Inferring Regular Languages & ω-Languages

Dana Fisman

based on joint works with

Dana Angluin, Udi Boker & Sarah Eisenstat

Synthesis

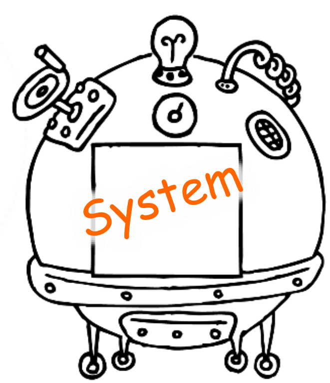
Challenges:

- *Hard to characterize using a logical calculous*
- *Complete bugless spec, really!?*

Specification
High Level
What?
Declarative
Ex: temporal logic

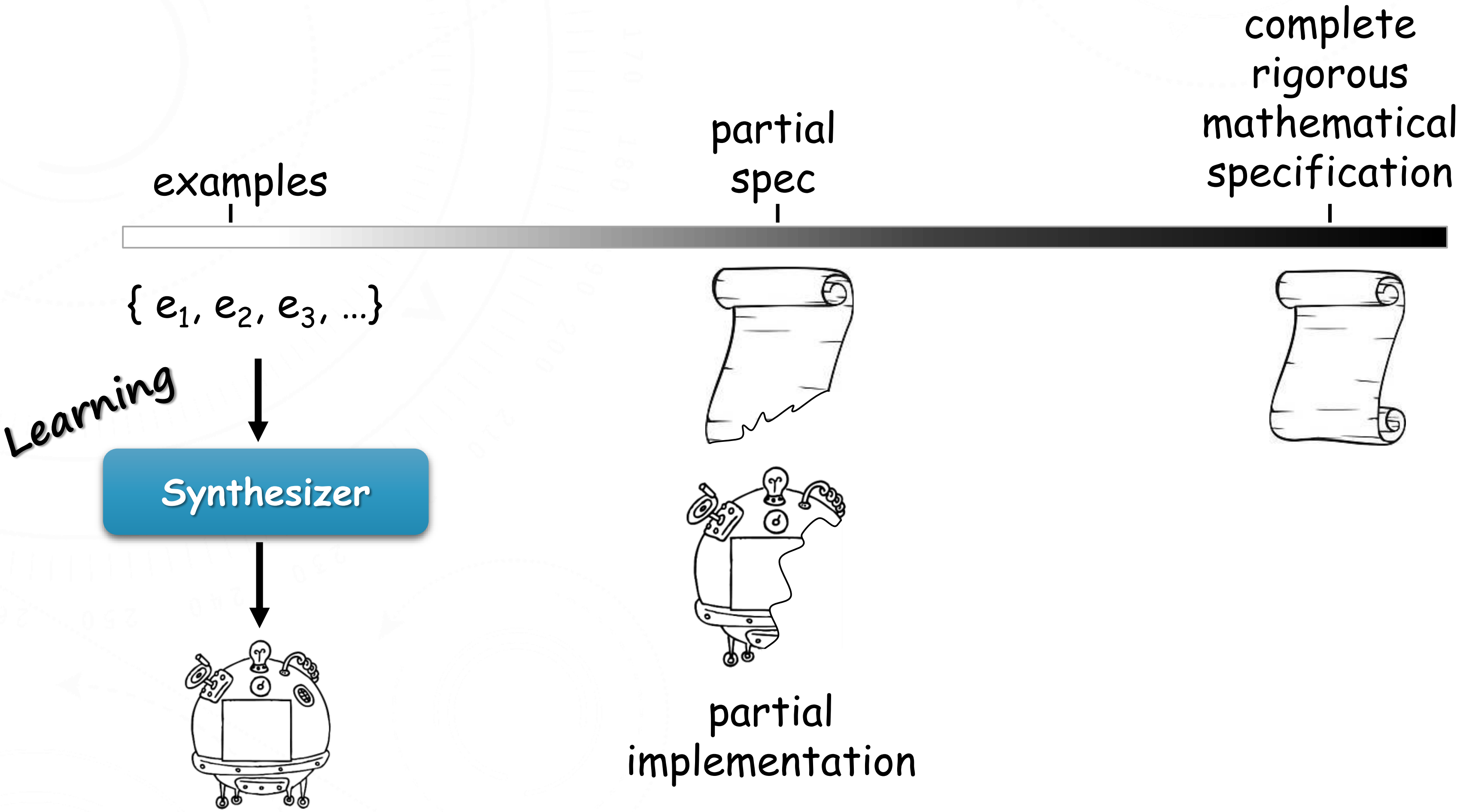
Synthesizer

Correct by construction



Implementation
Low Level
How?
Procedural/Executable
Ex: reactive system

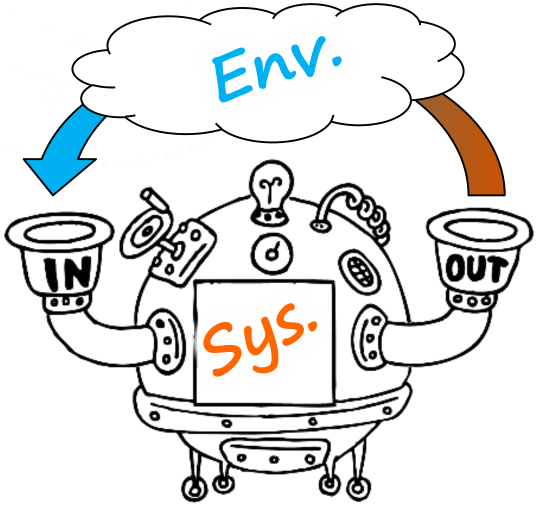
A specification scale



What kind of examples?

In the context of synthesizing reactive systems:

- The examples are words / strings describing computations / interfaces
- The learned concept is a set of such examples, presumably a regular language.
- For regular languages [Angluin, 1987] suggested L* algorithm.
- L* learns in polynomial time an unknown regular language using membership and equivalence queries.



L* - Active Learning with MQ and EQ

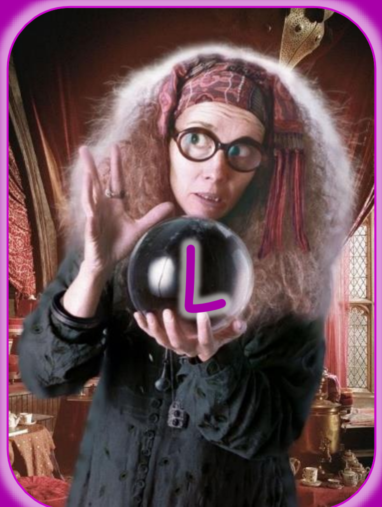
Learner

Is w in L ?

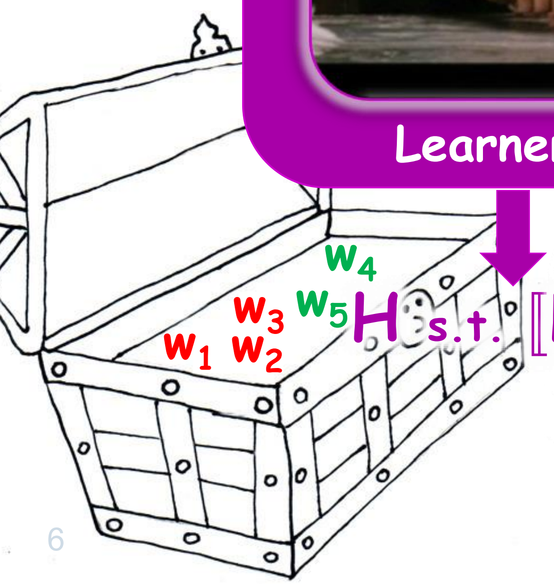
Yes / No

Is $[\![H]\!]$ same as L ?

Yes / No, c.e: w'



Teacher



w_4

w_3 w_5

w_1 w_2

H s.t. $[\![H]\!]=L$

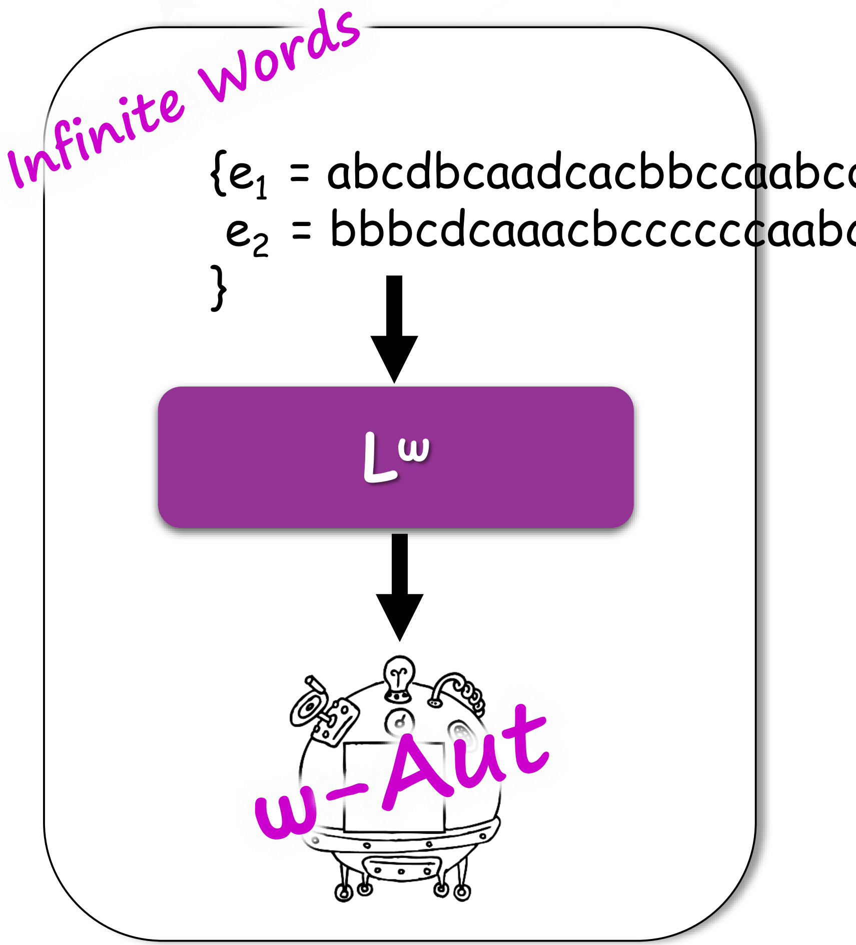
Usages of L*

- L* is an extremely popular algorithm.
  It has applications in many areas including AI, neural networks, geometry, data mining, verification and synthesis.

