Inferring Regular Languages & $\omega$-Languages

Dana Fisman
Ben-Gurion University

based on joint works with

Dana Angluin, Udi Boker & Sarah Eisenstat
Challenges:
- Hard to characterize using a logical calculous
- Complete bugless spec, really!? 

Specification
- High Level
- What?
- Declarative
- Ex: temporal logic

Synthesizer

Correct by construction

System

Implementation
- Low Level
- How?
- Procedural/Executable
- Ex: reactive system
A specification scale

{ e_1, e_2, e_3, ... }

Examples

Synthesizer

Partial implementation

Partial spec

Complete rigorous mathematical specification
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

Is \( w \) in \( L \)?
- Yes / No

Is \([H]\) same as \( L \)?
- Yes / No, c.e: \( w' \)

Learner

Teacher
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 $L^\omega$, an alg. that learns regular languages of infinite words ($\omega$-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

Finite Words

\{ e_1, e_2, e_3, \ldots \}

AL*

AFA

[Angluin, Eisenstat & Fisman IJCAI15]
### What are alternating automata?

<table>
<thead>
<tr>
<th>Transition Type</th>
<th>from state</th>
<th>upon reading</th>
<th>to state(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>s1</td>
<td>c</td>
<td>s2</td>
</tr>
<tr>
<td>Non-Deterministic</td>
<td>s1, s3</td>
<td>c</td>
<td>s2</td>
</tr>
<tr>
<td>Universal</td>
<td>s1, s3, s4</td>
<td>c</td>
<td>s1, s2</td>
</tr>
</tbody>
</table>

[Diagram showing transitions and states]
### What are alternating automata?

<table>
<thead>
<tr>
<th>Transition Type</th>
<th>from state</th>
<th>upon reading</th>
<th>to state(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>$s_1$</td>
<td>$c$</td>
<td>$s_2$</td>
</tr>
<tr>
<td>Non-Deterministic</td>
<td>$s_1$</td>
<td>$c$</td>
<td>$s_3$ or $s_4$</td>
</tr>
</tbody>
</table>

**Diagram:**

- **Deterministic:**
  - $s_1$ to $s_2$ upon reading $c$.

- **Non-Deterministic:**
  - $s_1$ to $s_3$ or $s_4$ upon reading $c$.

- **Universal:**
  - $s_1$ to $s_3$ and $s_4$ upon reading $c$.

- **Alternating:**
  - $s_1$ to $(s_3$ or $s_4$) upon reading $c$.
### What are alternating automata?

<table>
<thead>
<tr>
<th>Transition Type</th>
<th>from state</th>
<th>upon reading</th>
<th>to state(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>s1</td>
<td>c</td>
<td>s2</td>
</tr>
<tr>
<td>Non-Deterministic</td>
<td>s1</td>
<td>c</td>
<td>s3 or s4</td>
</tr>
<tr>
<td>Universal</td>
<td>s1</td>
<td>c</td>
<td>s3 and s4</td>
</tr>
</tbody>
</table>

What are alternating automata?

![Diagram showing transitions and states]
What are alternating automata?

<table>
<thead>
<tr>
<th>Transition Type</th>
<th>from state</th>
<th>upon reading</th>
<th>to state(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>s1</td>
<td>c</td>
<td>s2</td>
</tr>
<tr>
<td>Non-Deterministic</td>
<td>s1</td>
<td>c</td>
<td>s3 or s4</td>
</tr>
<tr>
<td>Universal</td>
<td>s1</td>
<td>c</td>
<td>s3 and s4</td>
</tr>
<tr>
<td>Alternating</td>
<td>s1</td>
<td>c</td>
<td>(s3 or s4) and s2</td>
</tr>
</tbody>
</table>
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 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.
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**

$L = aba^*$

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

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

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

$ab \sim_L abaaa$
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 AFA aren'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 $L_1, L_2, ..., L_n$ 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

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

May be exponentially bigger than
May be doubly exponentially bigger than

Same size relations in the residual case

[Denis et al. 2001]
[Duality]
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 table of residual languages

<table>
<thead>
<tr>
<th></th>
<th>ε</th>
<th>a</th>
<th>b</th>
<th>aa</th>
<th>ab</th>
<th>ab</th>
<th>bb</th>
<th>aaa</th>
<th>aab</th>
<th>aba</th>
<th>abb</th>
<th>baa</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

all the suffixes of ab that are in L i.e. ab⁻¹L

By Myhill-Nerode the number of distinct rows is finite.
The table of residual languages

The number of distinct columns is also finite.

