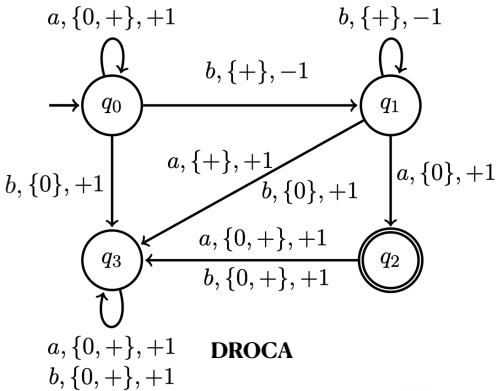
Minimal-equivalence queries:

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

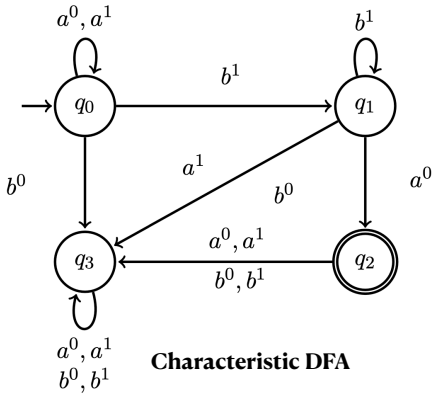
Membership different

Counter-value different

Characteristic DFA



DROCA

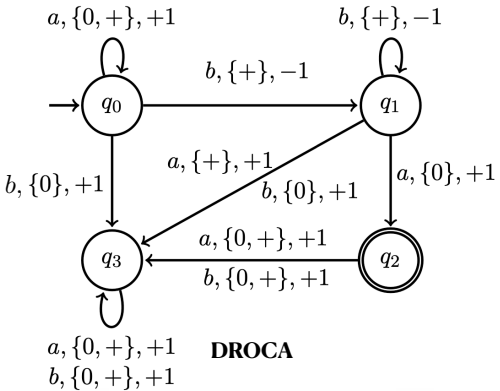


Characteristic DFA

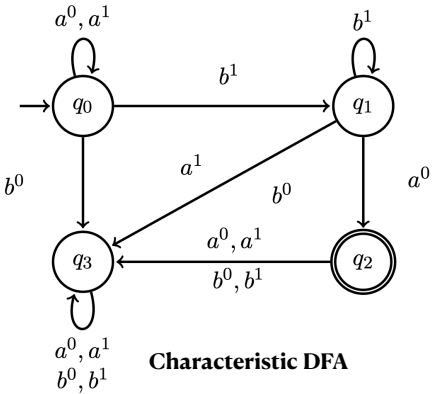
$$\mathscr{L} = \{a^n b^n a \mid n > 0\}$$

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

Characteristic DFA



DROCA



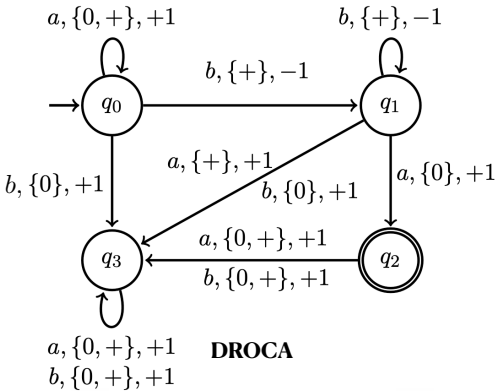
Characteristic DFA

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

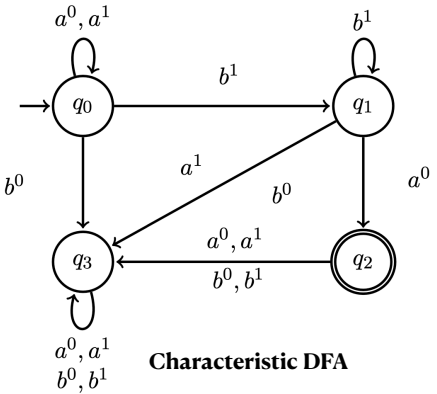
$\{a,b\}^* \rightarrow \{a^0, a^1, b^0, b^1\}^*$

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

Characteristic DFA



DROCA



Characteristic DFA

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

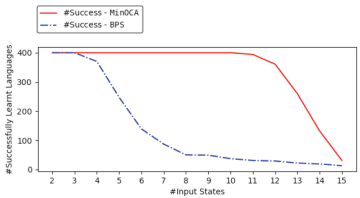
- The primary idea is to try and learn the characteristic DFA.

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

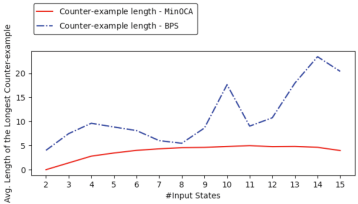
$a\ b\ a \rightarrow a^0\ b^1\ a^0$

Experimental Results

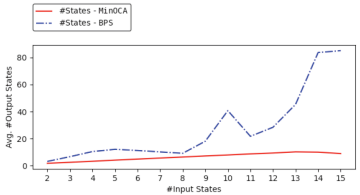
Comparison with existing method (Bruyère et al., 2022)



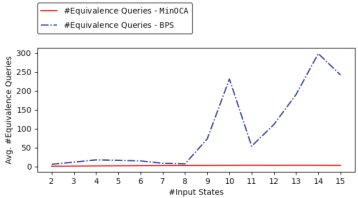
Number of successfully learnt languages (out of 400)



