

Learning weighted automata

Alexandra Silva



Active Automata Learning



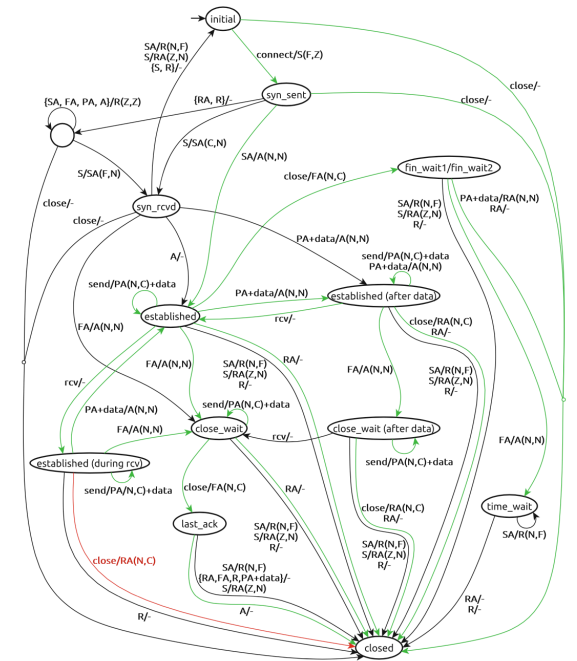
Black-box
interactions

Zoom-in
specific part
of code



Incrementally
build a model

Refine with
properties of
interest

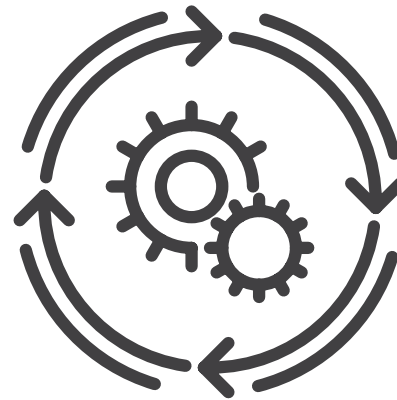


Active Automata Learning



Black-box
interactions

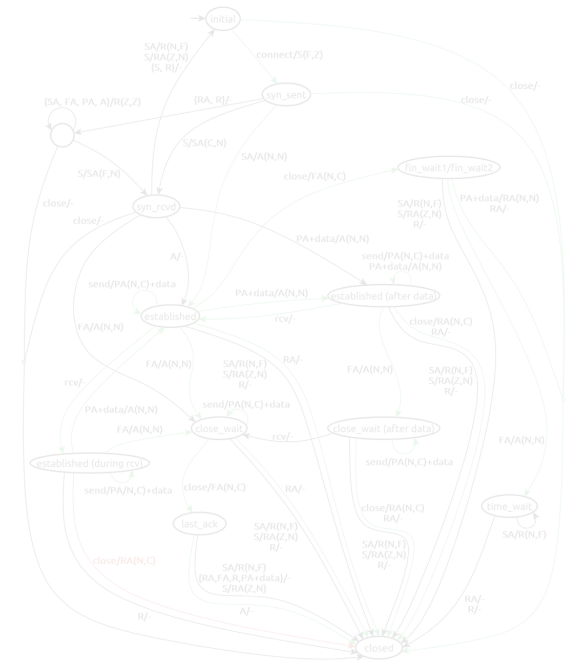
Zoom-in
specific part
of code



Automated

Incrementally
build a model

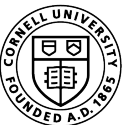
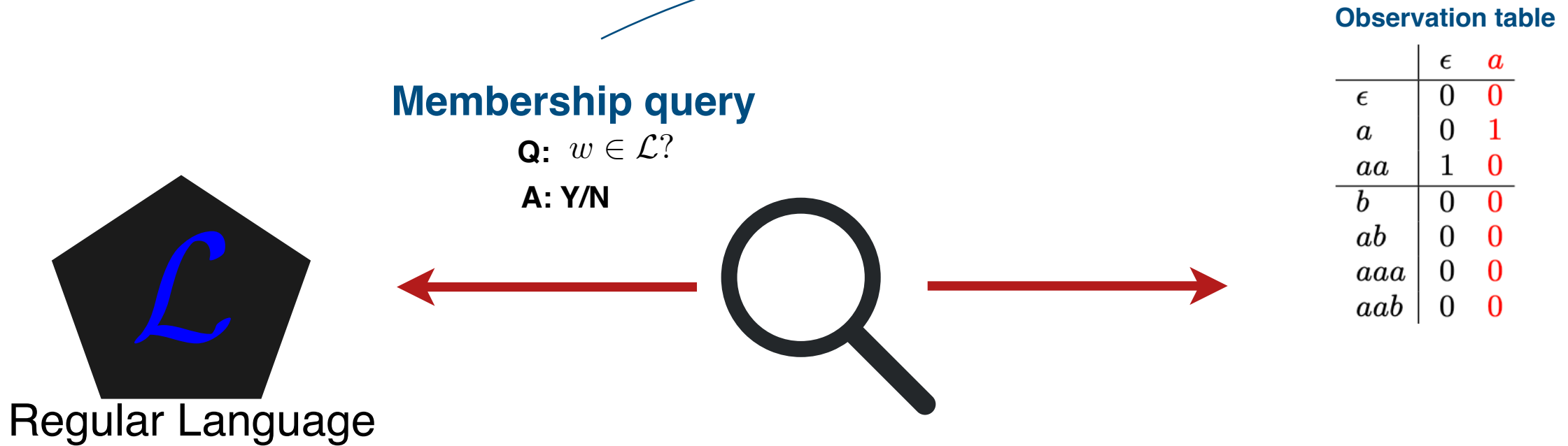
Refine with
properties of
interest



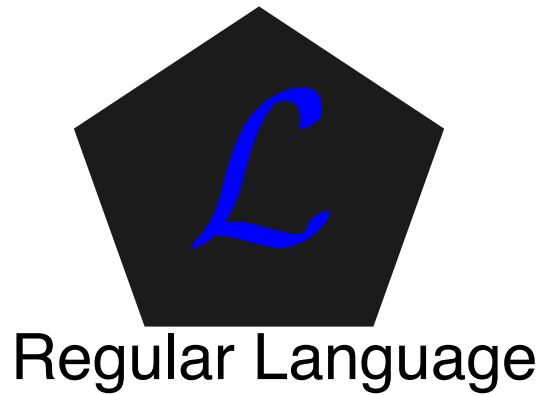
DFA Learning (L^* , Angluin'87)



DFA Learning (L^* , Angluin'87)



DFA Learning (L^* , Angluin'87)



Membership query

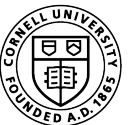
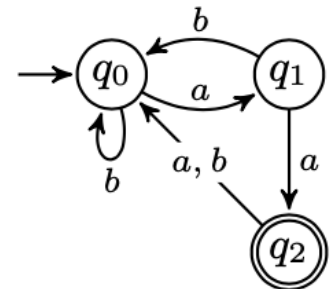
Q: $w \in \mathcal{L}$?