We call it the column index.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>aa</th>
<th>ab</th>
<th>ab</th>
<th>bb</th>
<th>aaa</th>
<th>aab</th>
<th>aba</th>
<th>abb</th>
<th>baa</th>
<th>bab</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Enumeration of all strings

The number of distinct columns is also finite.

We call it the column index.
### L* Data Structure

#### An Observation

**Table:**

<table>
<thead>
<tr>
<th>$s_i$</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
<th>$e_5$</th>
<th>$e_6$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$s_3$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_4$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_5$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Strings: candidate state representatives

Strings: experiments to distinguish states

$M_{i,j} = \begin{cases} 
1 & \text{if } s_i e_j \in L \\
0 & \text{otherwise} 
\end{cases}$
A closed observation table \( T = (S, E, M) \) is **closed** w.r.t a subset \( B \subseteq S \).

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 \( AL^* \).
According to $L^*$ i.e. when using DFAs

<table>
<thead>
<tr>
<th>$S$</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
<th>$e_5$</th>
<th>$e_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$a$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$ab$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$aa$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$aaa$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$aab$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Same

Same

Same

Same
N-Covered

According to $\text{NL}^*$ i.e. when using NFA's

<table>
<thead>
<tr>
<th>$S$</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
<th>$e_5$</th>
<th>$e_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$a$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$ab$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$aa$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$aaa$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$aab$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Expressible as bitwise-or of some rows in $B$

$b = (\varepsilon \lor a)$
According to \textit{AL*} i.e. when using \textit{AFAs}

\begin{align*}
\text{Expressible as a monotone combination of some rows in } B
\end{align*}

\begin{align*}
b &= (\varepsilon \land a) \\
ab &= (\varepsilon \land a) \lor aa
\end{align*}

\begin{tabular}{c|cccccc}
\textit{S} & \textit{e}_1 & \textit{e}_2 & \textit{e}_3 & \textit{e}_4 & \textit{e}_5 & \textit{e}_6 \\
\hline
\varepsilon & 1 & 0 & 0 & 1 & 1 & 1 \\
a & 0 & 1 & 0 & 0 & 1 & 1 \\
b & 0 & 0 & 0 & 0 & 1 & 1 \\
ab & 1 & 1 & 1 & 0 & 1 & 1 \\
aa & 1 & 1 & 1 & 0 & 0 & 1 \\
aaa & 1 & 0 & 0 & 1 & 1 & 1 \\
aab & 0 & 1 & 0 & 0 & 1 & 1 \\
\end{tabular}
From Tables to Automata

Closed and Minimal

<table>
<thead>
<tr>
<th>S</th>
<th>ε</th>
<th>e_2</th>
<th>e_3</th>
<th>e_4</th>
<th>e_5</th>
<th>e_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>aa</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>aaa</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>aab</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\varepsilon = (\varepsilon \land a) = (\varepsilon \land a) \lor aa = \varepsilon = a
\]
Need to solve

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

- **Given a set of Boolean vectors $S$, find a minimal**
  - unique union basis \textit{Poly time} [Bollig et al.]
  - Not monotone basis \textit{NP-complete} [NEW]

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

**Then both**
- $B_1 = \{001,010,101\}$
- $B_2 = \{110,101,101\}$

are minimal monotone bases.
The Learning Alg.

