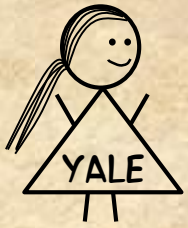
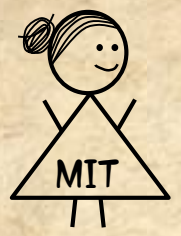
Learning Regular Languages via Alternating Automata





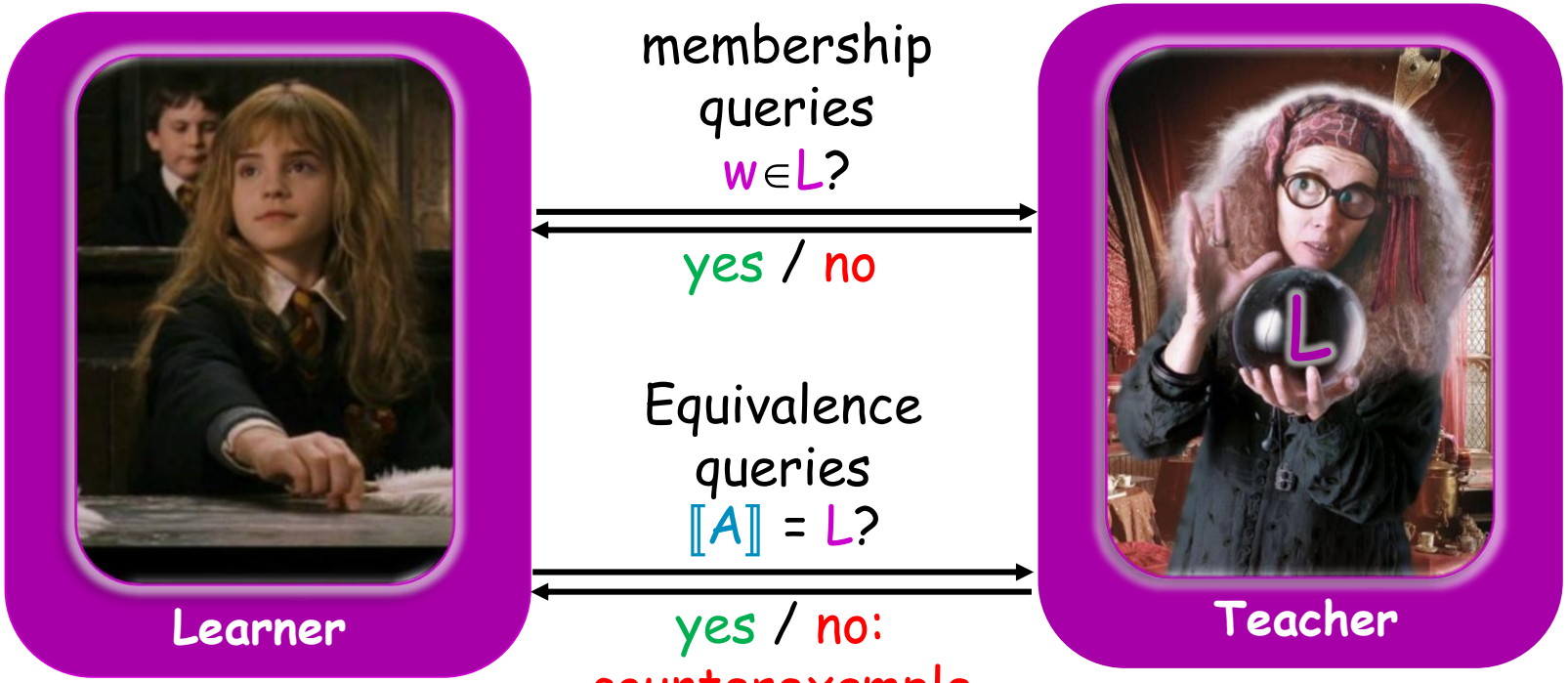
Dana Angluin, Sarah Eisenstat and Dana Fisman

IJCAI'15, Buenos Aires

The Problem

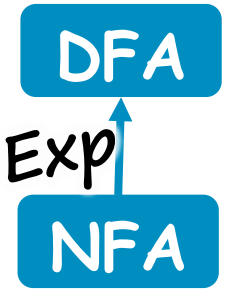
Learn an unknown regular language L using MQ and EQ

data mining
neural networks
geometry
verification
synthesis



Motivation and Goal questions

- [Angluin, 1987] proposed L* which uses deterministic finite automata DFA
- [Bollig et al., 2009], proposed NL* which uses non-deterministic finite automata NFA

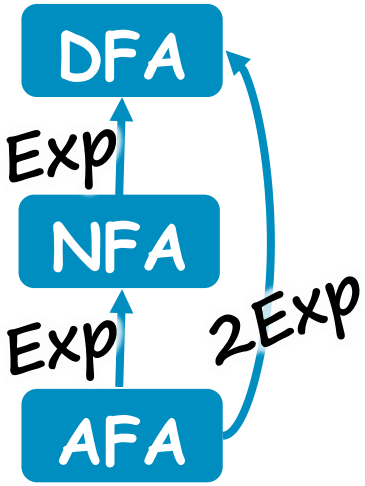


Motivation and Goal questions

Succinctness

- [Angluin, 1987] proposed L* which uses deterministic finite automata DFA
- [Bollig et al., 2009], proposed NL* which uses non-deterministic finite automata NFA

Can we generalize this to AL* which uses alternating finite automata AFA ?

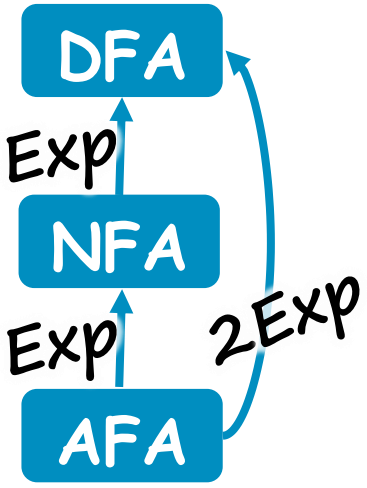


Motivation and Goal questions

Succinctness

- [Angluin, 1987] proposed L* which uses deterministic finite automata DFA
- [Bollig et al., 2009], proposed NL* which uses non-deterministic finite automata NFA

Can we generalize this to AL* which uses alternating finite automata AFA ?



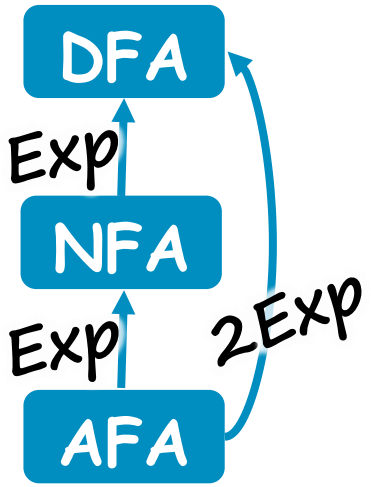
- [Bollig et al., 2009] showed that NL* outperforms L* on random RE (concatenation, iteration, union)

Motivation and Goal questions

Succinctness

- [Angluin, 1987] proposed L* which uses deterministic finite automata DFA
- [Bollig et al., 2009], proposed NL* which uses non-deterministic finite automata NFA

Can we generalize this to AL* which uses alternating finite automata AFA ?