- Usages of L* in verification and synthesis include:
  - Black-box checking [Peled et al.]
  - Assume-guarantee reasoning [Cobleigh et al.]
  - Specification mining [Ammons et al., Gabel et al., …]
  - Error localization [Chapman et al.]
  - Learning interfaces [Alur et al.]
  - Regular Model Checking [Habermehl & Vonjar]
  - …

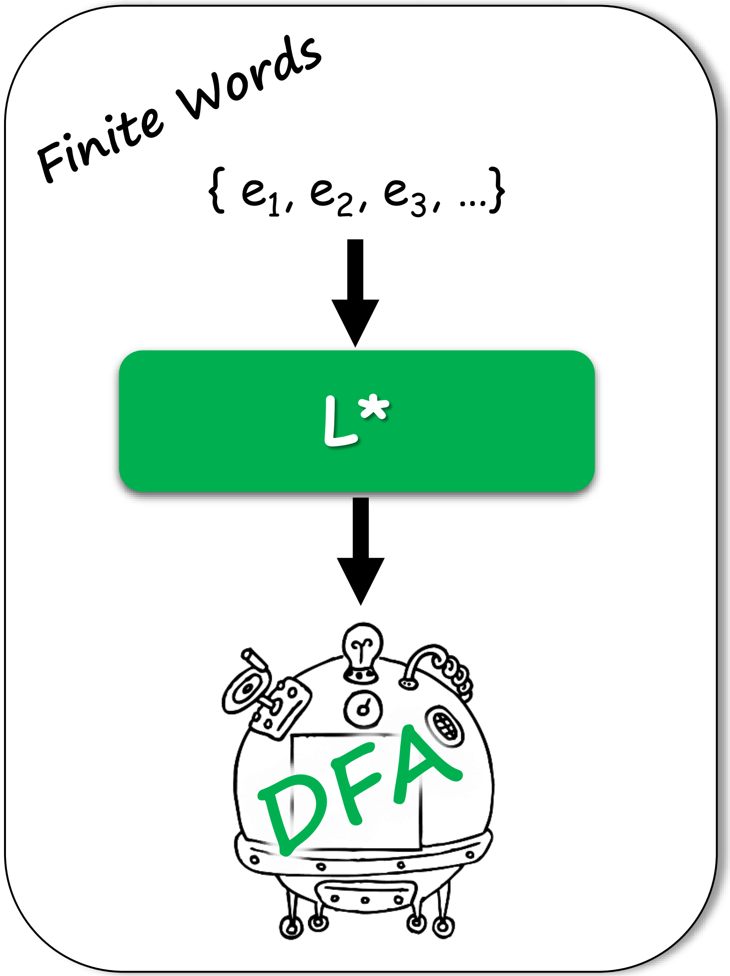
Challenge 1

- L* learns a regular language of finite words. Interesting properties of reactive systems e.g. (liveness and fairness) are not expressible by finite words.
- Can we extend L* to L^ω, an alg. that learns regular languages of infinite words (ω-words)?



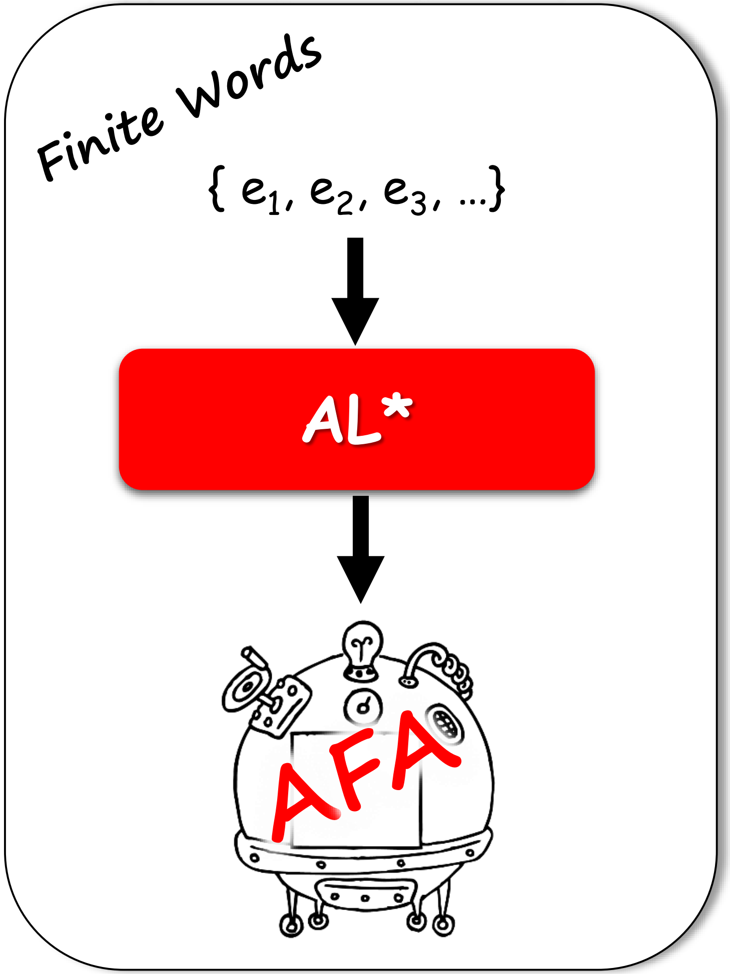
Challenge 2

- L* produces DFAs (deterministic finite automata), a well behaved representation, yet not a compact one.
- Can we learn more succinct representations, such as non-deterministic finite automata (NFA) or alternating automata (AFA)?

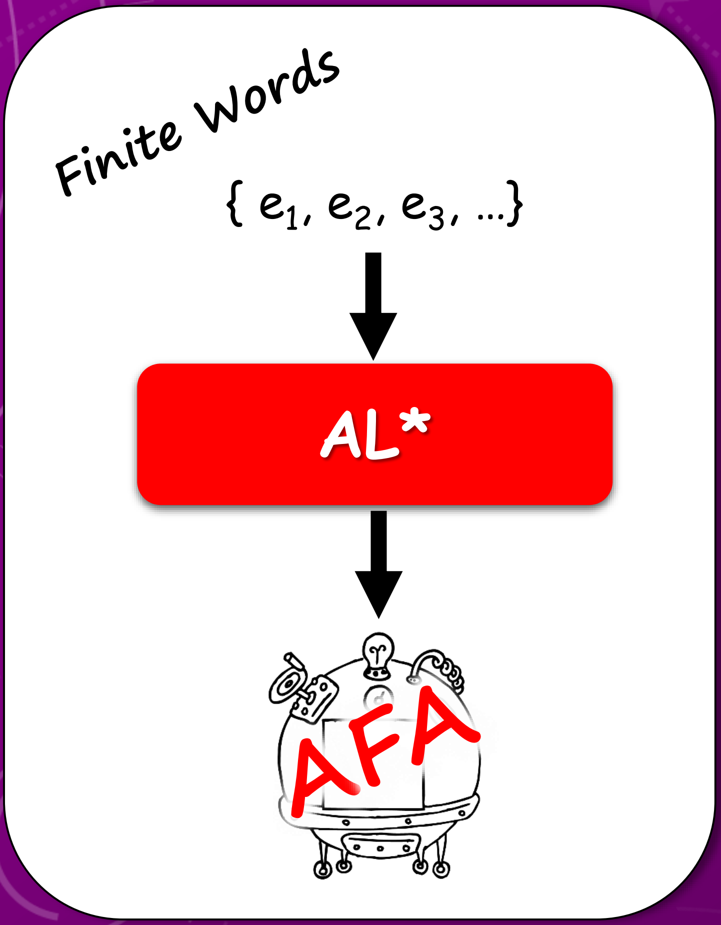


Challenge 2

- L* produces DFAs (deterministic finite automata), a well behaved representation, yet not a compact one.
- Can we learn more succinct representations, such as non-deterministic finite automata (NFA) or alternating automata (AFA)?

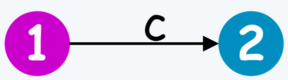


Learning alternating automata



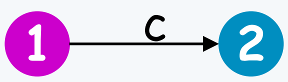
[Angluin, Eisenstat & Fisman IJCAI'15]

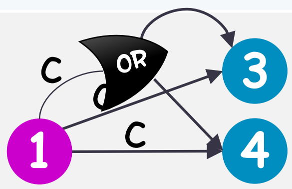
What are alternating automata?

Transition Type	from state	upon read-ing	to state(s)
Deterministic	s1	c	s2 

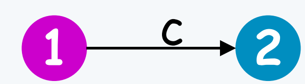
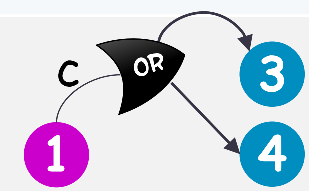
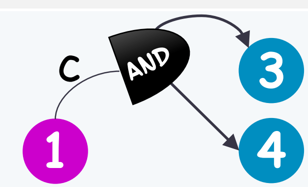
What are alternating automata?

Transition Type	from state	upon read-ing	to state(s)
Deterministic	s1	c	s2
Non-Deterministic	s1	c	s3 or s4

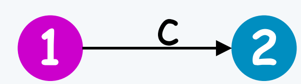
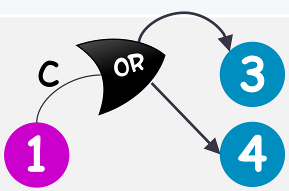
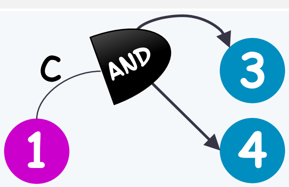
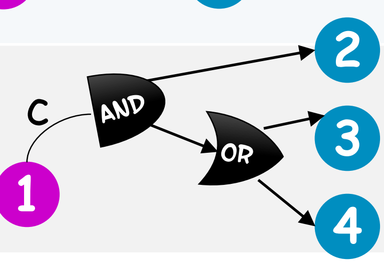




What are alternating automata?

