Categorical Automata Learning Framework

calf-project.org

Matteo Sammartino

S-REPLS 6
Our team

Alexandra Silva  
UCL

Matteo Sammartino  
UCL

Gerco van Heerdt  
UCL

Joshua Moerman  
Radboud University

Maverick Chardet  
ENS Lyon

Collaborators:

Bartek Klin  
Warsaw University

Michal Szynwelski  
Warsaw University
Automata learning

Learner

queries
answers

System black-box $S$

builds

automaton model of $S$
Automata learning

Learner

queries

answers

System black-box $S$

builds

automaton model of $S$

No formal specification available? Learn it!
L* algorithm (D. Angluin ’87)

Finite alphabet of system’s actions \( A \)
set of system behaviors is a regular language \( \mathcal{L} \subseteq A^* \)
L* algorithm (D. Angluin ’87)

Finite alphabet of system’s actions $A$

set of system behaviors is a regular language $\mathcal{L} \subseteq A^*$

Learner

Teacher $\mathcal{L}$
L* algorithm (D. Angluin ’87)

Finite alphabet of system’s actions $A$

set of system behaviors is a regular language $\mathcal{L} \subseteq A^*$

Q: $w \in \mathcal{L}$?
A: Y/N
L* algorithm (D. Angluin ’87)

Finite alphabet of system’s actions $A$

set of system behaviors is a regular language $\mathcal{L} \subseteq A^*$

Q: $w \in \mathcal{L}$?
A: Y/N

Q: $\mathcal{L}(H) = \mathcal{L}$?
A: Y/N + counterexample

$H = \text{hypothesis automaton}$
L* algorithm (D. Angluin ’87)

Finite alphabet of system’s actions $A$

set of system behaviors is a regular language $\mathcal{L} \subseteq A^*$

Q: $w \in \mathcal{L}$?
A: Y/N

Q: $\mathcal{L}(H) = \mathcal{L}$?
A: Y/N + counterexample

$H$ = hypothesis automaton

Learner

builds

Minimal DFA accepting $\mathcal{L}$

Teacher $\mathcal{L}$
The paper is organized as follows. In Section 2, we present an overview of the work developed in the various steps. In Section 3, we describe the learning algorithm works by incrementally building an automaton, it poses an equivalence query to the teacher. It terminates if for all prefixes of the teacher replies equivalence queries. As an example, and to set notation, consider the following.

\[ S, E \subseteq A^* \quad A = \{a, b\} \]

\[ row : S \cup S \cdot A \rightarrow 2^E \]

\[ row(s)(e) = 1 \iff se \in \mathcal{L} \]

**Figure 1.** Observation table

<table>
<thead>
<tr>
<th></th>
<th>(\epsilon)</th>
<th>(a)</th>
<th>aa</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\epsilon)</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(a)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(b)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Hypothesis automaton

An observation table

<table>
<thead>
<tr>
<th></th>
<th>(\epsilon)</th>
<th>(a)</th>
<th>(aa)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\epsilon)</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(a)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(b)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\(S, E \subseteq A^*\) \(A = \{a, b\}\)

Row function:

\[row: S \cup S \cdot A \rightarrow 2^E\]

\[row(s)(\epsilon) = 1 \iff se \in \mathcal{L}\]

Hypothesis automaton

- States: \(\{row(s) | s \in S\}\)
- Final states: \(\{row(s) | s \in S, row(s)(\epsilon) = 1\}\)
- Initial state: \(row(\epsilon)\)
- Transition function: \(row(s) \xrightarrow{a} row(sa)\)
The paper is organized as follows. In Section 2, we present an overview of the approach contains a description of NLambda, details of the implementation, algorithm for regular languages over finite alphabets, and then a simple example of execution. The algorithm is able to fill the table with membership queries (∈ S) and equivalence queries (s × E). The Teacher replies consisting of a hypothesis DFA (S, E, q0, F) and a single word w. Since w is not closed, the Teacher replies no, and consistent and a new hypothesis is constructed.