- [Bollig et al., 2009] showed that NL* outperforms L* on random RE

intersection

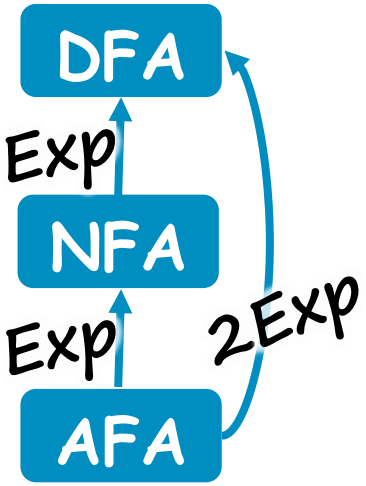
(concatenation, iteration, union)

Motivation and Goal questions

Succinctness

- [Angluin, 1987] proposed L* which uses deterministic finite automata DFA
- [Bollig et al., 2009], proposed NL* which uses non-deterministic finite automata NFA

Can we generalize this to AL* which uses alternating finite automata AFA ?



Understanding

- [Bollig et al., 2009] showed that NL* outperforms L* on random RE

intersection
(concatenation, iteration, union)

What are the tradeoffs between L*, NL* & AL*?

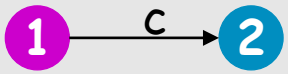
AUTOMATA TYPES

Same expressive power

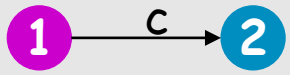
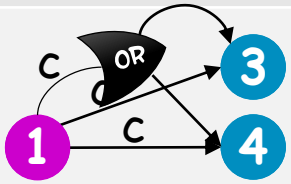
- Deterministic (DFA)
- Non-deterministic (NFA)
- Universal (UFA)
- Alternating (AFA)

Transition Types

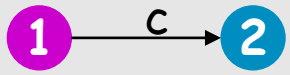
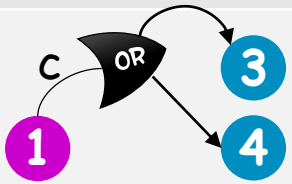
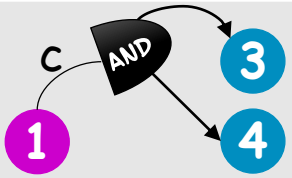
Transition Type	from state	upon read-ing	to state(s)
Deterministic	s1	c	s2



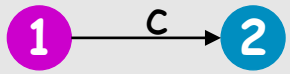
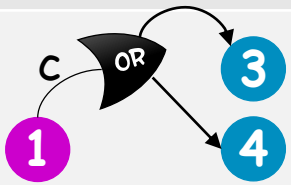
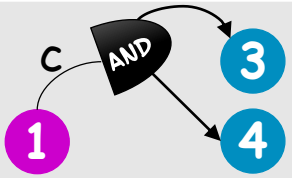
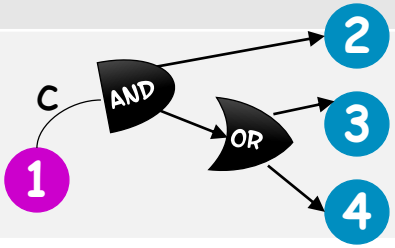
Transition Types

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

Transition Types

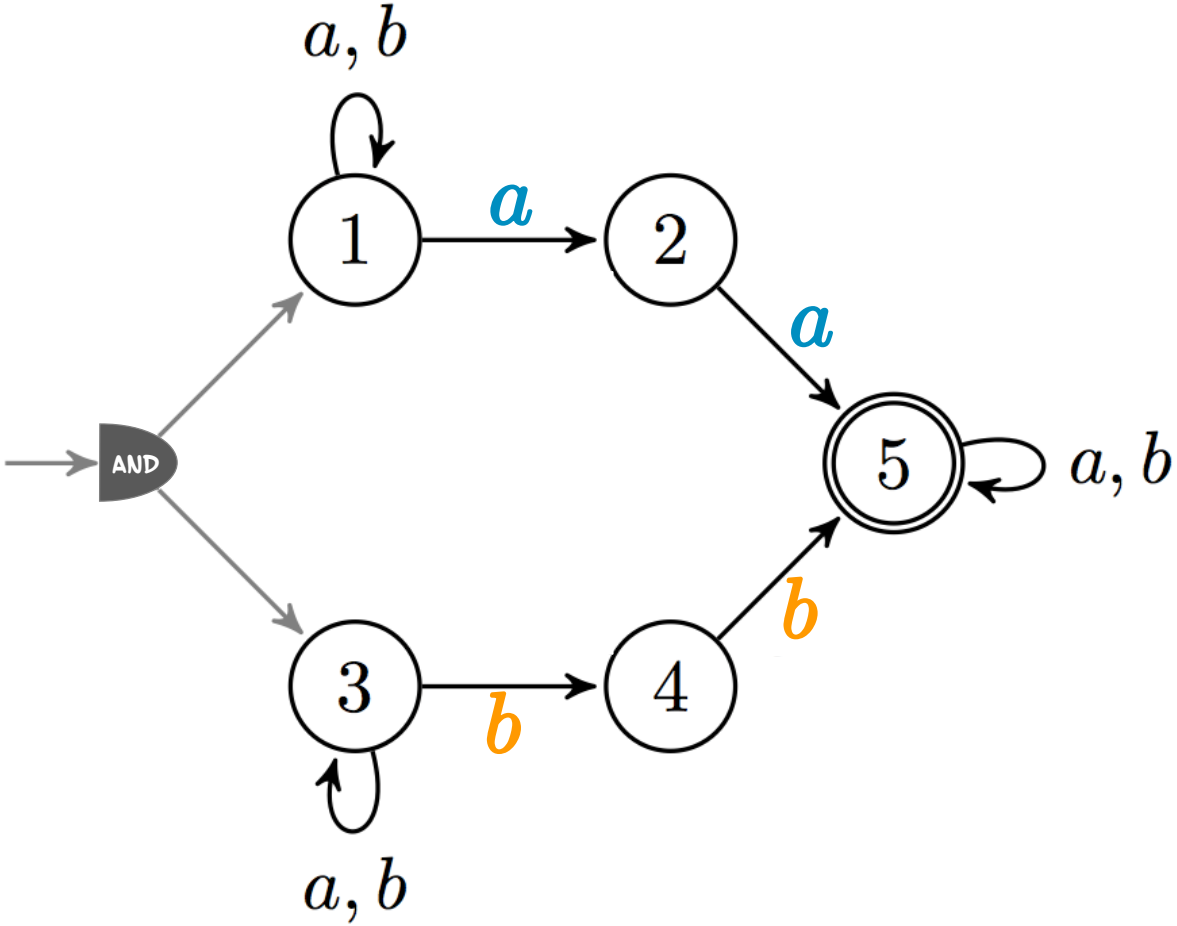
Transition Type	from state	upon read-ing	to state(s)
Deterministic	s1	c	s2 
Non-Deterministic	s1	c	s3 or s4 
Universal	s1	c	s3 and s4 

Transition Types

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.

