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 \( \mathcal{L} \subseteq A^* \)
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** $\mathcal{L} \subseteq A^*$

$\mathbf{Q: } w \in \mathcal{L}?$

$\mathbf{A: } \text{Y/N}$

$\mathbf{Q: } \mathcal{L}(H) = \mathcal{L}?$

$\mathbf{A: } \text{Y/N + counterexample}$

$H = \text{hypothesis automaton}$
**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 $\mathcal{L}$
A zoo of automata

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

Probabilistic
Non-deterministic
Weighted
Universal
Alternating
Register
Mealy Machines

Algorithms
Correctness proofs
involved and hard to check
A zoo of automata

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

Category theory comes to the rescue!

Algorithms
Correctness proofs

involved and hard to check
Category Theory

- Conceptual tools
- Correctness proof(s)
- Guidelines new algorithms
- Unveil connections
Category Theory

- Conceptual tools
- Correctness proof(s)
- Guidelines new algorithms
- Unveil connections
- No free lunch!
Automata

\[ X \rightarrow 2 \times X^A \]

DFA
Automata

\[ X \to 2 \times X^A \]

DFA

\[ X \to \mathbb{R} \times (\mathbb{R}^X)^A \]

WFA
Automata

\[ X \rightarrow 2 \times X^A \quad \text{DFA} \]

\[ X \rightarrow \mathbb{R} \times (\mathbb{R}^X)^A \quad \text{WFA} \]

\[ X \rightarrow FTXX \quad \text{Transition structure} \]

\[ X \rightarrow \text{Algebraic properties} \]
\[ X \rightarrow FTX \]

\[ X \rightarrow 2 \times X^A \]

\[ X \rightarrow \mathbb{R} \times (\mathbb{R}^X)^A \]

DFA

WFA
\[ X \rightarrow FTX \]

\[ X \rightarrow 2 \times X^A \quad X \rightarrow \mathbb{R} \times (\mathbb{R}^X)^A \]

DFA

\[ 2^{A^*} \]

 acceptance

WFA

\[ \mathbb{R}^{A^*} \]

Vector space
\[ X \rightarrow F T X \]

\[ X \rightarrow 2 \times X^A \]

**DFA**

\[ 2^{A^*} \]

Language equivalence

**WFA**

\[ \mathbb{R} \times (\mathbb{R}^X)^A \]

\[ \mathbb{R}^{A^*} \]

Weighted language equivalence or bisimilarity

Vector space
\[ X \rightarrow FTX \]

\[ X \rightarrow 2 \times X^A \]

\[ X \rightarrow \mathbb{R} \times (\mathbb{R}^X)^A \]

**DFA**

\[ 2^{A^*} \]

Language equivalence

**WFA**

\[ \mathbb{R}^{A^*} \]

Weighted language equivalence or bisimilarity

Proof methods for equivalence

Vector space
Up-to techniques

Algebraic structure → Better Proof Techniques
Up-to techniques

Algebraic structure \[\rightarrow\] Better Proof Techniques

HKC algorithm - Bonchi and Pous 2014
Example

\[ x \leftarrow z \rightarrow y \]

\[ u \leftarrow w \rightarrow v \]
Example

Build a bisimulation using powerset construction on the fly
Example

\[
\begin{align*}
(x, u) &+ (y, v+w) = (x+y, u+v+w) \\
\end{align*}
\]

Using bisimulations up to union
Another example

\[ x \xrightarrow{1} \overline{u} \]
\[ \overline{u} \xrightarrow{2} x + y + z \]

Using bisimulations up to congruence this yields to the HKC algorithm [Bonchi, Pous'13]
Another example

\[ x + y = u + y \quad (1) \]
\[ = y + z + y \quad (2) \]
\[ = y + z \]
\[ = u \quad (2) \]
Another example

\[
x + y = u + y \quad (1) \\
= y + z + y \quad (2) \\
= y + z \\
= u \quad (2)
\]

Bisimulations up-to \textit{congruence}

HKC algorithm of Bonchi & Pous
More examples

**Up-To Techniques for Weighted Systems. (TACAS ’17)**
Filippo Bonchi, Barbara König, Sebastian Küpper

**The Power of Convex Algebras (CONCUR’ 17)**
Filippo Bonchi, Alexandra Silva, Ana Sokolova

**Coinduction up-to in a fibrational setting (CSL-LICS 2014)**
Filippo Bonchi, Daniela Petrisan, Damien Pous, Jurriaan Rot
(\(S, E, \text{row}\)) is \textit{closed} if for all \(t \in S \cdot A\) there exists an \(s \in S\) such that \(\text{row}(t) = \text{row}(s)\).
Category Theory in learning

