Automata Learning: A Categorical Perspective

Bart Jacobs and Alexandra Silva

Institute for Computing and Information Sciences, Radboud University Nijmegen
{bart,alexandra}@cs.ru.nl
April 2, 2014

Dedicated to Prakash Panangaden on the occasion of his 60th birthday

Abstract. Automata learning is a known technique to infer a finite state machine from a set of observations. In this paper, we revisit Angluin's original algorithm from a categorical perspective. This abstract view on the main ingredients of the algorithm lays a uniform framework to derive algorithms for other types of automata. We show a straightforward generalization to Moore and Mealy machines, which yields an algorithm already know in the literature, and we discuss generalizations to other types of automata, including weighted automata.

1 Introduction

One of the topics Prakash Panangaden has always been interested in is learning. He is not only a great scholar himself, but he also has a great drive to spread his knowledge, through his many lectures and discussions with colleagues around the world. We, as authors, have enjoyed and been inspired by him and his work.

On a more technical level, learning is an active area in computer science, especially in artificial intelligence. It involves deducing a (minimal) machine from observations. In this paper we explore, redescribe, and generalize part of this research using a categorical perspective. In this way we apply Prakash's favourite language to an area that is close to him — since he is a member of McGill's Reasoning and Learning Lab.

Finite automata or state machines have a wide range of applications in Computer Science. One of their applications is in verification of software systems and security protocols. Typically, the behavior of the system is modeled by a finite state machine and then desired properties, encoded in an appropriate logic, are checked against the model. Models are unfortunately not always available and the rapid changes in the system require frequent adaptations. This has motivated a lot of research into inferring or learning a model from a given system just by observing its behavior or response to certain queries.

Automata learning, or regular inference [4], is a widely used technique for creating an automaton model from observations. The regular inference algorithms provide sequences of input, so called membership queries, to a system, and observe the responses to infer an automaton. In addition, equivalence queries check whether the inferred automaton is equivalent to the system being learned. The original algorithm [4], by Dana Angluin, works for deterministic finite automata, but since then it has been extended and generalized to other types of automata [5,21,1], including Mealy machines and I/O automata, and even a special class of context-free grammars.

F. van Breugel et al. (Eds.): Panangaden Festschrift, LNCS 8464, pp. 384–406, 2014.

Category theory provides an abstract framework to study structures in mathematics and computer science. Automata are prime examples of such structures and have been studied using both algebras and coalgebras (see *e.g.* [13] and [19]) in two somewhat independent research streams. In the last few years, strengths of both perspectives on automata are being combined fruitfully, leading to the derivation of new algorithms and results. In this paper, we again explore the power of abstraction and recast the main ingredients of Angluin's algorithm using basic categorical concepts, from algebra and coalgebra, which open the door to instantiations to other types of automata, in other categories, without having to reprove correctness of the algorithm.

In this paper we sketch the straightforward generalization from deterministic automata to Moore and Mealy machines, which yields an algorithm known in the literature [21,1,2], that had been developed inspired by Angluin's algorithm but without an explicit connection of the similarities and differences in both. Our abstract view provides this connection and opens the door to even further generalizations to other types of automata.

In the proof of minimality of the inferred automaton we have used a technique that goes back to Kalman and has since then been explored by a multitude of authors, including Prakash himself. Among his many research interests, Prakash's recent activities includes using Stone-type dualities to minimize automata. He has observed that there might be connections with automata learning (personal communication). This paper provides a first step towards exploring this connection.

Organization of the paper. The rest of the paper is organized as follows. In Section 2, we recall the basic ingredients of Angluin's algorithm for deterministic automata and show how we can recast them in a categorical language. In Section 3, we present the first generalization of the algorithm based on the categorical reformulation by varying the functor under consideration and provide a learning algorithm for Mealy/Moore automata. In Section 4, we show a different type of generalization by changing the base category in which the automata are considered from **Sets** to **Vect**, obtaining in this manner an algorithm for linear weighted automata.

2 Automata Learning: The Basic Algorithm

In this section, we explain the ingredients of Angluin's original algorithm for learning deterministic finite automata and rephrase them using basic categorical constructs as we proceed.

Let us first introduce some notation and basic definitions. Let A be a finite set of labels, often called an alphabet, and A^* the set of finite words or sequences of elements of A. We will use λ to denote the empty word and, given two words $u, v \in A^*$, uv denotes their concatenation.

A language over A is a subset of words in A^*, that is $L \in 2^{A^*}$. We will often switch between the equivalent representation of a language as a set of words and as its characteristic function. Given a language L and a word $u \in A^*$, we write $L(u)$ to denote 1 if $u \in L$ and 0 otherwise.

Given two languages U and V, we will denote by $U \cdot V$ (or simply UV) the concatenation of the two languages $U \cdot V = \{uv \mid u \in U, v \in V\}$. Given a language L and $a \in A$ we can define its right and left derivative by setting

$$a^{-1}L = \{u \mid au \in L\} \qquad \text{and} \qquad La^{-1} = \{u \mid ua \in L\}.$$

A language L is *prefix-closed* if $La^{-1} \subseteq L$, for all $a \in A$, and *suffix-closed* if $a^{-1}L \subseteq L$, for all $a \in A$. Note that every non-empty suffix or prefix-closed language must contain the empty word λ. We will use $\downarrow u$ (resp. $\uparrow u$) to denote the set of prefixes (resp. suffixes) of a word $u \in A^*$.

$$\downarrow u = \{w \in A^* \mid w \text{ is a prefix of } u\} \qquad \uparrow u = \{w \in A^* \mid w \text{ is a suffix of } u\}$$

For the rest of this paper we fix a language $\mathcal{L} \in 2^{A^*}$ to be learned: the master language. This learning means that we seek a finite deterministic automaton that accepts $\mathcal{L}$. Many definitions and results are parametric in $\mathcal{L}$ but we do not always make this explicit.

2.1 Observation Tables

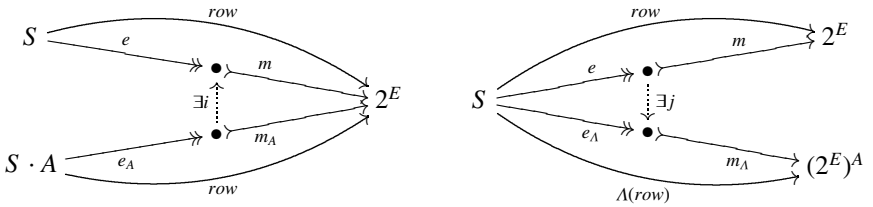
The algorithm of Angluin incrementally constructs an observation table with Boo-lean entries. The rows are labelled by words in $S \cup S \cdot A$, where S is a non-empty finite *prefix-closed* language, and the columns by a non-empty finite *suffix-closed* language E. For arbitrary $U, V \subseteq A^*$, define $row\colon U \to 2^V$ by $row(u)(v) = \mathcal{L}(uv)$. Formally, an observation table is a triple (S, E, row), where $row\colon (S \cup S \cdot A) \to 2^E$. Note that $\cup$ here is used for language union and not coproduct. Since row is fully determined by the language $\mathcal{L}$ we will from now on refer to an observation table as a pair (S, E), leaving the language $\mathcal{L}$ implicit.