A: Y/N

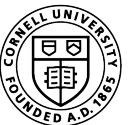
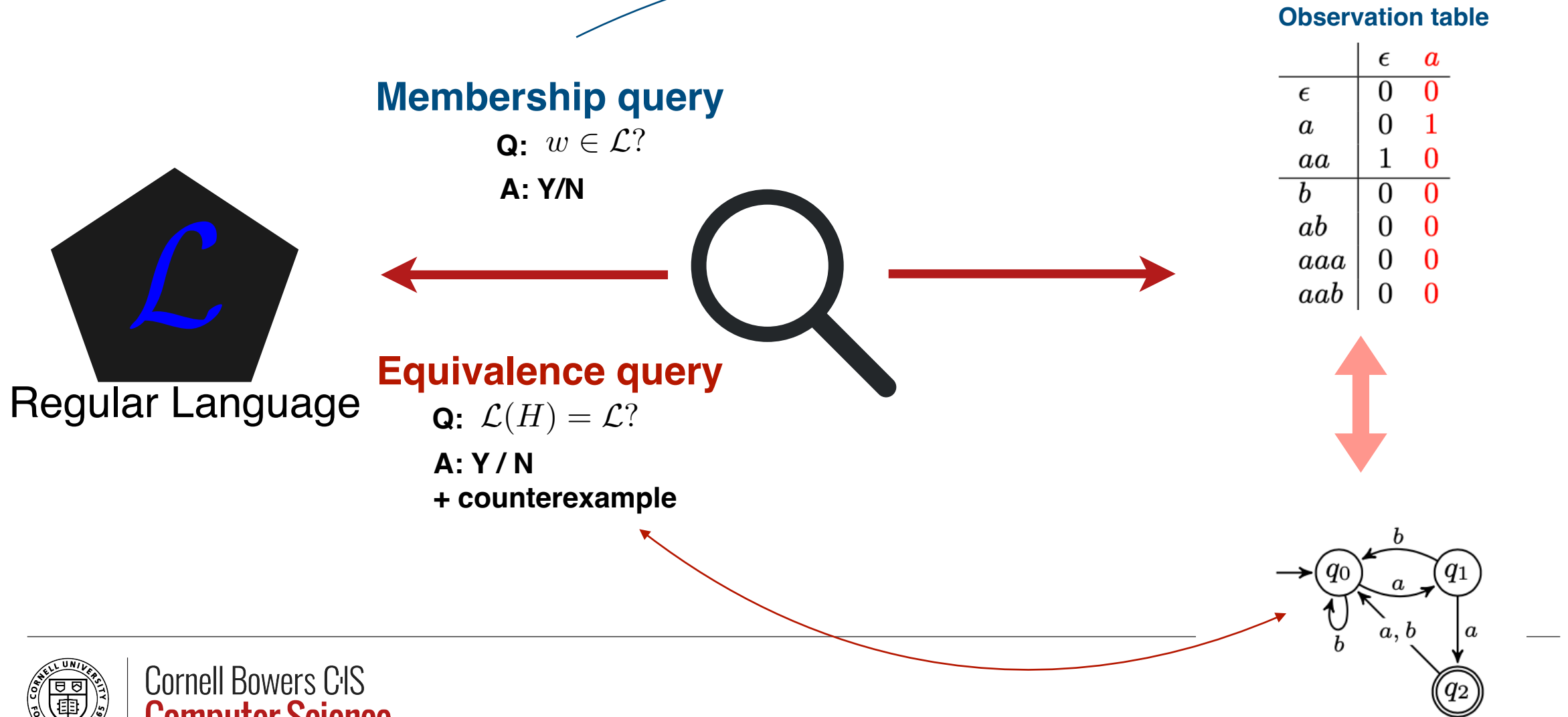


Observation table

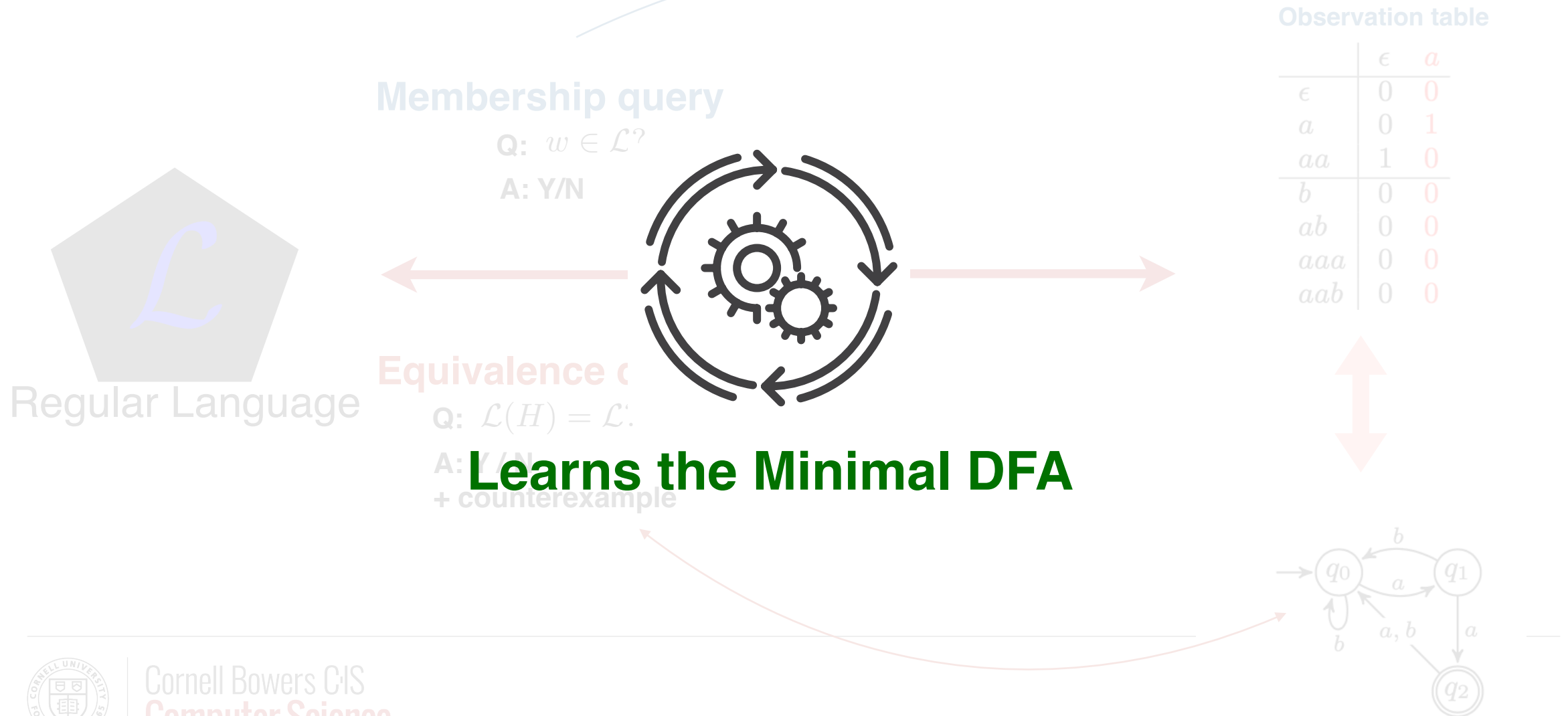
	ϵ	a
ϵ	0	0
a	0	1
aa	1	0
b	0	0
ab	0	0
aaa	0	0
aab	0	0



DFA Learning (L^* , Angluin'87)



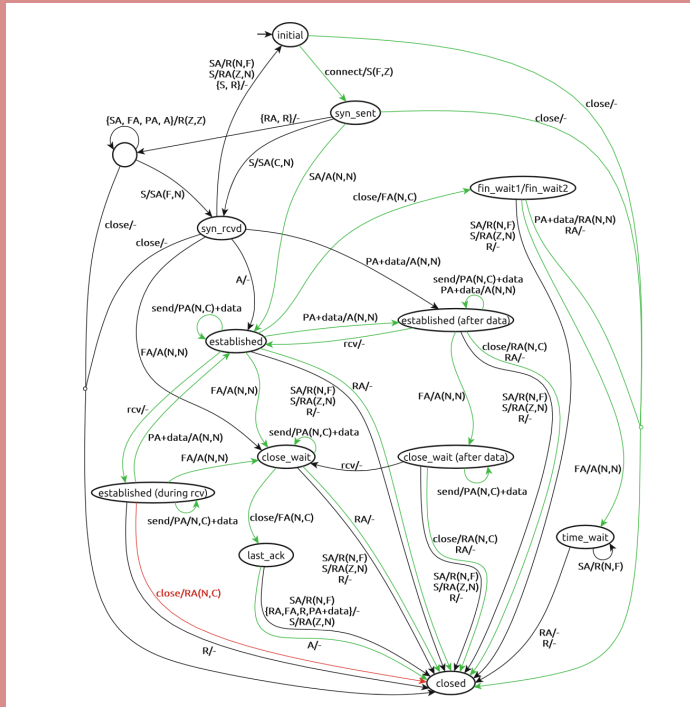
DFA Learning (L^* , Angluin'87)



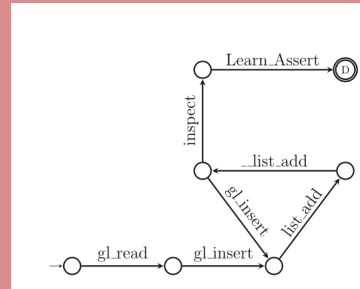
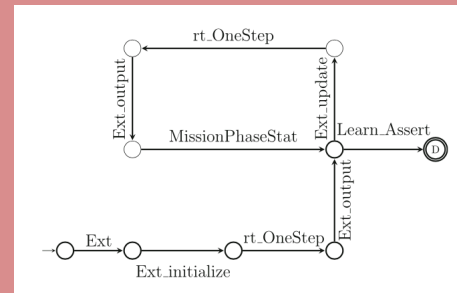
L^* at work

