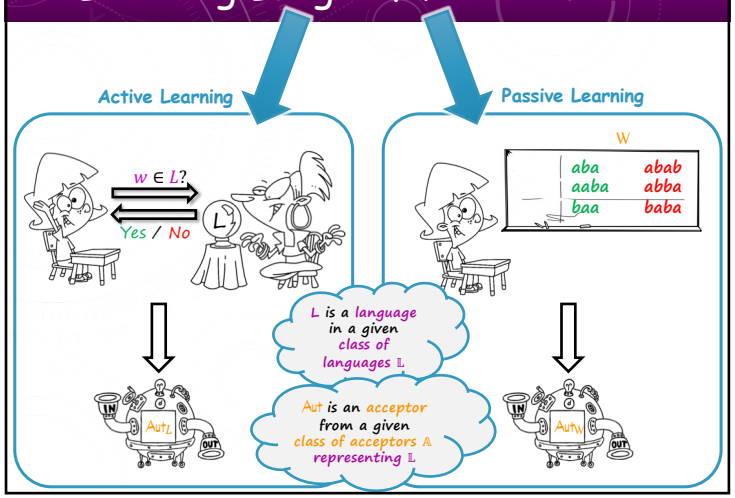
Learning Regular Languages of Infinite Words

Dana Fisman
Ben-Gurion University

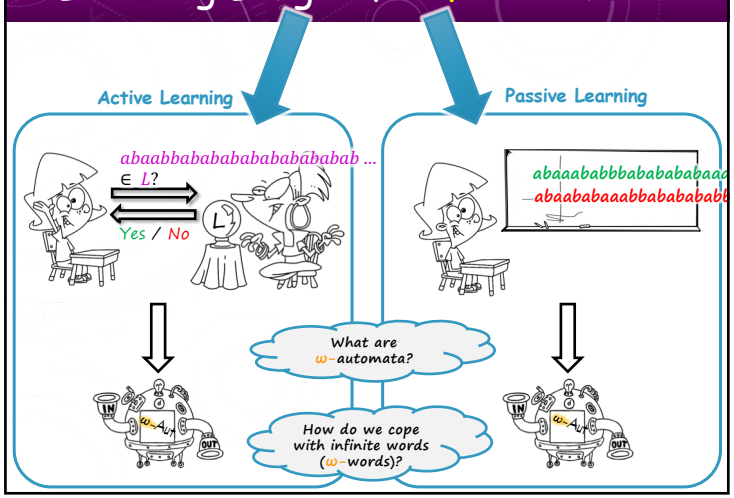
based on joint works with

Dana Angluin, Timos Antonopoulos, Udi Boker, Nevin George, Yaara Shoval

Learning Langs of Finite Words



Learning Langs of Infinite Words



Why infinite words?

- Languages of infinite words model reactive systems
- Verification questions on reactive systems can be modeled by reduction to question on automata on infinite words.

Reactive Systems

Transformational Systems

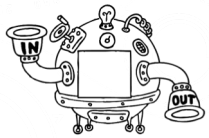
Processing …

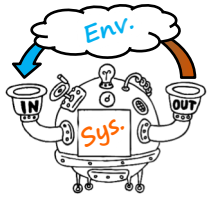
Input → Output

Reactive Systems

Processing …

Input → Output



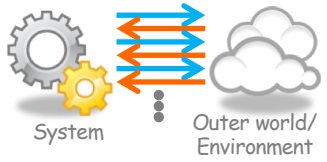


Reactive Systems

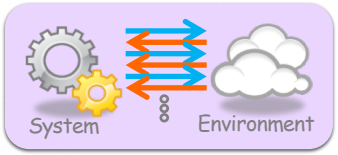
Reactive Systems

Processing …

Input → Output



Reactive Systems



Operating systems

Cyber-physical systems

discrete logic + continuous dynamics

Networking Systems

Memory Management System

Mail client

Medical devices

Web browser

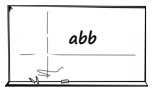
Executable biology

Smart phones

Talk Overview

ω-automata

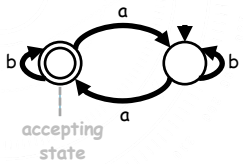
Learning Reg Langs



Learning of ω-reg langs

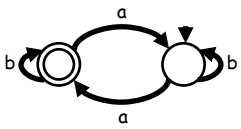
Automata

On words



A word is accepted if it the run on it ends in an accepting state.

On ω-words



A ω-word is accepted if it the run on it …

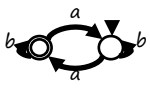
???

ω-Automata

ω-automaton $(\Sigma, S, s_0, \delta, \alpha)$

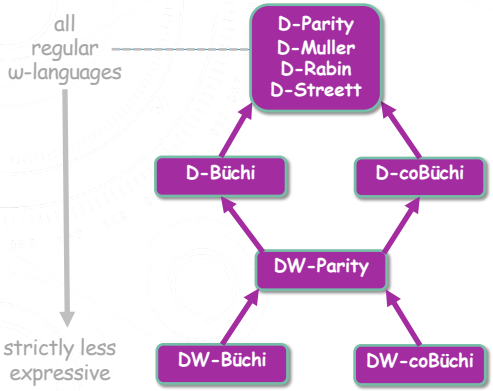
Alphabet, States, Transition Relation, Initial State, Acceptance Condition

- There are many ways to define acceptance condition for ω-Automata
  - Büchi
  - co-Büchi
  - Muller
  - Parity
  - Rabin
  - Streett
- All are defined using the notion of the states visited infinitely often during a run.