There are two crucial properties of the observation table that play a key role in the algorithm of [4] allowing for the construction of a deterministic automaton from an observation table: closedness and consistency.

Definition 1 (Closed and Consistent Table [4]). *An observation table (S, E) is* closed *if for all $t \in S \cdot A$ there exists an $s \in S$ such that $row(t) = row(s)$. An observation table (S, E) is* consistent *if whenever s_1 and s_2 are elements of S such that $row(s_1) = row(s_2)$, for all $a \in A$, $row(s_1 a) = row(s_2 a)$.*

In many categories each map $f\colon A \to B$ can be factored as $f = (A \twoheadrightarrow \bullet \rightarrowtail B)$, describing f as an epimorphism followed by a monomorphism. In the category **Sets** of sets and functions epimorphisms (resp. monomorphisms) are surjections (resp. injections). Using these factorizations we come to the following categorical reformulations.

Lemma 2. *An observation table (S, E) is closed (resp. consistent) if and only if there exists a necessarilly unique map i (resp. j) such that the diagram on the left (resp. right) commutes.*



Here, $\Lambda(row)$ is obtained by abstraction (Currying), so that $\Lambda(row)(s)(a) = row(sa)$.

Proof. Suppose the table is closed according to Definition 1. Then, for every $t \in S \cdot A$ there exists an $s \in S$ such that $row(s) = row(t)$. We define i by $i(e_A(t)) = e(s)$, using that e_A is epi/surjective. It remains to show that $m \circ i = m_A$.

$$\begin{aligned}
(m \circ i)(e_A(t)) &= (m \circ e)(s) && \text{definition of } i \\
&= row(s) && \text{definition of } row \\
&= row(t) && \text{closedness assumption} \\
&= m_A(e_A(t)) && \text{definition of } row.
\end{aligned}$$

The uniqueness of i is immediate using that m is monic.

Conversely, suppose that there exists i such that $m \circ i = m_A$ and let $t = ua \in S \cdot A$. Take s such that $e(s) = i(e_A(t))$ (which exists since e is epi). We need to show $row(s) = row(t)$.

$$\begin{aligned}
row(s) &= m(e(s)) && \text{factorization of } row \\
&= m(i(e_A(t))) && \text{assumption } e(s) = i(e_A(t)) \\
&= m_A(e_A(t)) && \text{assumption } m \circ i = m_A \\
&= row(t) && \text{definition of } row.
\end{aligned}$$

Suppose the table is consistent according to Definition 1. That is, if $s_1, s_2 \in S$ are such that $row(s_1) = row(s_2)$ then, for all $a \in A$, it holds that $row(s_1 a) = row(s_2 a)$. We define j by $j(e(s)) = e_\Lambda(s)$, using that e is epi. By definition, $j \circ e = e_\Lambda$. It remains to show that j is well-defined. Let s_1, s_2 be such that $e(s_1) = e(s_2)$. We need to show $e_\Lambda(s_1) = e_\Lambda(s_2)$.

$$\begin{aligned}
e(s_1) = e(s_2) &\Rightarrow row(s_1) = row(s_2) && \text{definition of } row \\
&\Rightarrow \forall_{a \in A} \cdot row(s_1 a) = row(s_2 a) && \text{consistency assumption} \\
&\Rightarrow \Lambda(row)(s_1) = \Lambda(row)(s_2) && \text{definition of } \Lambda \\
&\Rightarrow (m_\Lambda \circ e_\Lambda)(s_1) = (m_\Lambda \circ e_\Lambda)(s_2) && \text{factorization of } \Lambda(row) \\
&\Rightarrow e_\Lambda(s_1) = e_\Lambda(s_2) && m_\Lambda \text{ is monic.}
\end{aligned}$$

The uniqueness of j follows directly from using the fact that e is epi.

Conversely suppose that there exists j such that $j \circ e = e_\Lambda$ and let $s_1, s_2 \in S$ be such that $row(s_1) = row(s_2)$. We need to show $row(s_1 a) = row(s_2 a)$, for all $a \in A$ or, equivalently, $\Lambda(row)(s_1) = \Lambda(row)(s_2)$.

$$\begin{array}{rcll} \Lambda(row)(s_1) & = & m_\Lambda(e_\Lambda(s_1)) & \text{factorization of } \Lambda(row) \\ & = & m_\Lambda(j(e(s_1))) & \text{assumption } e_\Lambda = j \circ e \\ & = & m_\Lambda(j(e(s_2))) & \text{assumption } row(s_1) = row(s_2) \\ & = & m_\Lambda(e_\Lambda(s_2)) & \text{assumption } e_\Lambda = j \circ e \\ & = & \Lambda(row)(s_2) & \text{factorization of } \Lambda(row). \end{array}$$ □

Closed and consistent observation tables are important in the algorithm of [4] because they can be translated into a deterministic automaton. We first describe the construction concretely and subsequently more abstractly using our categorical reformulation.

Definition 3 (Automaton associated to a closed and consistent observation table [4]). *Given a closed and consistent table (S, E) one can construct a deterministic automaton $M(S, E) = (Q, q_0, \delta, F)$ where*

- *Q is a finite set of states, $F \subseteq Q$ is a set of final states and $q_0 \in Q$ is the initial state;*
- *$\delta\colon Q \times A \to Q$ is the transition function.*

These Q, F and δ are given by:

$$\begin{array}{rclcrcl} Q & = & \{row(s) \mid s \in S\} & & q_0 & = & row(\lambda) \\ F & = & \{row(s) \mid s \in S, row(s)(\lambda) = 1\} & & \delta(row(s), a) & = & row(sa). \end{array}$$

To see that this is a well-defined automaton we need to check three facts: that the initial state is indeed an element of Q; that F is a well-defined subset, or equivalently, a well-defined function of type $Q \to 2$; and that δ is a well-defined function of type $Q \times A \to Q$.

For the first, note that since S is a non-empty prefix-closed language, it must contain λ, so q_0 is an element of Q.

For the second and third points, suppose s_1 and s_2 are elements of S such that $(\star)$ $row(s_1) = row(s_2)$. We must show that

$$\lambda \in E \quad \text{and} \quad row(s_1) \in F \iff row(s_2) \in F \tag{1}$$

$$\delta(row(s_1), a) = \delta(row(s_2), a) \in Q, \text{ for all } a \in A. \tag{2}$$

Since E is non-empty and suffix-closed, it must also contain λ. We also have :

$$row(s_1) \in F \iff row(s_1)(\lambda) = 1 \overset{(\star)}{\iff} row(s_2)(\lambda) = 1 \iff row(s_2) \in F.$$

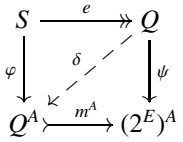
This concludes the proof of (1) above. Since the observation table is consistent, we have for each $a \in A$, that $(\star)$ implies $row(s_1 a) = row(s_2 a)$ and hence we can calculate

$$\delta(row(s_1), a) = row(s_1 a) = row(s_2 a) = \delta(row(s_2), a).$$

It remains to show that $row(s_1a) \in Q$. Since the table is closed, there exists an $s \in S$ such that $row(s) = row(s_1a)$. Hence, $row(s_1a) \in Q$ and (2) above holds.