L* at work

Model of Windows 8 TCP implementation



Learning Error Traces (function calls)



Combining Model Learning and Model Checking to Analyze TCP Implementations

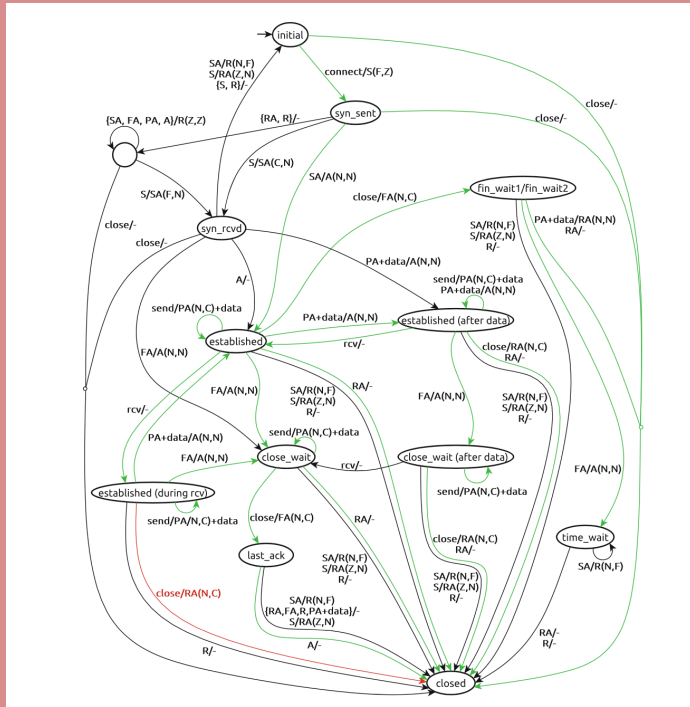
Learning the Language of Error

Martin Chapman¹, Hana Chockler¹ (✉), Pascal Kesseli², Daniel Kroening², Ofer Strichman³, and Michael Tautschnig⁴

Paul Fiterău-Broştean, Ramon Janssen, and Frits Vaandrager (✉)

L* at work

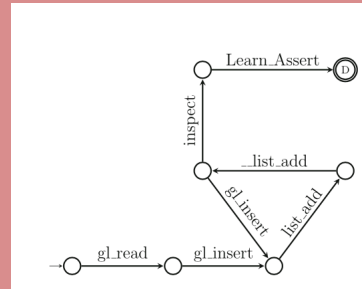
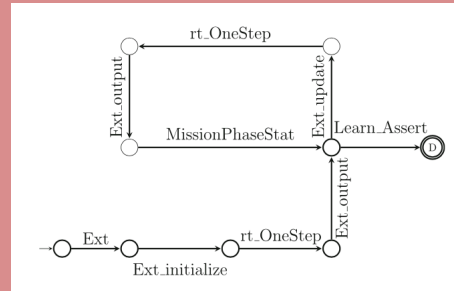
Model of Windows 8 TCP implementation



Combining Model Learning and Model Checking to Analyze TCP Implementations

Paul Fiterău-Broştean, Ramon Janssen, and Frits Vaandrager^(✉)

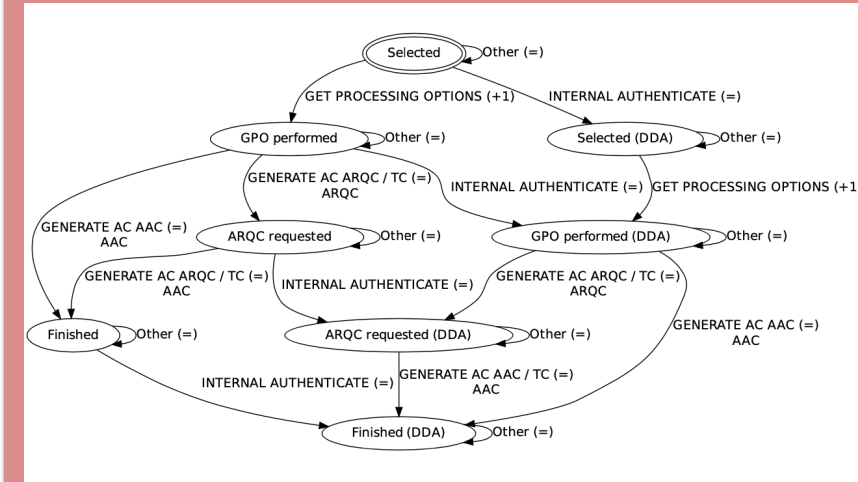
Learning Error Traces (function calls)



Learning the Language of Error

Martin Chapman¹, Hana Chockler^{1(✉)}, Pascal Kesseli², Daniel Kroening², Ofer Strichman³, and Michael Tautschnig⁴

Model of Visa Debit on Barclays card



Formal models of bank cards for free

Fides Aarts, Joeri de Ruiter, and Erik Poll

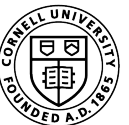
Active learning

Infer automaton through oracle (membership and equivalence queries)

Deterministic
finite automata

Gold, Angluin 80's

Weighted
automata



Active learning

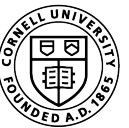
Infer automaton through oracle (membership and equivalence queries)

Deterministic
finite automata

Gold, Angluin 80's

Weighted
automata

$$L \in 2^{A^*}$$



Active learning

Infer automaton through oracle (membership and equivalence queries)

Deterministic
finite automata

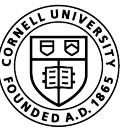
Gold, Angluin 80's

$$L \in 2^{A^*}$$

Weighted
automata

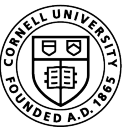
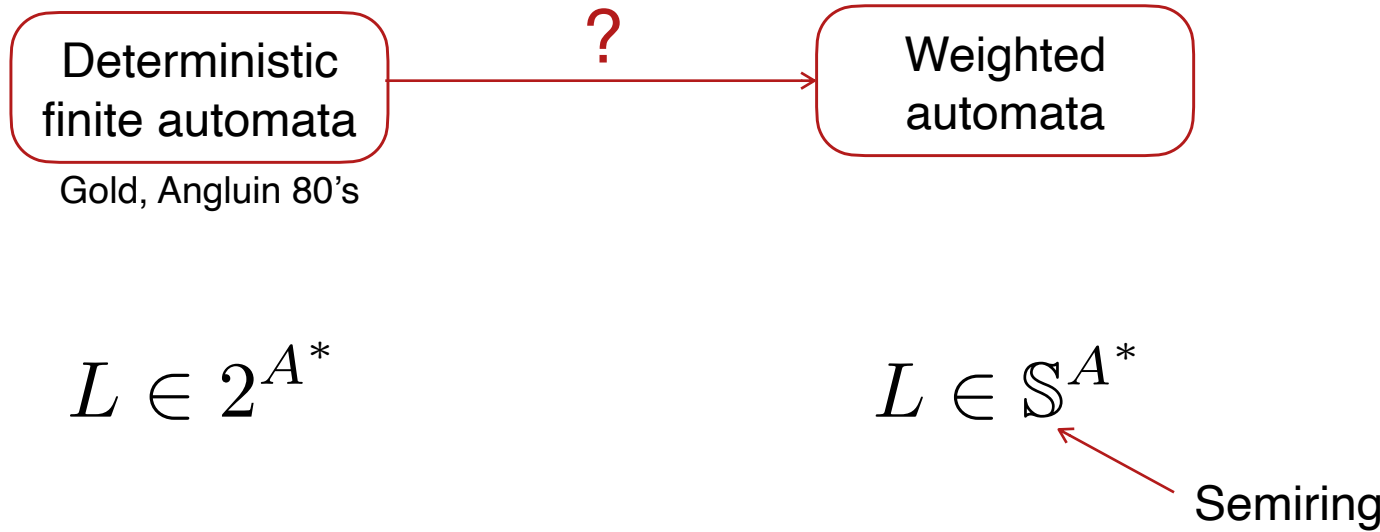
$$L \in \mathbb{S}^{A^*}$$

