

An Abstract Automata Learning Framework

Gerco van Heerdt¹

June 30, 2016

¹Supervisors: Frits Vaandrager, Alexandra Silva, Joshua Moerman

Active automata learning

- ▶ Active automata learning learns an automaton describing the behaviour of a system by providing inputs and observing outputs
- ▶ Enables verification methods that work on an automaton
- ▶ Allows to compare different implementations of e.g. a network protocol

Capturing systems more precisely requires more complex types of automata and accordingly more complicated learning algorithms.

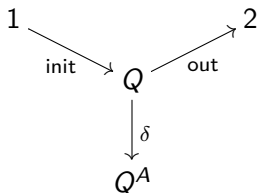
Idea: understanding the main concepts on an abstract level can help to develop and reason about new algorithms.

Deterministic automaton

Fix an *alphabet* A .

Definition

A *deterministic automaton* is a set of states Q with three functions:



$$1 = \{*\}$$

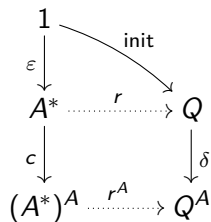
$$2 = \{0, 1\}$$

Q^A is the set of functions $A \rightarrow Q$

The 'automaton' of words

For each 'automaton' Q there is a *reachability map* $r: A^* \rightarrow Q$ that is unique in preserving the initial state and transition function.

$$c(u)(a) = ua$$

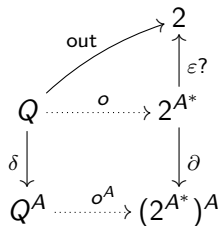


$$r(\epsilon) = \text{init}(\ast)$$

$$r(ua) = \delta(r(u))(a)$$

The 'automaton' of languages

For each 'automaton' Q there is an *observability map* $o: Q \rightarrow 2^{A^*}$ that is unique in preserving the outputs and transition function.



$$\epsilon?(l) = l(\epsilon)$$

$$\partial(l)(a)(u) = l(au)$$

$$o(q)(\epsilon) = \text{out}(q)$$

$$o(q)(au) = o(\delta(q)(a))(u)$$

Minimality

$$A^* \xrightarrow{r} Q \xrightarrow{o} 2^{A^*}$$

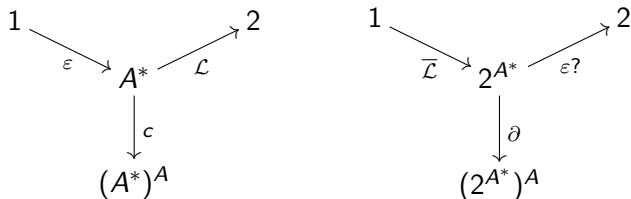
Q is

- ▶ *reachable* if r is surjective
- ▶ *observable* if o is injective
- ▶ *minimal* if it is reachable and observable

Languages

A language $\mathcal{L}: A^* \rightarrow 2$ can also be seen as a function $\bar{\mathcal{L}}: 1 \rightarrow 2^{A^*}$.

These turn A^* and 2^{A^*} into automata:



Given a language, the observability map of A^* is called the *total response* of the language. It coincides with the reachability map of 2^{A^*} and is given by

$$t_{\mathcal{L}}: A^* \rightarrow 2^{A^*} \qquad t_{\mathcal{L}}(u)(v) = \mathcal{L}(uv)$$

Language of an automaton

The language of an automaton Q is given by

$$\mathcal{L}_Q = A^* \xrightarrow{r} Q \xrightarrow{\text{out}} 2 \quad \overline{\mathcal{L}_Q} = 1 \xrightarrow{\text{init}} Q \xrightarrow{o} 2^{A^*}$$

Equipping A^* with the output \mathcal{L}_Q and 2^{A^*} with initial state $\overline{\mathcal{L}_Q}$,

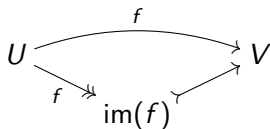
$$r: A^* \rightarrow Q \quad o: Q \rightarrow 2^{A^*}$$

are automaton homomorphisms. The composition $o \circ r$ is the total response $t_{\mathcal{L}_Q}: A^* \rightarrow 2^{A^*}$.