In our categorical reformulation of the construction of the automaton Q we use that epis/surjections and monos/injections in the category **Sets** form a factorization system (see *e.g.* [7]). This allows us to use the diagonal-fill-in property in the next result.

Lemma 4. *The transition function δ of the automaton associated with a closed and consistent observation table can be obtained as the unique diagonal in the following diagram,*



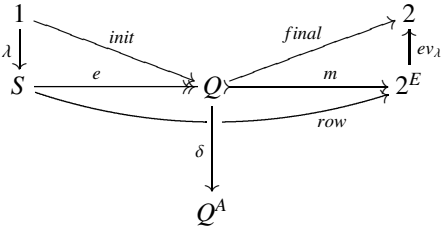
where $\begin{cases} \varphi &= \Lambda(i \circ e_A) \\ \psi &= m_\Lambda \circ j. \end{cases}$

Proof. The function δ obtained by diagonalization above satisfies:

$$\delta(e(s))(a) \;=\; \varphi(s)(a) \;=\; i(e_A(sa)).$$

This is the same as the above definition of δ, since $e(s)$ and $i(e_A(sa))$ represent, respectively, $row(s)$ and $row(sa)$. □

Definition 5 (Automaton associated with an observation table). *Let (S, E) be a closed and consistent observation table. The automaton $(Q, init, final, \delta)$ associated with the table is given in the following diagram.*



(3)

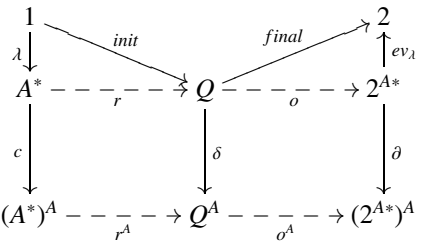
The initial state $init = row(\lambda)$ and the set of final states is given by evaluating the table in the λ column. These two functions exist because S and E are prefix- and suffix-closed, respectively. The transition function δ was defined in Lemma 4.

Next we give a categorical proof of the minimality result of [4].

Theorem 6. *The automaton associated with a closed and consistent observation table is minimal.*

Proof. An automaton is minimal if all states are reachable from the initial state and if no two different states recognize the same language (this property is referred to as observability).

Following a characterization that goes back to Kalman and then followed by other authors [16,6,9,8] these two properties can be nicely captured in the following diagram, where in the middle we have the automaton of Definition 5.



(4)

Let us define the unknown ingredients in the above diagram. On the left we have A^*, with a transition structure given by appending a letter to the end of the word:

$$c(u)(a) \;=\; ua.$$

The set A^*, together with the above transition structure, is the initial algebra of the functor $1 + A \times -$ on **Sets**. The map r exists and is unique by initiality; it sends $a_1 \cdots a_n$ to $\delta(\delta(\cdots\delta(\mathit{init})(a_1)\cdots)(a_{n-1}))(a_n)$.

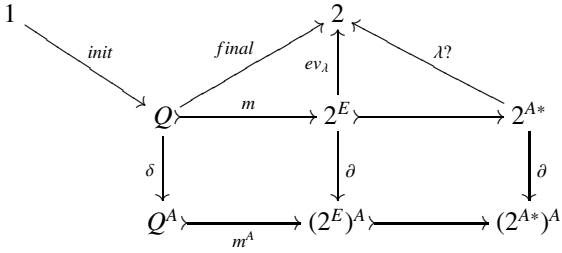
On the right we have 2^{A^*}, the set of languages over A, with a transition structure given by the Brzozowski/left derivative of a language:

$$\partial(L)(a) \;=\; a^{-1}L \;=\; \{u \mid au \in L\}.$$

The set 2^{A^*}, together with this transition function, is the final coalgebra of the functor $2 \times (-)^A$. The map of coalgebras $o\colon Q \to 2^{A^*}$ thus exists and is unique by finality. The map o assigns to every state the language it accepts.

Reachability and observability can now be rephrased in terms of properties of the functions r and o in (4): the automaton Q is *reachable* if $r\colon A^* \to Q$ is epic/surjective and it is *observable* if $o\colon Q \to 2^{A^*}$ is monic/injective.

To see that the automaton Q is minimal we extend the diagram above by including the auxiliary arrows of diagram (3). On the right we insert the mono $m\colon Q \rightarrowtail 2^E$ from Lemma 2 and complete the diagram:



(5)

Note that the transition function $\partial\colon 2^E \to (2^E)^A$ given by $\partial(L)(a) = a^{-1}L$ is well-defined because the subset $E \subseteq A^*$ is suffix-closed. Moreover, note that the inclusion $E \hookrightarrow A^*$

gives rise to an injection $2^E \rightarrowtail 2^{A^*}$, by taking images (here 2^- is the covariant powerset functor). This is a map of coalgebras $2^E \rightarrowtail 2^{A^*}$, making $(2^E, \partial)$ a subcoalgebra of the final coalgebra. This transition map ∂ in (5) satisfies:

$$\partial \circ row \;=\; \Lambda(row) \;:\; S \longrightarrow (2^E)^A. \tag{6}$$

where $\Lambda(row)$ is as in Lemma 2. The proof is easy:

$$\begin{aligned}(\partial \circ row)(s)(a)(v) \;=\; \partial(row(s))(a)(v) \;=\; a^{-1}(row(s))(v) \;&=\; row(s)(av)\\ &=\; \mathcal{L}(sav)\\ &=\; row(sa)(v)\\ &=\; \Lambda(row)(s)(a)(v).\end{aligned}$$

One can now see that the rectangle on the left in (5) commutes by precomposing its two maps $Q \rightrightarrows (2^E)^A$ with the epi $e\colon S \twoheadrightarrow Q$ from Lemma 2:

$$(\partial \circ m) \circ e \;=\; \partial \circ row \overset{(6)}{=} \Lambda(row) \;=\; m_\Lambda \circ e_\Lambda \;=\; m_\Lambda \circ j \circ e \;=\; (m^A \circ \delta) \circ e,$$

where the last equation uses the definition of δ from Lemma 4. Therefore, the unique map $Q \rightarrowtail 2^{A^*}$ to the final coalgebra is a mono, being a composite of two monos in (5), and we can conclude that the automaton Q is observable.

It remains to show that the automaton Q is reachable. This means that we must show that the map $r\colon A^* \to Q$ in diagram (4) is surjective/epic. We are done if we can show that $r \circ n = e\colon S \twoheadrightarrow Q$, where we write n for the inclusion map $S \hookrightarrow A^*$.