---

**Observation table**

<table>
<thead>
<tr>
<th></th>
<th>$\epsilon$</th>
<th>$a$</th>
<th>$aa$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$a$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

row: $S \cup S \cdot A \rightarrow 2^E$

$\text{row}(s)(\epsilon) = 1 \iff se \in \mathcal{L}$

**Hypothesis automaton**

states = \{row(s) | s ∈ S\}

final states = \{row(s) | s ∈ S \land \text{row}(s)(\epsilon) = 1\}

initial state

transition function row(s) $\overset{a}{\rightarrow}$ row(sa)

**Why is this correct?**
Table properties
Table properties

Closed

\( \forall t \in S \cdot A \ \exists s \in S \ \text{row}(t) = \text{row}(s). \)
### Table properties

\[ \text{row}(s) \xrightarrow{a} \text{row}(sa) \]

**Closed**

\[ \forall t \in S \cdot A \quad \exists s \in S \quad \text{row}(t) = \text{row}(s). \]
Table properties

\[\text{row}(s) \xrightarrow{a} \text{row}(sa)\]

next state exists

Closed

\[\forall t \in S \cdot A \; \exists s \in S \; \text{row}(t) = \text{row}(s).\]
Table properties

\[ row(s) \xrightarrow{a} row(sa) \]  

next state exists

Closed

\[ \forall t \in S \cdot A \quad \exists s \in S \quad row(t) = row(s). \]

Consistent

\[ \forall s_1, s_2 \in S \quad row(s_1) = row(s_2) \implies \forall a \in A \quad row(s_1a) = row(s_2a) \]
Table properties

\[ \forall t \in S \cdot A \quad \exists s \in S \quad \text{row}(t) = \text{row}(s). \]

Closed

next state exists

determinism

Consistent

\[ \forall s_1, s_2 \in S \quad \text{row}(s_1) = \text{row}(s_2) \implies \forall a \in A \quad \text{row}(s_1a) = \text{row}(s_2a) \]
Table properties

\[ \text{row}(s) \xrightarrow{a} \text{row}(sa) \]

next state exists

determinism

Closed

\[
\forall t \in S, A \quad \exists s \in S \quad \text{row}(t) = \text{row}(s).
\]

Fixed by extending the table

\[
\forall s_1, s_2 \in S \quad \text{row}(s_1) = \text{row}(s_2) \implies \forall a \in A \quad \text{row}(s_1a) = \text{row}(s_2a)
\]
Pros of L* …

Applications: Hardware verification, security/network protocols…

Generalizations: Mealy machines, I/O automata, …
A zoo of automata

- Non-deterministic
- Probabilistic
- Weighted
- Universal
- Alternating
- Register
- Mealy Machines
A zoo of automata

- Probabilistic
- Weighted
- Non-deterministic
- Universal
- Alternating
- Register
- Mealy Machines

Algorithms
Correctness proofs

involved and hard to check
A zoo of automata

Category theory comes to the rescue!

Algorithms
Correctness proofs

involved and hard to check

Non-deterministic
Probabilistic
Weighted
Universal
Alternating
Register
Abstract automata

Category $C = \text{universe of state-spaces}$

Endofunctor $F : C \to C = \text{automaton type}$

\[
\begin{array}{ccc}
FQ \\
\downarrow \delta_Q \\
\text{init}_Q & \text{Q} & \text{out}_Q \\
I & \rightarrow & Y
\end{array}
\]
Abstract automata

Category $\mathcal{C} = \text{universe of state-spaces}$

Endofunctor $F: \mathcal{C} \to \mathcal{C} = \text{automaton type}$

**DFAs**

$\mathcal{C} = \text{Set}$

$F = (-) \times A$

\[
\begin{tikzcd}
FQ \\
\downarrow \delta_Q \\
init_Q & Q & \text{out}_Q \\
I & Q & Y
\end{tikzcd}
\]
Abstract automata