If there is any automaton homomorphism $U \rightarrow V$ for automata U and V , then $\mathcal{L}_U = \mathcal{L}_V$.

Image factorisations

Each function $f: U \rightarrow V$ can be factorised as a surjective function followed by an injective function:



Diagonal fill-in

For each commutative diagram

$$\begin{array}{ccc} U & \xrightarrow{i} & V \\ f \downarrow & & \downarrow g \\ W & \xrightarrow{j} & X \end{array}$$

with i surjective and j injective, there is a unique diagonal $d: V \rightarrow W$ rendering commutative both triangles in

$$\begin{array}{ccc} U & \xrightarrow{i} & V \\ f \downarrow & \swarrow d & \downarrow g \\ W & \xrightarrow{j} & X \end{array}$$

$$d(i(u)) = f(u)$$

$$\begin{aligned} i(u_1) = i(u_2) &\Rightarrow g(i(u_1)) = g(i(u_2)) \\ &\Rightarrow j(f(u_1)) = j(f(u_2)) \\ &\Rightarrow f(u_1) = f(u_2) \end{aligned}$$

Factorising automaton homomorphisms

If $f: U \rightarrow V$ is an automaton homomorphism, we can equip $\text{im}(f)$ with an automaton structure:

$$\begin{array}{ccc} U & \xrightarrow{f} & \text{im}(f) \\ \delta \downarrow & & \downarrow \delta \\ U^A & & V \\ f^A \downarrow & \delta \swarrow & \downarrow \delta \\ \text{im}(f)^A & \xrightarrow{\quad} & V^A \end{array}$$

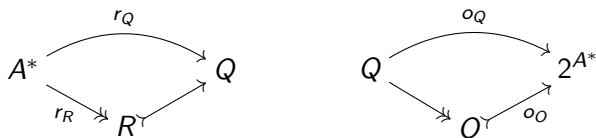
init = $1 \xrightarrow{\text{init}} U \xrightarrow{f} \text{im}(f)$
out = $\text{im}(f) \xrightarrow{\quad} V \xrightarrow{\text{out}} 2$

With these definitions, the function $f: U \rightarrow \text{im}(f)$ and the inclusion $\text{im}(f) \rightarrow V$ are automaton homomorphisms.

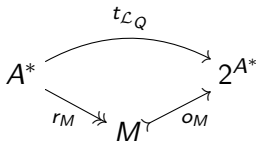
The diagonal property also works.

Theoretical minimisation

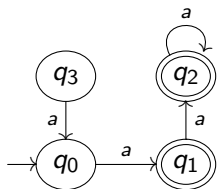
Removing the unreachable states from an automaton Q can be described as taking the image of the reachability map while merging equivalent states takes the image of the observability map:



We can even minimise in one step by taking the image of the total response:



Total response as a table



$$t_{\mathcal{L}}: A^* \rightarrow 2^{A^*}$$
$$t_{\mathcal{L}}(u)(v) = \mathcal{L}(uv)$$

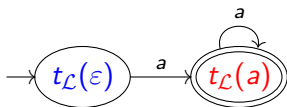
	ε	a	aa	aaa	\dots
ε	0	1	1	1	\dots
a	1	1	1	1	\dots
aa	1	1	1	1	\dots
\vdots	\vdots	\vdots	\vdots	\vdots	\ddots

$$M = \{t_{\mathcal{L}}(u) \mid u \in A^*\}$$

$$\text{init}(\ast) = t_{\mathcal{L}}(\varepsilon)$$

$$\text{out}(l) = l(\varepsilon)$$

$$\delta(t_{\mathcal{L}}(u))(a) = t_{\mathcal{L}}(ua)$$



Automaton wrapper

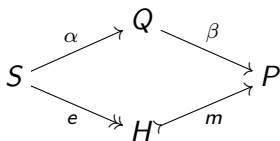
Definition

A *wrapper* for an automaton Q is a tuple (S, P, α, β) , where $\alpha: S \rightarrow Q$ and $\beta: Q \rightarrow P$ are functions.

$$S \xrightarrow{\alpha} Q \xrightarrow{\beta} P$$

The function α **selects** states of Q while β **classifies** them.