(S, E, row) is closed if for all $t \in S \cdot A$ there exists an $s \in S$ such that $\text{row}(t) = \text{row}(s)$. 
Category Theory in learning

(S, E, row) is closed if for all \( t \in S \cdot A \) there exists an \( s \in S \) such that \( \text{row}(t) = \text{row}(s) \).
Category Theory in learning

$(S, E, \text{row})$ is closed if for all $t \in S \cdot A$ there exists an $s \in S$ such that $\text{row}(t) = \text{row}(s)$.

Can we develop $L^*$ for infinite (nominal) sets?
Infinite alphabets

\[ \mathcal{L}_n = \{ww \mid w \in A^*, |w| = n\} \]

\[ \mathcal{L}_1 = \{aa, bb, cc, dd, \ldots\} \]

\[ A_5 = \]

infinite automaton
Infinite alphabets

\[ \mathcal{L}_n = \{ww \mid w \in A^*, |w| = n\} \quad \text{A infinite} \]

\[ \mathcal{L}_1 = \{aa, bb, cc, dd, \ldots\} \]

\[ A_5 = \]

\[ \text{infinite automaton} \quad \text{but with a finite representation} \]
Nominal automata

Nominal sets

name binding
alpha-equivalence

…..
Nominal automata

Nominal sets

Infinite sets

name binding
alpha-equivalence

…..
Nominal automata

Nominal sets

name binding
alpha-equivalence

…..

Infinite sets with symmetries

Finitely representable
Nominal automata

Nominal sets

name binding
alpha-equivalence

Infinite sets with symmetries

Finitely representable

Automata theory over nominal sets
Motivation

Nominal automata

\[
\begin{align*}
X &= \{s_0\} + A + \{t\} \\
B &\text{seen as a coalgebra in}\ Nom \\
A &\text{infinite}
\end{align*}
\]

\[
\{ w \in A^* \mid \exists a.a \text{ occurs twice in } w \}\]
Motivation

Nominal automata

\[ \exists a.a \text{ occurs twice in } w \]

\[ \{ w \in A^* \mid \exists a.a \text{ occurs twice in } w \} \]

\[ A \text{ infinite} \]

finite representation
Nominal automata

\begin{center}
\begin{tikzpicture}
\node[state] (s0) {$s_0$};
\node[state] (sa) [right of=s0] {$s_a$};
\node[state,accepting] (t) [right of=sa] {$t$};
\path[->]
(s0) edge node {$a$} (sa)
(sa) edge node {$a$} (t)
(s0) edge[loop above] node {$a$} (s0)
(sa) edge[loop above] node {$b\#a$} (sa)
(s0) edge[loop below] node {$a$} (s0);
\end{tikzpicture}
\end{center}

finite representation

canonical permutations

transition closed under permutations equivariant

\[ X = \{s_0\} + A + \{t\} \]

\[ \pi : A \rightarrow A \]

\[ S_a \xrightarrow{a} S_{\pi a} \]

\[ S_a \xrightarrow{a} t \implies S_{\pi a} \xrightarrow{\pi a} t \]
Nominal automata

finite representation

canonical permutations

transition closed under permutations equivariant

\[ X = \{s_0\} + \mathbb{A} + \{t\} \]

\[ \pi : \mathbb{A} \rightarrow \mathbb{A} \]

\[ S_a \rightarrow S_{\pi a} \]

\[ S_a \xrightarrow{a} t \Rightarrow S_{\pi a} \xrightarrow{\pi a} t \]
Nominal automata

### DFA in Nom

\[ X \rightarrow 2 \times X^A \]

\[ \pi : A \rightarrow A \]

\[ S_a \rightarrow S_{\pi a} \]

transition closed under permutations
equivariant

\[ S_a \xrightarrow{a} t \Rightarrow S_{\pi a} \xrightarrow{\pi a} t \]
Challenges

\[ L^* \text{ LEARNER} \]

