Learning Weighted Automata over Principal Ideal Domains

Gerco van Heerdt Clemens Kupke
Jurriaan Rot Alexandra Silva

April 29, 2020

$\mathtt{L}^\star$ 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**

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

L* setup for DFAs

Finite alphabet A

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

L* 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*

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

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

$$
\begin{array}{r|cc}
 & \overbrace{\varepsilon \quad \mathrm{a}}^{E} \\
\hline
\varepsilon & 1 \quad 0 \\
\mathrm{a} & 0 \quad 1 \\
\mathrm{aa} & 1 \quad 0 \\
\hline
\mathrm{aaa} & 0 \quad 1
\end{array}
$$

		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* algorithm (variation) for DFAs

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

$$
\begin{array}{r l|cc}
 & & \multicolumn{2}{c}{\overbrace{\qquad\qquad}^{E}} \\
 & & \varepsilon & \mathsf{a} \\ \hline
 & \varepsilon & 1 & 0 \\
S & \mathsf{a} & 0 & 1 \\
 & \mathsf{aa} & 1 & 0 \\ \hline
S \cdot A & \mathsf{aaa} & 0 & 1
\end{array}
$$

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

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

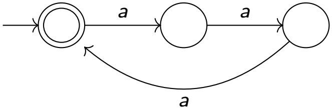
Initially $S = E = \{\varepsilon\}$

Repeat until no more counterexamples:

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

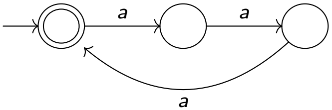
$\mathtt{L}^\star$ for DFAs, example

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

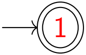


$\mathtt{L}^\star$ for DFAs, example

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

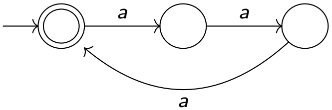


	ε
ε	1
a	0

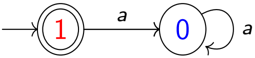


L* for DFAs, example

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



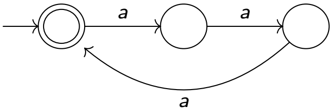
	ε
ε	1
a	0
aa	0



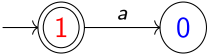
Counterexample: aaa

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

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

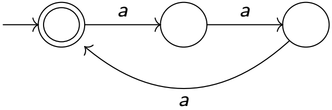


	ε	a	aa	aaa
ε	1	0	0	1
a	0	0	1	0
aa	0	1	0	0

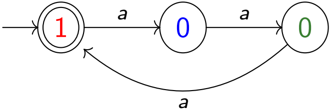


L* for DFAs, example

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



	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

$\mathcal{S}$ semiring, FQ free semimodule over Q

$\boxed{\text{DFA}}$	$\boxed{\text{WFA}}$
initial state in Q	initial state in FQ
$Q \to 2 \times Q^A$	$Q \to \mathcal{S} \times (FQ)^A$

DFAs vs WFAs

$\mathcal{S}$ semiring, FQ free semimodule over Q

$\boxed{\text{DFA}}$	$\boxed{\text{WFA}}$
initial state in Q	initial state in FQ
$Q \to 2 \times Q^A$	$Q \to \mathcal{S} \times (FQ)^A$

$$\begin{array}{c} Q \\ \downarrow \\ 2 \times Q^A \end{array} \qquad \begin{array}{c} Q \\ \downarrow \\ \mathcal{S} \times (FQ)^A \end{array}$$

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

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

Algorithm setup for WFAs

Membership queries:

return output value associated with word

Algorithm setup for WFAs

Membership queries:
return output value associated with word

Equivalence queries:
submit hypothesis WFA,
counterexample = word on which outputs differ

Algorithm setup for WFAs

Membership queries:

return output value associated with word

Equivalence queries:

submit hypothesis WFA,
counterexample = word on which outputs differ

Cells:

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

Algorithm setup for WFAs

Membership queries:

return output value associated with word

Equivalence queries:

submit hypothesis WFA,
counterexample = word on which outputs differ

Cells:

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

Closedness:

each lower row a linear combination of upper rows

General (weighted) L*

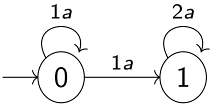
Initially $S = E = \{\varepsilon\}$

Repeat until no more counterexamples:

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

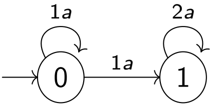
Example over $\mathbb{Q}$

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

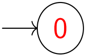


Example over $\mathbb{Q}$

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

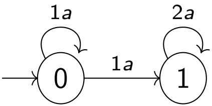


	ε
ε	0
a	1

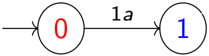


Example over $\mathbb{Q}$

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

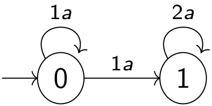


	ε
ε	0
a	1
aa	3

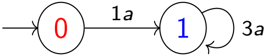


Example over $\mathbb{Q}$

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



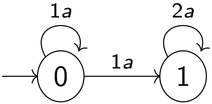
	ε
ε	0
a	1
aa	3



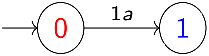
Counterexample: aaa

Example over $\mathbb{Q}$

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

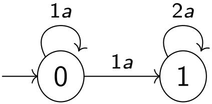


	ε	a	aa	aaa
ε	0	1	3	7
a	1	3	7	15
aa	3	7	15	31

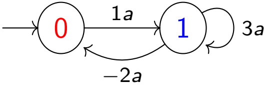


Example over $\mathbb{Q}$

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



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



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

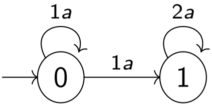
1. Progress measure and bound
   - Set size of semimodule spanned by table
   - $\leq$ determinisation of correct automaton
2. Ascending chain condition
   - Hankel matrix size $\leq$ determinisation of correct automaton

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

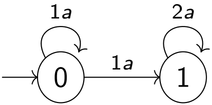
WFAs over $\mathbb{N}$: termination issue

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

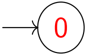


WFAs over $\mathbb{N}$: termination issue

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

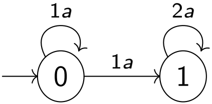


	ε	a
ε	0	1
a	1	3

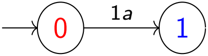


WFAs over $\mathbb{N}$: termination issue

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

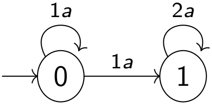


	ε	a
ε	0	1
a	1	3
aa	3	7

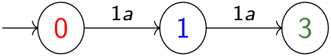


WFAs over $\mathbb{N}$: termination issue

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



	ε	a
ε	0	1
a	1	3
aa	3	7
aaa	7	15



Analysis: overweight finite automaton

Infinite chain of strict semimodule embeddings

- Ascending chain condition fails
- Hankel matrix not even finitely generated

Contribution: WFAs over PID

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

Integral domain: commutative ring,

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

Contribution: WFAs over PID

Principal ideal domain = integral domain with all ideals principal

Integral domain: commutative ring,

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

All ideals principal: generated by one element

Contribution: 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 finitely generated free 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 finitely generated free 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 :)
     (basis of the union of the chain included in a chain element)

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 :)
     (basis of the union of the chain included in a chain element)
3. Procedure to determine/fix closedness
   - Solve equations via Smith normal form (exists for PIDs)

Future work

Even bigger classes of semirings

Future work

Even bigger classes of semirings

Stronger negative results

Future work

Even bigger classes of semirings

Stronger negative results

Conditions on the monad