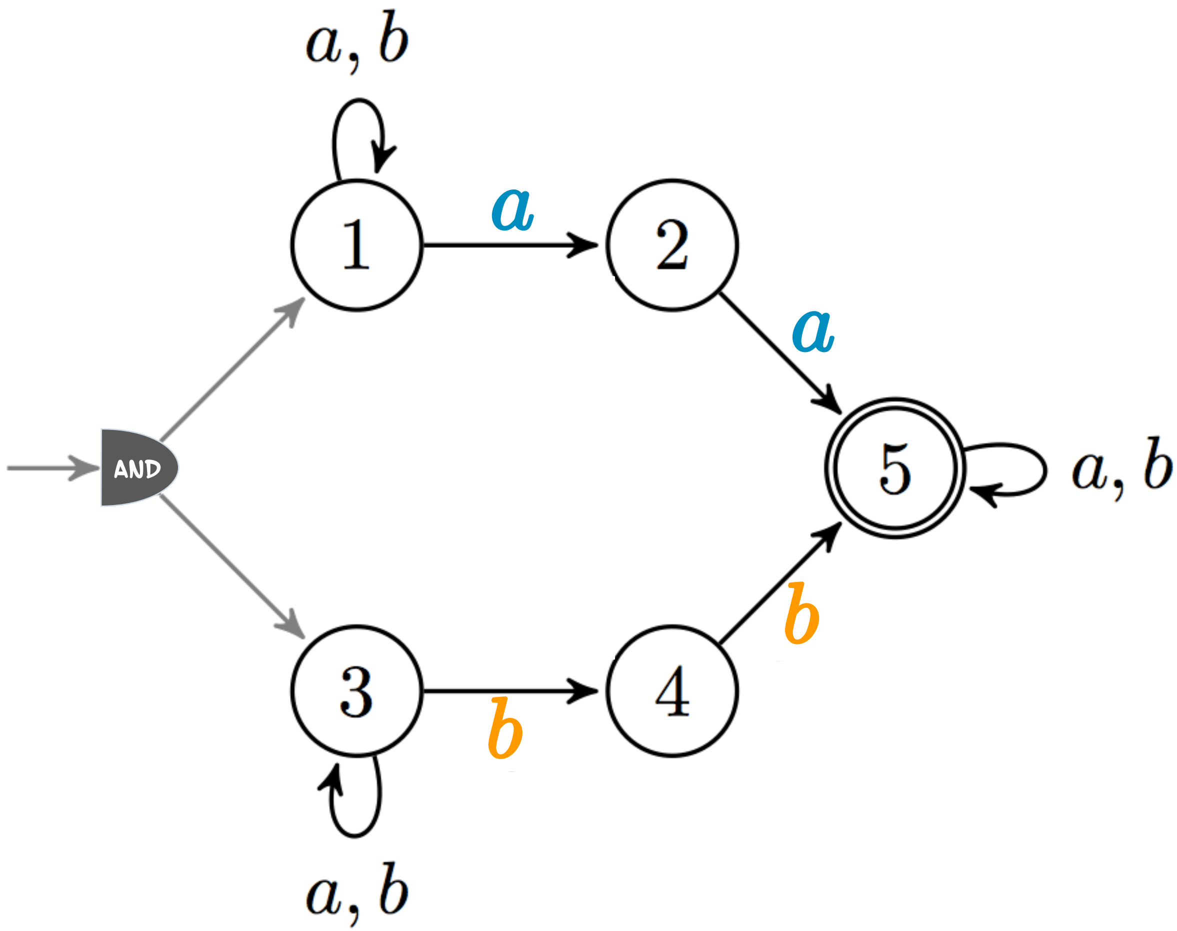
Transition Type	from state	upon read-ing	to state(s)
Deterministic	s1	c	s2 
Non-Deterministic	s1	c	s3 or s4 
Universal	s1	c	s3 and s4 

What are alternating automata?

Transition Type	from state	upon read-ing	to state(s)
Deterministic	s1	c	s2 
Non-Deterministic	s1	c	s3 or s4 
Universal	s1	c	s3 and s4 
Alternating	s1	c	(s3 or s4) and s2 

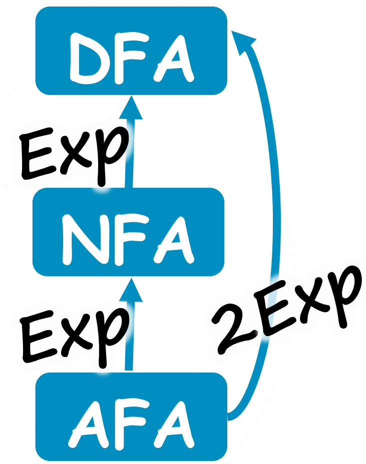
Alternating Automaton - Ex.

$\Sigma = \{a,b\}$



Accepts the language $\Sigma^* aa \Sigma^* \cap \Sigma^* bb \Sigma^*$

What are they good for?



- AFAs are a succinct representation
- The PSL formula

```
always (print-to-both ->
          ([*], print-a-start, busy[*3..], print-a-end) &
          ([*], print-b-start, busy[*3..], print-b-end))
```

can be stated by a 12 state AFA but the minimal DFA requires 115 states.

- Natural means to model conjunctions and disjunctions as well as existential and universal quantification
- 1-to-1 translations from temporal logics
- Working at the alternating level enables better structured algorithms, and is the common practice in industry verification tools.

Foundation of L* - Residuality

The residual of language L with respect to word u is the set of all words v such that uv in L

$$u^{-1}L = \{\, v \mid uv \in L \,\}$$

Example

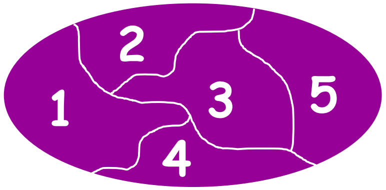
$L = aba^*$

$a^{-1}L = ba^*$
$ab^{-1}L = a^*$
$abaaa^{-1}L = a^*$
$b^{-1}L = \varnothing$

If $u^{-1}L = v^{-1}L$ we say that $u \sim_L v$.

$ab \sim_L abaaa$

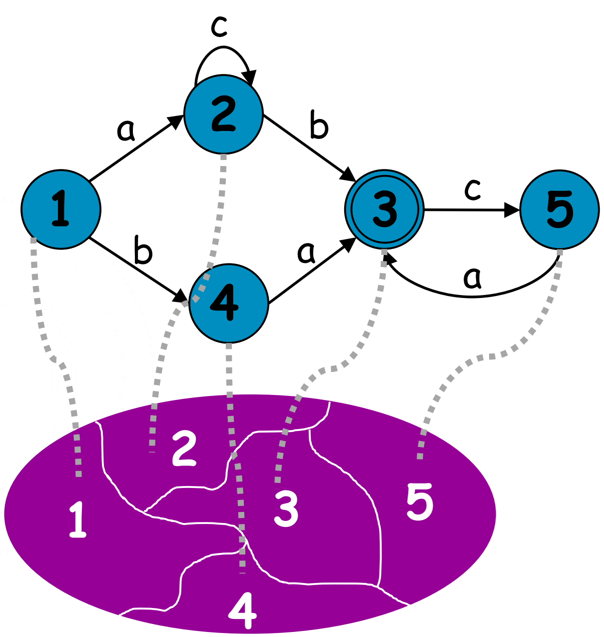
The residuality index is the number of equivalence classes of $\sim_L$



Myhill-Nerode THM

Every regular language L has a finite number of residual languages.

The minimal DFA has one state for every residual language of L !!!



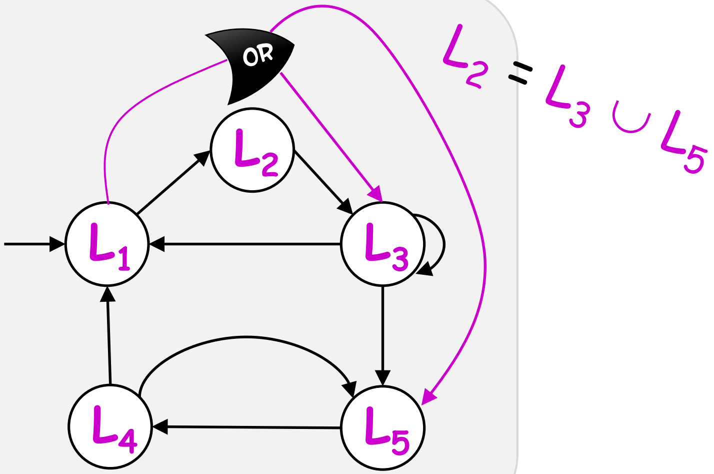
Challenge

NFAs and AFAs don't have the residually property, in general.

Residual NFAs

- Dennis et al. [STACS' 01] defined residual NFAs (NRFA)
- These are NFAs where each state corresponds to a residual language

Suppose $L_1, L_2, \ldots, L_n$ are all the residual languages of L
If for some L_i, we have
$$L_i = L_j \cup L_k$$
then we can remove the i^{th} state, and use non-determinism to capture it.



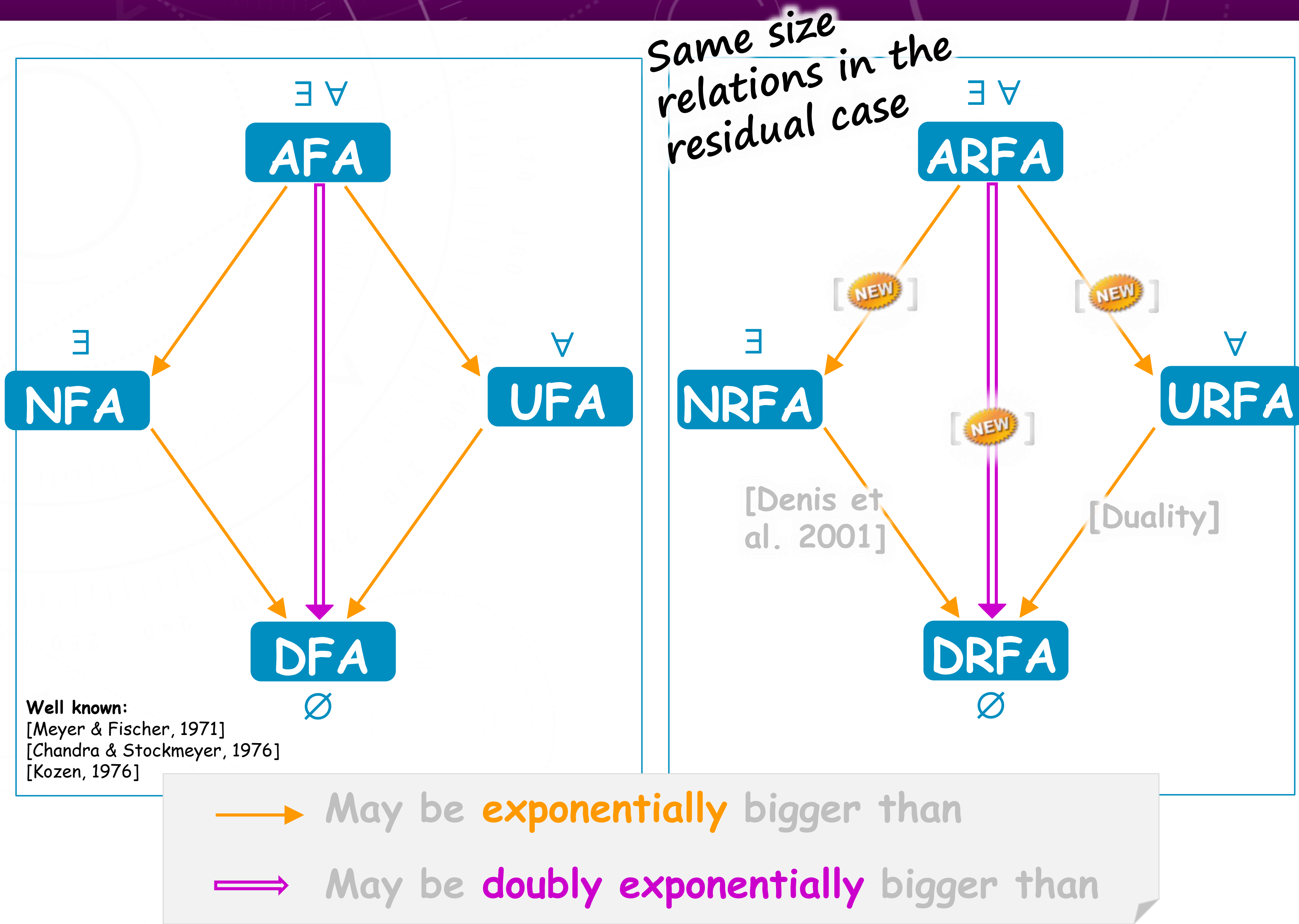
Residual NFAs

- Dennis et al. showed/provided
  - Every regular language is recognized by a unique (canonical) NRFA which has a minimal number of states and a maximal number of transitions.
  - There may be exponential gaps between the minimal DFA, the canonical NRFA and the minimal NFA.
- Bollig et al. [IJCAI'09] extended L* to NL* (learns NRFA)