The *potential hypothesis* is the factorisation of $\beta \circ \alpha$:



Writing ξ for $\beta \circ \alpha$, we take $H = \text{im}(\xi)$ and

$$e(s) = \xi(s)$$

$$m(h) = h$$

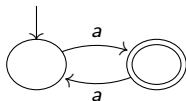
Automaton wrapper

Example: observation table

Let $S, E \subseteq A^*$. A wrapper for Q can be defined by the compositions

$$S \xrightarrow{\sigma} A^* \xrightarrow{r} Q \xrightarrow{o} 2^{A^*} \xrightarrow{\pi} 2^E$$
$$\sigma(s) = s \qquad \pi(l)(e) = l(e)$$

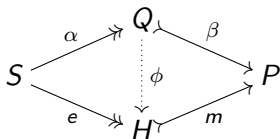
For example, if Q is the automaton on the left, $S = \{\varepsilon, a, aa\}$, and $E = \{a, aa\}$, then $\xi: S \rightarrow 2^E$ is given by the table on the right:



	a	aa
ε	1	0
a	0	1
aa	1	0

Recovering Q

If α is surjective and β injective, then the diagonal ϕ below is an isomorphism:

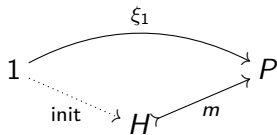
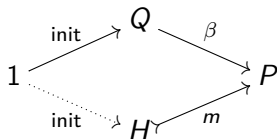


The hypothesis automaton H should be such that ϕ is an automaton isomorphism.

Initialisation

Definition

The wrapper is *initialised* if there is a function $\text{init}: 1 \rightarrow H$ making the left diagram commute.



The wrapper is initialised if and only if

$$\exists s \in S [\xi(s) = \xi_1(*)].$$

Initialisation

Example

For a wrapper

$$S \xrightarrow{\sigma} A^* \xrightarrow{r} Q \xrightarrow{o} 2^{A^*} \xrightarrow{\pi} 2^E$$

$\xi_1: 1 \rightarrow 2^E$ is just the restriction of $\overline{\mathcal{L}_Q}: 1 \rightarrow 2^{A^*}$ to E :

$$\xi_1 = 1 \xrightarrow{\text{init}} Q \xrightarrow{o} 2^{A^*} \xrightarrow{\pi} 2^E$$

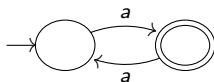
$o \circ \text{init}$ is the language accepted by the initial state of Q , so

$$\xi_1(*) (e) = \mathcal{L}_Q(e)$$

Initialisation

Example

Fix $E = \{a, aa\}$ and let Q be as below.



The function $\xi_1: 1 \rightarrow 2^E$ can be seen as a row:

$$\begin{array}{c|cc} & a & aa \\ \hline 1 & 1 & 0 \end{array}$$

A wrapper is initialised if and only if its table contains this row. Thus, below on the left we have initialisation, but not on the right.

$$\begin{array}{c|cc} & a & aa \\ \hline aa & 1 & 0 \end{array}$$

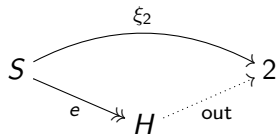
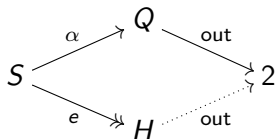
$$\begin{array}{c|cc} & a & aa \\ \hline a & 0 & 1 \end{array}$$

The canonical way of ensuring initialisation is having $\varepsilon \in S$.