Category $\mathcal{C} = \text{universe of state-spaces}$

Endofunctor $F : \mathcal{C} \rightarrow \mathcal{C} = \text{automaton type}$

DFAs

$\mathcal{C} = \text{Set}$

$F = (-) \times A$

\[
\begin{array}{c}
\quad Q \times A \\
\downarrow \delta_Q \\
\quad \text{init}_Q \\
\quad I \\
\end{array}
\]

\[
\begin{array}{c}
\quad Q \\
\quad \text{out}_Q \\
\quad Y \\
\end{array}
\]
Abstract automata

Category $C =$ universe of state-spaces

Endofunctor $F : C \to C =$ automaton type

DFAs

$C = \text{Set}$

$F = (-) \times A$

\[
\begin{align*}
Q \times A & \quad \downarrow \delta_Q \\
\text{init}_Q & \quad Q & \quad \text{out}_Q \\
1 & \quad Q & \quad Y
\end{align*}
\]
Abstract automata

Category $C = \text{universe of state-spaces}$

Endofunctor $F : C \rightarrow C = \text{automaton type}$

DFAs

$C = \text{Set}$

$F = (-) \times A$

\[
\begin{align*}
Q \times A & \\
\downarrow \delta_Q & \\
\text{init}_Q & \hspace{1cm} Q \hspace{1cm} \text{out}_Q \\
1 & \hspace{1cm} Y \\
q_0 \in Q
\end{align*}
\]
Abstract automata

Category $C =$ universe of state-spaces

Endofunctor $F: C \to C =$ automaton type

DFAs

$C = \text{Set}$

$F = (-) \times A$

\[
\begin{array}{c}
Q \times A \\
\downarrow \delta_Q \\
\text{init}_Q \rightarrow Q \\
\text{out}_Q \\
1 \rightarrow Q \rightarrow 2 \\
q_0 \in Q
\end{array}
\]
Abstract automata

Category \( C = \) universe of state-spaces

Endofunctor \( F : C \to C = \) automaton type

DFAs
\( C = \text{Set} \)
\( F = (-) \times A \)

\[
\begin{align*}
Q \times A & \quad \delta_Q \downarrow \\
\text{init}_Q & \quad Q & \quad \text{out}_Q \\
1 & \quad 2
\end{align*}
\]

\( q_0 \in Q \quad F \subseteq Q \)
Abstract learning

Abstract observation data structure
Abstract learning

Abstract observation data structure

approximates

Target minimal automaton

\[ F_Q \]
\[ \delta_Q \]
\[ \text{init}_Q \]
\[ Q \]
\[ \text{out}_Q \]

\[ I \]

\[ Y \]
Abstract learning

Abstract observation data structure

Hypothesis automaton

Target minimal automaton

approximates

abstract closedness and consistency
In this section we work towards a general correctness theorem. We then show how it applies to the examples.

Definition 14

By a learning algorithm and consistency, then ensures that if $\mathcal{T}, \mathcal{F}$ is (isomorphic to) $\mathcal{E}$, which is the state space of the target minimal automaton, as in Example 2, our reasoning hints at a correctness criterion:

$$\mathcal{P} \mathcal{I} \mathcal{E} \mathcal{D} \mathcal{L} \mathcal{E} \mathcal{A} \mathcal{M} \mathcal{T} \mathcal{E}$$

For a wrapper, the net. If $\mathcal{P} \mathcal{T}, \mathcal{F}$ is the following. Let $\mathcal{I}$ be as in (3) and define $\mathcal{J}$.

Example 2, our reasoning hints at a correctness criterion:

If $\mathcal{E}$$\mathcal{P} \mathcal{I} \mathcal{E} \mathcal{D} \mathcal{L} \mathcal{E} \mathcal{A} \mathcal{M} \mathcal{T} \mathcal{E}$, which by (the dual of) Proposition 12.