We prove this equation $r \circ n = e$ via the mono $m\colon Q \rightarrowtail 2^E$, and show that $m \circ r \circ n = m \circ e = row\colon S \to 2^E$. We do so by induction on the length of strings $u \in S$. Thus:

$$\begin{aligned}(m \circ r \circ n)(\lambda) \;&=\; m(r(\lambda))\\ &=\; m(init)\\ &=\; m(e(\lambda))\\ &=\; row(\lambda)\\ (m \circ r \circ n)(ua) \;&=\; (m \circ r)(ua)\\ &=\; (m \circ r)(c(u)(a))\\ &=\; ((m \circ r)^A)(c(u))(a)\\ &=\; ((m \circ r)^A \circ c)(u)(a)\\ &\overset{(4)}{=}\; (m^A \circ \delta \circ r)(u)(a)\\ &\overset{(5)}{=}\; (\partial \circ m \circ r)(u)(a)\\ &=\; (\partial \circ m \circ r \circ n)(u)(a) \qquad \text{since } S \text{ is prefix-closed}\\ &\overset{(IH)}{=}\; (\partial \circ row)(u)(a)\\ &\overset{(6)}{=}\; \Lambda(row)(u)(a)\\ &=\; row(ua).\end{aligned}$$

□

2.2 The Learning Algorithm

We present the algorithm of [4] in Figure 1. In the algorithm, there is a teacher which has the capacity of answering two types of questions: yes/no to the query on whether a word belongs to the master language and yes/no to the question whether a certain guess of the automaton accepting the master language is correct. In the case of a negative answer of the latter question, the teacher also provides a counter-example. The learner builds an observation table by asking the teacher queries of membership of words of increasing length. Once the table is closed and consistent, the learner tries to guess the master language. We explain every step by means of an example, over the alphabet $A = \{a, b\}$.

Input: Minimally Adequate Teacher of the master language $\mathcal{L}$.
Output: Minimal automaton accepting $\mathcal{L}$.

```
1:  function LEARNER
2:      S ← {λ} ; E ← {λ}.
3:      repeat
4:          while (S, E) is not closed or not consistent do
5:              if (S, E) is not consistent then
6:                  find s1, s2 ∈ S, a ∈ A, and e ∈ E such that
7:                      row(s1) = row(s2) and L(s1 a e) ≠ L(s2 a e)
8:                  E ← E ∪ {ae}.
9:              end if
10:             if (S, E) is not closed then
11:                 find s1 ∈ S, a ∈ A such that
12:                     row(s1 a) ≠ row(s), for all s ∈ S
13:                 S ← S ∪ {s1 a}.
14:             end if
15:         end while
16:         Make the conjecture M(S, E).
17:         if the Teacher replies no to the conjecture, with a counter-example t then
18:             S ← S ∪ ↓t.
19:         end if
20:     until the Teacher replies yes to the conjecture M(S, E).
21:     return M(S, E).
22: end function
```

Fig. 1. Angluin's algorithm for deterministic finite automata [4]

Imagine the Learner receives as input a Teacher for the master language

$$\mathcal{L} = \{u \in \{a, b\}^* \mid \text{the number of } a\text{'s in } u \text{ is divisible by } 3\}.$$

In the first step of the while loop it builds a table for $S = \{\lambda\}$ and $E = \{\lambda\}$.

Step 1

		λ
S	λ	1
$S \cdot A$	a	0
	b	1

(S, E) consistent? ✓
(S, E) closed? No, $row(a) = (\lambda \mapsto 0) \neq (\lambda \mapsto 1) = row(\lambda)$.
Then, $S \leftarrow S \cup \{a\}$ and we go to **Step 2**.

We extend the row index set S so we get a new observation table and we again check for closedness and consistency.

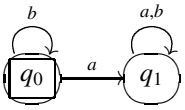
Step 2

	λ
λ	1
a	0
b	1
aa	0
ab	0

(S, E) consistent? ✓

(S, E) closed? ✓

Then, we guess the automaton:



where $q_0 = row(\lambda) = (\lambda \mapsto 0)$

$q_1 = row(a) = (\lambda \mapsto 1)$

Teacher replies with counter-example aaa.

$S \leftarrow S \cup \{a, aa, aaa\}$ and we go to **Step 3**.

In the second step we managed to build a closed and consistent table which enabled us to make a first guess on the automaton. The guess was wrong so the teacher provided a counter-example, which we use to extend the row index set, generating a larger table.

Step 3

	λ
λ	1
a	0
aa	0
aaa	1
b	1
ab	0
aab	0
$aaaa$	0
$aaab$	1

(S, E) consistent?

No, $row(a) = row(aa)$ but $row(aa) \neq row(aaa)$.

Then $E \leftarrow E \cup \{a\}$ and we go to (**Step 3.1**).

In the third step the test of consistency failed for the first time and hence we extend the column index set E from $\{\lambda\}$ to $\{\lambda, a\}$. This extension allows to distinguish states (that is, rows of the table) that were indistinguishable in the previous step though they could be differentiated after an a step.

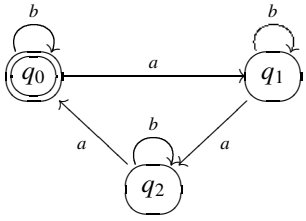
Step 3.1

	λ	a
λ	1	0
a	0	0
aa	0	1
aaa	1	0
b	1	0
ab	0	0
aab	0	1
$aaaa$	0	0
$aaab$	1	0

(S, E) consistent? ✓

(S, E) closed? ✓

We make another guess:



The teacher replies **yes**.

In the last step, we again constructed a closed and consistent table, which allowed us to make another guess of the automaton accepting the master language. This second guess yielded the expected automaton.

3 Angluin's Algorithm for Moore and Mealy Machines

Moore automata generalize deterministic automata by replacing the subset of final states by a function $o\colon Q \to B$, where B is a set of outputs. Mealy automata are another variation where instead of having the outputs associated to the states, each transition has an associated input and output letter. In a nutshell, here are the three types of automata we have encountered so far.

$o\colon Q \to 2$	$o\colon Q \to B$	
$\delta\colon Q \to Q^A$	$\delta\colon Q \to Q^A$	$\delta\colon Q \to (Q \times B)^A$
Deterministic automata	Moore automata	Mealy automata

Note that using the isomorphism $(Q \times B)^A \cong Q^A \times B^A$ we can also reduce Mealy to Moore automata (with a higher order output set).

Definition 1 of closedness and consistency does not depend at all on the fact that the output set is 2. More interestingly also the categorical proof of minimality (Theorem 6) is not specific for deterministic automata but can be carried over to an arbitrary Moore automaton. Hence, we can straightforwardly use all the categorical definitions and results above for an arbitrary output set B. This allow us to define what a closed and consistent table for a Moore/Mealy automaton is and derive the algorithm that infers the automaton recognizing the behavior of a Moore/Mealy automaton.