← Semiring



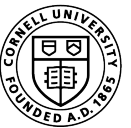
Active learning

Infer automaton through oracle (membership and equivalence queries)



L^* for DFAs

L^* learns minimal DFA for a *regular language* assuming an oracle that answers:

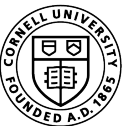


L^* for DFAs

L^* learns minimal DFA for a *regular language* assuming an oracle that answers:

Membership queries

$$w \in \mathcal{L}?$$



L^* for DFAs

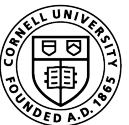
L^* learns minimal DFA for a *regular language* assuming an oracle that answers:

Membership queries

$$w \in \mathcal{L}?$$

Equivalence queries

$$\mathcal{L}(H) = \mathcal{L}?$$



L^* for DFAs

L^* learns minimal DFA for a *regular language* assuming an oracle that answers:

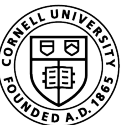
Membership queries

$$w \in \mathcal{L}?$$

Equivalence queries

$$\mathcal{L}(H) = \mathcal{L}?$$

↖ No = counter-example



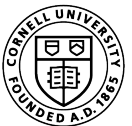
L^* for DFAs

$$S, E \subseteq A^*$$

		E	
		ε	a
S	ε	1	0
	a	0	1
	aa	1	0
$S \cdot A$	aaa	0	1

$\mathcal{L} = \{a^n \mid n \text{ is even}\}$

$aa \cdot a \notin \mathcal{L}$



L* for DFAs

$$S, E \subseteq A^*$$

		E	
		ε	a
S	ε	1	0
	a	0	1
	aa	1	0
$S \cdot A$	aaa	0	1

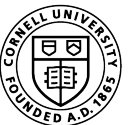
$\mathcal{L} = \{a^n \mid n \text{ is even}\}$

$aa \cdot a \notin \mathcal{L}$

$$S, E \leftarrow \{\varepsilon\}$$

Repeat until no more counterexamples:

1. Close table (changing S)
2. Equivalence query
3. Add suffixes of counterexample to E



L^* for DFAs, example

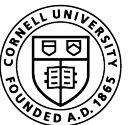
$$A = \{a\}$$

$$a^n \in \mathcal{L} \iff n \equiv 0 \pmod{3}$$

$$S, E \leftarrow \{\varepsilon\}$$

Repeat until no more counterexamples:

1. Close table (changing S)
2. Equivalence query
3. Add suffixes of counterexample to E

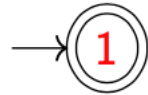


L^* for DFAs, example

$$A = \{a\}$$

$$a^n \in \mathcal{L} \iff n \equiv 0 \pmod{3}$$

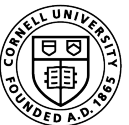
	ε
ε	1
a	0



→ $S, E \leftarrow \{\varepsilon\}$

Repeat until no more counterexamples:

1. Close table (changing S)
2. Equivalence query
3. Add suffixes of counterexample to E

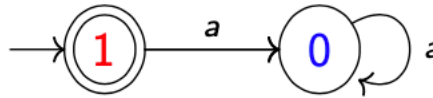


L^* for DFAs, example

$$A = \{a\}$$

$$a^n \in \mathcal{L} \iff n \equiv 0 \pmod{3}$$

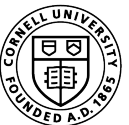
	ε
ε	1
a	0
aa	0



$$S, E \leftarrow \{\varepsilon\}$$

Repeat until no more counterexamples:

- ➔
1. Close table (changing S)
 2. Equivalence query
 3. Add suffixes of counterexample to E

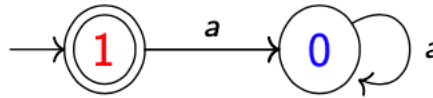


L* for DFAs, example

$$A = \{a\}$$

$$a^n \in \mathcal{L} \iff n \equiv 0 \pmod{3}$$

	ϵ
ϵ	1
a	0
aa	0

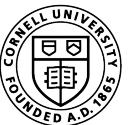


Counterexample: aaa

$$S, E \leftarrow \{\epsilon\}$$

Repeat until no more counterexamples:

1. Close table (changing S)
- 2. Equivalence query
3. Add suffixes of counterexample to E

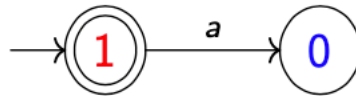


L* for DFAs, example

$$A = \{a\}$$

$$a^n \in \mathcal{L} \iff n \equiv 0 \pmod{3}$$

	ϵ	a	aa	aaa
ϵ	1	0	0	1
a	0	0	1	0
aa	0	1	0	0

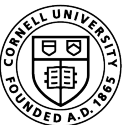


Counterexample: aaa

$$S, E \leftarrow \{\epsilon\}$$

Repeat until no more counterexamples:

1. Close table (changing S)
2. Equivalence query
3. Add suffixes of counterexample to E

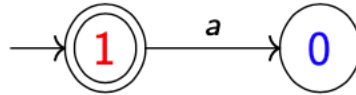


L* for DFAs, example

$$A = \{a\}$$

$$a^n \in \mathcal{L} \iff n \equiv 0 \pmod{3}$$

	ϵ	a	aa	aaa
ϵ	1	0	0	1
a	0	0	1	0
aa	0	1	0	0

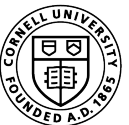


Counterexample: aaa

$$S, E \leftarrow \{\epsilon\}$$

Repeat until no more counterexamples:

1. Close table (changing S)
2. Equivalence query
3. Add suffixes of counterexample to E

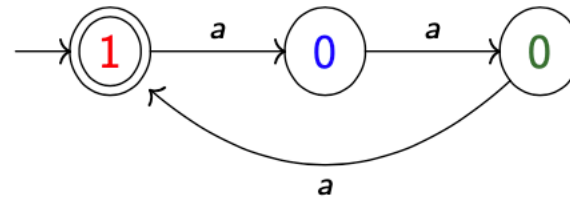


L* for DFAs, example

$$A = \{a\}$$

$$a^n \in \mathcal{L} \iff n \equiv 0 \pmod{3}$$

	ε	a	aa	aaa
ε	1	0	0	1
a	0	0	1	0
aa	0	1	0	0
aaa	1	0	0	1

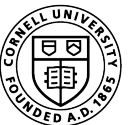


Counterexample: aaa

$$S, E \leftarrow \{\varepsilon\}$$

Repeat until no more counterexamples:

- ➔ 1. Close table (changing S)
- 2. Equivalence query
- 3. Add suffixes of counterexample to E

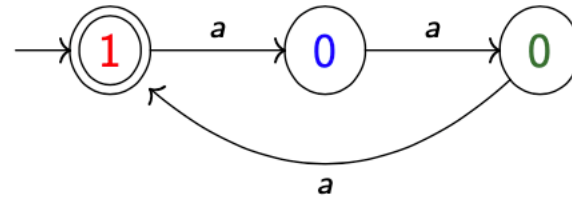


L* for DFAs, example

$$A = \{a\}$$

$$a^n \in \mathcal{L} \iff n \equiv 0 \pmod{3}$$

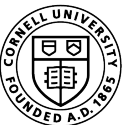
	ε	a	aa	aaa
ε	1	0	0	1
a	0	0	1	0
aa	0	1	0	0
aaa	1	0	0	1



$$S, E \leftarrow \{\varepsilon\}$$

Repeat until no more counterexamples:

1. Close table (changing S)
- 2. Equivalence query
3. Add suffixes of counterexample to E

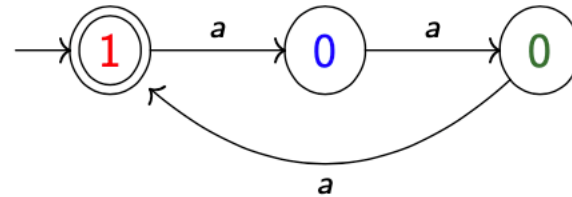


L* for DFAs, example

$$A = \{a\}$$

$$a^n \in \mathcal{L} \iff n \equiv 0 \pmod{3}$$

	ε	a	aa	aaa
ε	1	0	0	1
a	0	0	1	0
aa	0	1	0	0
aaa	1	0	0	1

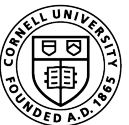


Terminates!