1. \( S, E \leftarrow \{ \varepsilon \} \)
2. repeat
   3. while \( (S, E) \) is not closed or not consistent
   4. if \( (S, E) \) is not closed
      5. find \( s_1 \in S, a \in A \) such that \( row(s_1a) \neq row(s), \) for all \( s \in S \)
      6. \( S \leftarrow S \cup \{ s_1a \} \)
   7. if \( (S, E) \) is not consistent
      8. find \( s_1, s_2 \in S, a \in A, \) and \( e \in E \) such that \( row(s_1) = row(s_2) \) and \( L(s_1ae) \neq L(s_2ae) \)
      9. \( E \leftarrow E \cup \{ ae \} \)
10. Make the conjecture \( M(S, E) \)
11. if the Teacher replies no, with a counter-example \( t \)
    12. \( S \leftarrow S \cup \text{prefixes}(t) \)
13. until the Teacher replies yes to the conjecture \( M(S, E) \).
14. return \( M(S, E) \)
This table indicates that ... a pair determined by the language table (over queries. As an example, and to set notation, consider the following

In this section, we give an overview of the work developed in the

2. Overview of the approach

11 Make the conjecture

9

LEARNER

Given an observation table $$S, E \leftarrow \{ \epsilon \}$$

1 repeat

2 while $$(S, E)$$ is not closed or not consistent

3 if $$(S, E)$$ is not closed

4 find $$s_1 \in S$$, $$a \in A$$ such that

5 $$\text{row}(s_1a) \neq \text{row}(s)$$, for all $$s \in S$$

6 $$S \leftarrow S \cup \{s_1a\}$$

7 if $$(S, E)$$ is not consistent

8 find $$s_1, s_2 \in S$$, $$a \in A$$, and $$e \in E$$ such that

9 $$\text{row}(s_1) = \text{row}(s_2)$$ and $$L(s_1ae) \neq L(s_2ae)$$

10 $$E \leftarrow E \cup \{ae\}$$

11 if the Teacher replies no, with a counter-example $$t$$

12 $$S \leftarrow S \cup \text{prefixes}(t)$$

13 until the Teacher replies yes to the conjecture $$M(S, E)$$.

14 return $$M(S, E)$$
Challenges

L* LEARNER
1 \begin{align*}
S, E & \leftarrow \{\epsilon\} \\
\text{repeat} & \\
\text{while } (S, E) \text{ is not closed or not consistent} & \\
\text{if } (S, E) \text{ is not closed} & \\
& \text{find } s_1 \in S, a \in A \text{ such that} \\
& \text{row}(s_1a) \neq \text{row}(s), \text{ for all } s \in S \\
& S \leftarrow S \cup \{s_1a\} \\
\text{if } (S, E) \text{ is not consistent} & \\
& \text{find } s_1, s_2 \in S, a \in A, \text{ and } e \in E \text{ such that} \\
& \text{row}(s_1) = \text{row}(s_2) \text{ and } L(s_1ae) \neq L(s_2ae) \\
& E \leftarrow E \cup \{ae\} \\
\text{Make the conjecture } M(S, E) & \\
\text{if the Teacher replies no, with a counter-example } t & \\
& S \leftarrow S \cup \text{prefixes}(t) \\
\text{until the Teacher replies yes to the conjecture } M(S, E). & \\
\text{return } M(S, E).
\end{align*}

- range over infinite sets
- finding witnesses potentially requires checking infinite rows
This table indicates that the learning algorithm works by incrementally building an observation table as a pair which the teacher replies whether which the teacher will reply whether no, with a counter-example \( t \) or not; and consistent and a new hypothesis which answers two types of queries:

- Membership queries
- 

For this to be well-defined, we need to have a DFA accepting a certain (unknown) language. As an example, and to set notation, consider the following language which should be accepted by the Teacher replies yes to the conjecture \( M(S, E) \). The new table is closed, but not consistent: rows

\[ \text{find } s_1, s_2 \in S, a \in A, \text{ and } e \in E \text{ such that } \]

\[ \text{row}(s_1) = \text{row}(s_2) \text{ and } \mathcal{L}(s_1ae) \neq \mathcal{L}(s_2ae) \]

\[ E \leftarrow E \cup \{ae\} \]

\[ \text{Make the conjecture } M(S, E) \]

\[ \text{if the Teacher replies no, with a counter-example } t \]

\[ S \leftarrow S \cup \text{prefixes}(t) \]

\[ \text{until the Teacher replies yes to the conjecture } M(S, E). \]

\[ \text{return } M(S, E) \]

\[ \text{Step } 1 \]

\[ S, E \leftarrow \{\epsilon\} \]

\[ \text{repeat} \]

\[ \text{while } (S, E) \text{ is not closed or not consistent} \]

\[ \text{if } (S, E) \text{ is not closed} \]

\[ \text{find } s_1 \in S, a \in A \text{ such that } \]