Responsiveness

Definition

The wrapper is *responsive* if there is a function $\text{out}: H \rightarrow 2$ making the left diagram commute.



The wrapper is responsive if and only if

$$\forall s, t \in S [\xi(s) = \xi(t) \implies \xi_2(s) = \xi_2(t)].$$

Responsiveness

Example

For a wrapper

$$S \xrightarrow{\sigma} A^* \xrightarrow{r} Q \xrightarrow{o} 2^{A^*} \xrightarrow{\pi} 2^E$$

$\xi_2: S \rightarrow 2$ is just the restriction of $\mathcal{L}_Q: A^* \rightarrow 2$ to S :

$$\xi_2 = S \xrightarrow{\sigma} A^* \xrightarrow{r} Q \xrightarrow{\text{out}} 2$$

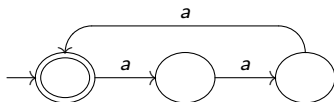
$\text{out} \circ r$ is just the language accepted by Q , so

$$\xi_2(s) = \mathcal{L}_Q(s)$$

Responsiveness

Example

Fix $S = \{\varepsilon, a\}$ and suppose Q is as given below.



The function $\xi_2: S \rightarrow 2$ can be seen as a column:

ε	1
a	0

A table is responsive if and only if all two elements of S that are equal in the table are equal in this column. Thus, the table on the left is responsive, but the one on the right is not.

	a	aa
ε	0	0
a	0	1

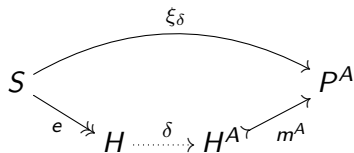
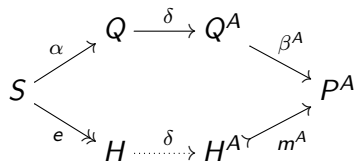
	a	$aaaa$
ε	0	0
a	0	0

The canonical way of ensuring responsiveness is having $\varepsilon \in E$.

Dynamism

Definition

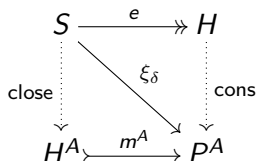
The wrapper is *dynamical* if there is a function $\delta: H \rightarrow H^A$ making the left diagram commute.



This property can be split into two parts.

Closedness and consistency

The wrapper is dynamical if and only if it is closed and consistent.



The wrapper is closed if and only if

$$\forall s \in S, a \in A \exists t \in S [\xi(t) = \xi_\delta(s)(a)].$$

It is consistent if and only if

$$\forall s, t \in S [\xi(s) = \xi(t) \implies \xi_\delta(s) = \xi_\delta(t)].$$

Dynamism

Example

For a wrapper

$$S \xrightarrow{\sigma} A^* \xrightarrow{r} Q \xrightarrow{o} 2^{A^*} \xrightarrow{\pi} 2^E$$

$\xi_\delta: S \rightarrow (2^E)^A$ is given by

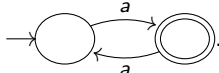
$$\xi_\delta = S \xrightarrow{\sigma} A^* \xrightarrow{r} Q \xrightarrow{\delta} Q^A \xrightarrow{o^A} (2^{A^*})^A \xrightarrow{\pi^A} (2^E)^A$$

With some effort, using that r preserves the transition function, we can derive the expected definition:

$$\xi_\delta(s)(a)(e) = \mathcal{L}_Q(sae)$$

Closedness

Example

Let Q be given by . The function $\xi_\delta: S \rightarrow (2^E)^A$

can be seen as a set of rows: one for each word $sa \in S \cdot A$. We display these rows below the table, so e.g. for $S = \{\varepsilon\}$ (left) or $S = \{\varepsilon, a\}$ (right):

	ε	a
ε	0	1
a	1	0

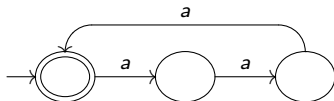
	ε	a
ε	0	1
a	1	0
a	1	0
aa	0	1

A table is closed if and only if the rows in the lower part can all be found in the upper part. The left table is not closed. Adding the missing row by adding a to S fixes this.