Proposition 12.

Assume that indeed exists. Because $\mathcal{E}$ is (isomorphic to) $\mathcal{E}$, which is the state space of the target minimal automaton, as in Example 2, our reasoning hints at a correctness criterion:

$$\mathcal{P} \mathcal{I} \mathcal{E} \mathcal{D} \mathcal{L} \mathcal{E} \mathcal{A} \mathcal{M} \mathcal{T} \mathcal{E}$$

In this section we work towards a general correctness theorem. We then show how it applies to the examples.

Theorem approximates

Target minimal automaton

$\mathcal{F} \mathcal{Q}$ $\delta_{\mathcal{Q}}$

$\mathcal{I}$ $\mathcal{Q}$ $\mathcal{Y}$

Hypothesis automaton

$\mathcal{F} \mathcal{H}$ $\delta_{\mathcal{H}}$

$\mathcal{I} \mathcal{H}$ $\mathcal{Y}$

abstract closedness and consistency

General correctness theorem

Guidelines for implementation
Abstract learning

Target minimal automaton

\[ \begin{array}{c}
FQ \\
\downarrow \delta_Q \\
\text{init}_Q \\
I \\
\rightarrow Q \\
\rightarrow \text{out}_Q \\
\rightarrow Y
\end{array} \]

General correctness theorem

Guidelines for implementation


Gerco van Heerdt, Matteo Sammartino, Alexandra Silva
Degrees of freedom
Degrees of freedom

Change base category

<table>
<thead>
<tr>
<th>Set</th>
<th>DFAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nom</td>
<td>Nominal automata</td>
</tr>
<tr>
<td>Vect</td>
<td>Weighted automata</td>
</tr>
</tbody>
</table>
Degrees of freedom

Change base category

| Set    | DFAs          |
| Nom    | Nominal automata |
| Vect   | Weighted automata |

Side-effects (via monads)

<table>
<thead>
<tr>
<th>Powerset</th>
<th>NFAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Powerset with intersection</td>
<td>Universal automata</td>
</tr>
<tr>
<td>Maybe monad</td>
<td>Partial automata</td>
</tr>
</tbody>
</table>
Degrees of freedom

Change base category
- Set: DFAs
- Nom: Nominal automata
- Vect: Weighted automata

Change main data structure
- Observation tables
- Discrimination trees

Side-effects (via monads)
- Powerset: NFAs
- Powerset with intersection: Universal automata
- Maybe monad: Partial automata
Degrees of freedom

Change base category

- Set: DFAs
- Nom: Nominal automata
- Vect: Weighted automata

Change main data structure

- Learning Nominal Automata (POPL '17)
  Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, Michal Szynwelski
- Discrimination trees

Side-effects (via monads)

- Powerset: NFAs
- Powerset with intersection: Universal automata
- Maybe monad: Partial automata
Degrees of freedom

Change base category

- Set: DFAs
- Nom: Nominal automata
- Vect: Weighted automata

Change main data structure

- Discrimination trees

Learning Nominal Automata (POPL ’17)
Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, Michal Szynwelski

Learning Automata with Side-effects (arXiv:1704.08055)
Gerco van Heerdt, Matteo Sammartino, Alexandra Silva

Side-effects (via monads)

- Powerset: NFAs
- Powerset with intersection: Universal automata
- Maybe monad: Partial automata
Connections with other algorithms

Automaton type

- Automata Learning algorithms
- Minimization algorithms
- Testing algorithms

Optimizations
Abstraction helps a lot ...
Abstraction helps a lot ... but there is no free lunch!
Ongoing and future work

- **Tool** to learn control + data-flow models (as *nominal automata*)

- Applications:
  - Specification mining
  - Network verification, with [Amazon](https://www.amazon.com)
  - Verification of cryptographic protocols
  - Ransomware detection