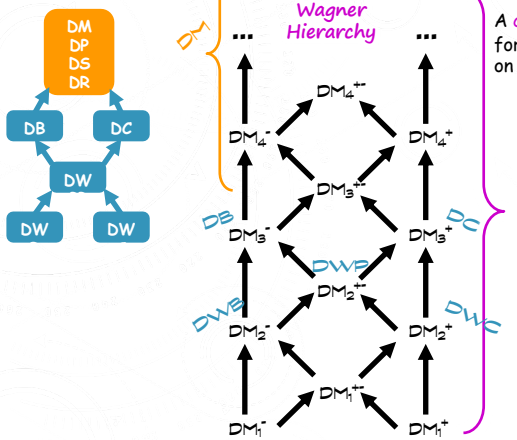


ω-automata - Expressiveness

- Some acceptance criteria are equally expressive, some are strictly less expressive than others.

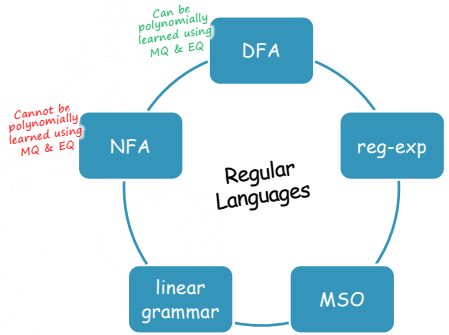


Hierarchy of regular ω-languages [Wagner 75]



A complexity measure for ω-languages defined on det. Muller aut. (DM)

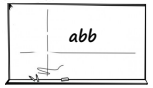
REPRESENTATION MATTERS



Talk Overview

ω-automata

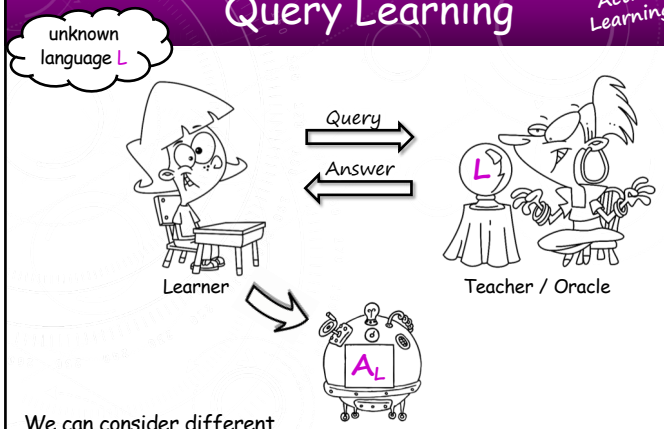
Query Learning of Reg Langs



[Angluin '87]

Query Learning

Active Learning

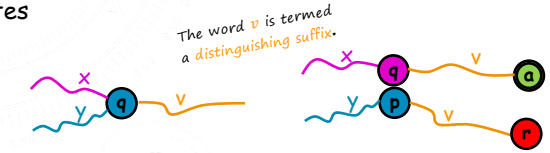


We can consider different types of queries, e.g. membership queries (MQ) or equivalence queries (EQ).

- Regular languages are polynomially learnable using MQ and EQ [L*, Angluin 87].
- Are not polynomially learnable using only MQ
- Are not polynomially learnable using only EQ

The idea behind many learning algs.

- Start with the smallest automaton adhering to data
- Add states when you realize you must
- If $xv \in L$ and $yv \notin L$ then x and y must lead to different states



- We say that $x \not\sim_L y$ (as defined next).
- The relation $\sim_L$ plays a key role in learning languages !!!

The Syntactic Right Congruence

Let $x, y \in \Sigma^*$.

Let $L \subseteq \Sigma^*$:

$x \sim_L y$ *iff* $\forall z \in \Sigma^*.\ xz \in L \iff yz \in L$

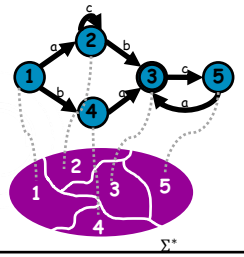
Let $L \subseteq \Sigma^\omega$:

$x \sim_L y$ *iff* $\forall z \in \Sigma^\omega.\ xz \in L \iff yz \in L$

Myhill-Nerode THM

For every regular language L the number of equivalence classes under $\sim_L$ is finite.

The minimal DFA has one state for every equivalence class of L.