$$S, E \leftarrow \{\varepsilon\}$$

➔ **Repeat until no more counterexamples:**

1. Close table (changing S)
2. Equivalence query
3. Add suffixes of counterexample to E



DFAs vs WFAs

$\mathbb{S}$ semiring (e.g. $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{Z}$, $\mathbb{N}$, 2), FQ free semimodule over Q

DFA

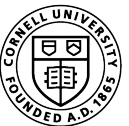
initial state in Q

$$\begin{array}{c} Q \\ \downarrow \\ 2 \times Q^A \end{array}$$

WFA

initial state in FQ

$$\begin{array}{c} Q \\ \downarrow \\ \mathbb{S} \times (FQ)^A \end{array}$$



DFAs vs WFAs

$\mathbb{S}$ semiring (e.g. $\mathbb{R}, \mathbb{Q}, \mathbb{Z}, \mathbb{N}, 2$), FQ free semimodule over Q

DFA

initial state in Q

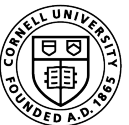
$$\begin{array}{c} Q \\ \downarrow \\ 2 \times Q^A \end{array}$$

WFA

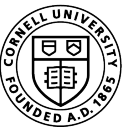
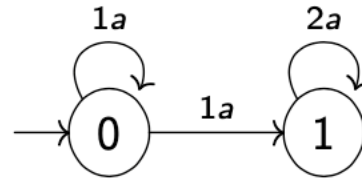
initial state in FQ

$$\begin{array}{c} Q \\ \downarrow \\ \mathbb{S} \times (FQ)^A \end{array}$$

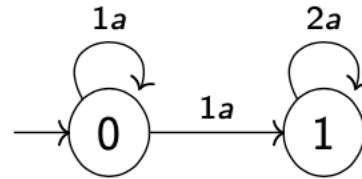
Accepts weighted language $A^* \rightarrow \mathbb{S}$
Multiplies along the path with final output
Sums over paths



DFAs vs WFAs



DFAs vs WFAs

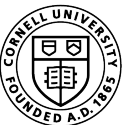


$$\mathcal{L}(\varepsilon) = 0$$

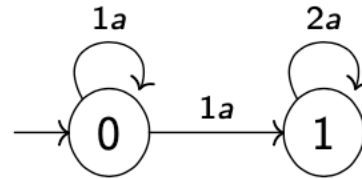
$$\mathcal{L}(a) = 1 \cdot 0 + 1 \cdot 1 = 1$$

$$\mathcal{L}(aa) = 1 \cdot 1 \cdot 0 + 1 \cdot 1 \cdot 1 + 1 \cdot 2 \cdot 1 = 3$$

$$\mathcal{L}(aaa) = 1 \cdot 1 \cdot 1 \cdot 0 + 1 \cdot 1 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot 2 \cdot 1 + 1 \cdot 2 \cdot 2 \cdot 1 = 7$$



DFAs vs WFAs



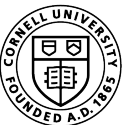
$$\mathcal{L}(\varepsilon) = 0$$

$$\mathcal{L}(a) = 1 \cdot 0 + 1 \cdot 1 = 1$$

$$\mathcal{L}(aa) = 1 \cdot 1 \cdot 0 + 1 \cdot 1 \cdot 1 + 1 \cdot 2 \cdot 1 = 3$$

$$\mathcal{L}(aaa) = 1 \cdot 1 \cdot 1 \cdot 0 + 1 \cdot 1 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot 2 \cdot 1 + 1 \cdot 2 \cdot 2 \cdot 1 = 7$$

$$\mathcal{L}(a^n) = 2^n - 1$$



Learning WFAs

Table:

Cells have values from semiring rather than 0,1

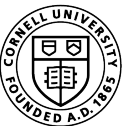
Membership Queries:

Return weight associated with word

Equivalence Queries:

Submit hypothesis WFA

Counterexample = work on which weight differs

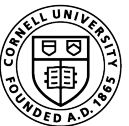


Learning WFAs

$S, E \leftarrow \{\varepsilon\}$

Repeat until no more counterexamples:

1. Close table (changing S)
2. Equivalence query
3. Add suffixes of counterexample to E



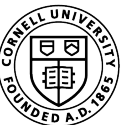
Learning WFAs

$$S, E \leftarrow \{\varepsilon\}$$

Repeat until no more counterexamples:

1. Close table (changing S)
2. Equivalence query
3. Add suffixes of counterexample to E

each lower row a linear combination of upper rows



Learning WFAs

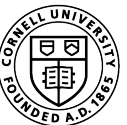
$$S, E \leftarrow \{\varepsilon\}$$

Repeat until no more counterexamples:

1. Close table (changing S)
2. Equivalence query
3. Add suffixes of counterexample to E

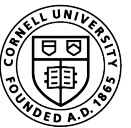
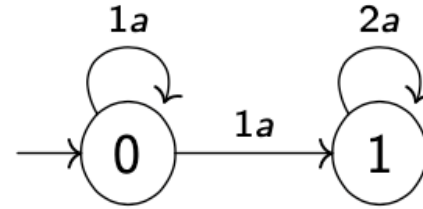
each lower row a linear combination of upper rows

Requirement for semiring: solving linear systems of equations should be computable.



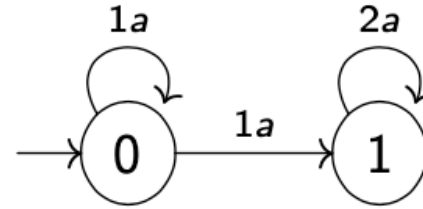
Learning WFAs, example

$$\mathcal{L}(a^n) = 2^n - 1$$

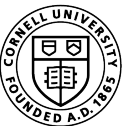
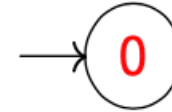


Learning WFAs, example

$$\mathcal{L}(a^n) = 2^n - 1$$

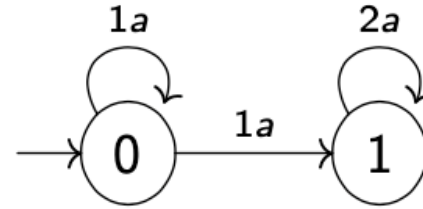


	ε
ε	0
a	1

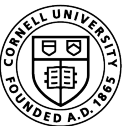
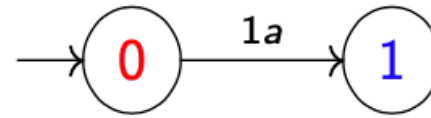


Learning WFAs, example

$$\mathcal{L}(a^n) = 2^n - 1$$

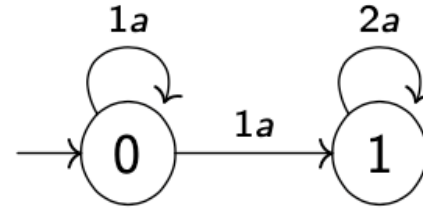


	ϵ
ϵ	0
a	1
aa	3

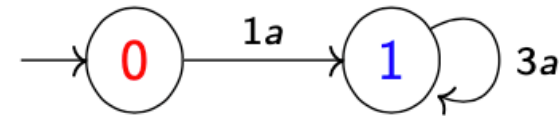


Learning WFAs, example

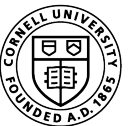
$$\mathcal{L}(a^n) = 2^n - 1$$



	ε
ε	0
a	1
aa	3

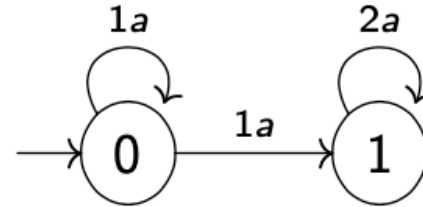


Counterexample: aaa

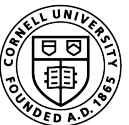
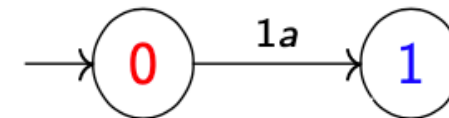


Learning WFAs, example

$$\mathcal{L}(a^n) = 2^n - 1$$

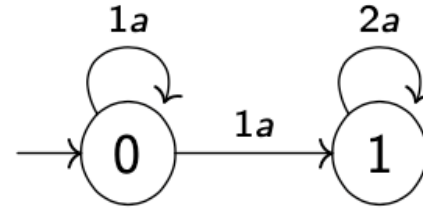


	ε	a	aa	aaa
ε	0	1	3	7
a	1	3	7	15
aa	3	7	15	31

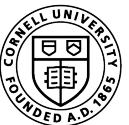
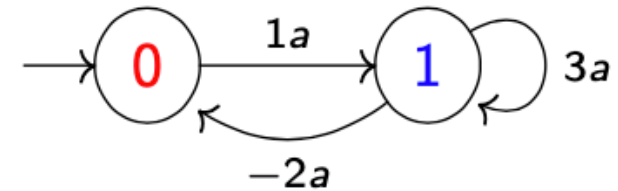


Learning WFAs, example

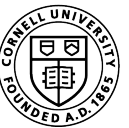
$$\mathcal{L}(a^n) = 2^n - 1$$



	ε	a	aa	aaa
ε	0	1	3	7
a	1	3	7	15
aa	3	7	15	31

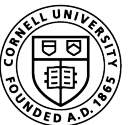


Does it terminate?