Average length of the longest counter-example

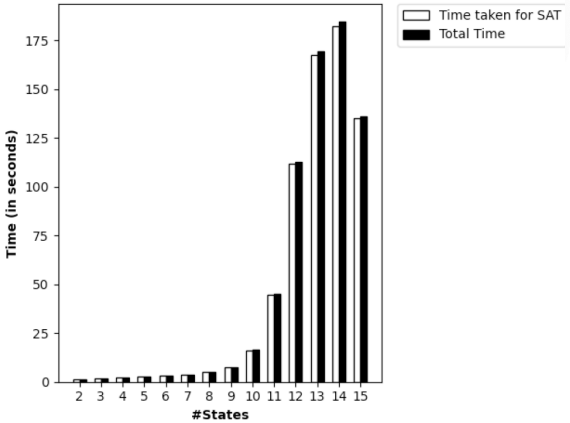


Average number of states in the learnt DROCA

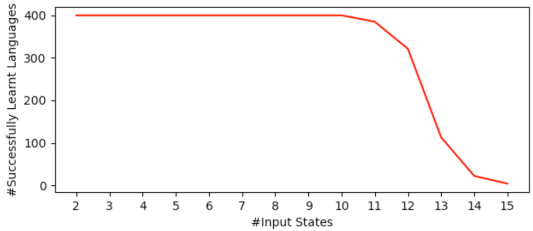


Average number of equivalence queries used for learning

Experimental results

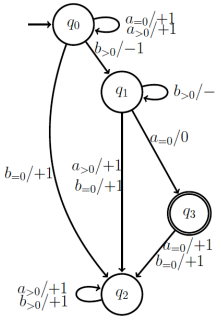


Total time vs Time taken for SAT by minOCA

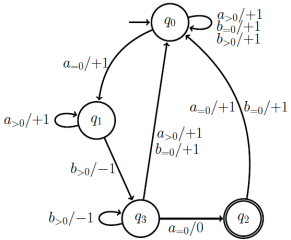


Number of successfully learnt DROCAs by minOCA with a timeout of 5 minutes

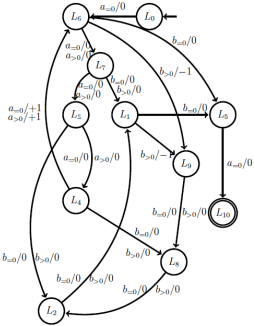
Example-1: DROCAs learnt by minOCA and BPS



(a) Input DROCA



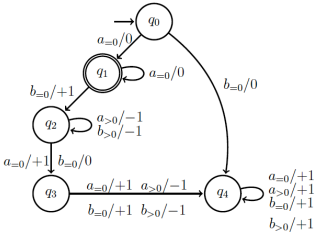
(b) Learnt by MinOCA



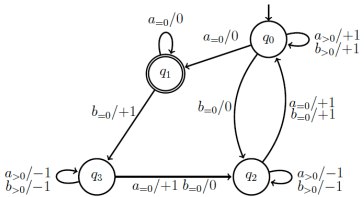
(c) Learnt by BPS

Fig. 2.17. The input DROCA recognises the language $\{a^n b^n a \mid n > 0\}$.

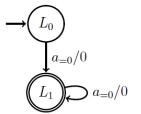
Example-2: DROCAs learnt by minOCA and BPS



(a) Input DROCA



(b) Learnt by MinOCA



(c) Learnt by BPS

Fig. 2.18. The input DROCA recognises the language $\{w \in \{a, b\}^* \mid w \text{ does not contain a } b\}$.

Conclusions

- The algorithm terminates after polynomially many queries.
- Learns an DROCA number of states less than or equal to the teacher DROCA.
- This can be used in learning a minimal visibly OCA if the teachers OCA is a visibly OCA.
- The minimisation problem for visibly OCA is NP-Complete (Michaliszyn and Otop, 2022, Gauwin et al., 2020).
- There is no polynomial time algorithm for learning a minimal counter-synchronous DROCA.

Future directions

- Does there exist a polynomial time algorithm for learning OCA?
    - The learnt model can contain a polynomially many states compared to the minimal OCA.
- Can we have a practical algorithm for learning OCA without using a SAT solver?

References I


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


Marijn J.H. Heule and Sicco Verwer.
"Exact DFA Identification Using SAT Solvers"
International Colloquium on Gramatical Inference, pages 66-79, 2010.


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.

References II

Oliver Gauwin, Anca Muscholl, and Michael Raskin.
"Minimization of visibly pushdown automata is NL-Complete"
Logical Methods in Computer Science, Volume 16, Issue 1, pages 14:1-14:9, 2020.

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.

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

References III

Jakub Michaliszyn and Jan Otop.
"Learning visibly pushdown automata under accessible stack"
47th International Symposium on Mathematical Foundations of Computer Science, pages 74:1-74:16, 2022.

Daniele Dell'Erba, Yong Li, and Sven Schewe.
"DFAminer: Mining minimal separating dfas from labelled samples"
26th International Symposium on Formal Methods, 2024 (to appear).

Marcell Vazquez-Chanlatte, Vint Lee, and Ameesh Shah.
"dfa-identify", August 2021.
URL: https://github.com/mvcisback/dfa- identify .

Thank You !