The infinite matrix of a language

Let $\Sigma = \{a, b\}$, $L \subseteq \Sigma^*$:

Enumeration of all strings

L	ε	a	b	aa	ab	ba	bb	aaa	aab	aba	abb	baa
ε	1	0	0	1	1	0	0	0	1	0	1	0
a	0	1	0	0	1	0	1	0	1	1	0	1
b	0	1	0	1	1	0	1	1	1	1	0	1
aa	1	1	1	0	0	1	1	1	0	1	1	1
ab	1	0	0	1	1	0	0	0	1	0	1	0
ba	0	1	0	0	1	0	1	0	1	1	0	1
bb	0	0	1	0	1	1	0	0	1	0	1	1
aaa	0	1	1	0	1	1	0	1	1	0	1	1
aab	1	0	0	1	1	1	1	0	0	1	1	1
aba	0	1	1	0	0	0	0	1	1	0	0	0
abb	1	1	0	0	1	1	0	0	1	0	1	0
baa	0	0	1	1	0	0	1	0	0	1	1	0

$bb \notin L$
$aab \in L$

Enumeration of all strings

The infinite matrix of a language

Let $\Sigma = \{a, b\}$, $L \subseteq \Sigma^*$:

Enumeration of all strings

L	ε	a	b	aa	ab	ba	bb	aaa	aab	aba	abb	baa
ε	1	0	0	1	1	0	0	0	1	0	1	0
a	0	1	0	0	1	0	1	0	1	1	0	1
b	0	1	0	1	1	0	1	1	1	1	0	1
aa	1	1	1	0	0	1	1	1	0	1	1	1
ab	1	0	0	1	1	0	0	0	1	0	1	0
ba	0	1	0	0	1	0	1	0	1	1	0	1
bb	0	0	1	0	1	1	0	0	1	0	1	1
aaa	0	1	1	0	1	1	0	1	1	0	1	1
aab	1	0	0	1	1	1	1	0	0	1	1	1
aba	0	1	1	0	0	0	0	1	1	0	0	0
abb	1	1	0	0	1	1	0	0	1	0	1	0
baa	0	0	1	1	0	0	1	0	0	1	1	0

If $\varepsilon \sim_L ab$ then their rows must be the same!

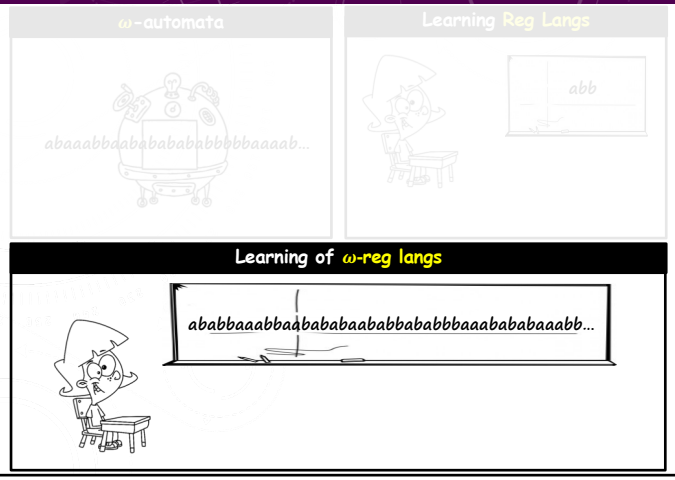
Enumeration of all strings

By Myhill-Nerode the number of distinct rows is finite.

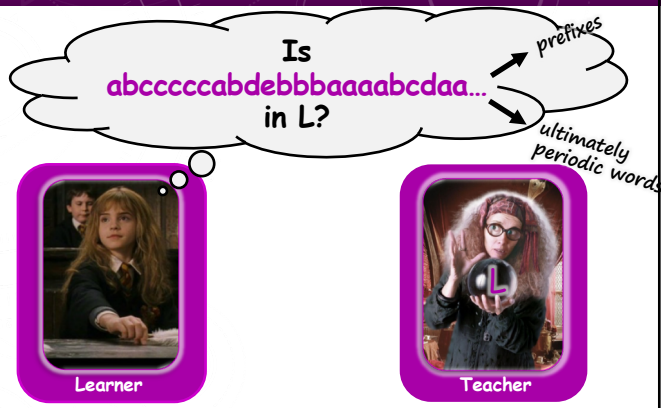
How does L* work?

- It maintains an observation table
- Where it tries to find all different rows of the infinite table.
- It asks MQ to gradually extends the table
- When the table meets certain criteria ("is closed") it is possible to extract from it an automaton and ask EQ.
- Smart processing of a counterexample to an EQ will reveal at least one new row to the table.
- If n is the number of states of the smallest DFA for L
- Then when the table is closed and has as n distinct rows the next EQ will be the smallest DFA.

Talk Overview



Coping with ω-words



Coping with ω-words

Coping with ω-words

Is
wingardium laviosa$^{\omega}$ in L?

wingardium laviosa laviosa laviosa laviosa laviosa ...

ultimately periodic words (Lasso words)

Learner

Teacher

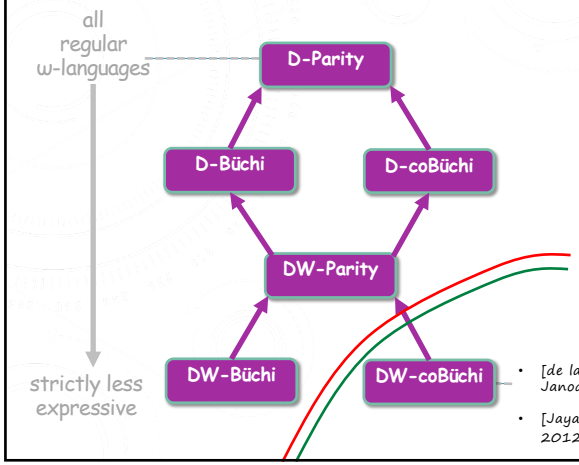
Coping with ω-words

Is
wingardium laviosa$^{\omega}$ in L?