Questions

- Can we extend the notion of residually to AFAs?
- Will exponential gaps remain?
- Can we define a canonical one?
- Can we learn ARFAs?

Succinctness



The learning algorithm

- L* uses a data structure termed an observation table.
- AL* generalizes NL* and L* and the notion of a complete/minimal observation table.
- As shown next…

The infinite table of residual languages

Enumeration of all strings

ε a b aa ab ab bb aaa aab aba abb baa

		0	0	1	1	0	0	0	1	0	1	0	0	1	0
			0	0	1	0	1	0	1	1	0	1	0	0	1
		1	0	1	1	0	1	1	1	1	0	1	1	1	1
aa	1	1	1	0	0	1	1	1	0	1	1	1	0	0	0
ab	1	0	0	1	1	0	0	0	1	0	1	0	0	1	0
	0	1	0	0	1	0	1	0	1	1	0	1	0	0	1
		0	1	0	1	1	0	0	1	0	1	1	0	0	1
			1	0	1	1	0	1	1	0	1	1	1	1	1
	1	0	0	1	1	1	1	0	0	1	1	1	0	0	0
aba	0	1	1	0	0	0	0	1	1	0	0	0	1	0	1
abb	1	1	0	0	1	1	0	0	1	0	1	0	1	1	0
baa	0	0	1	1	0	0	1	0	0	1	1	0	1	1	0
bab	1	0	0	1	0	1	0	1	0	1	1	1	0	0	1

all the suffixes of *ab* that are in *L*
i.e. $ab^{-1}L$

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

The *infinite* table of residual languages

The number of distinct columns is also finite.

We call it the column index.

Enumeration of all strings

a b aa ab ab bb aaa aab aba abb baa

Enumeration of all strings

	1	0	0	1	1	0	0	0	1	0	1	0	0	1	0
	0	1	0	0	1	0	1	0	1	1	0	1	0	0	1
b	0	1	0	1	1	0	1	1	1	1	0	1	1	1	1
aa	1	1	1	0	0	1	1	1	0	1	1	1	0	0	0
ab	1	0	0	1	1	0	0	0	1	0	1	0	0	1	0
ba	0	1	0	0	1	0	1	0	1	1	0	1	0	0	1
bb	0	0	1	0	1	1	0	0	1	0	1	1	0	0	1
aaa	0	1	1	0	1	1	0	1	1	0	1	1	1	1	1
aab	1	0	0	1	1	1	1	0	0	1	1	1	0	0	0
aba	0	1	1	0	0	0	0	1	1	0	0	0	1	0	1
abb	1	1	0	0	1	1	0	0	1	0	1	0	1	1	0
baa	0	0	1	1	0	0	1	0	0	1	1	0	1	1	0
bab	1	0	0	1	0	1	0	1	0	1	1	1	0	0	1

L* Data Structure

An Observation Table:

Strings: experiments to distinguish states

Strings: candidate state represent-atives

M	e_1	e_2	e_3	e_4	e_5	e_6	...
s_1	1	0	0	1	1		
s_2	0	1	0	0	1		
s_3	0	1	0	1	1		
s_4	1	1	1	0	0		
s_5	1	0	0	1	1		
$\vdots$							

$$M_{i,j} = \begin{cases} 1 & \text{if } s_i e_j \in L \\ 0 & \text{otherwise} \end{cases}$$

Closed Table

An observation table T = (S,E,M)
is closed w.r.t a subset B $\subseteq$ S

S	e_1	e_2	e_3	e_4	e_5	e_6	
ε	1	0	0	1	1		
a	0	1	0	0	1		
b	1	0	0	1	1		covered
ab	1	0	0	1	1		covered
aa	1	1	1	0	0		
aaa	1	0	0	1	1		covered
aab	0	1	0	0	1		covered

B

If it satisfies

1) Initialization: $\varepsilon \in$ B
2) Consecution: B$\Sigma \subseteq$ S
3) Coverage: all rows not in B are covered by some row in B

The definition of covers differs for L*, NL* and AL*.

D-Covered

According to **L***
i.e. when
using **DFA**s

S	e_1	e_2	e_3	e_4	e_5	e_6
ε	1	0	0	1	1	
a	0	1	0	0	1	
b	0	1	0	0	1	
ab	1	0	0	1	1	
aa	1	1	1	0	0	
aaa	1	0	0	1	1	
aab	0	1	0	0	1	

Same
Same
Same
Same

N-Covered

According to **NL***
i.e. when
using **NFAs**

S	e_1	e_2	e_3	e_4	e_5	e_6
ε	1	0	0	1	1	
a	0	1	0	0	1	
b	1	1	0	1	1	
ab	1	1	1	0	1	
aa	1	1	1	0	0	
aaa	1	0	0	1	1	
aab	0	1	0	0	1	

Expressible as bitwise-or of some rows in B

$b = (\varepsilon \vee a)$

A-Covered

According to **AL***
i.e. when
using **AFA**s

S	e_1	e_2	e_3	e_4	e_5	e_6
ε	1	0	0	1	1	
a	0	1	0	0	1	
b	0	0	0	0	1	
ab	1	1	1	0	1	
aa	1	1	1	0	0	
aaa	1	0	0	1	1	
aab	0	1	0	0	1	

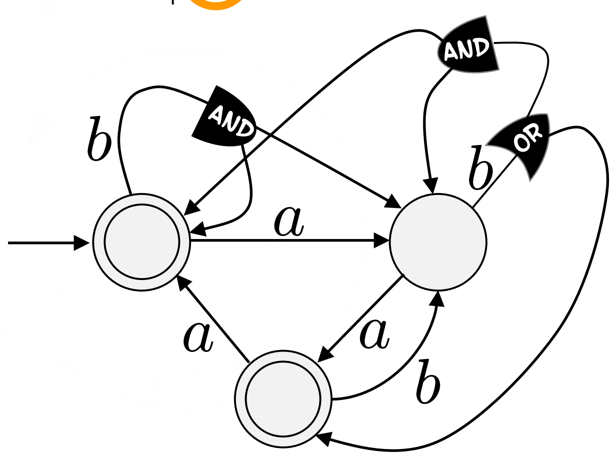
Expressible as
a monotone combination
of some rows in B

$$b = (\varepsilon \wedge a)$$
$$ab = (\varepsilon \wedge a) \vee aa$$

From Tables to Automata

Closed and Minimal

S	ε	e_2	e_3	e_4	e_5	e_6	
ε	1 ✓	0	0	1	1		
a	0	1	0	0	1		
b	0	0	0	0	1		$= (\varepsilon \wedge a)$
ab	1	1	1	0	1		$= (\varepsilon \wedge a) \vee aa$
aa	1 ✓	1	1	0	0		
aaa	1	0	0	1	1		$= \varepsilon$
aab	0	1	0	0	1		$= a$



Need to solve

- How to decide
  - Is row s a union of rows in B? Poly time [Bollig et al.]
  - Is s a monotone combination of rows in B? Poly time [NEW]

- Given a set of Boolean vectors S, find a **minimal**

unique union basis Poly time [Bollig et al.]

Not unique monotone basis NP-complete [NEW]

No obvious canonical rep. for ARFAs

Let
$S = \{0,1\}^3$

Then both
$B_1 = \{001,010,101\}$
$B_2 = \{110,101,011\}$
are minimal monotone bases.

The Learning Alg.

Algorithm 1: xL^* for x $\in \{$D, N, U, A$\}$

oracles : MQ, EQ
members: Observation table $\mathcal{T} = (S, E, M)$,
Candidate states set P
methods : *Is*x*Closed*, *Is*x*Minimal*, x
In$\mathbb{B}_{\text{X}}$, x*ExtractAut*
$S = \langle\epsilon\rangle, E = \langle\epsilon\rangle, P = \langle\epsilon\rangle$ and M_ϵ
repeat
$(a_1,$
if a_1
else
$($ $)$
if $a_2 =$ "*no*" **then**
P.*RemoveString*(s_2)
else
$\mathcal{A} = \mathcal{T}$.x*ExtractAut*$(P)$
$(a_3, s_3) = \text{EQ}(\mathcal{A})$
if $a_3 =$ "*no*" **then**
$\mathcal{T}$.x*Find&AddCols*(s_3)
$=$ "*yes*"
$\mathcal{A}$

Start with
ta

If the table is
not closed, e.g.
s_1 is missing,
th
U
is

If the table is
not minimal,
e.g. s_2 is
redundant, then

Start with basis:
$\{\varepsilon\}$

THM: Every
counterexample
yields at least one
new column

Ask an equivalence query.

If true, return.
Otherwise, use the given counterexample to find some columns to add, and add them.