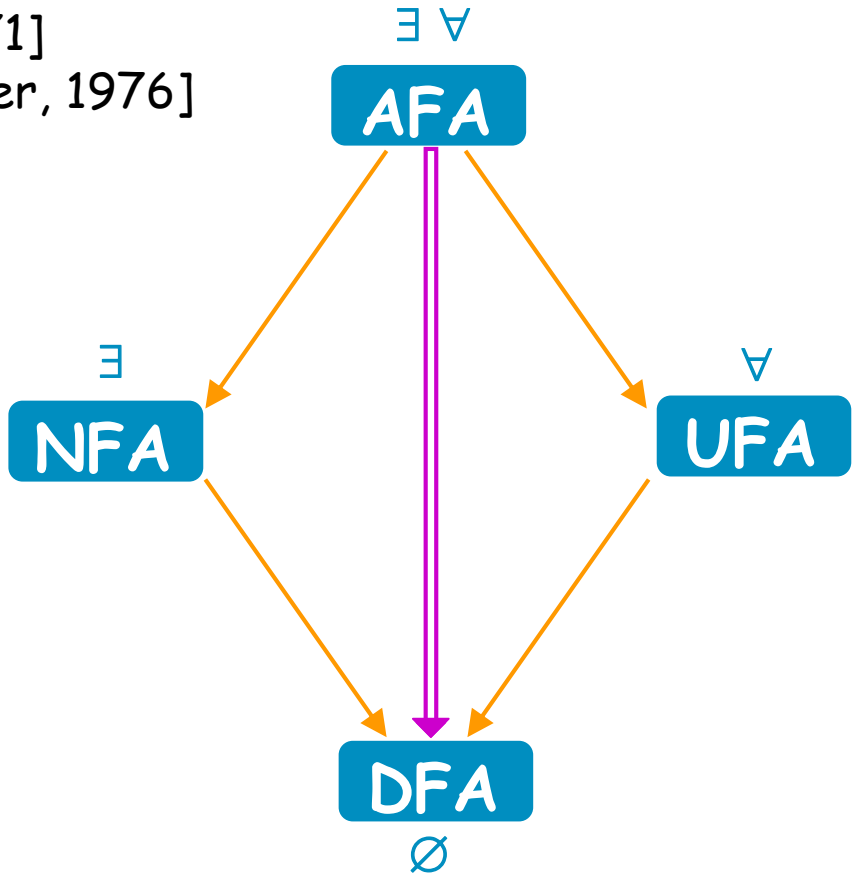
$\Sigma = \{a,b\}$



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

Succinctness

Well known:
[Meyer & Fischer, 1971]
[Chandra & Stockmeyer, 1976]
[Kozen, 1976]



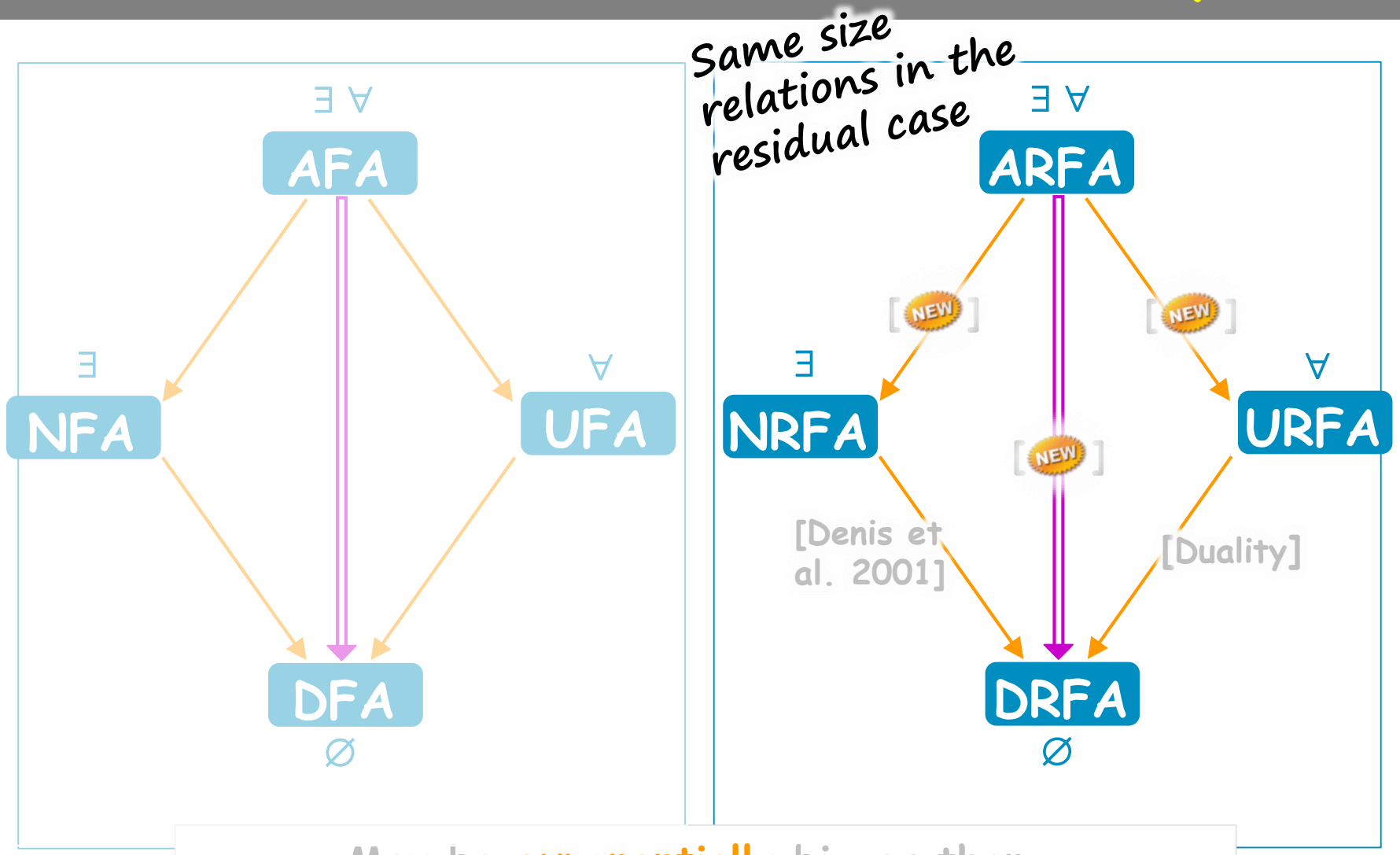
→ May be **exponentially** bigger than

⟹ May be **doubly exponentially** bigger than

Residuality

- To be precise [Bollig et al.] used residual NFA (NRFA), introduced by [Denis et al.]
- We thus first extended the idea of residual NFA to residual AFA (ARFA)
- Details in the paper and poster…

Succinctness & Residuality



L*, NL*, UL*, AL*

- Observation tables
- Closed and minimal tables
- Monotone, union and intersection bases
- Finding an adequate basis

An Observation Table

Strings: experiments to distinguish states

Strings: candidate state representatives

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		
$\vdots$							

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

The Complete infinite Observation Table

Enumeration of all strings

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

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

We call it the row index.

The Complete infinite Observation Table

The number of distinct columns is also finite.

We call it the column index.