ultimately periodic words (Lasso words)

THM:
Two regular ω-languages are equivalent
iff
they agree on the set of lasso words

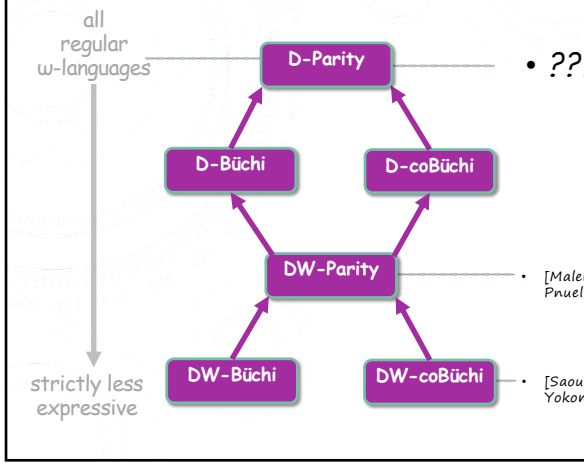
Learning ω-langs. from prefixes



- [de la Higuera & Janodet, 2004]
- [Jayasrirani et al, 2012]

35

Learning ω-langs. from lassos



- ???
- [Maler & Pnueli, 1995]
- [Saoudi & Yokomori, 1993]

36

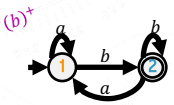
Difficulties

Regular vs. ω-Regular

The minimal DFA for a regular language L has one state for every equivalence class of $\sim_L$

For $x, y \in \Sigma^*$ we say that $x \sim_L y$ iff
$\forall z \in \Sigma^*. xz \in L \Leftrightarrow yz \in L$

$L = \Sigma^*(b)^+$

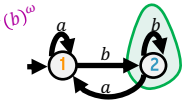


ε is a distinguishing suffix

The minimal DMA/DPA/... for an ω-regular lang. L may need more states than the eq. cls. of $\sim_L$

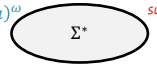
For $x, y \in \Sigma^*$ we say that $x \sim_L y$ iff
$\forall z \in \Sigma^\omega. xz \in L \Leftrightarrow yz \in L$

$L = \Sigma^*(b)^\omega$



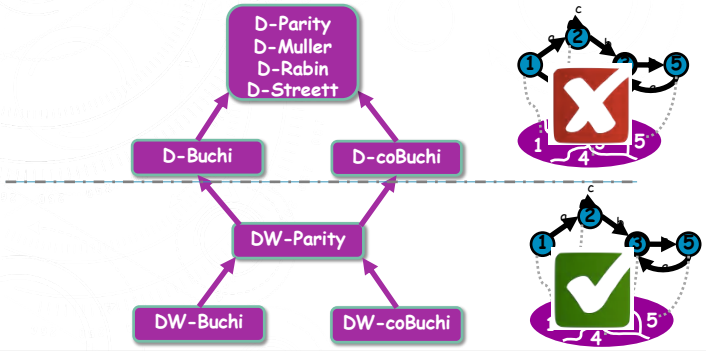
$x = ua$
$y = ub$
$z_1 = ba(b)^\omega$
$z_2 = baa(ba)^\omega$

No distinguishing suffixes!!!



Challenge

- Most learning algs. for reg langs work due to the Myhill-Nerode thm.
- The major difficulty in learning ω-reg languages is a lack of a corr. Myhill-Nerode theorem for non-weak ω-automata (of all types)



Ways Around #1

[Farzan et al. '08]

$L_\$$

- It follows from

> THM:
> Two regular ω-languages are equivalent
> iff
> they agree on the set of lasso words

that an ω-reg. lang. L can be represented by the reg. lang $L_\$ = \{ u\$v \mid uv^\omega \in L \}$ [Calbrix, Nivat & Podelski 1993]

- [Farzan et al. '08] proposed to learn N-Büchi using L* and a DFA for $L_\$$
- Unfortunately an N-Büchi with n states => DFA for $L_\$$ with $2^n + 3^{n^2}$ states [Kuperberg, Pinault & Pous 2019]

Ways Around #2

[Angluin & F. '14]

Finding a Myhill Nerode Rep.

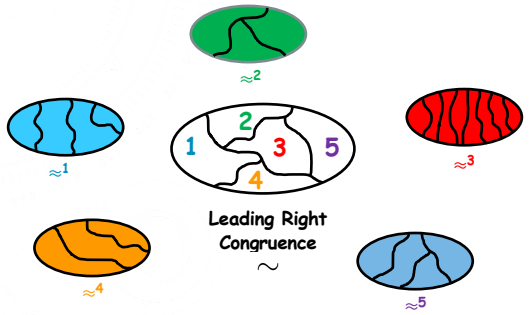
- Look for an alternative representation or reg-ω langs, that does have a Myhill-Nerode Theorem.
- In the quest for a representation for ω-langs for which a Myhill-Nerode THM exists [Maler and Staiger 1997] introduced

Family of Right Congruences (FORC)

Family of Right Congruences [MS97]