\[ \text{row}(s_1a) \neq \text{row}(s), \text{ for all } s \in S \]

\[ S \leftarrow S \cup \{s_1a\} \]

\[ \text{if } (S, E) \text{ is not consistent} \]

\[ \text{find } s_1, s_2 \in S, a \in A, \text{ and } e \in E \text{ such that } \]

\[ \text{row}(s_1) = \text{row}(s_2) \text{ and } \mathcal{L}(s_1ae) \neq \mathcal{L}(s_2ae) \]

\[ E \leftarrow E \cup \{ae\} \]

range over infinite sets

finding witnesses potentially requires checking infinite rows

\( t \) has only finitely many prefixes, but an infinite \( S \) is necessary
Challenges

L* LEARNER
1 $S, E \leftarrow \{\epsilon\}$
2 repeat
3  while $(S, E)$ is not closed or not consistent
4    if $(S, E)$ is not closed
5      find $s_1 \in S, a \in A$ such that $row(s_1a) \neq row(s)$, for all $s \in S$
6      $S \leftarrow S \cup \{s_1a\}$
7    if $(S, E)$ is not consistent
8      find $s_1, s_2 \in S, a \in A, e \in E$ such that $row(s_1) = row(s_2)$ and $L(s_1ae) \neq L(s_2ae)$
9      $E \leftarrow E \cup \{ae\}$
10     Make the conjecture $M(S, E)$
11     if the Teacher replies no, with a counter-example $t$
12       $S \leftarrow S \cup \text{prefixes}(t)$
13     until the Teacher replies yes to the conjecture $M(S, E)$.
14 return $M(S, E)$

range over infinite sets
finding witnesses potentially requires checking infinite rows
t has only finitely many prefixes, but an infinite S is necessary
no finite automaton accepts $L_1$
The learning algorithm works by incrementally building an observation table as a pair determined by the language does not contain the words table (over transition table).

In this section, we give an overview of the work developed in the

2. Overview of the approach

Figure 1.

The algorithm has

2.1 Simple example of execution

The language

\[ L = \{ a^n b^{2n} | n \geq 0 \} \]

provides a procedure to learn the minimal deterministic finite automaton. Therefore, line 12 of the algorithm requires checking infinite rows.

\[ \text{LEARNER} \]

1. \( S, E \leftarrow \{ \epsilon \} \)
2. repeat
3. \hspace{1em} while \((S, E)\) is not closed or not consistent
4. \hspace{2em} if \((S, E)\) is not closed
5. \hspace{3em} find \( s_1 \in S, a \in A \) such that
6. \hspace{3em} \( \text{row}(s_1a) \neq \text{row}(s) \), for all \( s \in S \)
7. \hspace{2em} \( S \leftarrow S \cup \{ s_1a \} \)
8. \hspace{2em} if \((S, E)\) is not consistent
9. \hspace{3em} find \( s_1, s_2 \in S, a \in A, \text{ and } e \in E \) such that
10. \hspace{3em} \( \text{row}(s_1) = \text{row}(s_2) \text{ and } L(s_1ae) \neq L(s_2ae) \)
11. \hspace{3em} \( E \leftarrow E \cup \{ ae \} \)
12. Make the conjecture \( M(S, E) \)
13. \hspace{1em} if the Teacher replies no, with a counter-example \( t \)
14. \hspace{2em} \( S \leftarrow S \cup \text{prefixes}(t) \)
15. until the Teacher replies yes to the conjecture \( M(S, E) \).
16. return \( M(S, E) \)

(P1) the sets \( S, S \cdot A \) and \( E \) admit a finite representation up to permutations;
(P2) row is such that \( \text{row}(\pi(s))(\pi(e)) = \text{row}(s)(e) \), for all \( s \in S \) and \( e \in E \).

Observation table admits a finite symbolic representation.
Nominal L*

6' \quad S \leftarrow S \cup \text{orb}(sa)
9' \quad E \leftarrow E \cup \text{orb}(ae)
12' \quad E \leftarrow E \cup \text{prefixes}(\text{orb}(t))

only 3 lines changed!
Nominal $L^*$

6' $S \leftarrow S \cup \text{orb}(sa)$
9' $E \leftarrow E \cup \text{orb}(ae)$
12' $E \leftarrow E \cup \text{prefixes}(\text{orb}(t))$

only 3 lines changed!

not really... all definitions have to be adapted to nominal/equivariant.
Nominal $L^*$

6' $S \leftarrow S \cup \text{orb}(sa)$
9' $E \leftarrow E \cup \text{orb}(ae)$
12' $E \leftarrow E \cup \text{prefixes}(\text{orb}(t))$

only 3 lines changed!

not really… all definitions have to be adapted to nominal/equivariant.

