Learning Weighted Automata over Principal Ideal Domains

Jurriaan Rot, Radboud University
joint work with Gerco van Heerdt, Clemens Kupke, Alexandra Silva

MOVES seminar, 14 May 2020

Overview

Background

- Active learning: infer automaton through membership and equivalence queries
- Weighted automata: quantitative type of automata

Problem

What type of weighted automata can we learn?

$\mathtt{L}^\star$ setup for DFAs

Finite alphabet A

System behaviour captured by a **regular language** $\mathcal{L} \subseteq A^*$

$\mathtt{L}^\star$ learns *minimal* DFA for $\mathcal{L}$

$\mathtt{L}^\star$ setup for DFAs

Finite alphabet A

System behaviour captured by a **regular language** $\mathcal{L} \subseteq A^*$

$\mathtt{L}^\star$ learns *minimal* DFA for $\mathcal{L}$ assuming an *oracle* that answers

- **Membership queries**

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

$\mathtt{L}^\star$ setup for DFAs

Finite alphabet A

System behaviour captured by a **regular language** $\mathcal{L} \subseteq A^*$

$\mathtt{L}^\star$ learns *minimal* DFA for $\mathcal{L}$ assuming an *oracle* that answers

- **Membership queries**

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

- **Equivalence queries**

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

  Negative result $\implies$ *counterexample*

L* algorithm (variation) for DFAs

$S, E \subseteq A^*$ induce a table

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

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

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

$\mathtt{L}^\star$ algorithm (variation) for DFAs

$S, E \subseteq A^*$ induce a table

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

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

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

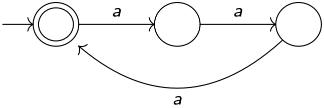
Initially $S = E = \{\varepsilon\}$

Repeat until no more counterexamples:

1. *Close* table
2. Query equivalence for *corresponding hypothesis*
3. Add suffixes of counterexample to E

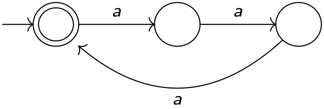
$\mathtt{L}^\star$ for DFAs, example

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

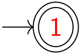


$\mathtt{L}^{\star}$ for DFAs, example

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

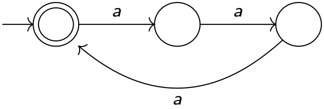


	ε
ε	1
a	0

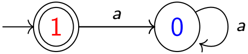


L* for DFAs, example

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

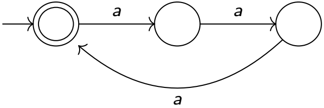


	ε
ε	1
a	0
aa	0

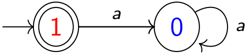


$\mathtt{L}^{\star}$ for DFAs, example

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



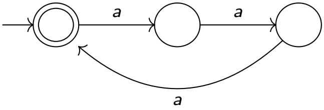
	ε
ε	1
a	0
aa	0



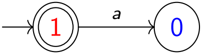
Counterexample: aaa

L* for DFAs, example

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

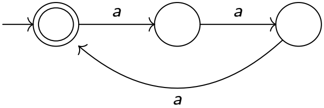


	ε	a	aa	aaa
ε	1	0	0	1
a	0	0	1	0
aa	0	1	0	0

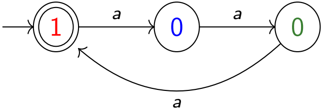


L* for DFAs, example

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



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



DFAs vs WFAs

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

$\boxed{\text{DFA}}$

initial state in Q

$$\begin{tikzcd} Q \arrow[d] \\ 2 \times Q^A \end{tikzcd}$$

$\boxed{\text{WFA}}$

initial state in FQ

$$\begin{tikzcd} Q \arrow[d] \\ \mathbb{S} \times (FQ)^A \end{tikzcd}$$

DFAs vs WFAs

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

$\boxed{\text{DFA}}$

initial state in Q

$$\begin{tikzcd} Q \arrow[d] \\ 2 \times Q^A \end{tikzcd}$$

$\boxed{\text{WFA}}$

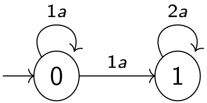
initial state in FQ

$$\begin{tikzcd} Q \arrow[d] \\ \mathbb{S} \times (FQ)^A \end{tikzcd}$$

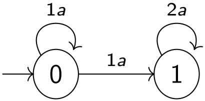
Interpretation: **weighted language** $A^* \to \mathbb{S}$