$(\sim, \approx_1, \approx_2, \approx_3, \approx_4, \approx_5)$

Leading — Progress

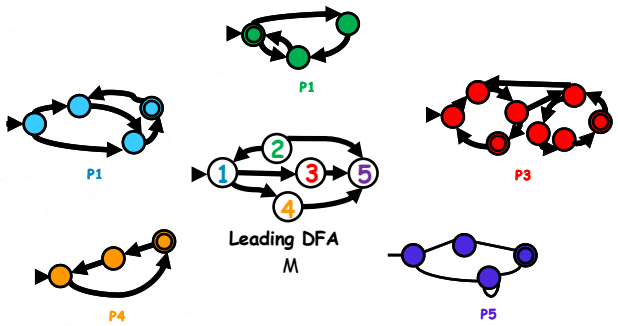


Plus some restrictions

Family of DFAs (FDFA) [AF14]

$(M, P_1, P_2, P_3, P_4, P_5)$

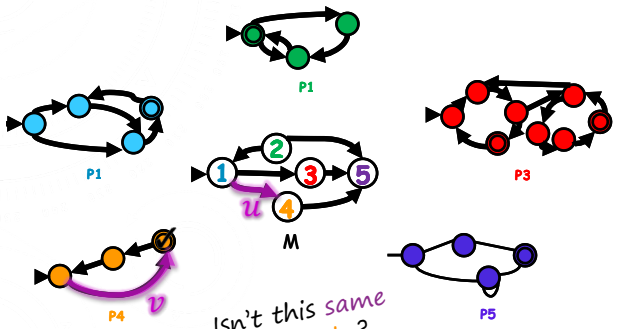
Leading | Progress



Restrictions removed

Periodic FDFA — FDFA Acceptance

$(u, v) \stackrel{?}{\in} (M, P_1, P_2, P_3, P_4, P_5)$

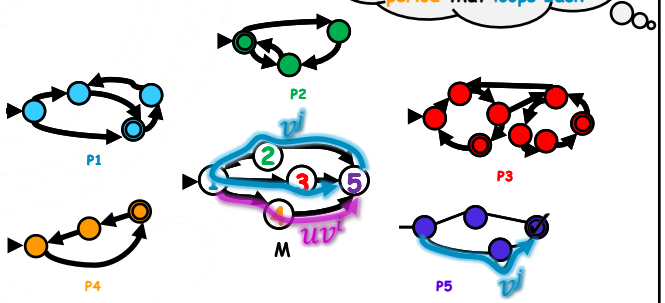


Isn't this same idea as $L_\$$?

Recurrent FDFA

$(u, v) \stackrel{?}{\in} (M, P_1, P_2, P_3, P_4, P_5)$

Normalization seeks for the smallest repetition of the period that loops back

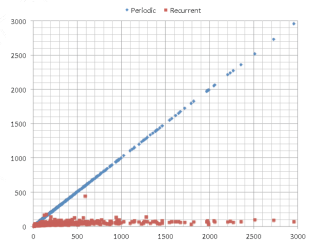


Recurrent FDFA can be **exponentially smaller** than Periodic FDFA ($L_\$$)!

Learning Result

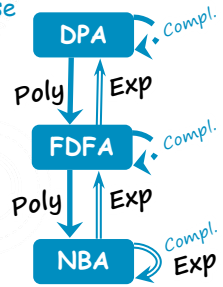
A learning algorithm L^ω that learns the full class of regular ω-languages using recurrent FDFAs

- Worst-case time complexity polynomial in $L_\$$
- Polynomial if the safety part is known
- Preforms very well on random targets



FDFAs as Acceptors of ω-Langs

- Have a Myhill-Nerode characterization
- Boolean operations are in LOGSPACE (union, intersection, complementation)
- Decision problems are in NLOGSPACE (emptiness, inclusion, equivalence)
- Succinctness-wise

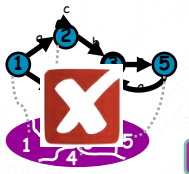


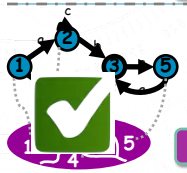
Ways Around #3

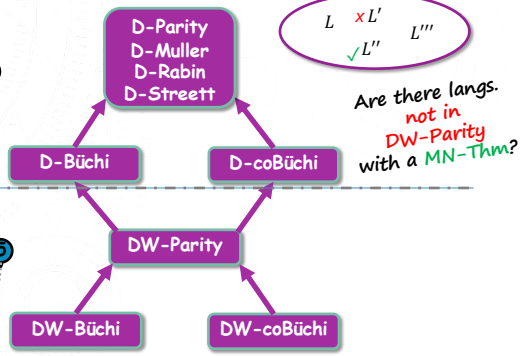
[Angluin & F. '18]
[Angluin, F. & Shoval '20]

Which Sub-Classes have a MN Thm?

What do we mean by this?







D-Parity

Are there langs. not in DW-Parity with a MN-Thm?

How informative is $\sim_L$?

If $\sim_L$ has one equivalence class we say it is zero-informative (or trivial)

If ∃ det. aut with 1-1 correspondence to $\sim_L$ we say it is fully-informative

Questions

- Are there other subsets of ω-regular langs. that have a fully informative right congruence?
- Can we fully characterize languages with a trivial right congruence?