Back to finite words

Theorem

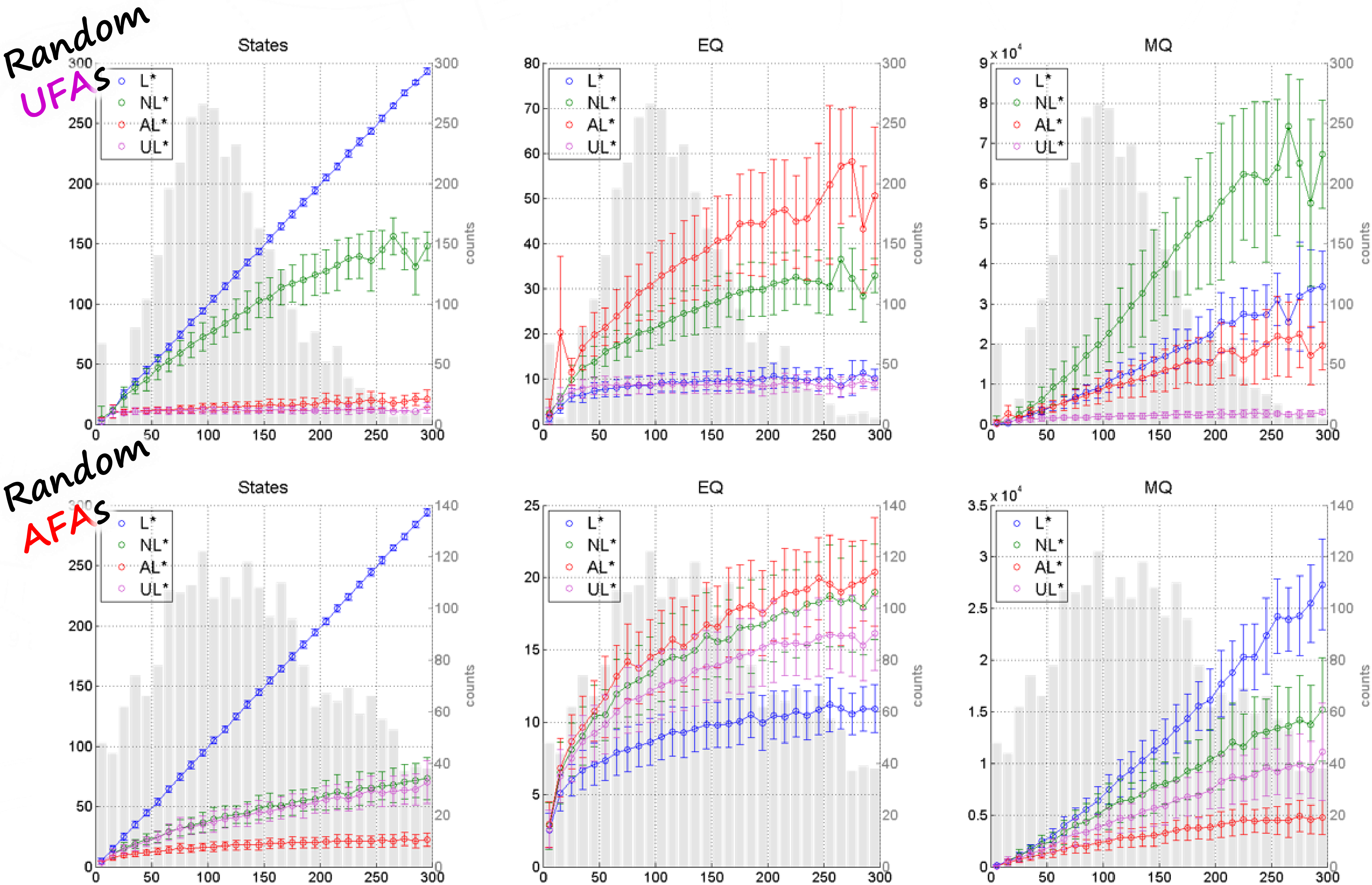
The algorithm AL* returns an AFA for the unknown language after at most

- m equivalence queries
- $O(|\Sigma|mnc)$ membership queries
- poly(m, n, c, $|\Sigma|$) time

	L*	NL*	AL*
EQ	n	$O(n^2)$	m
MQ	$O(\|\Sigma\|cn^2)$	$O(\|\Sigma\|cn^3)$	$O(\|\Sigma\|cnm)$

where
n = row index
m = column index
c = length of longest c.e.

Finite words - Empirical results



Finite words - Empirical results

Rough Summary:

- In terms of #states generated,

  AL* is always preferable

- In terms of #MQ,

  xL* outperforms the others when targets are xFAs

- In terms of #EQ,

  L* is always preferable

Open questions & further directions

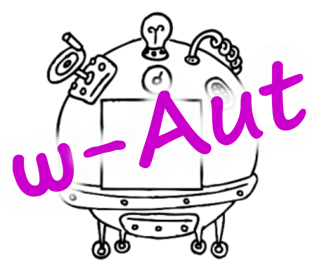
- Generalization to Boolean Automata ($\wedge\vee\neg$)
- Heuristics combining xL* s
- Understanding of Residual AFAs
  - Properties of ARFAs
  - **Theorem**: The algorithm AL* returns an AFA for the unknown language
  - **Conjecture**: The algorithm AL* returns an ARFA for the unknown language

Learning regular ω-languages

Infinite Words

$\{e_1$ = abcdbcaadcacbbccaabcdaaabbbccdddeeaaaabab
e_2 = bbbcdcaaacbcccccaabcdababababababaccabab
$\}$

L^ω



[Angluin & Fisman ALT'14]

Coping with ω-words

Is
abcccccabdebbbaaaabcdaa...
in L?

prefixes

ultimately periodic words (Lasso words)

Learner

Teacher

Coping with ω-words

Is

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

ultimately periodic words (Lasso words)

Learner

Teacher

Coping with ω-words

Is

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

$\underbrace{\text{wingardium laviosa}^{\omega}}_{\text{wingardium laviosa laviosa laviosa laviosa laviosa} \ldots}$

ultimately periodic words (Lasso words)

Learner

Teacher

Coping with ω-words

Is

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

ultimately
perio...
...ords
...ords)

THM:

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

ω-automata

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

Alphabet

States

Initial State

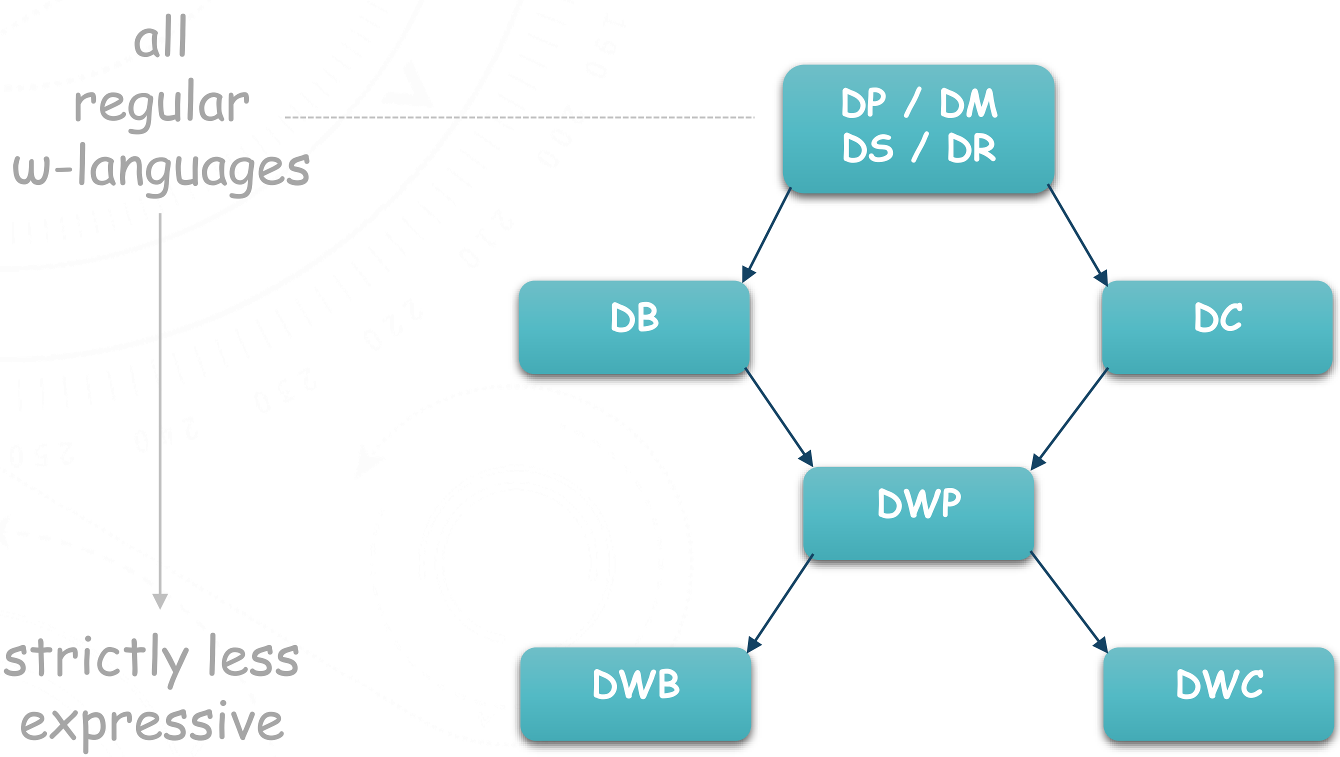
Transition Relation

Acceptance Condition

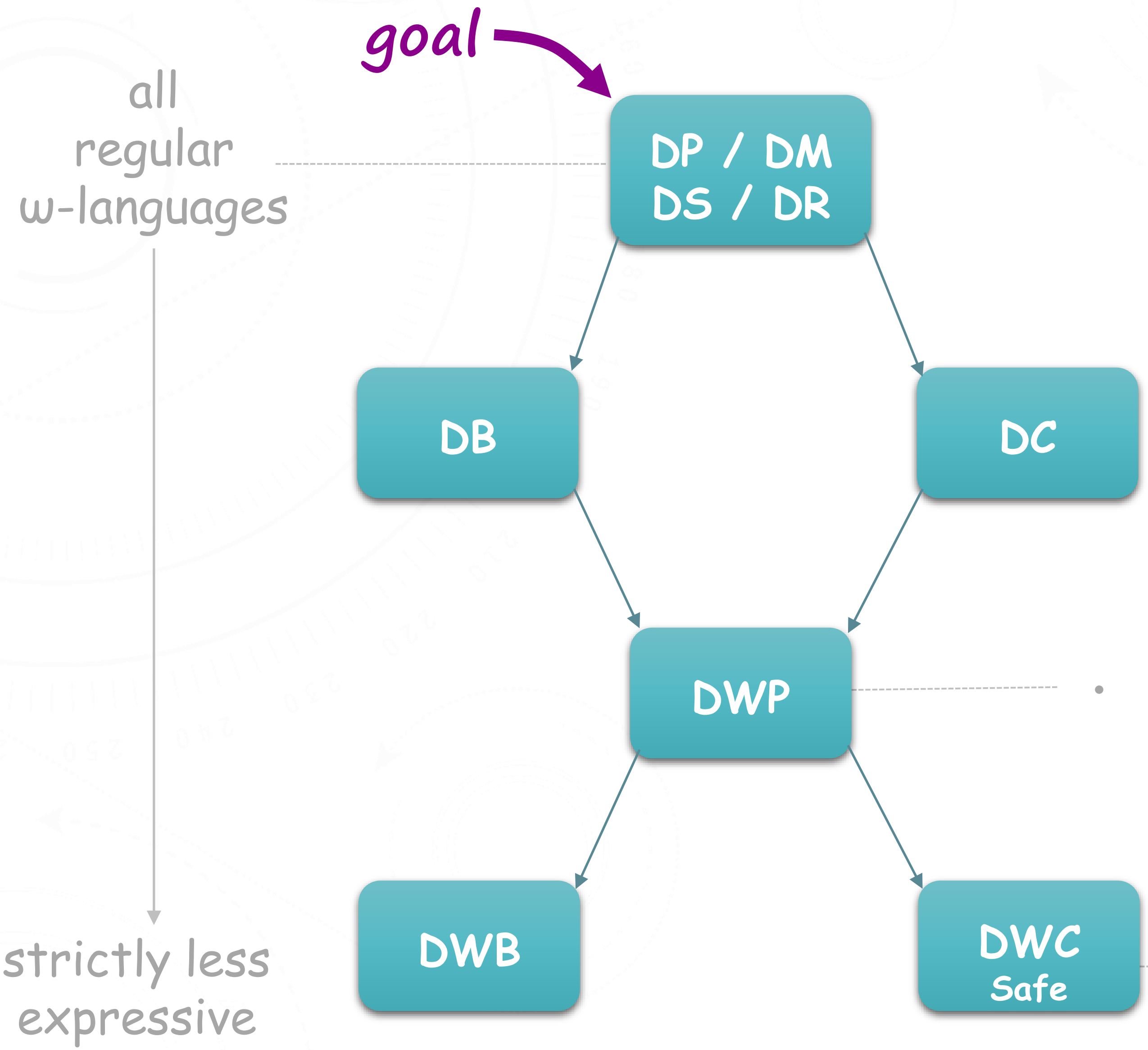
- There are many ways to define acceptance condition for ω-Automata
  - Büchi
  - Muller
  - Rabin
  - co-Büchi
  - Parity
  - Streett
- Roughly speaking, all are defined using the notion of the states visited infinitely often during a run.

ω-automata - Expressiveness

- Some acceptance criteria are equally expressive, some are strictly less expressive than others.
- Overall picture looks like this:



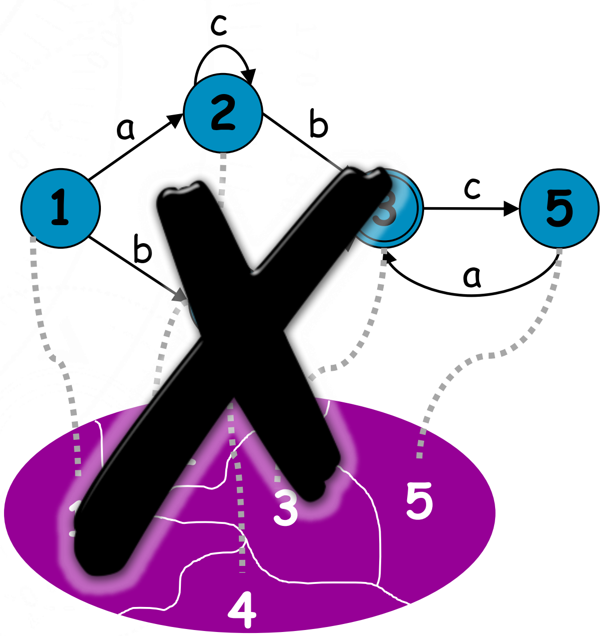
Previous work on learning ω-langs.



- [Maler & Pnueli, 1995] } From Lassos
- [de la Higuera & Janodet, 2004]
- [Jayasrirani et al, 2012] } From Prefixes
- [Saoudi & Yokomori, 1993] } From Lassos

Challenges

- L* works due to the Myhill-Nerode thm.
- The major difficulty in learning ω-languages is a lack of a corresponding Myhill-Nerode theorem for ω-automata (of all types)



Challenges

- It turns out that an ω-regular language can be represented by a regular language $L_\$$ of finite words [Calbrix, Nivat, Podelski 93]
- And thus one can use L* to learn this representation [Farzan et al. 2008]
- However, this representation is quite big: Büchi with n states => DFA for $L_\$$ with $2^n + 2^{2n^2+n}$

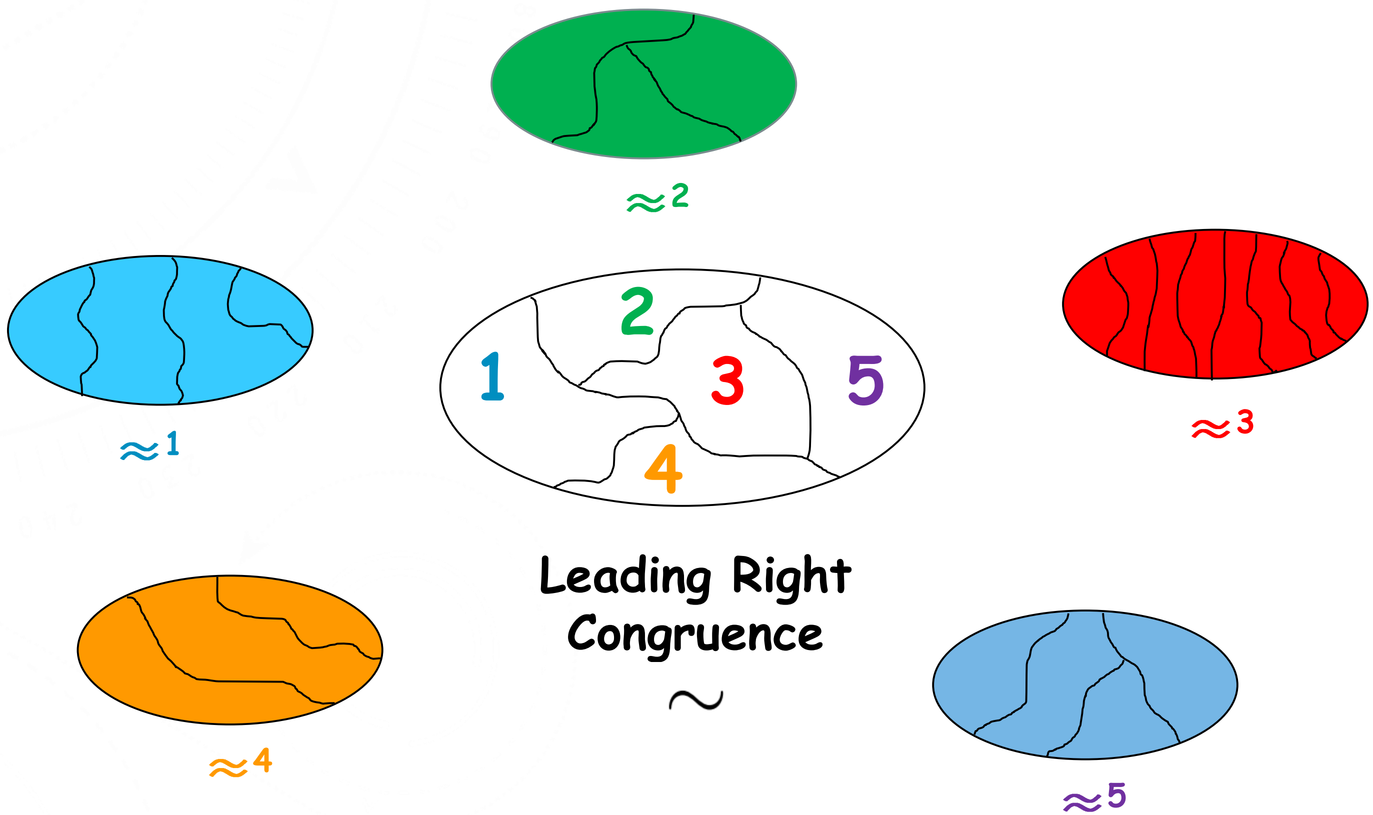
2014
2012
2008
2005
1994
1993
1987 L*
1962 ω-aut.

The way out

A new representation: Family of DFAs and a new canonical rep Recurrent FDFAs based on families of FORCs [Maler & Staiger, 95] and the syntactic FORC which has a Myhill-Nerode theorem

Family of Right Congruences [MS97]

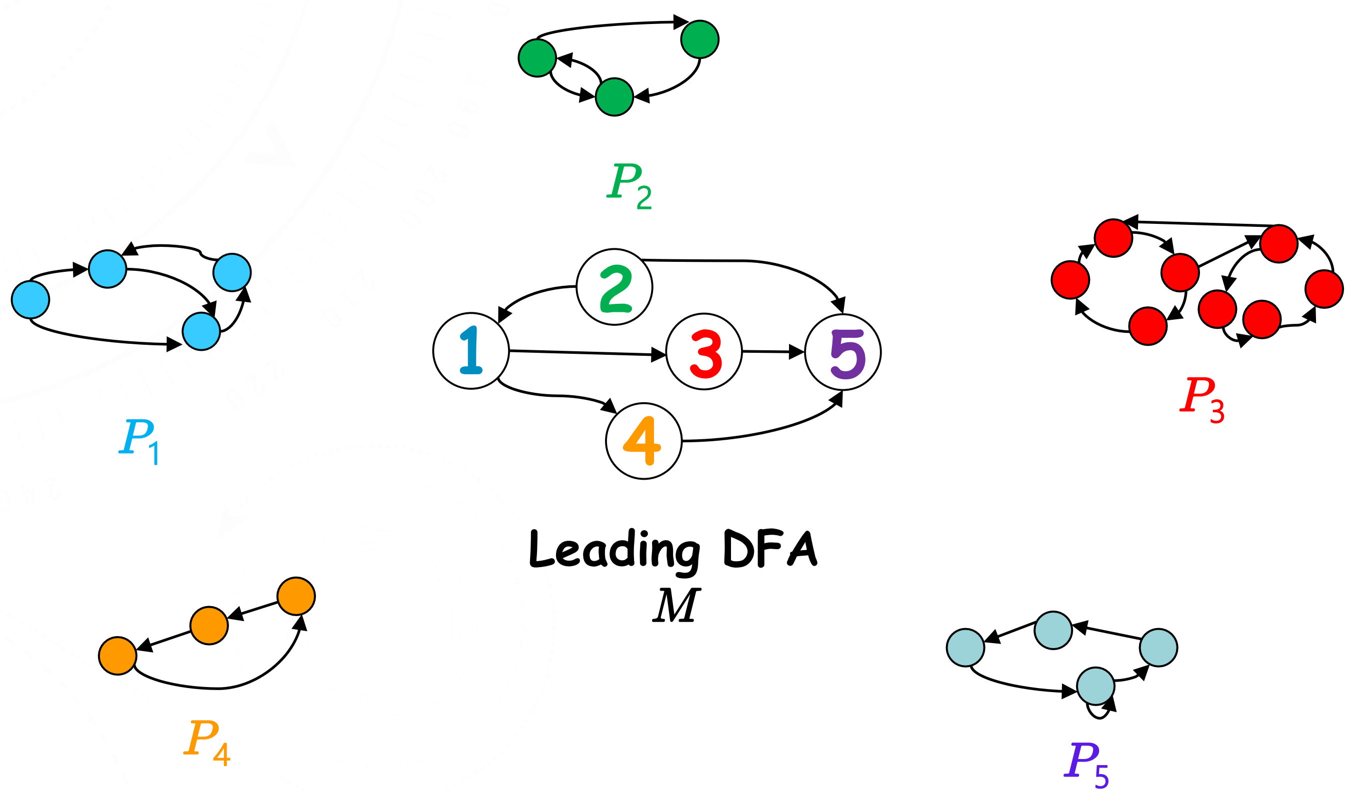
$$(\underbrace{\sim}_{\text{Leading}},\underbrace{\approx_1,\approx_2,\approx_3,\approx_4,\approx_5}_{\text{Progress}})$$