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

- **oracles**: MQ, EQ
- **members**: Observation table $T = (S, E, M)$, Candidate states set $P$
- **methods**: IsxClosed, IsxMinimal, x

$S = \langle \varepsilon \rangle$, $E = \langle \varepsilon \rangle$, $P = \langle \varepsilon \rangle$ and $M_\varepsilon$

Repeat

- $(a_1, s_1)$
  - if $a_1 = \text{false}$
    - $S$.AddString($a_1$)
  - else
    - $E$.AddString($a_1$)

- if $a_2 = \text{"no"}$ then
  - $P$.RemoveString($s_2$)
- else
  - $A = T.xExtractAut(P)$

  - $(a_3, s_3) = EQ(A)$
  - if $a_3 = \text{"no"}$ then
    - $T.xFind&AddCols(s_3)$
  - else
    - $A = \text{"yes"}$

**THM**: Every counterexample yields at least one new column.
Theorem

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

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

\begin{tabular}{|c|c|c|}
\hline
L* & NL* & AL* \\
\hline
EQ & n & $O(n^2)$ & m \\
MQ & $O(|\Sigma|cn^2)$ & $O(|\Sigma|cn^3)$ & $O(|\Sigma|cn^m)$ \\
\hline
\end{tabular}

where
- $n = \text{row index}$
- $m = \text{column index}$
- $c = \text{length of longest c.e.}$
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** ($\land \lor \neg$)
- **Heuristics** combining $\times L^*$'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 $\omega$-languages

\[
\begin{align*}
\{e_1 &= \text{abcdbcaadca}cbbccaaabcdaaabb \text{bcccddddeeeaaababcdd} \\
&= e_2 = \text{bbbc}dcaaaacbc \\
&= \text{ccccccccc}aabcdababababababababababcabababc \}
\end{align*}
\]

$\{ \omega \}$

$\omega\text{-Aut}$

[Angluin & Fisman ALT'14]
Coping with \( \omega \)-words

Is \[ \text{abccccccabdebbbaaaabcdaa...} \]
in \( L \)?

Learner

Teacher

prefixes

ultimately periodic words (Lasso words)
Coping with ω-words

Is wingardium laviosa\textsuperscript{ω} in L?

Learner

Teacher

ultimately periodic words (Lasso words)
Coping with ω-words

Is \( \text{wingardium laviosa}^\omega \) in \( L \)?

\( \text{wingardium laviosa laviosa laviosa laviosa laviosa ...} \)

Learner

Teacher

ultimately periodic words (Lasso words)
Coping with $\omega$-words

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

**THM:**

Two regular $\omega$-languages are equivalent iff they agree on the set of lasso words.
There are many ways to define acceptance condition for \textit{ω-Automata}:

- Büchi
- co-Büchi
- Muller
- Parity
- Rabin
- Streett

Roughly speaking, all are defined using the notion of the states visited infinitely often during a run.
Some acceptance criteria are equally expressive, some are strictly less expressive than others.

Overall picture looks like this:
Previous work on learning $\omega$-langs.

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

All regular $\omega$-languages are strictly less expressive.
L* works due to the Myhill-Nerode thm.
The major difficulty in learning \( \omega \)-languages is a lack of a corresponding Myhill-Nerode theorem for \( \omega \)-automata (of all types)
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 $2^n + 2^{2n^2+n}$
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]

\[ \sim, \sim_1, \sim_2, \sim_3, \sim_4, \sim_5 \]

Leading

Progress

Leading Right Congruence

\[ \sim \]

Plus some restriction (details omitted)
Family of DFAs (FDFA)

\[(M, P_1, P_2, P_3, P_4, P_5)\]

Leading DFA \(M\)

That restriction is removed
FDFA Acceptance

\[(u,v) \in \mathcal{F} \subseteq [M, P_1, P_2, P_3, P_4, P_5]\]
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)

- Syntactic FORC
  - May be \(\exp\) bigger than \(L\$

- Recurrent FDFA
  - At most \(\text{poly}\) bigger than \(L\$
  - May be \(\text{quadr.}\) bigger than \(L\$

- \([\text{Calbrix, Nivat & Podelski '93]}\]
- \([\text{Maler & Staiger '95]}\]
A learning algorithm $L^\omega$ that learns the full class of regular $\omega$-languages using recurrent FDFAs

- Worst-case time complexity polynomial in $L$
- Performs very well on random targets
- 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 $\omega$-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 Antonopoulos
Further Directions

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

On going work with

Rajeev Alur
Merci beaucoup pour votre attention !

Commentaires / Questions ?