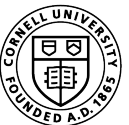
Does it terminate?

L* for DFAs: termination consequence of regular languages having finitely many Myhill-Nerode classes = minimal DFA



Does it terminate?

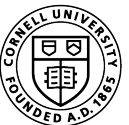
L* for DFAs: termination consequence of regular languages having finitely many Myhill-Nerode classes = minimal DFA



Does it terminate?

L* for DFAs: termination consequence of regular languages having finitely many Myhill-Nerode classes = minimal DFA

L* for WFAs: depends on the semiring

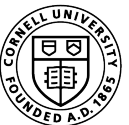


Does it terminate?

L* for DFAs: termination consequence of regular languages having finitely many Myhill-Nerode classes = minimal DFA

L* for WFAs: depends on the semiring

***Field:** terminates as a consequence of basis existence, our algorithm instantiates one of Bergadano and Varricchio (1996)*



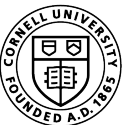
Does it terminate?

L* for DFAs: termination consequence of regular languages having finitely many Myhill-Nerode classes = minimal DFA

L* for WFAs: depends on the semiring

***Field:** terminates as a consequence of basis existence, our algorithm instantiates one of Bergadano and Varricchio (1996)*

***Boolean semiring:** WFAs = NFAs our algorithm instantiates one of Bollig et al (2009)*



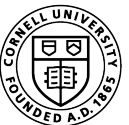
Does it terminate?

L* for DFAs: termination consequence of regular languages having finitely many Myhill-Nerode classes = minimal DFA

L* for WFAs: depends on the semiring

***Field:** terminates as a consequence of basis existence, our algorithm instantiates one of Bergadano and Varricchio (1996)*

***Boolean semiring:** WFAs = NFAs our algorithm instantiates one of Bollig et al (2009)*



Does it terminate?

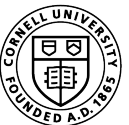
L* for DFAs: termination consequence of regular languages having finitely many Myhill-Nerode classes = minimal DFA

L* for WFAs: depends on the semiring

***Field:** terminates as a consequence of basis existence, our algorithm instantiates one of Bergadano and Varricchio (1996)*

***Boolean semiring:** WFAs = NFAs our algorithm instantiates one of Bollig et al (2009)*

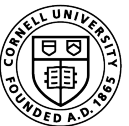
NFAs have no minimal representative!



Learning WFAs

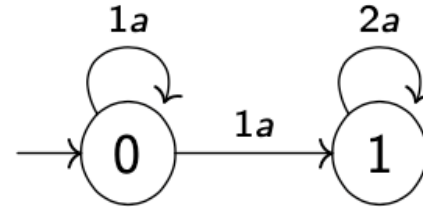
Does it terminate for any semiring?

No.

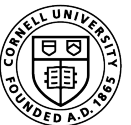


The bad semiring

$$\mathcal{L}(a^n) = 2^n - 1$$

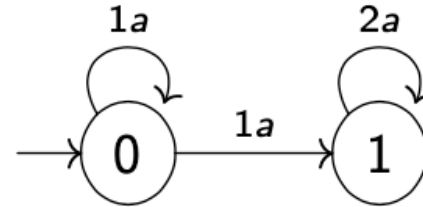


Learning over $\mathbb{Q}$

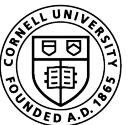
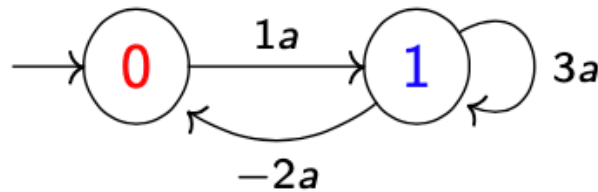


The bad semiring

$$\mathcal{L}(a^n) = 2^n - 1$$

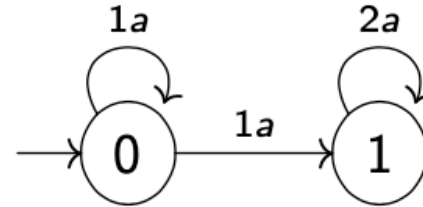


Learning over $\mathbb{Q}$

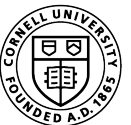
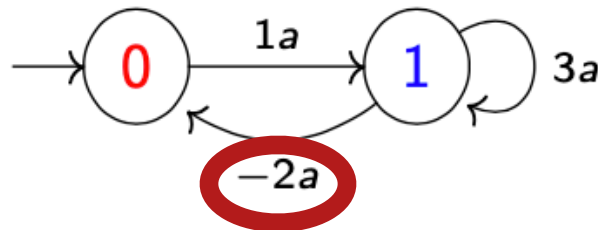


The bad semiring

$$\mathcal{L}(a^n) = 2^n - 1$$

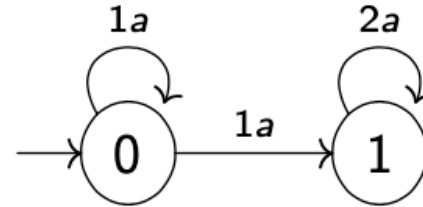


Learning over $\mathbb{Q}$

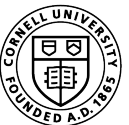
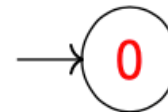


The bad semiring

$$\mathcal{L}(a^n) = 2^n - 1$$

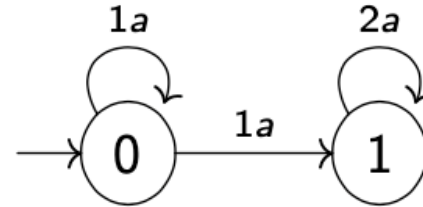


	ε	a
ε	0	1
a	1	3

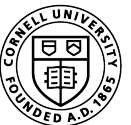
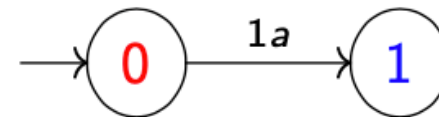


The bad semiring

$$\mathcal{L}(a^n) = 2^n - 1$$

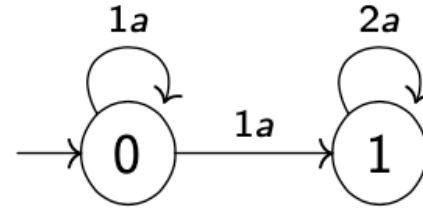


	ε	a
ε	0	1
a	1	3
aa	3	7

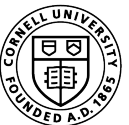
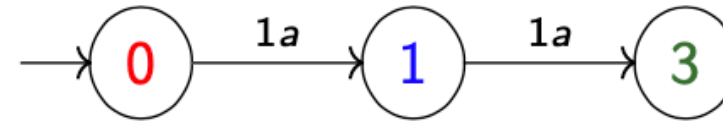


The bad semiring

$$\mathcal{L}(a^n) = 2^n - 1$$

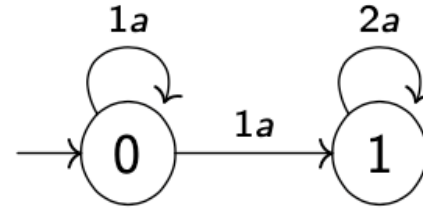


	ϵ	a
ϵ	0	1
a	1	3
aa	3	7
aaa	7	15

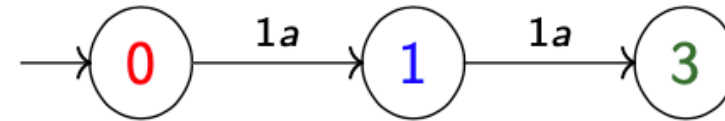


The bad semiring

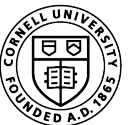
$$\mathcal{L}(a^n) = 2^n - 1$$



	ϵ	a
ϵ	0	1
a	1	3
aa	3	7
aaa	7	15



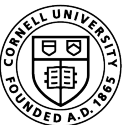
...