Plus some restriction (details omitted)

Family of DFAs (FDFA)

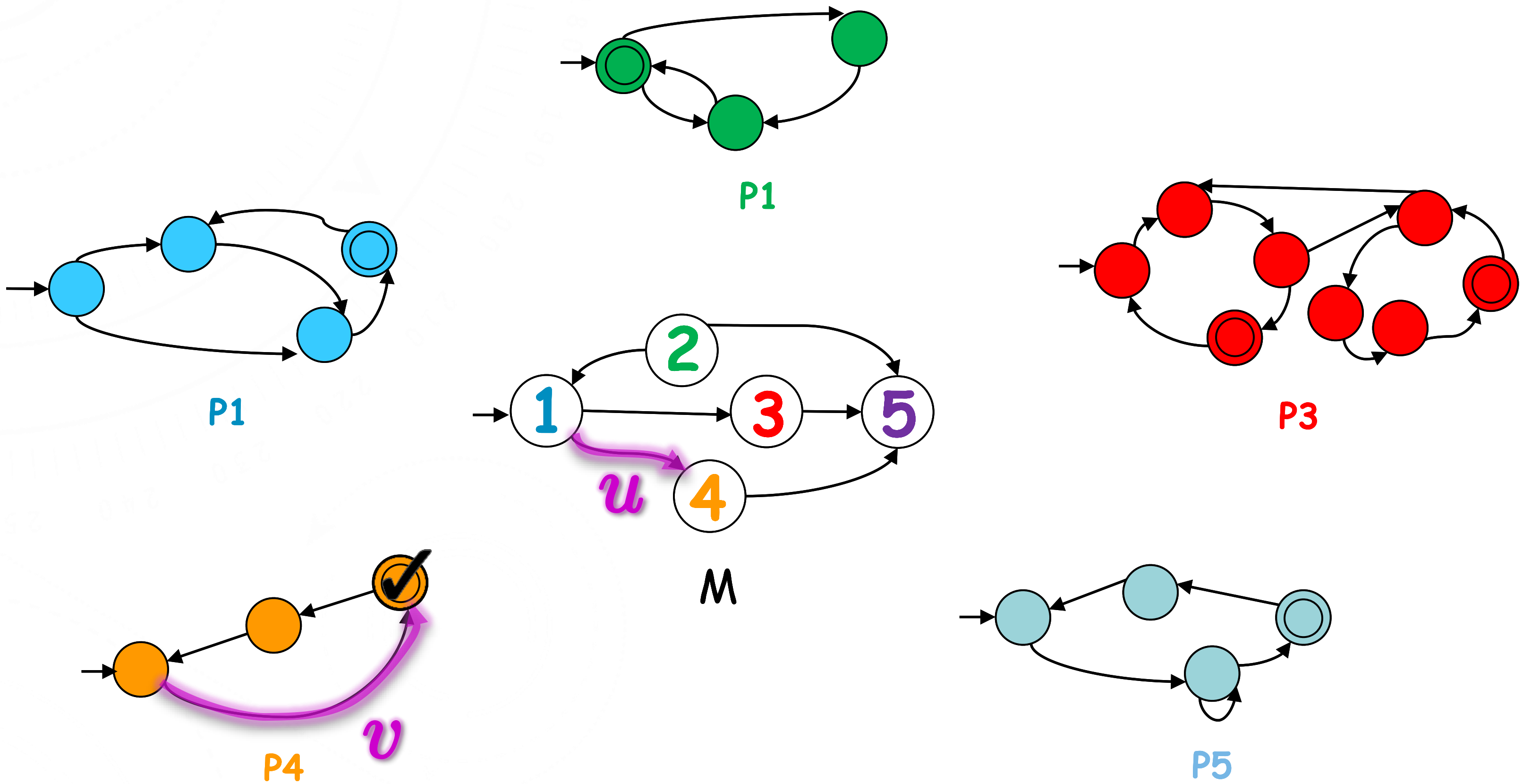
$$(\underbrace{M}_{\text{Leading}}, \underbrace{P_1, P_2, P_3, P_4, P_5}_{\text{Progress}})$$



That restriction is removed

FDFA Acceptance

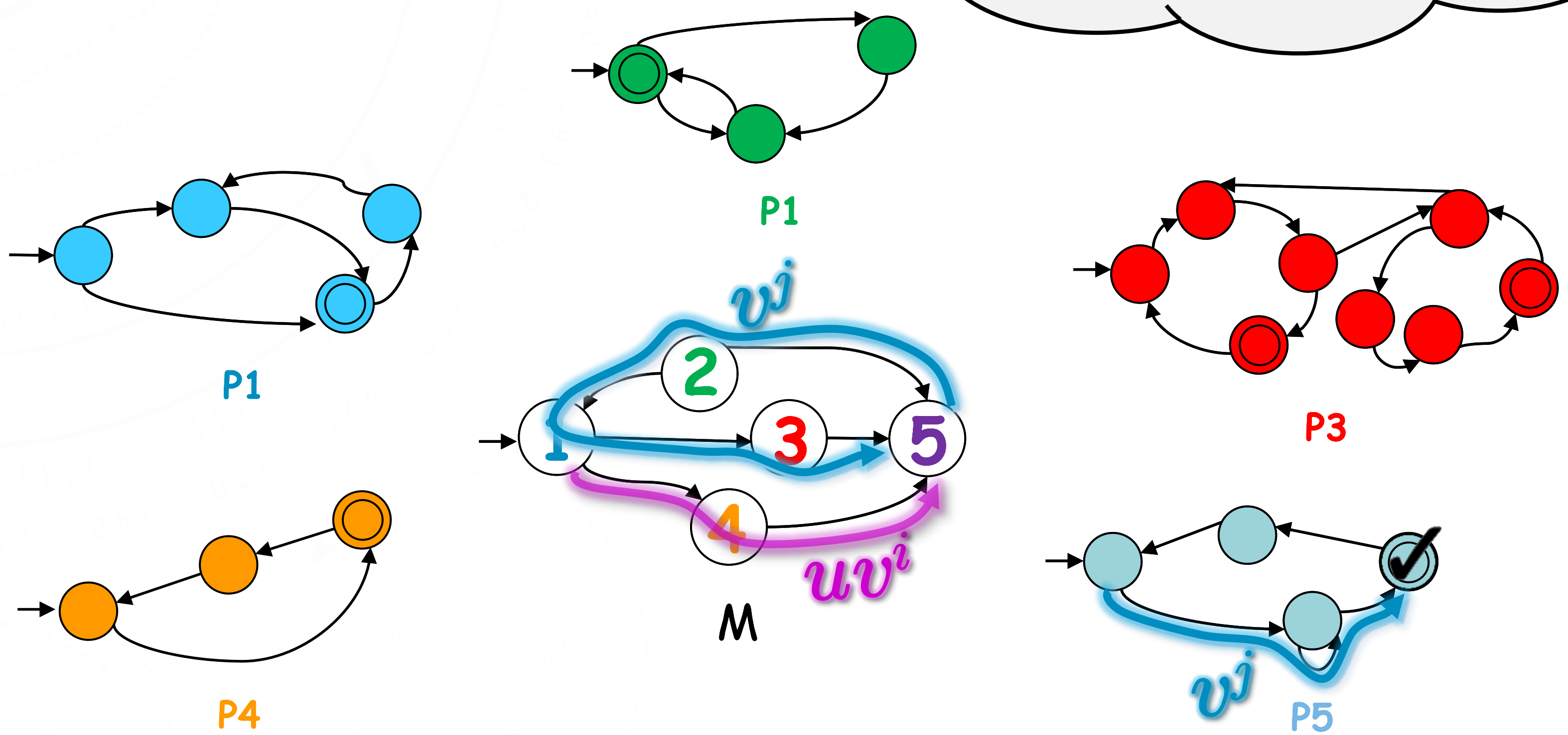
$$(u,v)\overset{?}{\in}[\![\overbrace{M, P_1, P_2, P_3, P_4, P_5}^{\mathcal{F}}]\!]$$