Enumeration of all strings

Enumeration of all strings

	a	b	aa	ab	ab	bb	aaa	aab	aba	abb	baa
	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

Closed Table

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

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

If it satisfies

1) Initialization: ε ⊆ B
2) Consecution: BΣ ⊆ S
3) Coverage: all rows not in B are covered by B

Closed Table

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

	S	e_1	e_2	e_3	e_4	e_5	e_6	
B	ε	1	0	0	1	1		
B	a	0	1	0	0	1		
B	b	0	1	0	0	1		covered
B	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

If it satisfies

1) Initialization: ε ⊆ B
2) Consecution: BΣ ⊆ S
3) Coverage: all rows not in B are x-covered by B

x ∈ {D,N,U,A}

That is, the definition differs for L*, NL*, UL* and AL*.

D-Covered

D. Fisman

According to L*
i.e. when
using DFAs

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		Same
b	0	1	0	0	1		Same
ab	1	0	0	1	1		
aa	1	1	1	0	0		Same
aaa	1	0	0	1	1		Same
aab	0	1	0	0	1		

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

$$b = (\varepsilon \vee a)$$

U-Covered

D. Fisman

According to UL* i.e. when using UFAs

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	0	0	1	1	
aa	1	1	1	0	0	
aaa	1	0	0	1	1	
aab	0	1	0	0	1	

Expressible as bitwise-and of some rows in B

$$b = (\varepsilon \wedge a)$$

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 **montone combination** of some rows in $\mathcal{B}$

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

Minimal Table

Smaller is better!

Not interesting!

closed

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

trivially closed

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

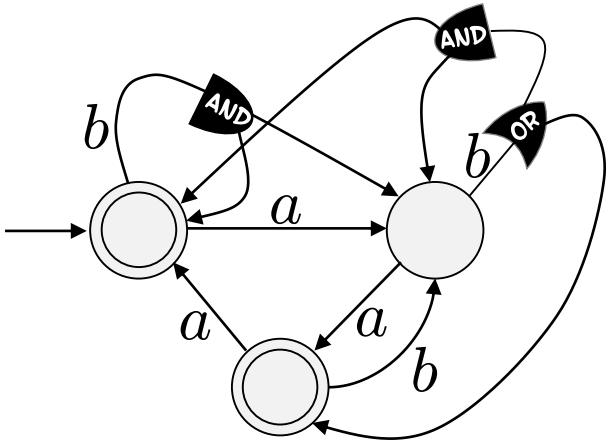
Let T be closed w.r.t. B.

T is x-minimal w.r.t. B if forall $b \in B$, b is not x-covered by any subset of $B \setminus \{b\}$.

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 row s an intersection of rows in B? Poly time [Duality]
  - 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.]
  - unique intersection basis Poly time [Duality]
  - Not unique monotone basis NP-complete [NEW]

The Learning Alg.

Start with table :

	ε
ε	0

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*Find&AddCols*,
In$\mathbb{B}_{\text{X}}$, x*ExtractAut*

$S = \langle\epsilon\rangle$, $E = \langle\epsilon\rangle$, $P = \langle\epsilon\rangle$ and $M_{\epsilon,\epsilon} = \text{MQ}(\epsilon)$.
repeat
  $(a_1, s_1) = \mathcal{T}.\text{Is}\text{x}\text{Closed}(P)$
  if $a_1 =$ *"no"* **then**
    $P.\text{AddString}(s_1)$
  else
    $(a_2, s_2) = \mathcal{T}.\text{Is}\text{x}\text{Minimal}(P)$
    if $a_2 =$ *"no"* **then**
      $P.\text{RemoveString}(s_2)$
    else
      $\mathcal{A} = \mathcal{T}.\text{x}\text{ExtractAut}(P)$
      $(a_3, s_3) = \text{EQ}(\mathcal{A})$
      if $a_3 =$ *"no"* **then**
        $\mathcal{T}.\text{x}\text{Find\&AddCols}(s_3)$
until $a_3 =$ *"yes"*
return $\mathcal{A}$

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*Find&AddCols*,
In$\mathbb{B}_{\mathrm{X}}$, X*ExtractAut*

$S = \langle\epsilon\rangle$, $E = \langle\epsilon\rangle$, $P = \langle\epsilon\rangle$ and $M_{\epsilon,\epsilon} = \mathrm{MQ}(\epsilon)$.

repeat
  $(a_1,$
  if a_1
  else
    if $a_2 =$ *"no"* **then**
      P.*RemoveString*(s_2)
    else
      $\mathcal{A} = \mathcal{T}$.X*ExtractAut*$(P)$
      $(a_3, s_3) = \mathrm{EQ}(\mathcal{A})$
      if $a_3 =$ *"no"* **then**
        $\mathcal{T}$.X*Find&AddCols*(s_3)
until $a_3 =$ *"yes"*
return $\mathcal{A}$

Start with basis:

$\{\varepsilon\}$

The Learning Alg.

If the table is **not closed**, e.g. s_1 is missing, then **add it**.

Until the table is **closed**.

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*Find&AddCols*,
In$\mathbb{B}_X$, X*ExtractAut*

$S = \langle\epsilon\rangle$, $E = \langle\epsilon\rangle$, $P = \langle\epsilon\rangle$ and $M_{\epsilon,\epsilon} = \text{MQ}(\epsilon)$.

repeat
  $(a_1, s_1) = \mathcal{T}.\textit{Is}\text{X}\textit{Closed}(P)$
  if $a_1 =$ *"no"* **then**
    $P.\textit{AddString}(s_1)$
  else
    $(a_2, s_2) = \mathcal{T}.\textit{Is}\text{X}\textit{Minimal}(P)$
    if $a_2 =$ *"no"* **then**
      $P.\textit{RemoveString}(s_2)$
    else
      $\mathcal{A} = \mathcal{T}.\text{X}\textit{ExtractAut}(P)$
      $(a_3, s_3) = \text{EQ}(\mathcal{A})$
      if $a_3 =$ *"no"* **then**
        $\mathcal{T}.\text{X}\textit{Find\&AddCols}(s_3)$
until $a_3 =$ *"yes"*
return $\mathcal{A}$

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*Find&AddCols*,
In$\mathbb{B}_{\text{X}}$, X*ExtractAut*
$S = \langle\epsilon\rangle$, $E = \langle\epsilon\rangle$, $P = \langle\epsilon\rangle$ and $M_{\epsilon,\epsilon} = \text{MQ}(\epsilon)$.
repeat
  $(a_1, s_1) = \mathcal{T}.\textit{Is}\text{X}\textit{Closed}(P)$
  if $a_1 =$ *"no"* **then**
    $P.\textit{AddString}(s_1)$
  else
    $(a_2, s_2) = \mathcal{T}.\textit{Is}\text{X}\textit{Minimal}(P)$
    if $a_2 =$ *"no"* **then**
      $P.\textit{RemoveString}(s_2)$
    else
      $\mathcal{A} = \mathcal{T}.\text{X}\textit{ExtractAut}(P)$
      $(a_3, s_3) = \text{EQ}(\mathcal{A})$
      if $a_3 =$ *"no"* **then**
        $\mathcal{T}.\text{X}\textit{Find\&AddCols}(s_3)$
until $a_3 =$ *"yes"*
return $\mathcal{A}$

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

Until the table is minimal.