Characterization Result [AF18]

- A regular ω-language L has a trivial right congr. iff L is prefix-independent.
- If L has a trivial right congr. then L is a liveness property, but the other direction is not necessarily true.
- The class of languages with a trivial right congr. is closed under intersection, union and complementation.
- A regular ω-language has a trivial right congr. iff
  $L = \Sigma^*(R_1{}^\omega + R_2{}^\omega + \ldots + R_k{}^\omega)$
  for some regular languages $R_1, R_2, \ldots, R_k$.

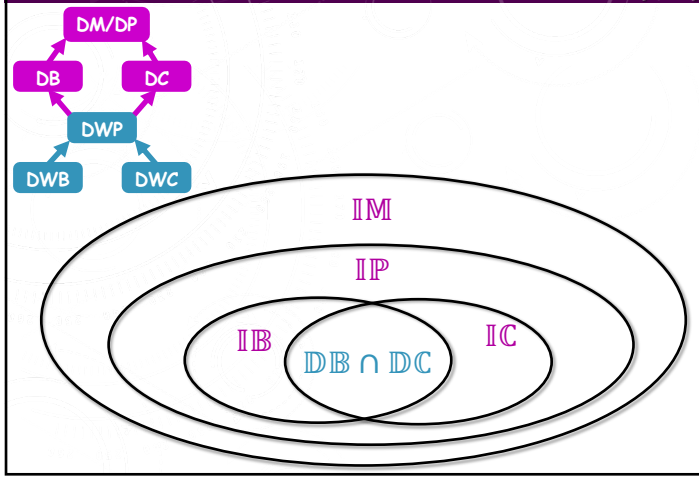
Notations

𝕀𝔹, 𝕀ℙ, 𝕀𝕄, 𝕀ℂ

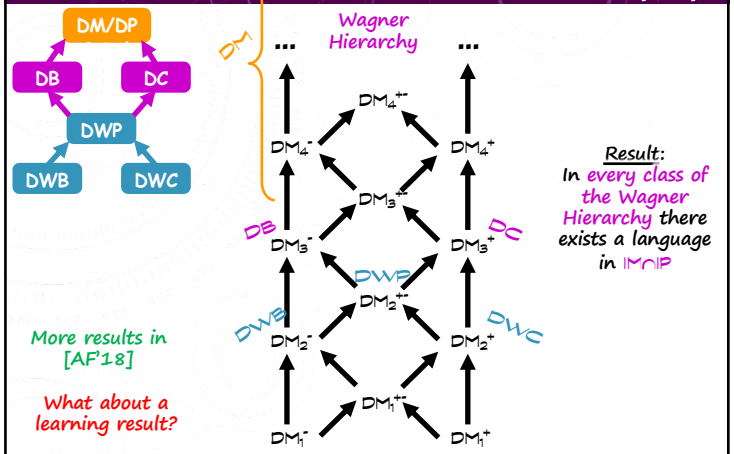
The "isomorphic classes"

- 𝕀𝔹 is the class of languages L where there is a 1-1 relation between $\sim_L$ and the number of states in the minimal DBW (Det. Buchi. Aut) for L
- 𝕀ℂ is … DCW (Det. coBuchi Aut)
- 𝕀ℙ is … DPW (Det. Parity Aut)
- 𝕀𝕄 is … DMW (Det. Muller Aut)

Expressiveness Result I [AF18]



Expressiveness Result II [AF'18]



More results in [AF'18]

What about a learning result?

Result: In every class of the Wagner Hierarchy there exists a language in IM∩IP

Passive Learning Informative Langs [AFS'20]

identifiable in the limit using polynomial time and data

Passive Learning

A class of acceptors $\mathbb{A}$ recognizing a class of languages $\mathbb{L}$ is

(I)

$W \subseteq \Sigma^* \times \{0,1\}$

Alg

Runs in polynomial time

A_W

A_W agrees with W

s.t.

W_L is termed a characteristic set for L

$W_L \subseteq W'$

Alg

A_L

A_L recognizes L !!!

(II)

L $\Rightarrow$ $W_L \subseteq \Sigma^* \times \{0,1\}$

$\forall L \in \mathbb{L}$

W_L agrees with L

W_L is of size polynomial in L

Results

Negative

Theorem

The classes $\mathbb{NBW}$ (Non-det Büchi), $\mathbb{NPW}$ (Non-det Parity), $\mathbb{NMW}$ (Non-det Muller), $\mathbb{NCW}$ (Non-det coBüchi) are **not** identifiable using polynomial time and data.

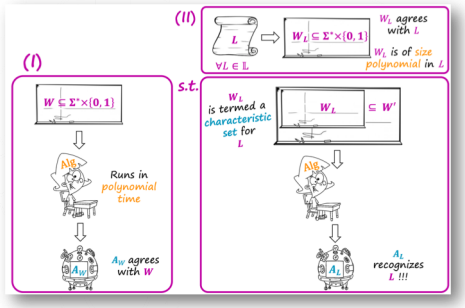
Positive

Theorem

The classes $\mathbb{IB}$ (Isomorphic Büchi), $\mathbb{IP}$ (Isomorphic Parity), $\mathbb{IM}$ (Isomorphic Muller), $\mathbb{IC}$ (Isomorphic coBüchi) **are** identifiable using polynomial time and data.

