Learning
Regular
ω-Languages

Dana Angluin and
Dana Fisman
Overview

- Motivation
- Background (ω-Automata)
- Previous work on learning regular ω-languages
- Why is it difficult to extend \( L^* \) [Angluin] to ω-languages?
  - \( L^* \) works due to the Myhill-Nerode Theorem
  - No such theorem exists for traditional ω-automata
- Three non-traditional representations for ω-langs
  - Periodic, Syntactic, Recurrent

Learning Algorithm for all three.
However ...

FDFA = Family of DFAs
Motivation
Motivation

What are $\omega$-Languages?

$\omega$-Languages are sets of infinite words (also called $\omega$-words).

What are they good for?

Regular $\omega$-Languages are the main means to model reactive systems.
What are reactive systems?

Systems with infinite ongoing behavior

A possible computation is an $\omega$-word

All possible computations are an $\omega$-language
Reasoning about reactive systems

- **Models:**
  - $\omega$-Automata
  - Temporal logic

- **Questions:**
  - Verification
  - Synthesis

- **Solving Methods:**
  - Reductions to problems about $\omega$-Automata (intersection, complementation, emptiness, etc.)

- **Practice:**
  - Tools by IBM, Intel, Cadence, Synopsis
  - Users: Intel, Nokia, AMD, TI, Apple...
Why learn ω-languages?

- **Synthesis:**
  - One can synthesize a reactive system from a given *temporal logic formula*
  - However, formulating *temporal logic formulas* characterizing a system might be difficult
  - Whereas classifying scenarios into *good* and *bad* is much easier

- **Compositional Verification:**
  - Current practice uses *L*\(^*\). Thus *drastically limiting* the *expressive power* of the obtained assumptions.
ω-Automata
A word is accepted if the run on it ends in an accepting state.

A $\omega$-word is accepted if the run on it ...
There are many ways to define acceptance condition for \( \omega \)-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:
Among the most expressive is a Muller aut.

A Muller acceptance condition is a set of sets of states $\alpha = \{F_1, F_2, \ldots, F_k\}$

It accepts a $\omega$-word, if the set of states visited infinitely often is exactly one of the $F_i$'s.

$L = \{ w \mid w \text{ has finitely many } a \text{'s} \}$
Previous Work on Learning regular $\omega$-Languages
How to cope with \( \omega \)-words?

- Some **finite** representations is needed.

- Two approaches:
  - Consider only **finite prefixes** of infinite words
  - Use **ultimately periodic words**, i.e. pairs of finite words \((u,v)\) standing for \(uv^\omega\)
How to cope with ω-words?

- Some finite representations is needed.
- Two approaches:
  - Consider only finite prefixes of infinite words
  - Use ultimately periodic words, i.e. pairs of finite words \((u, v)\) standing for \(uv\omega\)

It follows from McNaugton’s Theorem that two regular \(\omega\)-languages are equivalent iff they agree on the set of ultimately periodic words.
de la Higuera and Janodet, 2004

- positive results for polynomially learning in the limit _safe_ regular $\omega$-languages from _prefixes_
- negative results for learning any strictly subsuming class of $\omega$-languages from _prefixes_

Jayasrirani et al., 2012

- extended this to learning _bi-$\omega$-words_.
Previous Works (ultimately periodic words)

- Saoudi and Yokomori, 1993
  - Consider **ultimately periodic words**, and provide an algorithm for learning in the limit local \( \omega \)-languages, and **safe** \( \omega \)-languages.

- Maler and Pnueli, 1995
  - Provide an extension to \( L^* \), using **ultimately periodic words**, for the class of \( \omega \)-languages which are recognizable by **weak parity automata**.
No 1-to-1 correspondence between the syntactic right congruence and $\omega$-automata
Why is it so 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**!
The Syntactic Right Congruence

For finite words

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

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
For finite words:

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

For \( \omega \)-words:

\[ x \sim_L y \text{ iff } \forall u, v \in \Sigma^*. \ x u v^\omega \in L \iff y u v^\omega \in L \]
For ω-words:

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

Example:

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

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

Regardless of \( x \).

Thus \( \sim_L \) has just one equivalence class.
No 1-to-1 Correspondence!

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

Clearly, a Muller automaton for \( L \) needs at least two states.

While \( \sim_L \) has only one equivalence class, an \( \omega \)-automaton for \( L \) requires at least two states.
A non-traditional representation for $\omega$-languages for which the desired 1-to-1 correspondence exists
Family of Right Congruences

- In the quest for a representation for ω-langs for which a Myhill-Nerode THM exists [Maler and Staiger 1997] introduced

  Family of Right Congruences (FORC)

- **Idea:**
  - Consider only ultimately periodic words $uv^\omega$
  - Assume they are given as pairs $(u,v)$
  - Have a main/leading right congruence for the prefixes
  - And for every class of possibilities a right congruence for the allowed periods
Family of Right Congruences [MS97]

\((\sim, \approx_1, \approx_2, \approx_3, \approx_4, \approx_5)\)

Leading

Progress

Leading Right Congruence

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

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

That restriction is removed
FDFA Acceptance

\((u, v) \in \left[ M, P_1, P_2, P_3, P_4, P_5 \right] \)
\[ L = \{ w \mid w \text{ has finitely many } a's \} \]
Three Canonical FDFAs

3 canonical FDFAs:

- **Periodic** \(^\text{[corr. to } L^d\text{ of Calbrix, Nivat & Podelski, 1993]}\)
  - The simplest to understand

- **Syntactic** \(^\text{[Maler & Staiger, 1997]}\)
  - The coarsest possible FORC