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*Find&AddCols*,
In$\mathbb{B}_{\text{X}}$, X*ExtractAut*

$S = \langle\epsilon\rangle$, $E = \langle\epsilon\rangle$, $P = \langle\epsilon\rangle$ and $M_{\epsilon,\epsilon} = \text{MQ}(\epsilon)$.

repeat
  $(a_1, s_1) = \mathcal{T}.\textit{Is}\text{X}\textit{Closed}(P)$
  if $a_1 =$ *"no"* **then**
    $P.\textit{AddString}(s_1)$
  else
    $(a_2, s_2) = \mathcal{T}.\textit{Is}\text{X}\textit{Minimal}(P)$
    if $a_2 =$ *"no"* **then**
      $P.\textit{RemoveString}(s_2)$
    else
      $\mathcal{A} = \mathcal{T}.\text{X}\textit{ExtractAut}(P)$
      $(a_3, s_3) = \text{EQ}(\mathcal{A})$
      if $a_3 =$ *"no"* **then**
        $\mathcal{T}.\text{X}\textit{Find\&AddCols}(s_3)$
  $a_3 =$ *"yes"*
return $\mathcal{A}$

The table is now closed and minimal.

Extract the respective AFA.

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*Find&AddCols*,
In$\mathbb{B}_{\text{X}}$, X*ExtractAut*

$S = \langle\epsilon\rangle$, $E = \langle\epsilon\rangle$, $P = \langle\epsilon\rangle$ and $M_{\epsilon,\epsilon} = \text{MQ}(\epsilon)$.

repeat
  $(a_1, s_1) = \mathcal{T}.\textit{Is}\text{X}\textit{Closed}(P)$
  if $a_1 =$ *"no"* **then**
    $P.\textit{AddString}(s_1)$
  else
    $(a_2, s_2) = \mathcal{T}.\textit{Is}\text{X}\textit{Minimal}(P)$
    if $a_2 =$ *"no"* **then**
      $P.\textit{RemoveString}(s_2)$
    else
      $\mathcal{A} = \mathcal{T}.\text{X}\textit{ExtractAut}(P)$
      $(a_3, s_3) = \text{EQ}(\mathcal{A})$
      if $a_3 =$ *"no"* **then**
        $\mathcal{T}.\text{X}\textit{Find\&AddCols}(s_3)$
  $= $ *"yes"*
  $\mathcal{A}$

Ask an equivalence query.

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

Find Columns to Add

Closed and Minimal

Counter example w = $abcde$

	ε	e_2	e_3	e_4	e_5	e_6	$abcde$	$bcde$	cde	de	e
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	1	1	0	1
aab	1	1	1	0	0

$B = \{\varepsilon, a, aa\}$

$B\Sigma = \{b, ab, aaa, , aab\}$

Find Columns to Add

Closed and Minimal

Counter example w = $abcde$

	ε	e_2	e_3	e_4	e_5	e_6	$abcde$	$bcde$	cde	de	e
ε	1	0	0	1	1		1	0	1	0	1
a	0	1	0	0	1		0	1	0	0	0
b	0	0	0	0	1		0	0	1	0	1
ab	1	1	1	0	1		1	1	1	1	0
aa	1	1	1	0	0		1	0	0	1	1
aaa	1	1	1	0	1		1	1	1	1	0
aab	1	1	1	0	0		1	0	1	1	0

B = {ε, a, aa }

BΣ = {b, ab, aaa, , aab }

Find Columns to Add

Closed and Minimal

Counter example $w = abcde$

	ε	e_2	e_3	e_4	e_5	e_6	$bcde$	cde	de	e
ε	1	0	0	1	1		0	1	0	1
a	0	1	0	0	1		1	0	0	0
b	0	0	0	0	1		0	1	0	1
ab	1	1	1	0	1		1	1	1	0
aa	1	1	1	0	0		0	0	1	1
aaa	1	1	1	0	1		1	1	1	0
aab	1	1	1	0	0		0	1	1	0

B = {ε, a, aa }

BΣ = {b, ab, aaa, , aab }

Find Columns to Add

Closed and Minimal

Counter example $w = abcde$

	ε	e_2	e_3	e_4	e_5	e_6	$bcde$	cde	e
ε	1	0	0	1	1		0	1	1
a	0	1	0	0	1		1	0	0
b	0	0	0	0	1		0	1	1
ab	1	1	1	0	1		1	1	0
aa	1	1	1	0	0		0	0	1
aaa	1	1	1	0	1		1	1	0
aab	1	1	1	0	0		0	1	0

new column

new column

new column

B = {ε, a, aa }

BΣ = {b, ab, aaa, , aab }

The Learning Alg.

Theorem

Every counterexample yields at least one new column

The Learning Alg.

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.

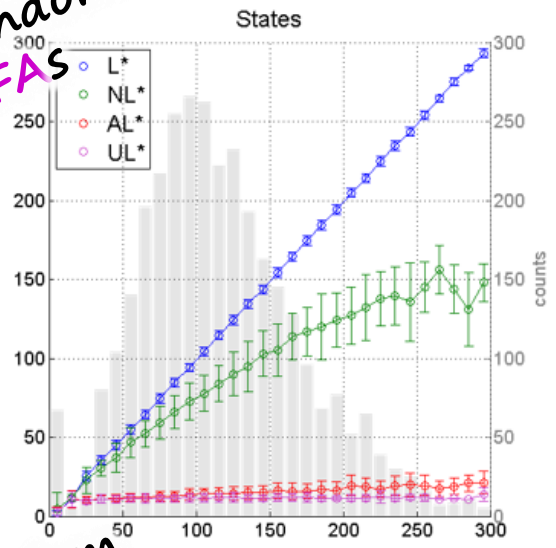
EMPIRICAL RESULTS

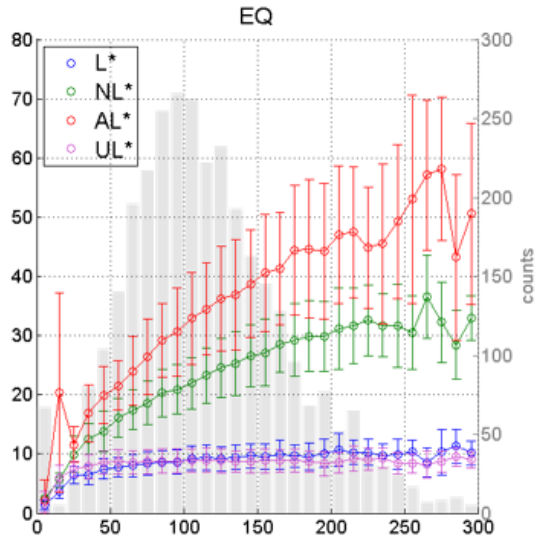
Comparison of L*, NL*, UL*, AL* in learning random DFAs, NFAs, UFAs, AFAs along