- multiply weights along paths and with final output
- sum over paths

WFA example over $\mathbb{Q}$

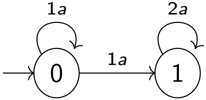


WFA example over $\mathbb{Q}$



$$\begin{aligned}
\mathcal{L}(\varepsilon) &= 0 \\
\mathcal{L}(a) &= 1 \cdot 0 + 1 \cdot 1 = 1 \\
\mathcal{L}(aa) &= 1 \cdot 1 \cdot 0 + 1 \cdot 1 \cdot 1 + 1 \cdot 2 \cdot 1 = 3 \\
\mathcal{L}(aaa) &= 1 \cdot 1 \cdot 1 \cdot 0 + 1 \cdot 1 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot 2 \cdot 1 + 1 \cdot 2 \cdot 2 \cdot 1 = 7
\end{aligned}$$

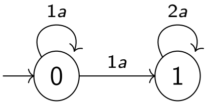
WFA example over $\mathbb{Q}$



$$
\begin{aligned}
\mathcal{L}(\varepsilon) &= 0\\
\mathcal{L}(a) &= 1\cdot 0 + 1\cdot 1 = 1\\
\mathcal{L}(aa) &= 1\cdot 1\cdot 0 + 1\cdot 1\cdot 1 + 1\cdot 2\cdot 1 = 3\\
\mathcal{L}(aaa) &= 1\cdot 1\cdot 1\cdot 0 + 1\cdot 1\cdot 1\cdot 1 + 1\cdot 1\cdot 2\cdot 1 + 1\cdot 2\cdot 2\cdot 1 = 7
\end{aligned}
$$

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

WFA example over $\mathbb{Q}$



$$\begin{aligned}
\mathcal{L}(\varepsilon) &= 0 \\
\mathcal{L}(a) &= 1 \cdot 0 + 1 \cdot 1 = 1 \\
\mathcal{L}(aa) &= 1 \cdot 1 \cdot 0 + 1 \cdot 1 \cdot 1 + 1 \cdot 2 \cdot 1 = 3 \\
\mathcal{L}(aaa) &= 1 \cdot 1 \cdot 1 \cdot 0 + 1 \cdot 1 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot 2 \cdot 1 + 1 \cdot 2 \cdot 2 \cdot 1 = 7
\end{aligned}$$

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

In fact: this is a weighted automaton over $\mathbb{N}$ as well.

Learning algorithm for WFAs

Membership queries:

return output value associated with word

Learning algorithm for WFAs

Membership queries:

return output value associated with word

Equivalence queries:

submit hypothesis WFA,
counterexample = word on which outputs differ

Learning algorithm for WFAs

Membership queries:

return output value associated with word

Equivalence queries:

submit hypothesis WFA,
counterexample = word on which outputs differ

Table cells:

output values in $\mathbb{S}$ instead of $0, 1$

Learning algorithm for WFAs

Membership queries:

return output value associated with word

Equivalence queries:

submit hypothesis WFA,
counterexample = word on which outputs differ

Table cells:

output values in $\mathbb{S}$ instead of $0, 1$

Closedness:

each lower row a linear combination of upper rows

General (weighted) $\mathtt{L}^\star$

Initially $S = E = \{\varepsilon\}$

Repeat until no more counterexamples:

1. *Close table*
2. Query equivalence for corresponding hypothesis
3. Add suffixes of counterexample to E

General (weighted) $\mathtt{L}^\star$

Initially $S = E = \{\varepsilon\}$

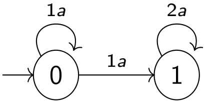
Repeat until no more counterexamples:

1. *Close table*
2. Query equivalence for corresponding hypothesis
3. Add suffixes of counterexample to E

Requirement on semiring $\mathbb{S}$: solving linear systems of equations should be computable.

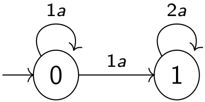
Example over $\mathbb{Q}$

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

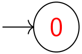


Example over $\mathbb{Q}$

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

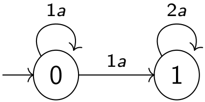


	ε
ε	0
a	1

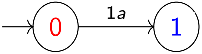


Example over $\mathbb{Q}$

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

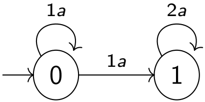


	ε
ε	0
a	1
aa	3

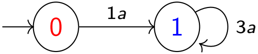


Example over $\mathbb{Q}$

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



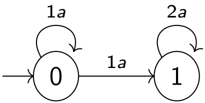
	ε
ε	0
a	1
aa	3



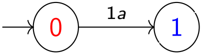
Counterexample: aaa

Example over $\mathbb{Q}$

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

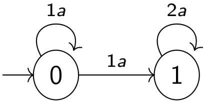


	ε	a	aa	aaa
ε	0	1	3	7
a	1	3	7	15
aa	3	7	15	31

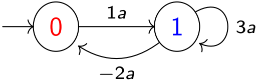


Example over $\mathbb{Q}$

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



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



Does it terminate?

