Learning Regular $\omega$-Languages

Dana Angluin and Dana Fisman
The Goal

Device an algorithm for learning an unknown regular ω-language $L$. 

Device an algorithm for learning an unknown regular $\omega$-language $L$ set of infinite words ($\omega$-words)
The Goal

Device an algorithm for learning an unknown regular \( \omega \)-language \( L \) accepted by a finite automaton.
The Goal

Device an algorithm for learning an unknown regular language $L$. 

Learner
The Goal

Device an algorithm for learning an unknown regular ω-language $L$. 

Learner

Teacher
The Goal

Device an algorithm for learning an unknown regular w-language L

Learner

Teacher

membership queries

equivalence queries
Coping with $\omega$-words

Is $abccccccabdebbbaaaabcdaa...$ in $L$?

Learner

Teacher
Coping with \(\omega\)-words

Is

\[abccccccabdebbbaaaabcdaa\ldots\]

in \(L\)?

- *prefixes*
- *ultimately periodic words (Lasso words)*

**Learner**

**Teacher**
Coping with ω-words

Is wingardium laviosa\(^\omega\) in L?

Learner

Teacher

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

Is wingardium laviosa$^\omega$ in L?

wingardium laviosa laviosa laviosa laviosa laviosa laviosa ...

ultimately periodic words (Lasso words)

Learner

Teacher
Coping with \( \omega \)-words

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

**THM:**

Two regular \( \omega \)-languages are equivalent iff they agree on the set of ultimately periodic words.
The Goal

Is $uv^\omega$ in $L$?
The Goal

Is $uv^\omega$ in $L$?

Learner

Yes / No

Teacher
The Goal

Is $uv^\omega$ in $L$?

Yes / No

Is $[H]$ same as $L$?
The Goal

Is $uv^\omega$ in $L$?
- Yes / No

Is $[H]$ same as $L$?
- Yes / No, c.e: $uv^\omega$
The Goal

Is $uv^\omega$ in $L$?

Yes / No

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

Yes / No, c.e: $uv^\omega$

Learner

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

Teacher
Motivation

- Same problem for regular (finitary) languages is solved by the $L^*$ algorithm [Angluin]
- $L^*$ has found applications in many areas including AI, neural networks, geometry, data mining, verification and synthesis and many more.
- Reactive systems concerns $\omega$-languages, using $L^*$ for this limits application to safety (and excludes liveness, fairness)
Previous Work on Learning $\omega$-Langs.

- DP / DM
- DS / DR
- DB
- DC
- DWP
- DWB
- DWC
- Safe

All regular $\omega$-languages

Strictly less expressive
Previous Work on Learning $\omega$-Langs.

- $\omega$-languages
- Safe
- DP / DM
- DS / DR
- DB
- DC
- DWP
- DWC
- DWC

From Prefixes

- [de la Higuera & Janodet, 2004]
- [Jayasrirani et al, 2012]
- [Saoudi & Yokomori, 1993]

From Lassos

all regular $\omega$-languages

strictly less expressive
Previous Work on Learning $\omega$-Langs.

- DP / DM
- DS / DR
- DB
- DC
- DWP
- DWB
- DWC
- Safe

From Prefixes:
- [Maler & Pnueli, 1995]
- [Jayasrirani et al, 2012]
- [Saoudi & Yokomori, 1993]

From Lassos:
- [de la Higuera & Janodet, 2004]

All regular $\omega$-languages strictly less expressive
Previous Work on Learning \( \omega \)-Langs.

The goal is to learn all regular \( \omega \)-languages.

From Prefixes:
- [de la Higuera & Janodet, 2004]
- [Jayasrirani et al., 2012]
- [Saoudi & Yokomori, 1993]

From Lassos:
- [Maler & Pnueli, 1995]

Strictly less expressive.
Previous Work on Learning $\omega$-Langs.

Turns out:
- $L^\$ - the pairs of good lasso words is a regular language on finite words [CNP93]

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

From Prefixes
- [Jayasrirani et al, 2012]

From Lassos
- [Saoudi & Yokomori, 1993]

DWP

DP / DM
DS / DR

DB

DC

DWB

DWC

Safe

all regular $\omega$-languages

strictly less expressive
Previous Work on Learning $\omega$-Langs.

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

From Prefixes
- [Jayasrirani et al., 2012]

Turns out:
- $L^*$ – the pairs of good lasso words is a regular language on finite words [CNP93]
- Thus can be learnt using $L^*$ itself [F. et al. 2008]
However ...

$L_\$ [Calbrix, Nivat & Podelski '93]

[Syntactic FORC

[Maler & Staiger '93]

[Recurrent FDFA

NEW]
However ...