- **Recurrent**
  - Simplification of the Syntactic
  - The smallest among the three

No contradiction here. The Recurrent is not a FORC. Save up is due to the removal of that restriction.
Three Canonical FDFAs

- All three canonical FDFAs use $\sim_L$ for the leading automaton.
- It is the smallest choice possible for the leading automaton [MS97].
- The canonical FDFAs differ in the definition of the right congruence for the progress automata.
The Progress Right Congruences

Periodic

\[ x \approx_{P}^u y \]

Syntactic

\[ x \approx_{S}^u y \]

Recurrent

\[ x \approx_{R}^u y \]
The Progress Right Congruences

**Periodic**

$x \cong_p^u y \iff u(xv) \omega \in L \iff u(yv) \omega \in L$

Considering periods starting at $u$, $x$ and $y$ should agree on all extensions

**Syntactic**

**Recurrent**
Possible Periods of $\lambda$

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

- Some are hard:
  - $1012$  
    - $1012$  $1012$
    - $1012$  $1012$
  - $12$  
    - $12$  $12$  $12$
    - $12$  $12$  $12$

The periodic progress DFA for $\lambda$ has 23 states!
The Progress Right Congruences

Periodic

Syntactic

Recurrent

\[ x \equiv^u_R y \quad \text{iff} \quad \forall v \in \Sigma^*. \quad uxv \sim_L u \quad \text{and} \quad u(xv)^\omega \in L \quad \iff \quad uyv \sim_L u \quad \text{and} \quad u(yv)^\omega \in L \]

\[ x \quad \text{and} \quad y \quad \text{should agree on extensions that loop back to} \quad u \quad \text{in the leading automaton} \]
Periods of $\lambda$ that loop back

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

The recurrent progress DFA for $\lambda$ has only 5 states!
More generally

- Generalizing to arbitrary $n$ we get the following lower bound.
Saturation (Consistency)

- But wait, wouldn’t you like to get the same result for 12 as you do for 12 12 12?

- Well... yes

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 Exact Acceptance

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

$$\mathcal{F} \{ (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

Syntactic/Recurrent FDFA are saturated under Normalized Acceptance.
Learning Algorithm for all 3 Canonical FDFAs
Learning FDFAs

- We now have a 1-to-1 correspondence between FDFA states and right congruences.

- Note that we don’t have oracles for the languages of the leading/progress DFAs. We have oracles for the \( \omega \)-language itself.
Main Idea

- Similar to $L^*$

- We learn the **leading** and **progress** DFAs simultaneously

- When we get a counter example, we **first** try to see if we can distinguish a new state in the **leading** DFA

- **Otherwise** we try to distinguish a state in the **respective progress** DFA
L* Data Structure

Observation Table

<table>
<thead>
<tr>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( s_3 )</th>
<th>( s_4 )</th>
<th>( s_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e_1 )</td>
<td>( e_2 )</td>
<td>( e_3 )</td>
<td>( \ldots )</td>
<td></td>
</tr>
</tbody>
</table>

\[
m_1 = \begin{cases} 
\text{True} & \text{if } s_1 e_1 \in L \\
\text{False} & \text{otherwise}
\end{cases}
\]
\( L^\omega \) Data Structures

**Leading Table**

<table>
<thead>
<tr>
<th>( e_1 )</th>
<th>( e_2 )</th>
<th>( e_3 )</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( u_2 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( u_3 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( u_4 )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Progress Table 1**

<table>
<thead>
<tr>
<th>( u_1 )</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( u_2 )</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( u_3 )</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( u_4 )</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**L^w Leading Table**

Same as in $L^*$ only that experiments $e_i$ are ultimately periodic words $u_i v_i^\omega$

<table>
<thead>
<tr>
<th></th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$m_1 = \begin{cases} 
\text{True} & \text{if } s_1 e_1 \in L \\
\text{False} & \text{otherwise}
\end{cases}$
## $L^\omega$ - Recurrent Progress Table

### Recurrent Observation Table

<table>
<thead>
<tr>
<th>$x$</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>($m_1, c_1$)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Membership:**

\[
m_1 = \begin{cases} 
\text{True} & \text{if } u(x_1 v_1)^\omega \in L \\
\text{False} & \text{otherwise}
\end{cases}
\]

**Cycle-Closing:**

\[
c_1 = \begin{cases} 
\text{True} & \text{if } ux_1 v_1 \sim u \\
\text{False} & \text{otherwise}
\end{cases}
\]
From tables to an algorithm

- How to define when two rows should be considered distinguished?

- Given a counter example, how do we find an experiment that distinguishes a new state?

<table>
<thead>
<tr>
<th>$u$</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>++</td>
<td>+ -</td>
<td>++</td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td>++</td>
<td>- -</td>
<td>++</td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Difficulty: when an experiment differentiates two strings with respect to membership, it might not close a cycle, thus should not necessarily imply the strings are in different equivalence classes ...
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, such a phenomenon can cause the learning algorithm for the syntactic/recurrent FDFAs to work as hard as the one for the periodic FDFA.

- The time complexity for all 3 families is thus polynomial in the size of the periodic FDFA.
Recap

- Learning ω-languages is important for verification of reactive systems.
- No learning alg. for traditional ω-automata since they don’t have a Myhill-Nerode theorem.
- We have provided a L*-style learning algorithm for all three canonical representations of ω-languages, based on the notion of FDFAs.
- While the size relations are worse case time complexity depends on the largest representation.
Future Direction

- Find smaller canonical representations?
- Find smallest FDFA?
- Learning the leading first?
- $L^*$-learning for (traditional) $\omega$-automata?
- Other types of learning for $\omega$-languages?