The algorithm terminates for some known cases of semirings $\mathbb{S}$, if the input language is recognised by a WFA over $\mathbb{S}$:

- any field; (variation on algorithm by Bergadano and Varricchio (1996))
- the Boolean semiring 2 (WFA are non-deterministic automata; variation on algorithm by Bollig et al (2009)).

Does it terminate?

The algorithm terminates for some known cases of semirings $\mathbb{S}$, if the input language is recognised by a WFA over $\mathbb{S}$:

- any field; (variation on algorithm by Bergadano and Varricchio (1996))
- the Boolean semiring 2 (WFA are non-deterministic automata; variation on algorithm by Bollig et al (2009)).

Burning question

Does it terminate for any semiring?

Does it terminate?

The algorithm terminates for some known cases of semirings $\mathbb{S}$, if the input language is recognised by a WFA over $\mathbb{S}$:

- any field; (variation on algorithm by Bergadano and Varricchio (1996))
- the Boolean semiring 2 (WFA are non-deterministic automata; variation on algorithm by Bollig et al (2009)).

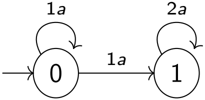
Burning question

Does it terminate for any semiring?

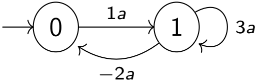
No.

The natural numbers

Recall the automaton:



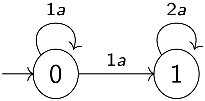
When learning over $\mathbb{Q}$, we get an automaton with a negative coefficient:



If we learn over $\mathbb{N}$, the algorithm doesn't terminate.

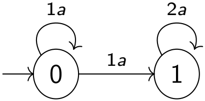
WFAs over $\mathbb{N}$: termination issue

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

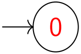


WFAs over $\mathbb{N}$: termination issue

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

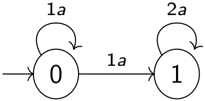


	ε	a
ε	0	1
a	1	3

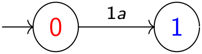


WFAs over $\mathbb{N}$: termination issue

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

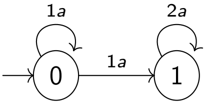


	ε	a
ε	0	1
a	1	3
aa	3	7

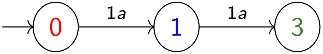


WFAs over $\mathbb{N}$: termination issue

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



	ε	a
ε	0	1
a	1	3
aa	3	7
aaa	7	15



Approximating the Hankel matrix

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

$$
\begin{array}{r|ccccc}
 & \varepsilon & a & aa & aaa & \dots \\
\hline
\varepsilon & 0 & 1 & 3 & 7 & \\
a & 1 & 3 & 7 & 15 & \\
aa & 3 & 7 & 15 & 31 & \dots \\
aaa & 7 & 15 & 31 & 63 & \\
\dots & & & \dots & &
\end{array}
$$

This is not finitely generated.

Termination of the general algorithm

Algorithm terminates assuming

- **progress measure with bound**

  Number, increases when rows separate via extra column

Termination of the general algorithm

Algorithm terminates assuming

- **progress measure with bound**

  Number, increases when rows separate via extra column
- **ascending chain condition** on Hankel matrix (table (A^*, A^*))

  Subsemimodule chains converge: if

  $$S_1 \subseteq S_2 \subseteq \cdots \subseteq H$$

  are subsemimodules, then there exists $n \in \mathbb{N}$ s.t.