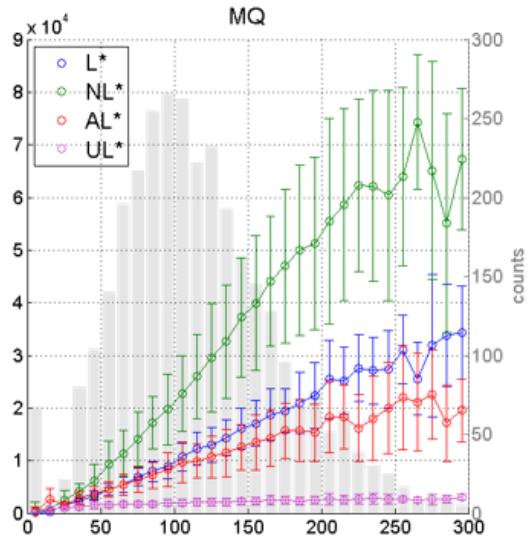
- # states
- # MQ
- # EQ

Empirical Results

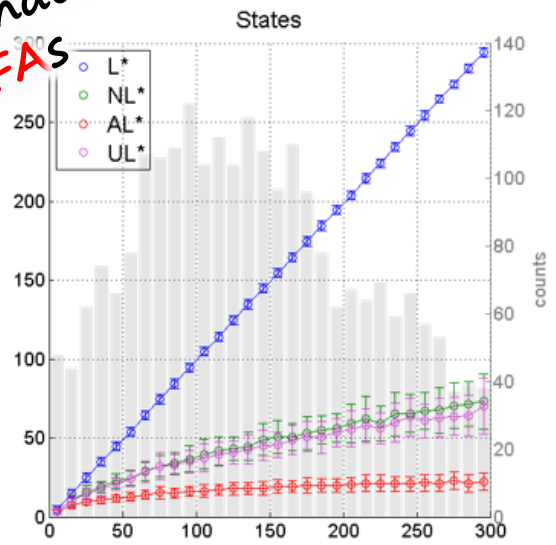
Random UFAs

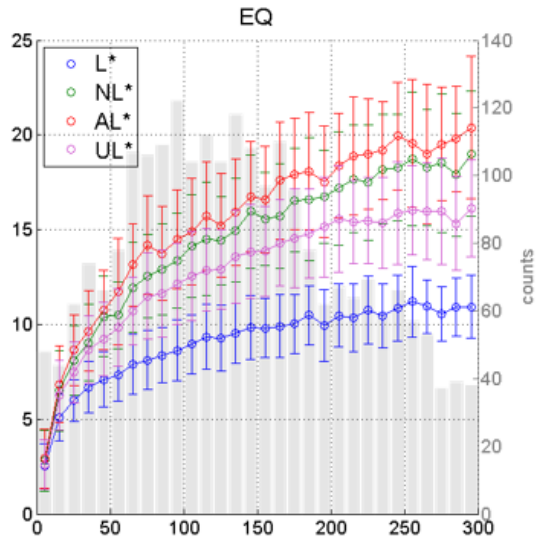


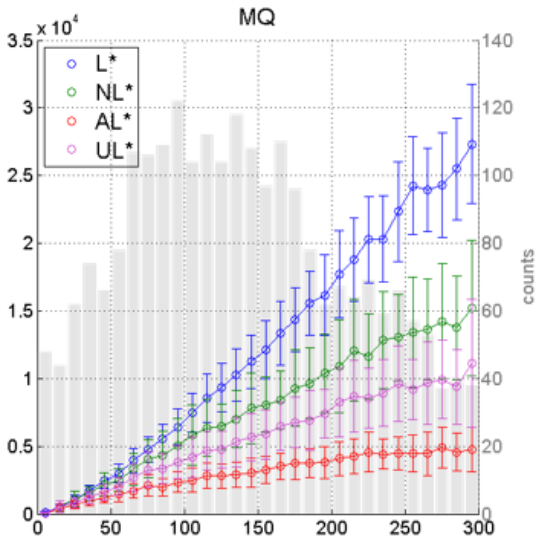




Random AFAs







Empirical Results

Rough Summary:

- In terms of #states generated,

  AL* is always preferable

- In terms of #MQ,

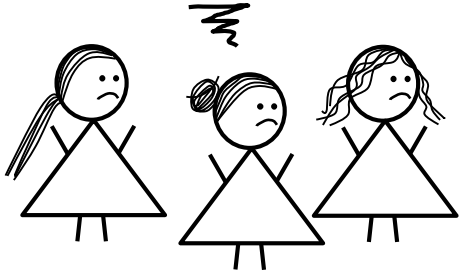
  xL* outperforms the others when targets are xFAs

- In terms of #EQ,

  L* is always preferable

Future Work

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

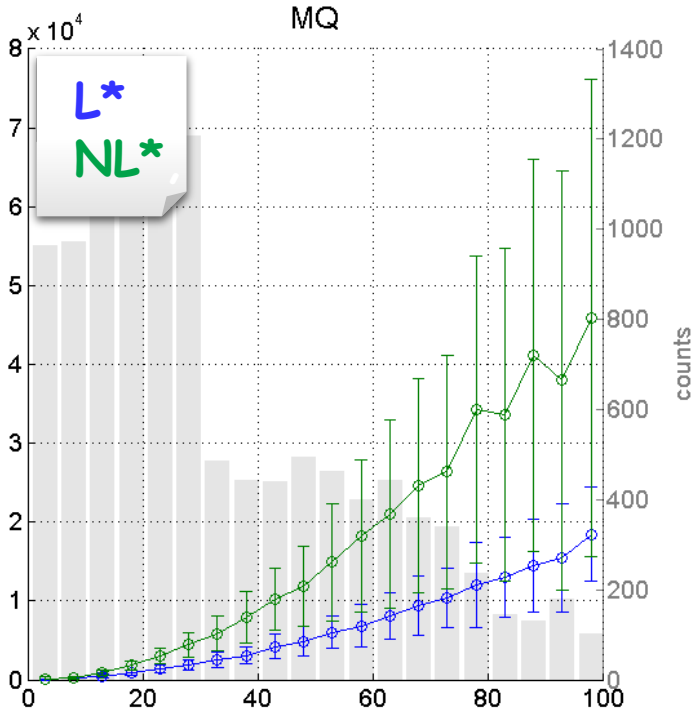


The End

Thank you for your attention!