Termination issue

The algorithm approximates the **Hankel matrix** of the language. Linear combinations of rows in:

	ϵ	a	aa	aaa	$\dots$
ϵ	0	1	3	7	
a	1	3	7	15	
aa	3	7	15	31	$\dots$
aaa	7	15	31	63	
$\dots$			$\dots$		

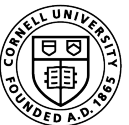


Termination issue

The algorithm approximates the **Hankel matrix** of the language. Linear combinations of rows in:

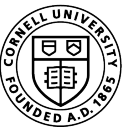
	ϵ	a	aa	aaa	$\dots$
ϵ	0	1	3	7	
a	1	3	7	15	
aa	3	7	15	31	$\dots$
aaa	7	15	31	63	
$\dots$			$\dots$		

This is not finitely generated (over commutative monoids)



Termination Conditions

Algorithm terminates assuming:



Termination Conditions

Algorithm terminates assuming:

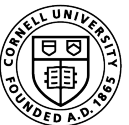
Progress measure with bound

number, increases when rows separate via extra column

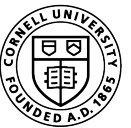
Ascending chain condition on Hankel matrix

Subsemimodule chains converge: if $S_1 \subseteq S_2 \subseteq \dots \subseteq H$ are subsemimodules, then $\exists n :$

$$S_n = S_{n+1} = S_{n+2} = \dots$$



Termination Argument

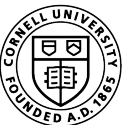


Termination Argument

Assume:

Progress measure with bound

Ascending chain condition on Hankel Matrix



Termination Argument

Assume:

Progress measure with bound

Ascending chain condition on Hankel Matrix

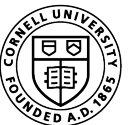
Then:

Modules generated by (S_n, A^*) form chain below Hankel matrix

Converges, from that point on closedness guaranteed

Bounded progress measure $\Rightarrow$ finitely many counterexamples

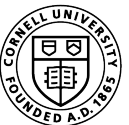
Counter example always leads to closedness defect or rows distinguished by new column



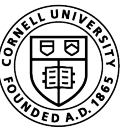
Summary

Main ingredients for effective terminating algorithm:

1. Progress measure with bound
2. Ascending chain condition on Hankel matrix
3. Procedure to determine/fix closedness: solvability of finite system of linear equations



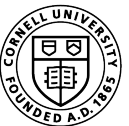
Learning over a field



Learning over a field

1. Progress measure with bound

- Dimension of vector space spanned by table
- $\leq$ minimal WFA size



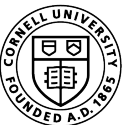
Learning over a field

1. Progress measure with bound

- Dimension of vector space spanned by table
- $\leq$ minimal WFA size

2. Ascending chain condition on Hankel matrix

- Vector space dimension increases with strict inclusion
- Minimal WFA size = Hankel matrix dimension



Learning over a field

1. Progress measure with bound

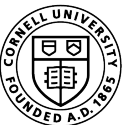
- Dimension of vector space spanned by table
- $\leq$ minimal WFA size

2. Ascending chain condition on Hankel matrix

- Vector space dimension increases with strict inclusion
- Minimal WFA size = Hankel matrix dimension

3. Procedure for closedness: solvability of finite system of linear equations

- Gaussian elimination



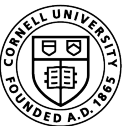
Learning over a finite semiring



Learning over a finite semiring

1. Progress measure with bound

- Set size of semimodule spanned by table
- $\leq$ determinization of correct automaton



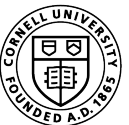
Learning over a finite semiring

1. Progress measure with bound

- Set size of semimodule spanned by table
- $\leq$ determinization of correct automaton

2. Ascending chain condition on Hankel matrix

- Hankel matrix size $\leq$ determinization of correct automaton



Learning over a finite semiring

1. Progress measure with bound

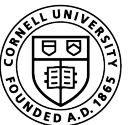
- Set size of semimodule spanned by table
- $\leq$ determinization of correct automaton

2. Ascending chain condition on Hankel matrix

- Hankel matrix size $\leq$ determinization of correct automaton

3. Procedure for closedness: solvability of finite system of linear equations

- Try all linear combinations of rows



Learning over a finite semiring

1. Progress measure with bound

- Set size of semimodule spanned by table
- $\leq$ determinization of correct automaton

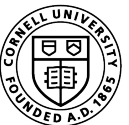
2. Ascending chain condition on Hankel matrix

- Hankel matrix size $\leq$ determinization of correct automaton

3. Procedure for closedness: solvability of finite system of linear equations

- Try all linear combinations of rows

Fields and finite semirings OK, can we do more?



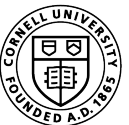
Principal ideal domains

Principal ideal domain = *integral domain* with *all ideals principal*

Integral domain: commutative ring, $ab=0 \Rightarrow a=0 \vee b=0$

All ideals principal: generated by one element

Examples: $\mathbb{Z}$, $\mathbb{Z}[i]$, $K[x]$ for K a field



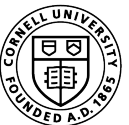
Modules over PID

A module is free if and only if it is torsion free: $pm=0 \Rightarrow p=0 \vee m=0$

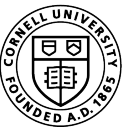
A submodule of a free and finitely generated module is

- Free and **finitely** generated
- With **smaller** (or equal) rank

If a finitely generated free module is a quotient of another, its rank is smaller or equal



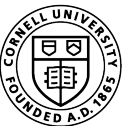
Learning over PIDs



Learning over PIDs

1. Progress measure with bound

- Rank of the module spanned by the table (which is free and f.g!)
- $\leq$ rank of Hankel matrix



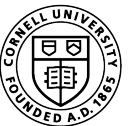
Learning over PIDs

1. Progress measure with bound

- Rank of the module spanned by the table (which is free and f.g!)
- $\leq$ rank of Hankel matrix

2. Ascending chain condition on Hankel matrix

- yes



Learning over PIDs

1. Progress measure with bound

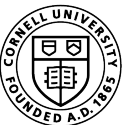
- Rank of the module spanned by the table (which is free and f.g!)
- $\leq$ rank of Hankel matrix

2. Ascending chain condition on Hankel matrix

- yes

3. Procedure for closedness: solvability of finite system of linear equations

- Solve equations via Smith normal form (exists for PIDs), some further assumptions on computability (hold for integers)



Learning over PIDs

1. Progress measure with bound

- Rank of the module spanned by the table (which is free and f.g!)
- $\leq$ rank of Hankel matrix

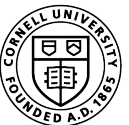
2. Ascending chain condition on Hankel matrix

- yes

3. Procedure for closedness: solvability of finite system of linear equations

- Solve equations via Smith normal form (exists for PIDs), some further assumptions on computability (hold for integers)

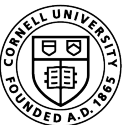
Learning algorithm terminates for the integers!



Key takeaway

Active learning algorithm for WFAs:

1. Termination depends on properties of the semiring: works for fields, finite semirings, integers, not naturals.
2. Solvability of finite systems of linear equations essential.



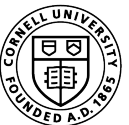
Key takeaway

Active learning algorithm for WFAs:

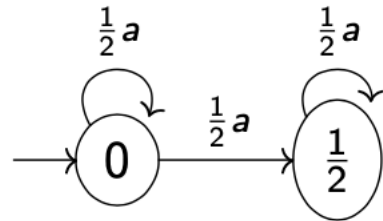
1. Termination depends on properties of the semiring: works for fields, finite semirings, integers, not naturals.
2. Solvability of finite systems of linear equations essential.