The master language $\mathcal{L}\colon A^* \to 2$ is now replaced by a weighted language (or formal power series) $\mathcal{L}\colon A^* \to B$. Here, we use our abstract view on automata. Every automaton has a canonical semantics and universe of behavior associated with it: the final coalgebra of the functor associated with the transition structure. In the case of deterministic automata the final coalgebra is precisely the set of formal languages 2^{A^*} and

in the case of Moore automata it is the set of weighted languages B^{A^*}. Note that for Mealy machines where the output set is B^A we then get as semantics B^{A^+}, where A^+ denotes the set of non-empty words over A.

Apart from the change in the type of the master language, the rest of the algorithm in Figure 1 is precisely the same and also the proof of minimality carries over since we have phrased it in the general setting using the final coalgebra.

Let us now illustrate the algorithm for Mealy machines using as example the language $L\colon A^+ \to B$, with $A = \{a, b\}$ and $B = \mathbb{N}$, given by, for $w \in A^+$,

$$L(w) = \begin{cases} |w|_a \mod 3 & \text{if } w = ua \\ 3 & \text{if } w = aab(bb)^n, n \in \mathbb{N} \\ 0 & \text{otherwise} \end{cases}$$

where $u \in A^*$ and $|u|_a$ denotes the number of a's in the word u.

For notational convenience we will use the following equivalent representation of *row*

$$\frac{\mathit{row}\colon S \to (B^A)^E}{\mathit{row}\colon S \to B^{E\cdot A}}$$

This last representation, $\mathit{row}\colon S \to B^{E\cdot A}$, starting with $E \cdot A = \{\lambda\} \cdot A = A$, is precisely what can be found in the existing algorithms for Mealy machines [21,1,2].

In the first step of the algorithm we build a table for $S = \{\lambda\}$ and $E \cdot A = A$.

Step 1

		a	b
S	λ	1	0
$S \cdot A$	a	2	0
	b	1	0

(S, E) consistent? ✓

(S, E) closed? No, $\mathit{row}(a) \neq \mathit{row}(\lambda)$.

Then, $S \leftarrow S \cup \{a\}$ and we go to **Step 2**.

We extend the row index set S so we get a new observation table and we again check for closedness and consistency.

Step 2

	a	b
λ	1	0
a	2	0
b	1	0
aa	0	0
ab	2	0

(S, E) consistent? ✓

(S, E) closed? No, $\mathit{row}(aa) \neq \mathit{row}(\lambda)$ and $\mathit{row}(aa) \neq \mathit{row}(a)$.

Then, $S \leftarrow S \cup \{aa\}$ and we go to **Step 3**.

We again extend the row index set S we check for closedness and consistency of the new table.

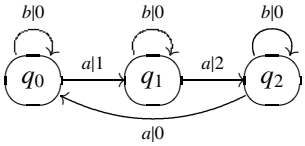
Step 3

	a	b
λ	1	0
a	2	0
aa	0	0
b	1	0
ab	2	0
aaa	1	0
aab	0	0

(S, E) consistent? ✓

(S, E) closed? ✓

Then, we guess the Mealy automaton $\mathcal{A}$:



Teacher replies with counter-example aab.

$S \leftarrow S \cup \{a, aa, aab\}$ and we go to **Step 4**.

We process the teacher's counter-example and analyze the resulting table. Note that the above example is of minimal length, but in fact this is not guaranteed: the teacher can reply with an arbitrary counter-example. Shorter counter-examples do not necessarily imply less steps in the algorithm. For instance, the teacher could have replied with $aabb$ and, in fact, this would cause the algorithm to terminate in one less step than it will now.

Step 4

	a	b
λ	1	0
a	2	0
aa	0	0
aab	0	0
b	1	0
ab	2	0
aaa	1	0
$aaba$	1	0
$aabb$	0	3

(S, E) consistent?

No, $row(aa) = row(aab)$ and $row(aab) \neq row(aabb)$.

Then $E \leftarrow E \cup \{b\}$ and we go to (**Step 4.1**).

In the fourth step the consistency check failed and therefore $E \cdot A$, the column index, gets extended from $\{\lambda\} \cdot A$ to $\{\lambda, b\} \cdot A$.

Step 4.1

	a	b	ba	bb
λ	1	0	1	0
a	2	0	2	0
aa	0	0	0	0
aab	0	0	0	3
b	1	0	1	0
ab	2	0	2	0
aaa	1	0	1	0
$aaba$	1	0	1	0
$aabb$	0	3	0	0

(S, E) consistent? ✓

(S, E) closed? No, $row(aabb) \neq row(s)$, for all $s \in S$.

Then, $S \leftarrow S \cup \{aabb\}$ and we go to **Step 5**.

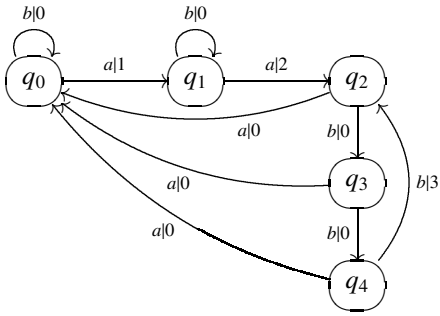
Step 5

	a	b	ba	bb
λ	1	0	1	0
a	2	0	2	0
aa	0	0	0	0
aab	0	0	0	3
$aabb$	0	3	0	0
b	1	0	1	0
ab	2	0	2	0
aaa	1	0	1	0
$aaba$	1	0	1	0
$aabba$	1	0	1	0
$aabbb$	0	0	0	0

(S, E) consistent? ✓

(S, E) closed? ✓

Then, we guess the Mealy automaton $\mathcal{A}$:



Teacher replies **yes**.

4 Further Generalizations: Linear Weighted Automata

In Section 3, we showed that the categorical perspective on Angluin's algorithm delivers without any extra effort an algorithm for Moore and Mealy automata. That generalization was obtained by observing that changes in the output set did not have any influence on the construction of the observation table. We will now consider yet another example in which we consider as output set $\mathbb{F}$, a field, and we construct an automaton where the state space is now a finite dimensional vector space over the field $\mathbb{F}$. These automata are known as linear weighted automata over the field $\mathbb{F}$ [12]. These examples illustrate two

possible avenues of generalization: in the Mealy/Moore case we changed the functor type of the automaton whereas here we change also the underlying category (in other words the state space of the automaton) and consider automata in the category **Vect** of vector spaces and linear maps.

Formally, a linear weighted automaton over a finite alphabet A and with outputs in a field $\mathbb{F}$ is a quadruple (V, v_0, δ, ϕ) where

- V is a finite dimensional vector space over $\mathbb{F}$ and $v_0 \in V$ is the initial vector.
- $\delta\colon V \to V^A$ is a linear map determining the transition structure.
- $\phi\colon V \to \mathbb{F}$ is a linear map assigning outputs in $\mathbb{F}$ to states.

That is, a linear weighted automaton is just a Moore automaton in the category of vector spaces and linear maps.