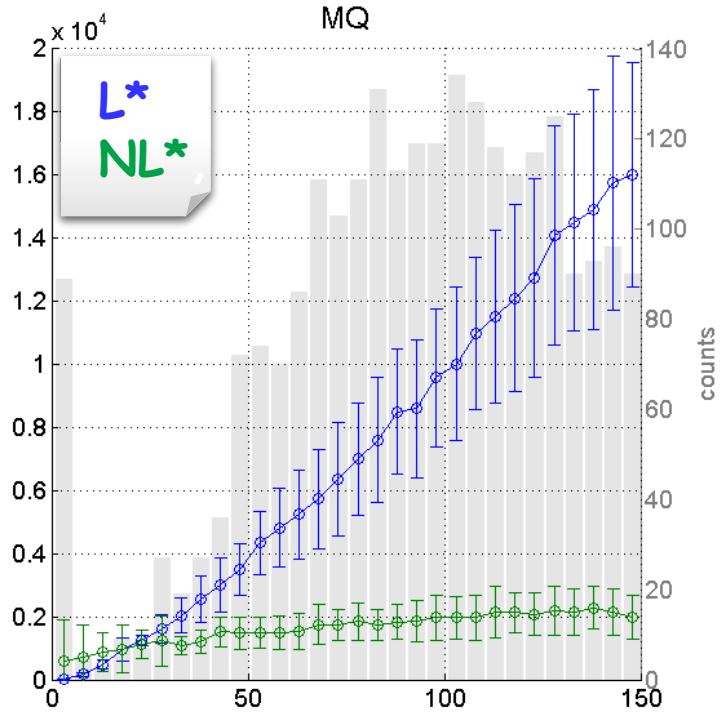
of membership queries

Random DFA targets



L* is better

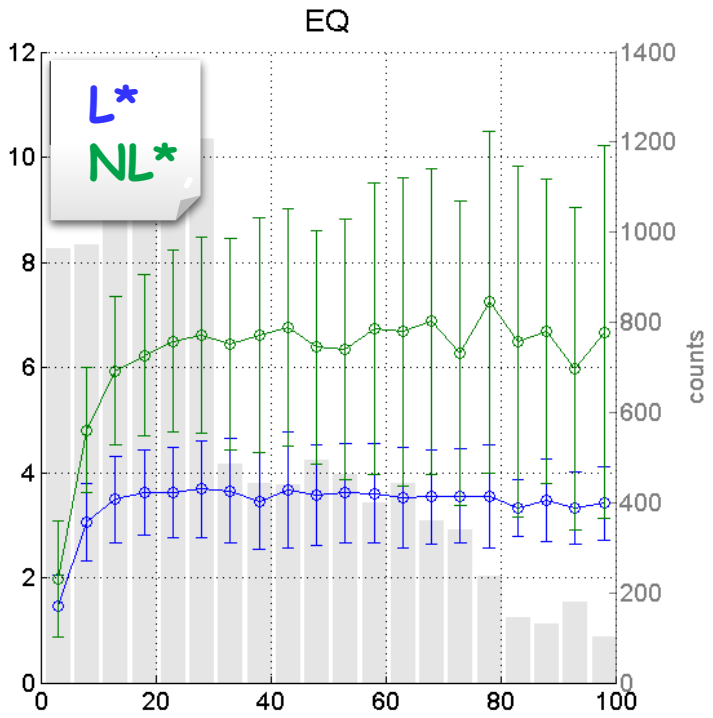
Random NFA targets



NL* is better

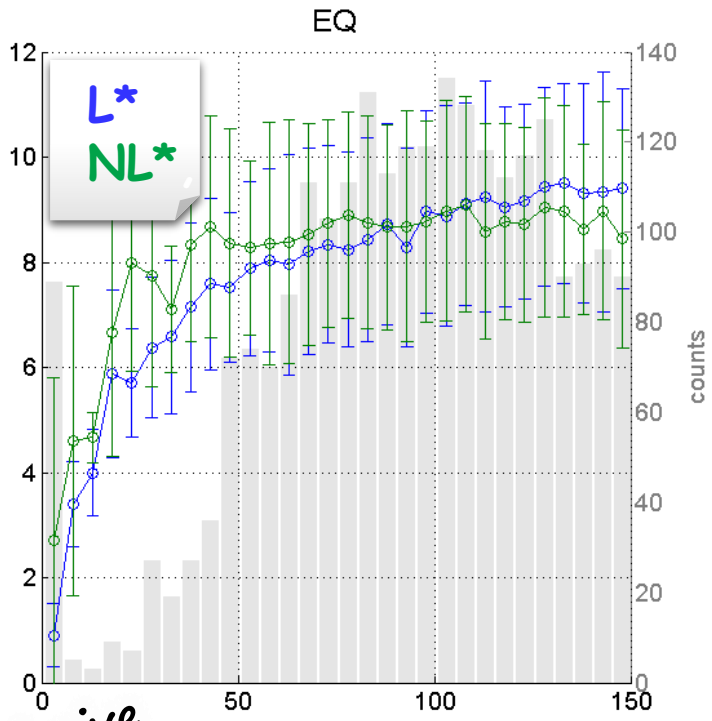
of equivalence queries

Random DFA targets



L* is better

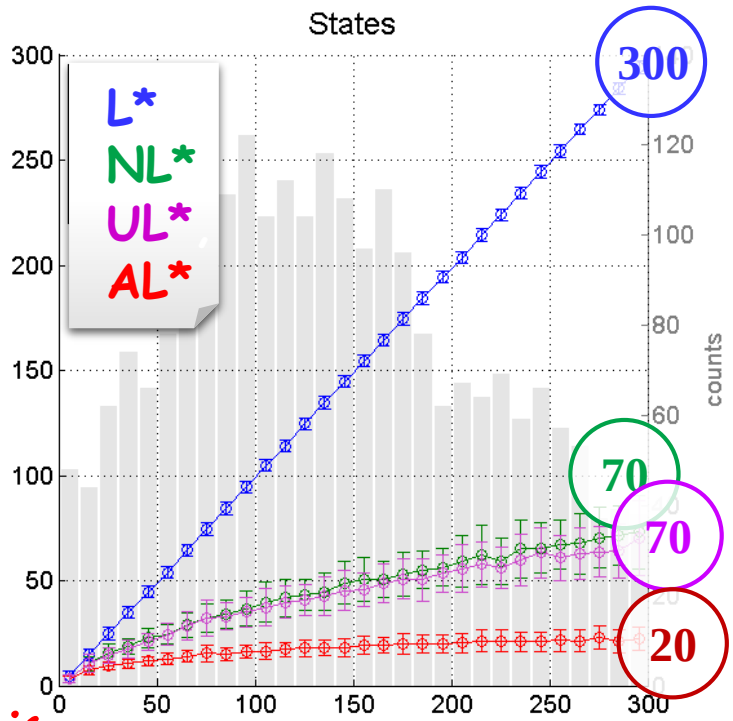
Random NFA targets



Inconclusive

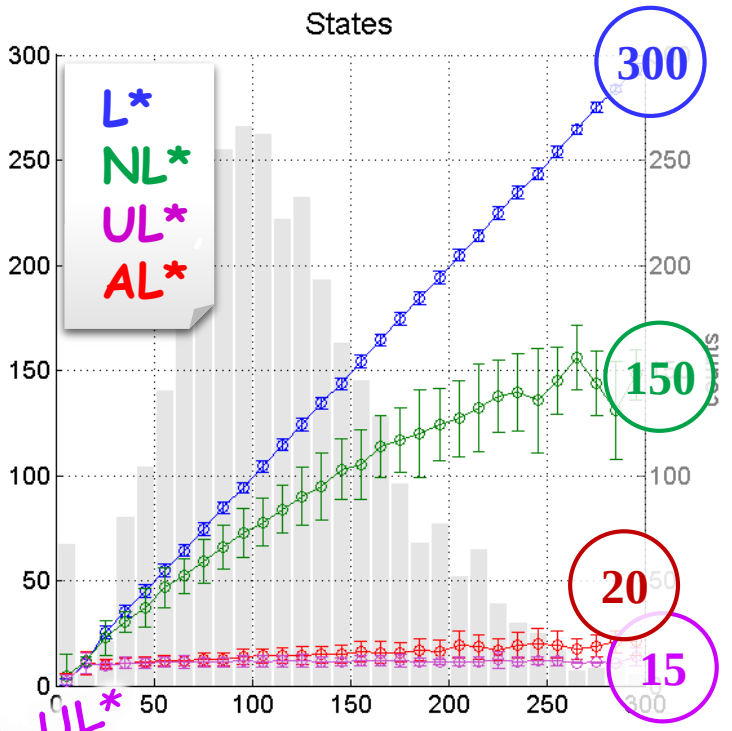
Resulting # of states

Random AFA targets