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

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

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

- Converges, from that point on closedness guaranteed

Termination argument

Assume

- **progress measure with bound**
- **ascending chain condition** on Hankel matrix

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

- Converges, from that point on closedness guaranteed

Abstract result $\implies$ counterexample leads to either

- closedness defect or
- rows distinguished by new column

Termination argument

Assume

- **progress measure with bound**
- **ascending chain condition** on Hankel matrix

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

- Converges, from that point on closedness guaranteed

Abstract result $\implies$ counterexample leads to either

- closedness defect or
- rows distinguished by new column

Bounded progress measure $\implies$ finitely many counterexamples

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

WFAs over field: no problem

1. Progress measure and bound
   - Dimension of vector space spanned by table
   - $\leq$ minimal WFA size

WFAs over field: no problem

1. Progress measure and bound
   - Dimension of vector space spanned by table
   - $\leq$ minimal WFA size
2. Ascending chain condition
   - Vector space dimension increases with strict inclusion
   - Minimal WFA size = Hankel matrix dimension

WFAs over field: no problem

1. Progress measure and bound
   - Dimension of vector space spanned by table
   - $\leq$ minimal WFA size
2. Ascending chain condition
   - Vector space dimension increases with strict inclusion
   - Minimal WFA size = Hankel matrix dimension
3. Procedure to determine/fix closedness
   - Gaussian elimination

WFAs over finite semiring: naive algorithm

1. Progress measure and bound
   - Set size of semimodule spanned by table
   - $\leq$ determinisation of correct automaton

WFAs over finite semiring: naive algorithm

1. Progress measure and bound
   - Set size of semimodule spanned by table
   - $\leq$ determinisation of correct automaton
2. Ascending chain condition
   - Hankel matrix size $\leq$ determinisation of correct automaton

WFAs over finite semiring: naive algorithm

1. Progress measure and bound
   - Set size of semimodule spanned by table
   - $\leq$ determinisation of correct automaton
2. Ascending chain condition
   - Hankel matrix size $\leq$ determinisation of correct automaton
3. Procedure to determine/fix closedness
   - Try all linear combinations of rows

WFAs over PID

Principal ideal domain = integral domain with all ideals principal

Integral domain: commutative ring,

$$ab = 0 \implies a = 0 \vee b = 0$$

WFAs over PID

Principal ideal domain = integral domain with all ideals principal

Integral domain: commutative ring,

$$ab = 0 \implies a = 0 \vee b = 0$$

All ideals principal: generated by one element

WFAs over PID

Principal ideal domain = integral domain with all ideals principal

Integral domain: commutative ring,

$$ab = 0 \implies a = 0 \vee b = 0$$

All ideals principal: generated by one element

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

PID free module properties

A module is free if and only if it is **torsion free**:

$$pm = 0 \implies p = 0 \vee m = 0$$

PID free module properties

A module is free if and only if it is **torsion free**:

$$pm = 0 \implies p = 0 \vee m = 0$$

A submodule of a free and finitely generated module is

- free and finitely generated
- with smaller (or equal) rank

PID free module properties

A module is free if and only if it is **torsion free**:

$$pm = 0 \implies p = 0 \vee m = 0$$

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

Progress measure for PIDs

Table modules are torsion free and thus free

Measure: **rank of table module**

Progress measure for PIDs

Table modules are torsion free and thus free

Measure: **rank of table module**

Bound: **Hankel matrix rank**

Progress measure for PIDs

Table modules are torsion free and thus free

Measure: **rank of table module**

Bound: **Hankel matrix rank**

Progress (general fact): for X, Y finite sets and

- $FX \xrightarrow{f} FY$ a surjective homomorphism
- that identifies some elements

we have $|X| > |Y|$

Learning WFAs over PIDs

1. Progress measure and bound
   - Rank of the module spanned by the table
   - $\leq$ rank of the Hankel matrix

Learning WFAs over PIDs

1. Progress measure and bound
   - Rank of the module spanned by the table
   - $\leq$ rank of the Hankel matrix
2. Ascending chain condition
   - Yes :)

Learning WFAs over PIDs

1. Progress measure and bound
   - Rank of the module spanned by the table
   - $\leq$ rank of the Hankel matrix
2. Ascending chain condition
   - Yes :)
3. Procedure to determine/fix closedness
   - Solve equations via Smith normal form (exists for PIDs), some further assumptions on computability (hold for integers)

Learning WFAs over PIDs

1. Progress measure and bound
   - Rank of the module spanned by the table
   - $\leq$ rank of the Hankel matrix
2. Ascending chain condition
   - Yes :)
3. Procedure to determine/fix closedness
   - Solve equations via Smith normal form (exists for PIDs), some further assumptions on computability (hold for integers)

So: the learning algorithm terminates for the integers!

Conclusion

Learning weighted automata

- Works for fields, finite semirings (known)
- also works for $\mathbb{Z}$
- does *not* terminate for $\mathbb{N}$.