Linear weighted automata recognize weighted languages $\mathbb{F}^{A^*}$ in the following way. A word $w = a_1 \cdots a_n \in A^*$ is assigned the weight $r \in \mathbb{F}$ if and only if

$$\phi(\delta(\delta(\cdots\delta(v_0)(a_1)\cdots)(a_{n-1}))(a_n)) = r.$$

More formally, $\mathbb{F}^{A^*}$ together with a transition structure given by

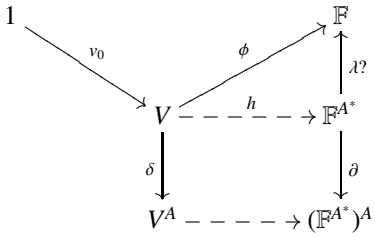
$$\lambda?(L) = L(\lambda) \qquad \partial(L)(a)(u) = L(au)$$

is the final coalgebra, in the category of vector spaces and linear maps, of the functor $\mathbb{F} \times (-)^A$. Note that $\mathbb{F}^{A^*}$ has a vector space structure given by pointwise sum and scalar multiplication. More precisely, given $r_1, \ldots, r_k \in \mathbb{F}$ and $L_1, \cdots, L_k \in \mathbb{F}^{A^*}$, we define:

$$(r_1 L_1 + \cdots + r_k L_k)(w) = r_1(L_1(w)) + \cdots + r_k(L_k(w))$$

where on the right the sum and scalar multiplication are the ones in $\mathbb{F}$. The functions $\lambda?$ and ∂ are also linear w.r.t. the vector space operations defined above.

Lemma 7. *Given a linear weighted automaton (V, v_0, δ, ϕ) there exists a unique linear homomorphism such that the following diagram commutes.*



Proof. For any $v \in V$, we define $h(v)$ by induction on $w \in A^*$.

$$h(v)(\lambda) = \phi(v) \qquad h(v)(aw) = h(\delta(v)(a))(w)$$

It is easy to see that h makes the above diagram commute. It remains to show uniqueness and linearity of h. Linearity follows by induction and using the linearity of ϕ and δ.

$$\begin{aligned}
h(r_1v_1 + \cdots + r_kv_k)(\lambda) &= \phi(r_1v_1 + \cdots + r_kv_k) \\
&= r_1\phi(v_1) + \cdots + r_k\phi(v_k) && \text{linearity of } \phi \\
&= r_1(h(v_1)(\lambda)) + \cdots + r_k(h(v_k)(\lambda)) \\
h(r_1v_1 + \cdots + r_kv_k)(aw) &= h(\delta(r_1v_1 + \cdots + r_kv_k)(a))(w) \\
&= h(r_1\delta(v_1)(a) + \cdots + r_k\delta(v_k)(a))(w) && \text{linearity of } \delta \\
&= r_1(h(\delta(v_1)(a))(w)) + \cdots + r_k(h(\delta(v_k)(a))(w)) && \text{induction hyp.} \\
&= r_1(h(v_1)(aw)) + \cdots + r_k(h(v_k)(aw))
\end{aligned}$$

For uniqueness, suppose there is a map g such that $\lambda? \circ g = \phi$ and $\partial \circ g = g^A \circ \delta$. Then, for any $v \in V$,

$$g(v)(\lambda) = \lambda?(g(v)) = \phi(v) = h(v)(\lambda)$$

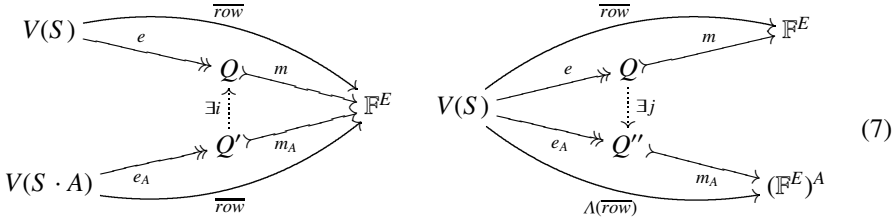
and, for any $a \in A$,

$$g(v)(aw) = (\partial(g(v))(a))(w) = g(\delta(v)(a))(w) \overset{\text{(IH)}}{=} h(\delta(v)(a))(w) = h(w)(aw). \quad \square$$

The master language is now an element of the final coalgebra $\mathbb{F}^{A^*}$. Our categorical definitions and results – closedness, consistency and minimality – are valid also for linear weighted automata. In the definitions of closedness and consistency we use the usual epi-mono factorization of linear maps. Using the matrix representation of linear maps, epimorphisms (resp. monomorphisms) will correspond to matrices of full column (resp. row) rank.

For convenience, in the algorithm, we still want to use the sets S and E and we will represent the table as $S \to \mathbb{F}^E$. This is possible since all vector spaces are freely generated and a linear map is determined by its value on the basis vectors. First, we just recall the definitions of closedness and consistency and we instantiate them concretely. We need some notation. Let $V(S)$ denote the free vector space generated by S (if S is finite $V(S) = \mathbb{F}^S$), whose elements we will frequently denote by finite formal sums $\sum_I r_i s_i$.

An observation table is now a linear map $\overline{row}\colon V(S) \to \mathbb{F}^E$. An observation table (S, E) is closed (resp. consistent) if and only if there exist necessarily unique linear maps i and j such that the diagram on the left (resp. right) commutes.



(7)

closed (in **Vect**) consistent (in **Vect**)

The existence of the maps i and j induce a (linear) transition structure $\delta\colon Q \to Q^A$ as defined in Lemma 4.

More concretely, in the linear setting, a table is closed if for all $t \in S \cdot A$, there exist $s_i \in S$ such that

$$\overline{row}(t) = \sum_I r_i \times \overline{row}(s_i).$$

A table is consistent if whenever

$$\sum_I r_i \times \overline{row}(s_i) = \sum_J r_j \times \overline{row}(t_j),$$

for $s_i, t_j \in S$, then, for all $a \in A$,

$$\sum_I r_i \times \overline{row}(s_i a) = \sum_J r_j \times \overline{row}(t_j a).$$

The map $\overline{row}$ has some special properties, related with the fact that $V(-)$ is actually a monad. We defined V above on sets but, V can also be defined on maps $f\colon U \to T$ as $V(f)\colon V(U) \to V(T)$, where

$$V(f)\left(\sum_I r_i u_i\right) = \sum_I r_i f(u_i).$$

This $V(-)$ is a functor and, of interest to us, a monad where the unit $\eta\colon S \to V(S)$ is given by the trivial unit linear combination. Given a map $f\colon S \to V$, where V is a vector space, let $\overline{f}\colon V(S) \to V$ denote its linearization given by

$$\overline{f}\left(\sum_I r_i s_i\right) = \sum_I r_i f(s_i).$$

The linearization of f is the unique linear map satisfying $\overline{f} \circ \eta = f$. We will use this uniqueness property below in the proofs.