Correctness, termination, … have to be re-proved!
Nominal L*

\begin{align*}
6' & \quad S \leftarrow S \cup \text{orb}(sa) \\
9' & \quad E \leftarrow E \cup \text{orb}(ae) \\
12' & \quad E \leftarrow E \cup \text{prefixes}(\text{orb}(t))
\end{align*}

only 3 lines changed!

not really... all definitions have to be adapted to nominal/equivariant.

Correctness, termination, ... have to be re-proved!
Categorical glasses

$(S, E, \text{row})$ is closed if for all $t \in S \cdot A$ there exists an $s \in S$ such that $\text{row}(t) = \text{row}(s)$. 
Categorical glasses

(S, E, row) is closed if for all $t \in S \cdot A$ there exists an $s \in S$ such that $\text{row}(t) = \text{row}(s)$. 
Categorical glasses

\[(S, E, \text{row}) \text{ is } \text{closed} \text{ if for all } t \in S \cdot A \text{ there exists an } s \in S \text{ such that } \text{row}(t) = \text{row}(s).\]

\[(S, E, \text{row}) \text{ is } \text{consistent} \text{ if whenever } s_1, s_2 \in S \text{ are such that } \text{row}(s_1) = \text{row}(s_2), \text{ for all } a \in A, \text{row}(s_1 a) = \text{row}(s_2 a).\]
(\(S, E, \text{row}\)) is **closed** if for all \(t \in S \cdot A\) there exists an \(s \in S\) such that \(\text{row}(t) = \text{row}(s)\).

(\(S, E, \text{row}\)) is **consistent** if whenever \(s_1, s_2 \in S\) are such that \(\text{row}(s_1) = \text{row}(s_2)\), for all \(a \in A\), \(\text{row}(s_1 a) = \text{row}(s_2 a)\).
(S, E, row) is closed if for all \( t \in S \cdot A \) there exists an \( s \in S \) such that \( \text{row}(t) = \text{row}(s) \).

Pretty.... but is it useful?

(S, E, row) is consistent if for all \( s_1, s_2 \in S \) are such that \( \text{row}(s_1) = \text{row}(s_2) \), for all \( a \in A, \text{row}(s_1 a) = \text{row}(s_2 a) \).
The power of abstraction

Definitions are the same

Proof of correctness is the same
The power of abstraction

Definitions are the same

Proof of correctness is the same

\[ X \rightarrow 2 \times X^A \]

DFA in Nom

\[ A^* \xrightarrow{\text{init}} Q \xrightarrow{\text{final}} 2^{A^*} \]

1 \[ \lambda \]

2 \[ \text{ev}_\lambda \]

\[ (A^*)^A \rightarrow Q^A \rightarrow (2^{A^*})^A \]
Abstract automata

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

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

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

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

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

DFAs

\( C = \text{Set} \)

\( F = (\cdot) \times A \)
Abstract automata

Category $C =$ universe of state-spaces

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

DFAs

$C = \text{Set}$

$F = (-) \times A$

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

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

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

DFAs

$C = \text{Set}$

$F = (-) \times A$

Diagram:

$Q \times A$

$\delta_Q$

init$_Q$

$Q$

out$_Q$

$1$

$Y$
Abstract automata

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

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

DFAs

$C = \text{Set}$

$F = (-) \times A$

\[
\begin{align*}
Q \times A & \xrightarrow{\delta_Q} Q \\
\downarrow & \\
\text{init}_Q & \rightarrow Q & \text{out}_Q \\
\uparrow & \\
1 & \rightarrow Q & Y \\
\end{align*}
\]

$q_0 \in Q$
Abstract automata

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

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

DFAs

$C = \text{Set}$

$F = (-) \times A$

\[
\begin{align*}
Q \times A & \\
\downarrow \delta_Q & \\
init_Q & \quad Q & \quad \text{out}_Q \\
1 & \quad 2 \\
q_0 \in Q
\end{align*}
\]
Abstract automata

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

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

DFAs

$C = \text{Set}$

$F = (\neg) \times A$

\[
\begin{aligned}
 & Q \times A \\
 & \downarrow \delta_Q \\
 & \text{init}_Q \quad Q \quad \text{out}_Q \\
 & \quad 1 \quad \quad 2 \\
 & q_0 \in Q \quad F \subseteq Q
\end{aligned}
\]
Abstract learning