AL* is better

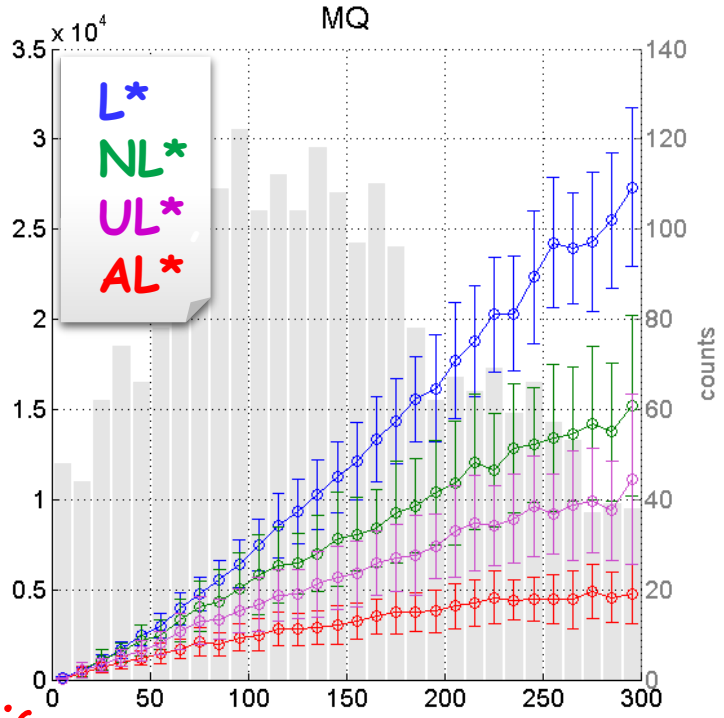
Random UFA targets



AL* & UL* are better

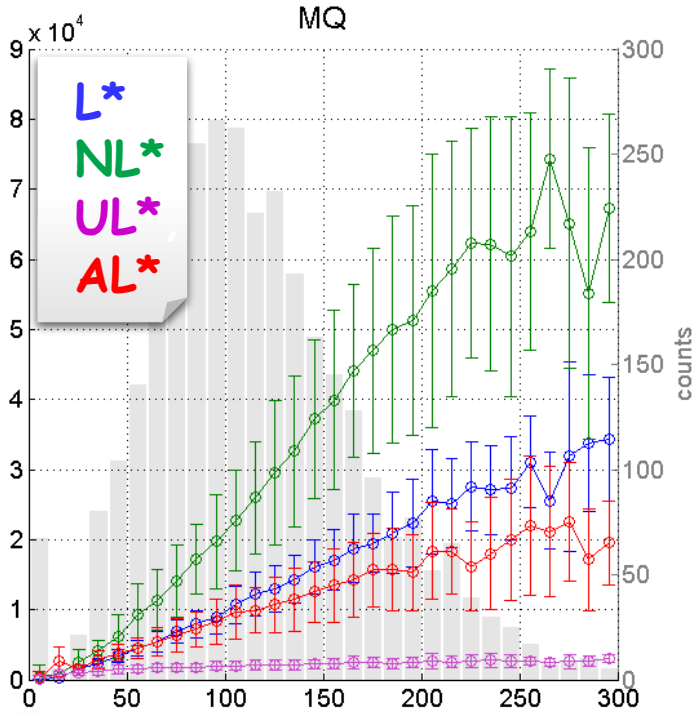
of membership queries

Random AFA targets



AL* is better

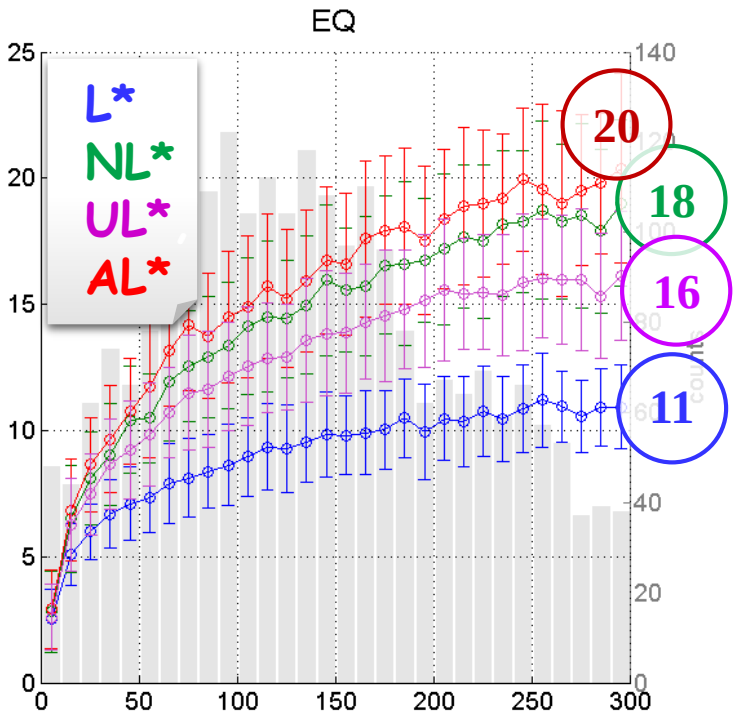
Random UFA targets



UL* is better

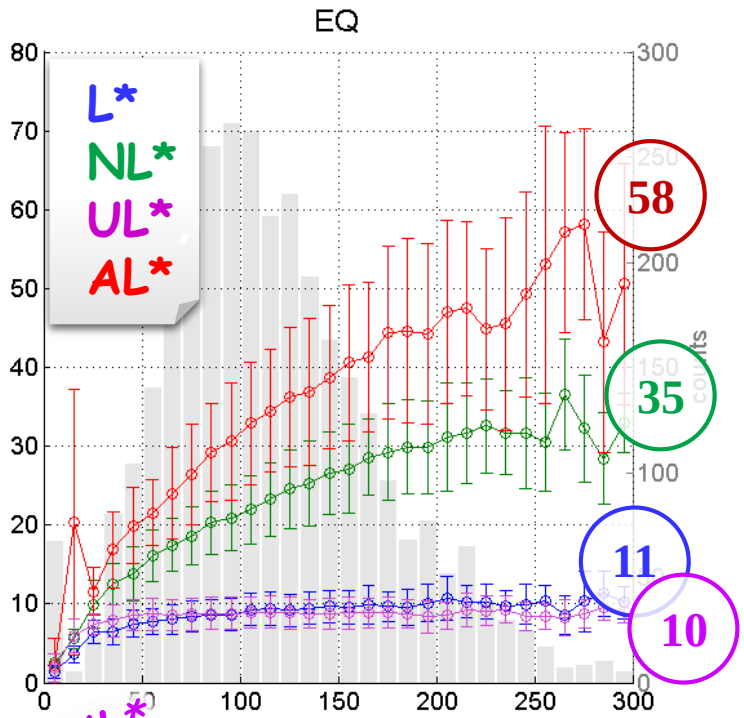
of equivalence queries

Random AFA targets



L* is better

Random UFA targets



L* & UL* are better