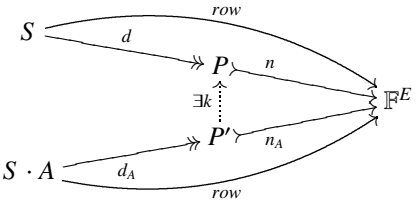
The map $\overline{row}$ in the diagrams above is the linearization of the map $row\colon S \to \mathbb{F}^E$ that only determines the values of the elements of S, which act as a basis for the vector space $V(S)$.

It might be interesting to observe that closedness in sets implies closedness in the linear setting, but consistency has a dual property: consistency in the linear setting implies consistency in sets. This is interesting for the algorithm: if closedness is already true for the table indexed by the basis vectors – $row\colon S \to \mathbb{F}^E$ - then it is also true for the linear view on the same table – $\overline{row}\colon V(S) \to \mathbb{F}^E$. On the other hand if $row\colon S \to \mathbb{F}^E$ is not consistent then $\overline{row}\colon V(S) \to \mathbb{F}^E$ is also not consistent. These correspondences are both interesting from an algorithmic point of view: note that the definition of consistency in the linear setting involves comparison of arbitrary linear combinations of rows, which is a rather expensive operation.

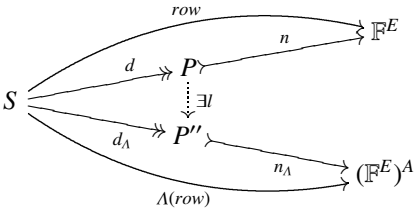
Lemma 8. *Let $row\colon S \to \mathbb{F}^E$ be an observation table and $\overline{row}\colon V(S) \to \mathbb{F}^E$ its linearization.*

① *If row is closed then so is* $\overline{row}$.
② *If* $\overline{row}$ *is consistent then so is row.*

Proof. Recall the definitions of $row\colon S \to \mathbb{F}^E$ being closed and consistent.

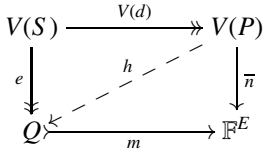


closed (in **Sets**)



consistent (in **Sets**)

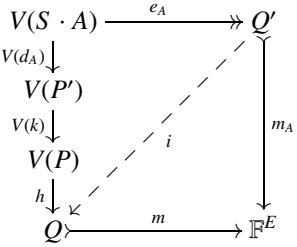
First, observe that we can always define $h\colon V(P) \to Q$, with Q as given in (7), as follows:



The commutativity of the above diagram follows by observing that

$$m \circ e \circ \eta = row \quad \text{and} \quad \overline{n} \circ V(d) \circ \eta = \overline{n} \circ \eta \circ d = n \circ d = row.$$

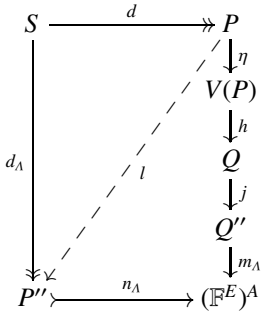
For ①, suppose *row* is closed and let us prove that $\overline{row}$ is also closed, according to (7). We define the map i as follows.



Remains to show the commutativity of the above diagram, which follows by observing that

$$m_A \circ e_A \circ \eta = \overline{row} \circ \eta = row \quad \text{and}$$
$$m \circ h \circ V(k) \circ V(d_A) \circ \eta = \overline{n} \circ \eta \circ k \circ d_A = n \circ k \circ d_A = n_A \circ d_A = row.$$

For ②, suppose $\overline{row}$ is consistent and let us now prove that row is also consistent. We define l as follows.



The commutativity of the above diagram follows by a simple calculation, using naturality and properties of j as given above.

$$\begin{aligned}
m_\Lambda \circ j \circ h \circ \eta \circ d &= m_\Lambda \circ j \circ h \circ V(d) \circ \eta && \text{naturality of } \eta \\
&= m_\Lambda \circ j \circ e \circ \eta && h \circ V(d) = e \\
&= m_\Lambda \circ e_\Lambda \circ \eta && j \circ e = e_\Lambda \\
&= \Lambda(\overline{row}) \circ \eta && m_\Lambda \circ e_\Lambda = \Lambda(\overline{row}) \\
&= \Lambda(row) \\
&= n_\Lambda \circ d_\Lambda && \text{factorization of } \Lambda(row)
\end{aligned}$$ □

The converses of ① and ② in the above lemma fail, as we point out in the example below.

We will illustrate the algorithm in the linear setting with the following example over $\mathbb{F} = \mathbb{R}$.

$$\mathcal{L}(u) = \begin{cases} 1 & \text{if } u = \lambda \\ 2|u|_b & \text{if } |u|_a \text{ is even} \\ 0 & \text{otherwise} \end{cases}$$

In the first step we build a table for $S = \{\lambda\}$ and $E = \{\lambda\}$.

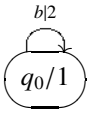
Step 1

	λ
λ	1
a	0
b	2

(S, E) consistent? ✓

(S, E) closed? ✓

Then, we guess the automaton:



where $q_0 = row(\lambda)$.

Teacher replies with counter-example aa.

$S \leftarrow S \cup \{a, aa\}$ and we go to **Step 2**.

Note that the above table would not be closed in **Sets** because $row(a) \neq row(\lambda)$. However, $row(a) = 0 \times row(\lambda)$ and hence the table is closed in the linear setting. Also note that we use the following conventions in the representation of the automaton: q/r denotes $\phi(q) = row(q)(\lambda) = r$ (the output of the state) and $q \xrightarrow{a|r} q'$ denotes $\delta(q)(a) = r \times q'$.

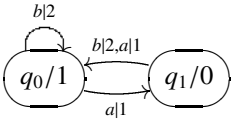
Step 2

	λ
λ	1
a	0
aa	1
b	2
ab	0
aaa	0
aab	2

(S, E) consistent? ✓

(S, E) closed? ✓

Then, we guess the automaton:



where $q_0 = row(\lambda)$, $q_1 = row(a)$

Teacher replies with counter-example ab.

$S \leftarrow S \cup \{a, ab\}$ and we go to **Step 3**.

Step 3

	λ
λ	1
a	0
aa	1
ab	0
b	2
ab	0
aaa	0
aab	2
aba	2
abb	0

(S, E) consistent?

No, $row(a) = row(ab)$ and $row(aa) \neq row(aba)$.

Then $E \leftarrow E \cup \{a\}$ and we go to (**Step 3.1**).

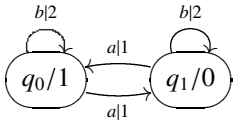
Step 3.1

	λ	a
λ	1	0
a	0	1
aa	1	0
ab	0	2
b	2	0
ab	0	2
aaa	0	1
aab	2	0
aba	2	0
abb	0	4

(S, E) consistent? ✓

(S, E) closed? ✓

Then, we guess the automaton:



where $q_0 = row(\lambda)$

$q_1 = row(a)$

Teacher replies **yes**.