Abstract observation data structure
Abstract learning

Abstract observation data structure approximates

Target minimal automaton

$\delta_Q$

$\text{init}_Q$

$\text{out}_Q$

$I$

$Y$
Abstract learning

Abstract observation data structure

Hypothesis automaton

approximates

abstract closedness and consistency

Target minimal automaton

\[ \text{Hypothesis automaton} \]

\[ \begin{align*}
F & \quad H \\
\downarrow \delta_H & \\
\text{init}_H & \quad \text{out}_H \\
I & \quad Y
\end{align*} \]

\[ \text{Target minimal automaton} \]

\[ \begin{align*}
\text{target minimal automaton} & \\
F & \quad Q \\
\downarrow \delta_Q & \\
\text{init}_Q & \quad \text{out}_Q \\
I & \quad Y
\end{align*} \]
Abstract learning

Abstract observation data structure

Hypothesis automaton

Target minimal automaton

abstract closedness and consistency

approximates

General correctness theorem

Guidelines for implementation
Abstract learning

Abstract observation data structure

approximates

abstract closedness and consistency

Hypothesis automaton

\[ \begin{align*}
F \delta_H & \downarrow \\
\delta_H & \downarrow \delta_H \\
\text{init}_H & \downarrow \text{init}_H \\
I & \downarrow \text{init}_H \\
\end{align*} \]

Target minimal automaton

\[ \begin{align*}
F Q & \downarrow \delta_Q \\
\delta_Q & \downarrow \delta_Q \\
\text{init}_Q & \downarrow \text{init}_Q \\
I & \downarrow \text{init}_Q \\
\end{align*} \]

General correctness theorem

Guidelines for implementation

Gerco van Heerdt, Matteo Sammartino, Alexandra Silva
Other automata & optimizations
Other automata & optimizations

Change base category

<table>
<thead>
<tr>
<th>Set</th>
<th>Nominal automata</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nom</td>
<td>Nominal automata</td>
</tr>
<tr>
<td>Vect</td>
<td>Weighted automata</td>
</tr>
</tbody>
</table>
Other automata & optimizations

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>

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>Double powerset</td>
<td>Alternating automata</td>
</tr>
<tr>
<td>Maybe monad</td>
<td>Partial automata</td>
</tr>
</tbody>
</table>
## Other automata & optimizations

<table>
<thead>
<tr>
<th></th>
<th>Change base category</th>
<th>Change main data structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>DFAs</td>
<td>Observation tables</td>
</tr>
<tr>
<td>Nom</td>
<td>Nominal automata</td>
<td>Discrimination trees</td>
</tr>
<tr>
<td>Vect</td>
<td>Weighted automata</td>
<td></td>
</tr>
</tbody>
</table>

### Side-effects (via monads)

- Powerset
- Powerset with intersection
- Double powerset
- Maybe monad
- NFAs
- Universal automata
- Alternating automata
- Partial automata
Other automata & optimizations

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

Change main data structure
- Nom: Nominal automata
- Set: DFAs
- Vect: Weighted automata

Other automata & optimizations
- NFAs
- Universal automata
- Alternating automata
- Partial automata

Side-effects (via monads)
- Powerset
- Powerset with intersection
- Double powerset
- Maybe monad

Learning Nominal Automata (POPL '17)
Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, Michal Szynwelski
Other automata & optimizations

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>

Change main data structure

- **Learning Nominal Automata (POPL ’17)**
  Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, Michal Szynwelski
- **Discrimination trees**
- **Learning Automata with Side-effects** *(arXiv:1704.08055)*
  Gerco van Heerdt, Matteo Sammartino, Alexandra Silva

Side-effects (via monads)

- Powerset
- Powerset with intersection
- Double powerset
- Maybe monad
- NFAs
- Universal automata
- Alternating automata
- Partial automata
Connections with other algorithms

- Automaton type
- Automata Learning algorithms
- Minimization algorithms
- Testing algorithms
- Optimizations
Ongoing and future work

• **Library & tool** to learn control + data-flow models (as **nominal automata**)

• Applications:
  
  • Specification mining
  
  • Network verification, with **Amazon**
  
  • Verification of cryptographic protocols
  
  • Ransomware detection
Ongoing and future work

Learning convex automata

Rich algebraic structure

Challenging analytical properties
Conclusions

Category theory is a good playground to understand and generalise algorithms
Conclusions

Category theory is a good playground to understand and generalise algorithms

Unveils connections and sets the scene

—

No free lunch
Questions?