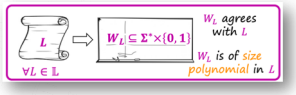
Polynomial Identification of $\mathbb{IB}, \ldots, \mathbb{IM}$

Goal:

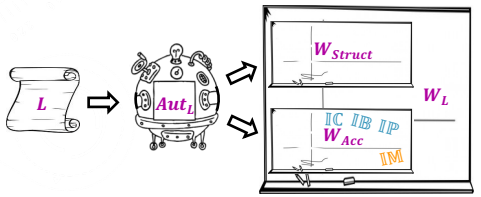


Constructing the characteristic set

Goal:



Scheme:



The Canonical Forest

W_{Acc} $\mathbb{IP}$

Theorem

Let $P = (\Sigma, Q, q_0, \delta, \kappa)$ be a DPA.

There exists a **canonical forest** $F^*(P)$ that is unique up to isomorphism

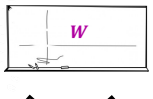
and has a lot of good properties

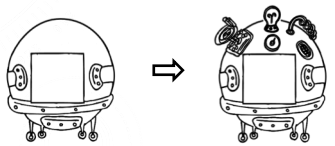
Can be computed in polynomial time

Inferring the automaton

Goal

Scheme



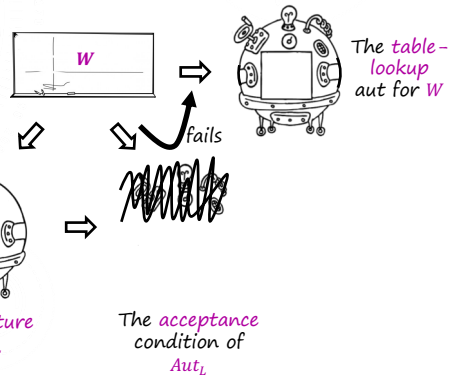


The structure of Aut_L

The acceptance condition of Aut_L

Inferring the automaton

Scheme



Active Learning Informative Langs

- As hard as learning the deterministic counterparts [Bohn & Lödding '21]

Theorem

There is a polynomial time active learning algorithm for $\mathbb{IX}$
iff
there is a polynomial time active learning algorithm for $\mathbb{DX}$

for $\mathbb{X} \in \{\mathbb{B}, \mathbb{C}, \mathbb{P}, \mathbb{M}\}$

- What about active learning of the classes $\mathbb{IB}, \mathbb{IC}, \mathbb{IP}, \mathbb{IM}$?

Ways Around #4

[Angluin, Antonopoulos & F. '20]
[Angluin, Antonopoulos, F. & George]

What about Non-Det Models?

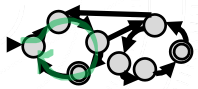
- **Q:** Why do we consider only deterministic models ?
- **A:** It is not hard to show that NBW are not polynomially predictable with MQs and EQs

> THM:
> Under standard cryptographic assumptions
> NBAs cannot be polynomially learned
> using MQs and EQs

- **Q:** Okay, so what about unambiguous Büchi Automata ?

Unambiguous Automata

DFA — Det. Finite Aut.



A single run on each word.

NFA — NonDet. Finite Aut.

Various runs on each word.

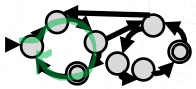
UFA — Unambiguous Finite Aut.

Various runs on each word.
At most one is accepting

Strongly Unambiguous Büchi Automata

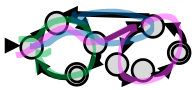
[CM03, BL10]

DBA — Det. Büchi



A single run on each word.

NBA — NonDet. Büchi



Various runs on each word.

UBA — Unambiguous Büchi

Various runs on each word.
At most one is initial & final, accepting

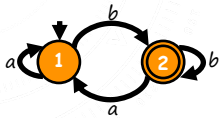
SUBA — Strongly Unambiguous Büchi

Various runs on each word.
At most one is ~~initial &~~ final.

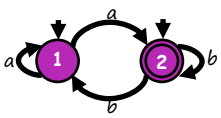
Strongly Unambiguous Büchi Automata

[CM03, BL10]

infinitely many b's



UBA



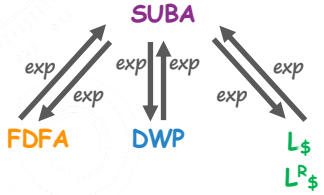
SUBA

Why SUBAs?

- Every ω-regular language can be accepted by a SUBA
- Equivalence and containment of SUBAs can be decided in polynomial time.
- SUBAs are **backward deterministic**
- Intuitively, one starts reading the word from somewhere in the loop and goes backward deterministically
- SUBAs may be exponentially smaller than DBAs and may be exponentially bigger than DMAs

SUBA - Succinctness

[CM03, BL10]



[AAF'20]

SUBA - Learning Results

The class of ω-regular languages is polynomially predictable with membership queries when the target language is represented by a SUBA

There is a polynomial time algorithm that learns all regular ω-languages, for target langs. represented by a SUBA using MQ and non-proper EQ

via Mod-2 Multiplicity Automata

Multiplicity Automata (MA)

- Multiplicity Automata (MA) are an algebraic generalization of automata, defined wrt. a field $\mathbb{F}$.
- They associate with every word a value from $\mathbb{F}$.