Syntactic FORC

May be exp. bigger than

$L_\$$

Recurrent FDFA

[Calbrix, Nivat & Podelski '93]

[Maler & Staiger '93]
However ...

Syntactic FORC \( L \$

- \text{May be} \quad \text{exp. bigger than}

- \text{May be} \quad \text{quadr. bigger than}

Recurrent FDFA

[Calbrix, Nivat & Podelski '93]

[Maler & Staiger '93]
Main Contribution

\[ L^\omega \ - \]

a learning algorithm
for all 3 representations
(periodic, syntactic, recurrent)
Why is it difficult?
Why is it difficult?

$L^*$ works due to the **Myhill-Nerode Theorem**, stating a **1-to-1** relationship between

- states of the **minimal DFA** *(deterministic finite automaton)*
- and the **syntactic right congruence** $\sim$ of a language.
Why is it difficult?

$L^*$ works due to the **Myhill-Nerode Theorem**, stating a **1-to-1** relationship between

- states of the **minimal DFA** (deterministic finite automaton)
- and the **syntactic right congruence** $\sim$ of a language.

But for $\omega$-langs, the syntactic right congruence is **not informative enough**!
For finite words

\[ x \sim_L y \text{ iff } \forall v \in \Sigma^*. \ xv \in L \iff yv \in L \]
The Syntactic Right Congruence

For finite words

\[ x \sim_L y \text{ iff } \forall v \in \Sigma^*. \; xv \in L \iff yv \in L \]

Example:

\[ L = \{ w \mid w \text{ has an even number of } a's \} \]
The Syntactic Right Congruence

For finite words

\[ x \sim_L y \text{ iff } \forall v \in \Sigma^*. \ xv \in L \iff yv \in L \]

Example:

\( L = \{ w \mid w \text{ has an even number of } a's \} \)

Then \( \sim_L \) has two equivalence classes:

- words with an odd number of \( a's \) and
- words with an even number of \( a's \)
The Syntactic Right Congruence

For finite words:

\[ x \sim_L y \iff \forall v \in \Sigma^*. \ xv \in L \iff yv \in L \]

For \( \omega \)-words:

\[ x \sim_L y \iff \forall w \in \Sigma^\omega. \ xw \in L \iff yw \in L \]
The Syntactic Right Congruence

For finite words:

\[ x \sim_L y \iff \forall v \in \Sigma^*. \ xv \in L \iff yv \in L \]

For \( \omega \)-words:

\[ x \sim_L y \iff \forall u, v \in \Sigma^*. \ xuv^\omega \in L \iff yuv^\omega \in L \]
For \( \omega \)-words:

\[ x \sim_L y \ \text{iff} \ \forall u,v \in \Sigma^*. \ xuv^\omega \in L \iff yuv^\omega \in L \]

Example:

\( L = \{ w \mid w \text{ has finitely many } a\}'s \)
The Syntactic Right Congruence

For \( \omega \)-words:

\[ x \sim_L y \iff \forall u, v \in \Sigma^*. \quad \text{xuv}^\omega \in L \iff \text{yuv}^\omega \in L \]

Example:

\( L = \{ w \mid w \text{ has finitely many } a's \} \)

Then \( \text{xuv}^\omega \in L \iff \text{uv}^\omega \) has finitely many \( a \)'s.
The Syntactic Right Congruence

For ω-words:

\[ x \sim_L y \quad \text{iff} \quad \forall u, v \in \Sigma^*. \quad xuv^\omega \in L \iff yuv^\omega \in L \]

Example:

\[ L = \{ w \mid w \text{ has finitely many } a's \} \]

Then \( xuv^\omega \in L \quad \text{iff} \quad uv^\omega \text{ has finitely many } a's. \]

Regardless of \( x. \)
The Syntactic Right Congruence

For $\omega$-words:

$$x \sim_L y \text{ iff } \forall u,v \in \Sigma^*. \ xuv^\omega \in L \iff yuv^\omega \in L$$

Example:

$$L = \{ w \mid w \text{ has finitely many } a' \text{’s} \}$$

Then $xuv^\omega \in L$ iff $uv^\omega$ has finitely many $a$’s.

Regardless of $x$. Thus $\sim_L$ has just one equivalence class.
The Syntactic Right Congruence

For $\omega$-words:

$$x \sim_L y \iff \forall u,v \in \Sigma^*. \ xuv^\omega \in L \iff yuv^\omega \in L$$

Example:

$L = \{ w \mid w \text{ has finitely many } a\text{'s} \}$

Then $xuv^\omega \in L \iff uv^\omega \text{ has finitely may } a\text{'s}.$

Regardless of $x$. Thus $\sim_L$ has just one equivalence class.