Consistency

Example

Let Q be given in



The function $\xi_\delta: S \rightarrow (2^E)^A$ can also be seen as a set of columns: one for each word $ae \in A \cdot E$. We can display these rows at the right of the table, so e.g. for $E = \{\varepsilon\}$ (left) or $E = \{\varepsilon, a\}$ (right):

	ε	a
a	0	0
aa	0	1

	ε	a	a	aa
a	0	0	0	1
aa	0	1	1	0

A table is consistent if and only all two rows that are equal in the left part are also equal in the right part. The left table is not consistent. We add the column that separates $\xi(a)$ from $\xi(aa)$ by adding a to E .

Observation structure

Apart from

$$\xi = S \xrightarrow{\alpha} Q \xrightarrow{\beta} P$$

we need to know the functions

$$\xi_1 = 1 \xrightarrow{\text{init}} Q \xrightarrow{\beta} P$$

$$\xi_2 = S \xrightarrow{\alpha} Q \xrightarrow{\text{out}} 2$$

$$\xi_\delta = S \xrightarrow{\alpha} Q \xrightarrow{\delta} Q^A \xrightarrow{\beta^A} P^A$$

These functions are referred to as the *observation structure* of the automaton wrapper.

Angluin's algorithm

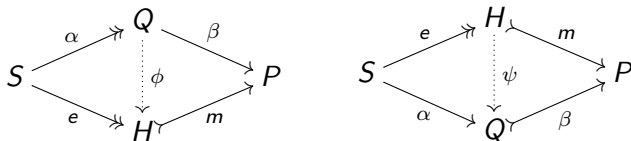
1. Initialize $S = E = \{\varepsilon\}$. The wrapper is always initialized and responsive.
2. Enforce closedness and consistency.
3. Construct the hypothesis.
4. Ask if the hypothesis is correct.
5. If it is, we are done; otherwise, obtain a counterexample, add all its prefixes to S , and go back to 2.

Termination relies on the fact that H increases in size after fixing a closedness/consistency defect and after processing a counterexample.

Results

We say that the wrapper reaches Q if α is surjective; it observes Q if β is injective.

- ▶ If the wrapper reaches Q , ϕ exists; if it observes Q , ψ exists:



- ▶ If the wrapper reaches Q , it is initialised and closed; if it observes Q , it is responsive and consistent.
- ▶ Assume the hypothesis can be constructed. If ϕ exists, it is an automaton homomorphism; if ψ exists, it is an automaton homomorphism.

Main results

Theorem

If the wrapper reaches and observes Q , then the hypothesis can be constructed and is isomorphic to Q .

Theorem

If Q is reachable and the wrapper observes Q and is initialised and closed, then it also reaches Q .

Theorem

If Q is observable and the wrapper reaches Q and is responsive and consistent, then it also observes Q .

Learning examples

Consider the familiar wrapper

$$S \xrightarrow{\sigma} A^* \xrightarrow{r} Q \xrightarrow{o} 2^{A^*} \xrightarrow{\pi} 2^E$$

and assume Q is minimal. To find Q up to automaton isomorphism, we have the following results:

- ▶ If $r \circ \sigma$ is surjective (every state is reached by a word in S), we only have to ensure responsiveness and consistency.
- ▶ If $\pi \circ o$ is injective (every pair of different states is distinguished by a word in E), we only have to ensure initialisation and closedness.

Reachability analysis

Suppose Q is a given automaton and we want to find its reachable equivalent R .