$f(w) = 3|w|_a + 7$

$$(\Sigma, v_I, \{\mu_\sigma\}_{\sigma\in\Sigma}, v_F)$$

Alphabet · Initial Vector · Transition Matrices · Final Vector

$w = aaba$

$$\begin{bmatrix} \vdots \\ v_I \\ \vdots \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \cdots & a & \cdots \\ & \mu_a & \\ \cdots & & \cdots \end{bmatrix} \begin{bmatrix} \cdots & a & \cdots \\ & \mu_a & \\ \cdots & & \cdots \end{bmatrix} \begin{bmatrix} \cdots & b & \cdots \\ & \mu_b & \\ \cdots & & \cdots \end{bmatrix} \begin{bmatrix} \cdots & a & \cdots \\ & \mu_a & \\ \cdots & & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ v_F \\ \vdots \end{bmatrix}^{\mathsf{T}}$$

$$\begin{bmatrix} \vdots \\ v_0 \\ \vdots \end{bmatrix} \quad \begin{bmatrix} \vdots \\ v_1 \\ \vdots \end{bmatrix} \quad \begin{bmatrix} \vdots \\ v_2 \\ \vdots \end{bmatrix} \quad \begin{bmatrix} \vdots \\ v_3 \\ \vdots \end{bmatrix} \quad \begin{bmatrix} \vdots \\ v_4 \\ \vdots \end{bmatrix} \quad c$$

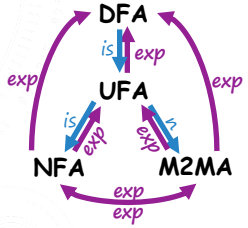
Mod-2 MA (M2MA)

- MA are Polynomially learnable with MQ and EQ [Beimel et al.'00]
- How can we use MA to represent languages?
- M2MA are MA over the field $GF(2) = \{0,1\}$ where sum and product are computed modulo 2.
- We can treat words associated with 1 as accepted, and words associated with 0 as rejected.

Advantages of M2MA

- M2MA are Polynomially learnable with MQ and EQ.
- A deterministic model.
- Complementation incurs only an additive blowup.
- Union and intersection incurs only a quadratic blowup.
- Equivalence can be decided in polynomial time.

Size Relations



Worst-case size bounds

	DFA	UFA	M2MA	NFA	(G)SUBA	NBA
LTL	$2^{2^{O(n)}}$ via UFA	$2^{O(n)}$ [AFFG]	←	←	$(2^n, n)$ [AFFG]	$n2^n$ [BK'08]
	→	→	$2^{\Omega(n)}$ [AFFG]	$2^{\Omega(n)}$ [AFFG]	→	$2^{\Omega(n)}$ [BK'08]
DBA	$n + n3^{n^2}$ [KPP'19]	←	←	←	↓	–
	$2^{\Omega(n \log n)}$ [AF'16]	→	$2^{\Omega(n)}$ [AFFG]	$2^{\Omega(n)}$ [AFFG]	$2^{\Omega(n)}$ [BL'10]	–
NBA	$2^n + 2^n 3^{n^2}$ [KPP'19]	←	←	$n + n3^{n^2}$ [KPP'19]	$(12n)^n$ [CM16]	–
	↑	↑	↑	↑	↑	–
SUBA	↑	$n + 2n^2$ [BL'10]	←	←	–	–
	$2^{\Omega(n)}$ [AAF'20]	→	$\Omega(n^2)$ [AFFG]	$\Omega(n^2)$ [AFFG]	–	–

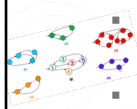
Upper Bound

Lower Bound

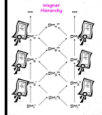
Worst-case size bounds

	DFA	UFA	M2MA	NFA	(G)SUBA	NBA
LTL	$2^{2^{O(n)}}$ via UFA →	$2^{O(n)}$ [AFFG] →	← $2^{\Omega(n)}$ [AFFG]	← $2^{\Omega(n)}$ [AFFG]	$(2^n, n)$ [AFFG] →	$n2^n$ [BK'08] $2^{\Omega(n)}$ [BK'08]
DBA	$n + n3^{n^2}$ [KPP'19] $2^{\Omega(n \log n)}$ [AF'16]	← →	← $2^{\Omega(n)}$ [AFFG]	← $2^{\Omega(n)}$ [AFFG]	↓ $2^{\Omega(n)}$ [BL'10]	– –
NBA	$2^n + 2^n 3^{n^2}$ [KPP'19] ↑	← ↑	← ↑	$n + n3^{n^2}$ [KPP'19] ↑	$(12n)^n$ [CM16] ↑	– –
SUBA	↑ $2^{\Omega(n)}$ [AAF'20]	$n + 2n^2$ [BL'10] →	← $\Omega(n^2)$ [AFFG]	← $\Omega(n^2)$ [AFFG]	– –	– –

Recap



- ω-regular languages can be learned using MQ and EQ via FDFAs

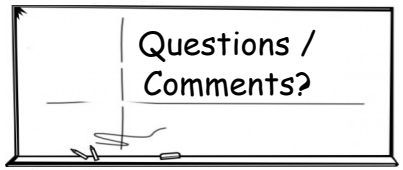


- If the safety part of the languages is known, then the liveness part can be polynomially learned
- There are arbitrary complex ω-regular language with a fully informative right congruence

- These languages can be learned in the limit using polynomial time and data
- Actively learning them is not easier than their non-fully informative counterparts

- ω-regular languages can be polynomially learned using MQ and EQ via SUBAs/M2MAs

Thank you for listening!



Thanks to Guy Ofek for his help with the graphics!