But clearly an $\omega$-automaton for $L$ requires at least two states.
Families of DFAs (FDFA)

- A non-traditional representation of $\omega$-automata
- Inspired by Maler & Staiger’s ‘97 families of right congruences (FORC)
- And their canonical representation of a Syntactic FORC
- For which they have shown a Myhill-Nerode Theorem
Family of Right Congruences [MS97]

(\sim, \sim_1, \sim_2, \sim_3, \sim_4, \sim_5)

Leading

Progress

\sim^2

\sim^1

\sim^3

\sim^4

\sim^5

Leading Right Congruence

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

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

Leading DFA

\[M\]
Family of DFAs (FDFA)

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

Leading DFA \(M\)

Leading

Progress
Family of DFAs (FDFA)

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

Leading

Progress

Leading DFA

\[M\]
Family of DFAs (FDFA)

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

Leading DFA

\(M\)

Leading

Progress

\(P_1\)

\(P_2\)

\(P_3\)
Family of DFAs (FDFA)

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

Leading DFA

Leading

Progress

\(P_1\)

\(P_2\)

\(P_3\)

\(P_4\)

Leading DFA

\(M\)
Family of DFAs (FDFA)

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

Leading DFA \(M\)

Leading

Progress

\(P_1\)

\(P_2\)

\(P_3\)

\(P_4\)

\(P_5\)
Family of DFAs (FDFA)

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

Leading DFA

That restriction removed.
FDFA Exact Acceptance

\[(u, v) \in [M, P_1, P_2, P_3, P_4, P_5] \]
FDFA Exact Acceptance

\[(u, v) \in \mathcal{F} = [M, P_1, P_2, P_3, P_4, P_5]\]
FDFA Exact Acceptance

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

Diagram:

- \(M\)
- \(P_1\)
- \(P_2\)
- \(P_3\)
- \(P_4\)
- \(P_5\)
FDFA Exact Acceptance

$u v^\omega \in \mathcal{F} \iff (u, v) \in [M, P_1, P_2, P_3, P_4, P_5]$
$L = \{ w \mid w \text{ has finitely many } a\text{'s}\}$
\( L = \{ w \mid w \text{ has finitely many } a\text{'s} \} \)

**FORC**

- No \( a\)'s
- Some \( a\)'s

\( \approx \bullet \approx 1 \)

**FDFA**

- \( P_1 \)
  - \( a \)
  - \( a,b \)

All prefixes are equally good
FORC and FDFA - Example

\[ L = \{ w \mid w \text{ has finitely many } a\text{'s}\} \]

**FORC**

- No a's
- Some a's
  \[ \approx 1 \]

**FDFA**

\[ P_1 \]

- \( a \rightarrow \)
- \( b \rightarrow \)
- \( a, b \rightarrow \)

- \( a, b \rightarrow \)
- \( 1 \rightarrow \)

- \( (abba, bbb) \) ✓
- \( (abba, bab) \) ✗

**Progress**

- All prefixes are equally good
4-state automaton
Some periods of $\lambda$ are easy to find:

- 11
- 22
- 100201
- 10020122
Some periods of $\lambda$ are easy to find:

- 11
- 22
- 100201
- 10020122

Some are hard:

- 1012
Some periods of $\lambda$ are **easy** to find:

- 11
- 22
- 100201
- 10020122

Some are **hard**:

- 1012 1012
- 1012 1012
Some periods of $\lambda$ are **easy** to find:

- 11
- 22
- 100201
- 10020122

Some are **hard**:

- 1012
- 1012 1012
- 1012 1012

The DFA for periods of $\lambda$ has 23 states

$\Rightarrow L_\$ > 23$
4-state automaton

- All periods that loop back are easy to find:
  - 11
  - 22
  - 100201
  - 10020122
**4-state automaton**

- **All periods that loop back are easy to find:**
  - 11
  - 22
  - 100201
  - 10020122

The DFA that accepts only periods of λ that loop back has only 5 states!
Saturation (Consistency)

- How can we use this without losing saturation?
Saturation (Consistency)

- How can we use this without losing saturation?

We say that a language is **saturated** if

\[ uv^\omega = xy^\omega \quad \text{implies} \quad (u,v) \in L \iff (x,y) \in L \]
Saturation (Consistency)

- How can we use this without losing saturation?