FDFA Normalized Acceptance

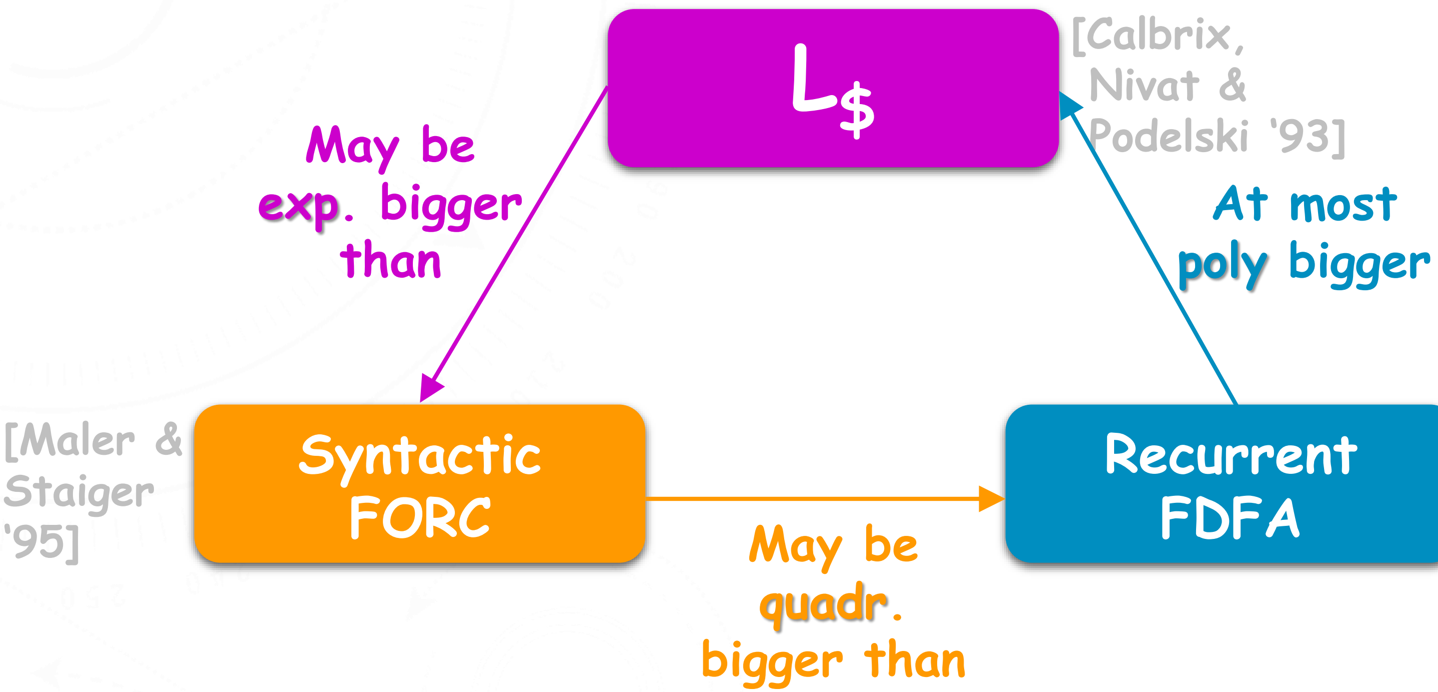
$$(u,v) \overset{?}{\in} [\![\overbrace{M, P_1, P_2, P_3, P_4, P_5}^{\mathcal{F}}]\!]$$

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



We term Recurrent FDFA the FDFA where progress DFA recognize only periods that loop back.

Results (1)

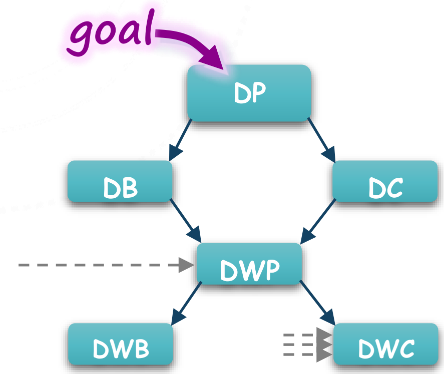


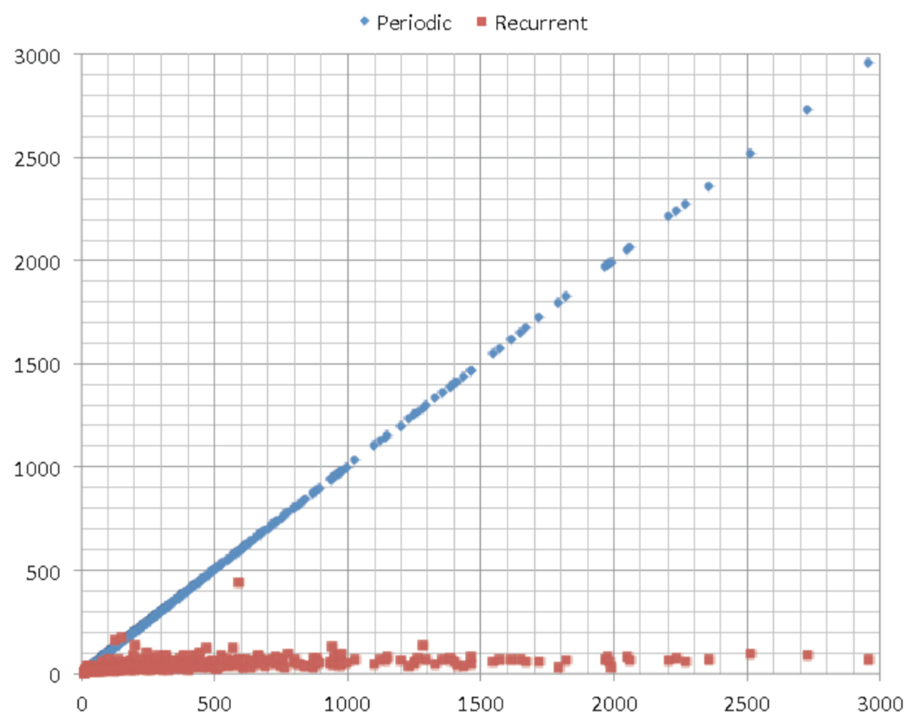
$L_\$$
[Calbrix, Nivat & Podelski '93]
May be exp. bigger than
At most poly bigger
[Maler & Staiger '95]
Syntactic FORC
Recurrent FDFA
May be quadr. bigger than

Results (2)

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

- ☹ Worst-case time complexity polynomial in $L_\$$
- ☺ Preforms very well on random targets

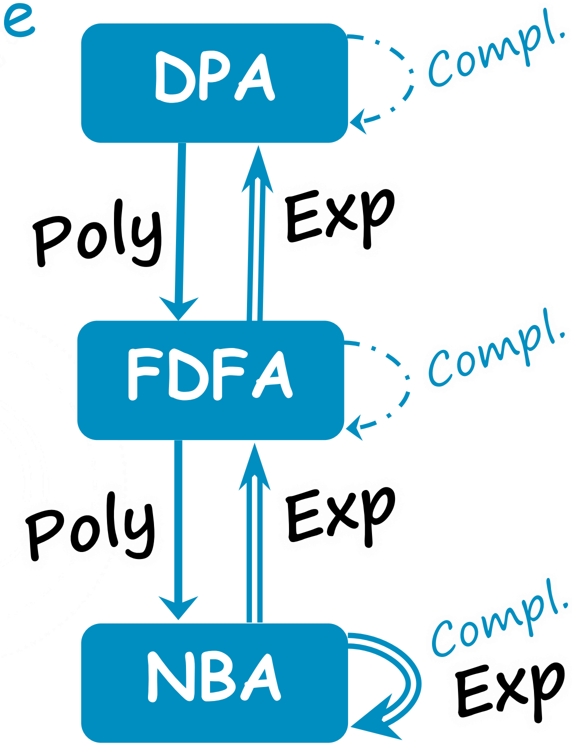




FDFAs as Acceptors of ω-Langs

[Angluin, Boker & Fisman FMCS'16]

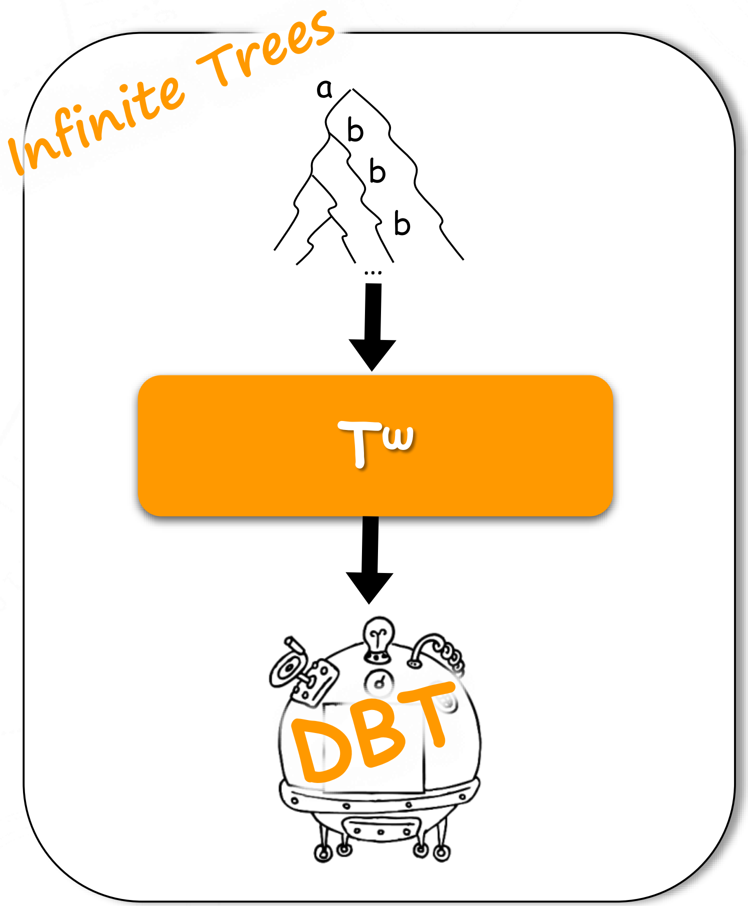
- Have a Myhill-Nerode characterization
- Boolean operations are in LOGSPACE
- Decision problems are in NLOGSPACE
- Succinctness-wise



Some open questions

- Polytime learning of a class of ω-Langs more expressive than DWP
- Saturation of FDFA is in PSPACE; currently no lower bound
- Find smaller canonical representations

Further Directions



On going work with

Dana Angluin &

Timos Antonopulos

Further Directions

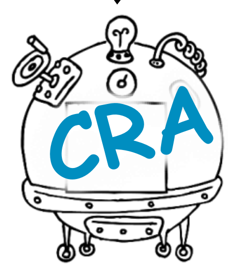
Regular Functions

{ f(abb) = 7,
  f(bbb) = 80,
  f(aaaaaa) = 12
}

↓

F*

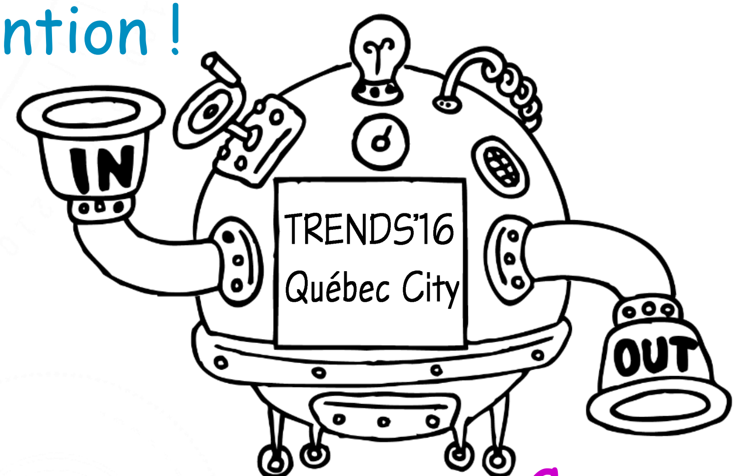
↓



On going work with

Rajeev Alur

Merci beaucoup pour votre attention !



Commentaires / Questions ?