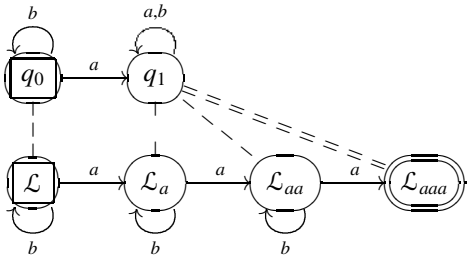
5 Discussion

We have presented the first steps towards a categorical understanding and generalization of Angluin's learning algorithm, originally defined for deterministic finite automata. The categorical reformulation enables us to explore two avenues of generalization: varying the functor (giving for instance different input/output for the automaton) and varying the category under study (changing for instance the type of computations involved). The variations we concretely considered in this paper were rather mild but interestingly enough yielded algorithms for Mealy/Moore automata and linear weighted automata.

The possibilities of further generalizations are vast. We would like to provide a categorical proof of termination of the algorithm. This requires a categorical understanding on how the algorithm determines which rows/columns need to be added in order for the observation table to be closed/consistent. Once this is understood, we expect that further variations on the functor can also be considered. We conjecture that there is a deep connection with the construction of the initial and final sequence of two functors. On the one hand, the functor whose initial algebra determines the *experiments/queries* needed to build the observation table. In the case we studied here the functor in question is $1 + A \times -$. On the other hand, the functor whose final coalgebra determines the notion of behavior. In all our examples, this was the functor $B \times (-)^A$. We expect that the connection between the two functors can be explained by duality, as Prakash also hinted to us (personal communication).

The production of counter-examples can also be a good place for exploring enhancements of the algorithm. In this subject, recent work on bisimulations will be of use, as we explain next. In order to return a counter-example, the teacher essentially tries to build a bisimulation relation between the guessed automaton $M(S, E)$ and the actual minimal automaton recognizing the master language. The latter is $\langle \mathcal{L} \rangle$ the subcoalgebra of the final coalgebra $(2^{A^*}, \langle \lambda?, \partial \rangle)$ generated by $\mathcal{L}$. Let us illustrate this with the first counter-example generated on page 393 in the example for deterministic automata. The

procedure of constructing a bisimulation containing the initial state of the guessed automaton and the master language is depicted below. The pairs added to the bisimulation are connected by a dashed line.



The double dashed line, in red, shows the first contradiction in the bisimulation construction: q_1 has to be related to $\mathcal{L}_{aaa}$ but they differ in their output. Hence the path leading to these implies that *aaa* is the counter-example returned.

Finding counter-examples can be optimized by using enhancements of the bisimulation method. In the case of deterministic automata, this was first observed in the 70's by Hopcroft, Karp, and Tarjan [14,3,15] and has since then been improved (see e.g [22,11,18]) and explored in other contexts, notably in concurrency theory [20,17,10].

References

1. Aarts, F.: Inference and abstraction of communication protocols. Master's thesis, Radboud University Nijmegen and Uppsala University (2009)
2. Aarts, F., de Ruiter, J., Poll, E.: Formal models of bank cards for free. In: ICST Workshops, pp. 461–468. IEEE (2013)
3. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The Design and Analysis of Computer Algorithms. Addison-Wesley (1974)
4. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. 75(2), 87–106 (1987)
5. Angluin, D., Csürös, M.: Learning Markov chains with variable memory length from noisy output. In: Freund, Y., Schapire, R.E. (eds.) COLT, pp. 298–308. ACM (1997)
6. Arbib, M.A., Manes, E.G.: Adjoint machines, state-behavior machines, and duality. Journal of Pure and Applied Algebra 6(3), 313–344 (1975)
7. Barr, M., Wells, C.: Toposes, Triples and Theories. Springer, Berlin (1985) Revized and corrected version available from URL, `www.cwru.edu/artsci/math/wells/pub/ttt.html`
8. Bonchi, F., Bonsangue, M., Hansen, H., Panangaden, P., Rutten, J., Silva, A.: Algebra-coalgebra duality in Brzozowski's minimization algorithm. In: ACM Transactions on Computational Logic (TOCL) (2014)
9. Bonchi, F., Bonsangue, M.M., Rutten, J.J.M.M., Silva, A.: Brzozowski's Algorithm (Co)Algebraically. In: Constable, R.L., Silva, A. (eds.) Kozen Festschrift. LNCS, vol. 7230, pp. 12–23. Springer, Heidelberg (2012)
10. Bonchi, F., Caltais, G., Pous, D., Silva, A.: Brzozowski's and up-to algorithms for must testing. In: Shan, C.-C. (ed.) APLAS 2013. LNCS, vol. 8301, pp. 1–16. Springer, Heidelberg (2013)

11. Bonchi, F., Pous, D.: Checking nfa equivalence with bisimulations up to congruence. In: Giacobazzi, R., Cousot, R. (eds.) POPL, pp. 457–468. ACM (2013)
12. Boreale, M.: Weighted bisimulation in linear algebraic form. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 163–177. Springer, Heidelberg (2009)
13. Eilenberg, S.: Automata, languages, and machines. Pure and Applied Mathematics. Elsevier Science (1974)
14. Hopcroft, J.E., Karp, R.M.: A linear algorithm for testing equivalence of finite automata. Technical report, Cornell University (1979)
15. Hopcroft, J.E.: An n log n algorithm for minimizing states in a finite automaton. Technical report, Stanford University, Stanford, CA, USA (1971)
16. Kalman, R.: On the general theory of control systems. IRE Transactions on Automatic Control 4(3), 110–110 (1959)
17. Pous, D., Sangiorgi, D.: Enhancements of the coinductive proof method. In: Sangiorgi, D., Rutten, J. (eds.) Advanced Topics in Bisimulation and Coinduction. Cambridge Tracts in Theoretical Computer Science, vol. 52, Cambridge University Press (November 2011)
18. Rot, J., Bonsangue, M., Rutten, J.M.M.: Coalgebraic bisimulation-up-to. In: van Emde Boas, P., Groen, F.C.A., Italiano, G.F., Nawrocki, J., Sack, H. (eds.) SOFSEM 2013. LNCS, vol. 7741, pp. 369–381. Springer, Heidelberg (2013)
19. Rutten, J.J.M.M.: Automata and coinduction (an exercise in coalgebra). In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 194–218. Springer, Heidelberg (1998)
20. Sangiorgi, D.: Beyond Bisimulation: The "up-to" Techniques. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2005. LNCS, vol. 4111, pp. 161–171. Springer, Heidelberg (2006)
21. Shahbaz, M., Groz, R.: Inferring Mealy Machines. In: Cavalcanti, A., Dams, D.R. (eds.) FM 2009. LNCS, vol. 5850, pp. 207–222. Springer, Heidelberg (2009)
22. De Wulf, M., Doyen, L., Henzinger, T.A., Raskin, J.-F.: Antichains: A new algorithm for checking universality of finite automata. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 17–30. Springer, Heidelberg (2006)