We say that a language is **saturated** if

\[ uv^\omega = xy^\omega \] implies \((u,v) \in L \iff (x,y) \in L\)

- To achieve saturation, we change the definition of **acceptance**.
FDFA Normalized Acceptance

\[(u, v) \in F \subseteq [M, P_1, P_2, P_3, P_4, P_5]\]
FDFA Normalized Acceptance

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

Normalization seeks for the smallest repetition of the period that loops back.
FDFA Normalized Acceptance

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

Normalization seeks for the smallest repetition of the period that loops back
FDFA Normalized Acceptance

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

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

$\mathcal{F} \ni (u,v) \in \{M, P_1, P_2, P_3, P_4, P_5\}$
FDFA Normalized Acceptance

\[(u,v) \in \left[M, P_1, P_2, P_3, P_4, P_5 \right]\]

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. It is saturated under normalized acceptance!
More generally

- Generalizing to arbitrary \( n \) we get the following lower bound

\[
\text{Syntactic FDFA} \quad \text{Periodic FDFA (L\$)} \quad \text{Recurrent FDFA}
\]

May be exp. bigger than
Algorithm 1: The Learner $L^\omega$

1. Initialize the leading table $\mathcal{T} = (S, \tilde{S}, E, T)$ with $S = \tilde{S} = \{\lambda\}$ and $E = \{(\lambda, \lambda)\}$.
2. CloseTable($\mathcal{T}$, ENT$_1$, DFR$_1$) and let $M = Aut_1(\mathcal{T})$.
3. forall $u \in \tilde{S}$ do
   4. Initialize the table for $u$, $\mathcal{T}_u = (S_u, \tilde{S}_u, E_u, T_u)$, with $S_u = \tilde{S}_u = E_u = \{\lambda\}$.
   5. CloseTable($\mathcal{T}_u$, ENT$_2^u$, DFR$_2^u$) and let $A_u = Aut_2(\mathcal{T}_u)$.
6. Let $(a, u, v)$ be the teacher’s response on the equivalence query $\mathcal{H} = (M, \{A_u\})$.
7. while $a = \text{“no”}$ do
   8. Let $(x, y)$ be the normalized factorization of $(u, v)$ with respect to $M$.
   9. Let $\tilde{x}$ be $M(x)$.
   10. if $MQ(x, y) \neq MQ(\tilde{x}, y)$ then
       11. $E = E \cup \text{FindDistinguishingExperiment}(x, y)$.
       12. CloseTable($\mathcal{T}$, ENT$_1$, DFR$_1$) and let $M = Aut_1(\mathcal{T})$.
       13. forall $u \in \tilde{S}$ do
           14. CloseTable($\mathcal{T}_u$, ENT$_2^u$, DFR$_2^u$) and let $A_u = Aut_2(\mathcal{T}_u)$.
   15. else
       16. $E_{\tilde{x}} = E_{\tilde{x}} \cup \text{FindDistinguishingExperiment}(\tilde{x}, y)$.
       17. CloseTable($\mathcal{T}_{\tilde{x}}$, ENT$_2^{\tilde{x}}$, DFR$_2^{\tilde{x}}$) and let $A_{\tilde{x}} = Aut_2(\mathcal{T}_{\tilde{x}})$.
       18. Let $(a, u, v)$ be the teacher’s response on equivalence query $\mathcal{H} = (M, \{A_u\})$.
19. return $\mathcal{H}$
Minimality?

- The Recurrent FDFA for a given $L$ is not necessarily the minimal FDFA for $L$.

- According to the normalized acceptance criterion a progress DFA $P_u$ should give correct results only to extensions that close a cycle to $u$. On other extensions it has freedom.

- This has the flavor of minimization with unknowns, which is NP complete.

- The Recurrent FDFA chooses to treat all don’t cares as rejecting.
Time Complexity

- Interestingly, [Klarlund, 1994] has shown that choosing a leading automaton which is more refined (bigger) than the syntactic right congruence, may yield an overall smaller FDFA.

- If we are given such a leading automaton, we can feed it to the learning algorithm, in which case it will yield the smaller FDFA.

- However, the same phenomenon can cause the learning algorithm for the syntactic/recurrent FDFAs to work as hard as the one for the periodic FDFA.
The worst case time complexity for all 3 families is thus polynomial in the size of the periodic FDFA.

Some positive results on sizes obtained via recurrent FDFA learner vs. periodic FDFA learner on randomly generated Muller automata.
Future Direction

- Find smaller canonical representations?
- Find smallest FDFA?
- Learning the leading first?
- L*-learning for (traditional) ω-automata?
- Other types of learning for ω-languages?
THE END

Thank you for your attention!

Comments or questions?

LEARNING REGULAR LANGUAGES VIA ALTERNATING AUTOMATA

Dana Angluin
Sarah Eisenstat and Dana Fisman

IJCAI'15
The worst case time complexity for all 3 families is thus polynomial in the size of the periodic FDFA.

Some positive results on sizes obtained via recurrent FDFA learner vs. periodic FDFA learner on randomly generated Muller automata.