How about probabilistic automata?

$$A^* \rightarrow [0, 1]$$

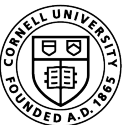
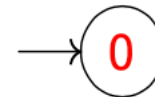


Probabilistic automata

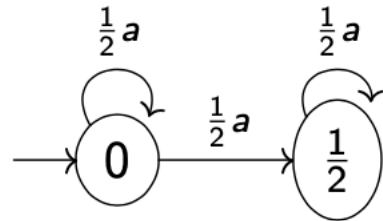


$$\mathcal{L}(a^n) = \frac{n}{2^{n+1}}$$

	ϵ
ϵ	0
a	0.25

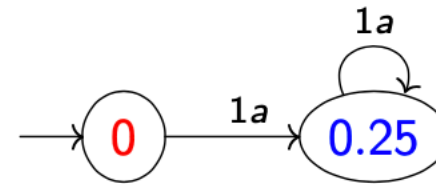


Probabilistic automata

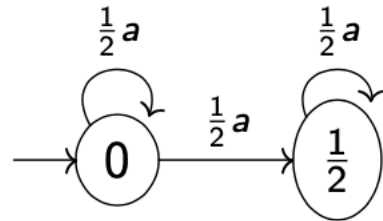


$$\mathcal{L}(a^n) = \frac{n}{2^{n+1}}$$

	ε
ε	0
a	0.25
aa	0.25

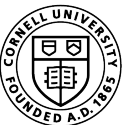


Probabilistic automata

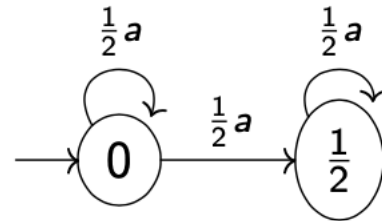


$$\mathcal{L}(a^n) = \frac{n}{2^{n+1}}$$

	ϵ	a
ϵ	0	0.25
a	0.25	0.25
aa	0.25	0.1875

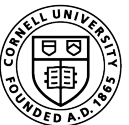


Probabilistic automata

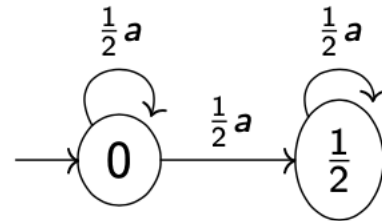


$$\mathcal{L}(a^n) = \frac{n}{2^{n+1}}$$

	ε	a
ε	0	0.25
a	0.25	0.25
aa	0.25	0.1875
aaa	0.1875	0.125

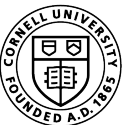


Probabilistic automata

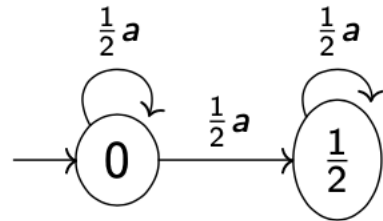


$$\mathcal{L}(a^n) = \frac{n}{2^{n+1}}$$

	ϵ	a	Ratio
ϵ	0	0.25	0
a	0.25	0.25	1
aa	0.25	0.1875	1.333...
aaa	0.1875	0.125	1.5



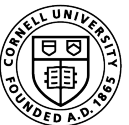
Probabilistic automata



$$\mathcal{L}(a^n) = \frac{n}{2^{n+1}}$$

	ϵ	a	Ratio
ϵ	0	0.25	0
a	0.25	0.25	1
aa	0.25	0.1875	1.333...
aaa	0.1875	0.125	1.5

Problem: rows of the table are not finitely generated

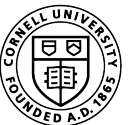


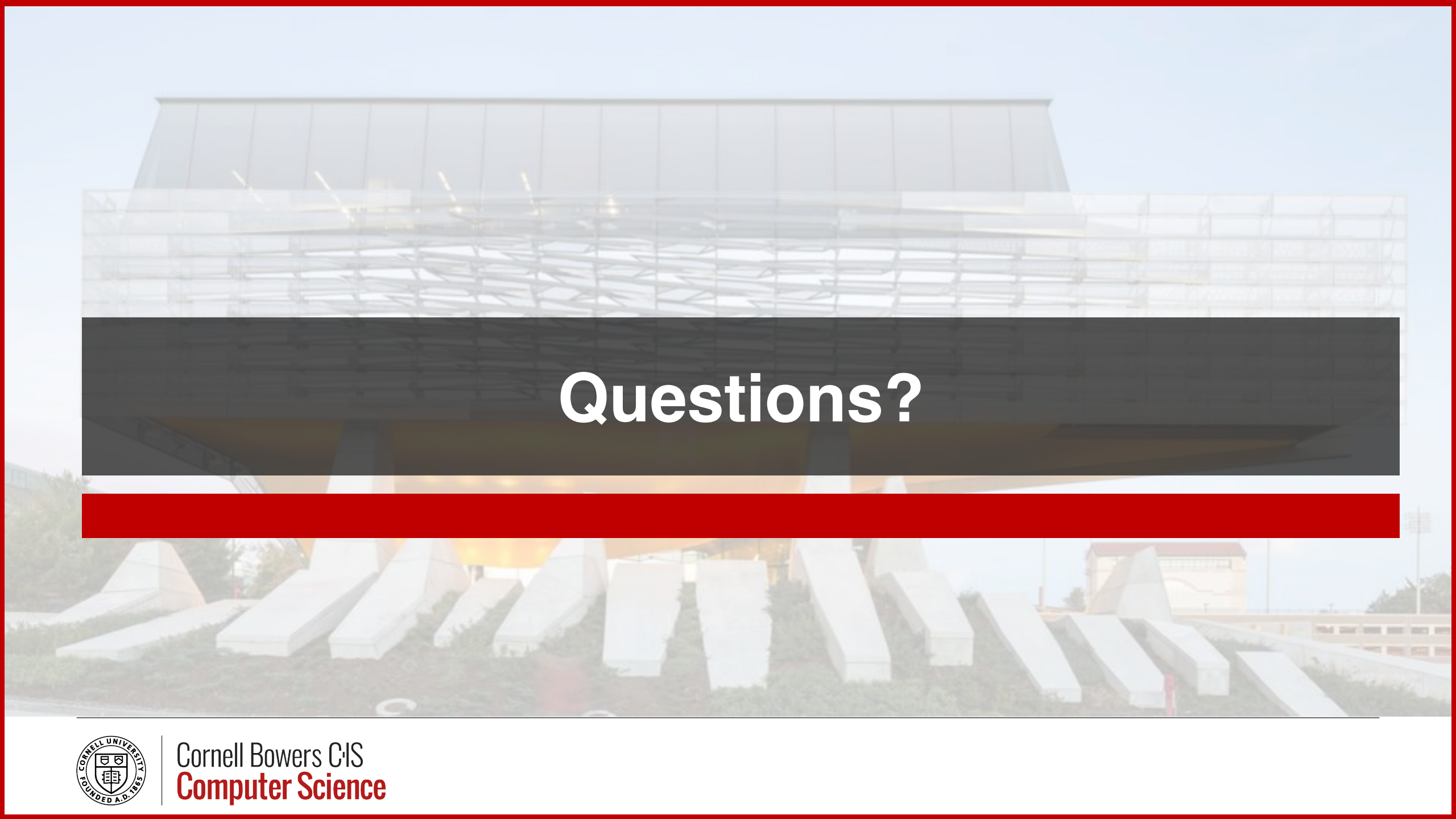
Probabilistic automata, partial solutions

Ongoing work with Leon Witzman

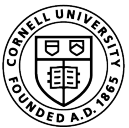
- If we know the number of states of target automaton, there exists an active learning algorithm (that uses at most exponential space);
- Even if the real automaton has rational coefficients the systems we solve might lack rational solutions (real algebraic numbers);
- Use a quadratic program to approximate solution;
- Counterexample handling (to avoid assuming number of states) unclear;
- Learning happens in convex sets, can we use this algebraic structure?

???



A photograph of the Cornell University Bowers Hall, a modern building with a large glass facade and a prominent concrete base. The building is set against a clear blue sky. A large black rectangular box is overlaid on the middle of the image, containing the word "Questions?". Below this box is a solid red horizontal bar.

Questions?



Cornell Bowers C-IS
Computer Science