Given $S \subseteq A^*$, we have a wrapper for R :

$$S \xrightarrow{\sigma} A^* \xrightarrow{r} R \longrightarrow Q$$

In fact, because $R \subseteq Q$, it observes R . Therefore, we are only concerned with initialisation and closedness.

- ▶ **Initialisation:** there must be a word in S reaching the initial state of Q .
- ▶ **Closedness:** all one-symbol successors of words in S must reach a state reached by a word in S .

Similar to the observation table situation, but there is one column that for a given $s \in S$ contains the state of Q reached after reading s .

Observability analysis

Suppose Q is a given automaton and we want to find its observable equivalent O .

Given $E \subseteq A^*$, we have a wrapper for O that reaches O :

$$Q \longrightarrow \gg O \xrightarrow{o} 2^{A^*} \xrightarrow{\pi} 2^E$$

Therefore, we are only concerned with responsiveness and consistency.

- ▶ **Responsiveness:** if two states are equivalent up to E , they must have the same direct output.
- ▶ **Consistency:** if two states are equivalent up to E , then their successors for each given input symbol must be equivalent up to E .

Similar to the observation table situation, but rows are labelled by states of Q .

Discrimination tree

Each time an element is added to E to distinguish two currently equal rows in a table, an additional query is needed for *every* row.

More localized distinguishment is admitted by a *discrimination tree*.

Given a set L of labels, define

$$\text{DT}_L ::= \text{Leaf}(L) \mid \text{Node}(A^*, \text{DT}_L^2).$$

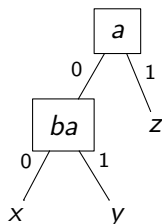
A tree $\tau \in \text{DT}_L$ induces a language classifier $\pi_\tau: 2^{A^*} \rightarrow L$ given by

$$\begin{aligned}\pi_{\text{Leaf}(l)}(f) &= l \\ \pi_{\text{Node}(v,c)}(f) &= \pi_{c(f(v))}(f)\end{aligned}$$

Discrimination tree

Example

The tree



classifies the language $\{b, ba\}$ into y .

Precomposing π_τ with the total response of a language \mathcal{L} yields a function $\text{sift}_\tau: A^* \rightarrow L$.

Calculating the label for a word $u \in A^*$ is called *sifting* u through τ . At each internal node with label v we test if $uv \in \mathcal{L}$.

Discrimination tree wrapper

For a wrapper

$$S \xrightarrow{\sigma} A^* \xrightarrow{r} Q \xrightarrow{o} 2^{A^*} \xrightarrow{\pi_\tau} L$$

we obtain the following definitions:

- ▶ **Initialisation:** there is an $s \in S$ such that $\text{sift}_\tau(s) = \text{sift}_\tau(\varepsilon)$.
- ▶ **Closedness:** for all $s \in S$ and $a \in A$ there is a $t \in S$ such that $\text{sift}_\tau(t) = \text{sift}_\tau(sa)$.
- ▶ **Responsiveness:** for all $s_1, s_2 \in S$ such that $\text{sift}_\tau(s_1) = \text{sift}_\tau(s_2)$ we have $\mathcal{L}(s_1) = \mathcal{L}(s_2)$.
- ▶ **Consistency:** for all $s_1, s_2 \in S$ such that $\text{sift}_\tau(s_1) = \text{sift}_\tau(s_2)$ we have $\text{sift}_\tau(s_1a) = \text{sift}_\tau(s_2a)$ for each $a \in A$.

If $s_1, s_2 \in S$ sift into the same leaf, but satisfy $\mathcal{L}(s_1v) \neq \mathcal{L}(s_2v)$ for some $v \in A^*$, we can turn the leaf into an internal node labelled v .

- ▶ Explicit category theoretical definitions and results
- ▶ Explains some conformance testing algorithms using wrapper results
- ▶ Instantiation to Moore and Mealy machines
- ▶ Lifting to the category of algebras for a monad
 - ▶ Instantiation to semilattice automata
 - ▶ Instantiation to automata with group actions, in particular nominal automata