

Learning to Coordinate

Coördineren kun je leren — dedicated to Farhad Arbab on the occasion of his retirement

Gerco van Heerdt¹, Bart Jacobs², Tobias Kappé¹, and Alexandra Silva¹

¹ Department of Computer Science, University College London

² Institute for Computing and Information Sciences, Radboud University Nijmegen

Abstract. Reo is a visual language of connectors that originated in component-based software engineering. It is a flexible and intuitive language, yet powerful and capable of expressing complex patterns of composition. The intricacies of the language resulted in many semantic models proposed for Reo, including several automata-based ones.

In this paper, we show how to generalize a known active automata learning algorithm — Angluin’s L^* — to Reo automata. We use recent categorical insights on Angluin’s original algorithm to devise this generalization, which turns out to require a change of base category.

1 Introduction

In the last two decades, with the widespread use and development of software, there has been a focus on promoting reusability of software code. Component-based software engineering and service-oriented computing are two examples of paradigms that were developed around this idea. Many languages appeared to enable flexible and expressive ways to compose software components. One of those languages is Reo — a language offering a visual approach, where *connectors* are used to compose components into a system. The language is modular, offering ways to *compositionally* build more complex connectors from basic ones, which makes it possible to capture intricate patterns of interaction such as input synchronization, mutual-exclusion, or state-dependent behavior.

Reo serves as a prime example of a language in which interaction is treated as a first-class concept that allows direct specification and manipulation of protocols. The treatment of interaction as a central concept and the development of rigorous mathematical tools and techniques for its study has occupied most of Farhad’s career. In this paper we make a modest contribution to Farhad’s toolkit — a generalization of Angluin’s algorithm to Reo automata, one of many semantic models developed for Reo.

Automata are used in modeling and verification of systems and protocols in Computer Science. 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. Unfortunately, models are not always available and rapid changes in the system require frequent adaptations. This has led to the development of *automata learning* algorithms, which enable inferring or learning a model from a given system just by observing its behavior or response to certain queries. One of the first algorithms was proposed by Dana Angluin [2] and though it only worked for deterministic finite automata it had an

interesting range of applications, such as in verification of software systems and security protocols (a recent survey can be found here [11]).

Category theory provides an abstract framework to study structures in mathematics and computer science. In this paper, we 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 and in other categories.

Section 2 is a refinement of a section we included in a previous paper [7]. Compared to [7], we added fully categorical characterizations of the data structures and *hypothesis automata* involved in Angluin’s algorithm. Using these ingredients, we made our proof of minimality of these automata completely abstract, which also fixes a gap present in the old proof. The application to Reo automata is entirely new and illustrates an important feature of the framework — the ability to have a learning algorithm in a different category. Angluin’s algorithm for Reo automata operates in the category **Posets**. This might not be surprising for Farhad and those familiar with the semantics of Reo and the idiosyncrasies of the semantics of interaction and concurrency — the order of actions (and signal flows) is an important part of correctly capturing the behavior of a Reo connector. However, this pleasantly surprised the authors, as it provided a simple yet non-trivial, outside the category **Sets**, application of the categorical understanding and generalization of Angluin’s algorithm. In order to obtain the algorithm we had to understand Reo automata as coalgebras for a functor. In this process, we show that they are essentially automata in the category **Posets**. We will define a poset of interactions to be used as the alphabet for the (categorical) Reo automaton — this highlights the algebraic structure of the actions (signal flows) in Reo and exposes the fact that interaction is a first-class concept.

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 change the base category in which the automata are considered from **Sets** to **Posets**, obtaining in this manner an algorithm for Reo automata [6].

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.

Let us first introduce some notation and basic definitions. Let A be a finite set of symbols, often called an alphabet, and A^* the set of finite words over A . We 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 often switch between the representation of a language as a set 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 left and right derivative by setting

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

A language L is *prefix-closed* (resp. *suffix-closed*) if $La^{-1} \subseteq L$ (resp. $La^{-1} \subseteq L$) for all $a \in A$. We use $\downarrow u$ (resp. $\uparrow u$) to denote the set of prefixes (resp. suffixes) of a word $u \in A^*$.

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

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

Angluin's algorithm incrementally constructs an observation table with Boolean entries. Rows are labeled by words in $S \cup S \cdot A$, where S is a finite *prefix-closed* language, and columns by a finite *suffix-closed* language E . Both S and E contain the empty word λ .

For arbitrary $U, V \subseteq A^*$, define $row: U \rightarrow 2^V$ by $row(u)(v) = \mathcal{L}(uv)$. Since row is fully determined by \mathcal{L} , we will from now on refer to an observation table as a pair (S, E) , leaving \mathcal{L} implicit. Formally, an observation table is a triple (S, E, row) , where $row: (S \cup S \cdot A) \rightarrow 2^E$. Note that \cup here is used for language union and not coproduct, but it will be convenient for us to split the function into $row: S \rightarrow 2^E$ and $row_A: S \cdot A \rightarrow 2^E$; handling the overlap between those efficiently is merely a practical consideration.

We can capture this structure more abstractly by observing that \mathcal{L} induces a unique coalgebra homomorphism $l: A^* \rightarrow 2^{A^*}$, as shown in the following diagram.

$$\begin{array}{ccc} A^* & \xrightarrow{c} & (A^*)^A \\ \mathcal{L} \swarrow & \downarrow l & \downarrow l^A \\ 2 & \xleftarrow{\lambda?} 2^{A^*} \xrightarrow{\partial} & (2^{A^*})^A \end{array}$$

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

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

On the bottom 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 map $\lambda?$ determines the inclusion of the empty word in the language: $\lambda?(L) = L(\lambda)$. The set 2^{A^*} , together with these two functions, is the final coalgebra of the functor $2 \times (-)^A$ on **Sets**. The map of coalgebras $l: A^* \rightarrow 2^{A^*}$ thus exists and is unique by finality. More concretely, it is given for all $u, v \in A^*$ by

$$l(u)(v) = \mathcal{L}(uv). \tag{1}$$

Note that we used the functional view on \mathcal{L} here. The set A^* , together with the map c and the empty word $\lambda: 1 \rightarrow A^*$, is the initial algebra of the functor $1 + A \times -$; we could equivalently define l through initiality by regarding \mathcal{L} as an element of 2^{A^*} .

In the following lemma we use the inclusion map $n: S \hookrightarrow A^*$ and the function $k: 2^{A^*} \rightarrow 2^E$ defined by $k(L)(e) = L(e)$ for every $L \in 2^{A^*}$ and $e \in E$. These can be seen as representations of the subsets S and E . Furthermore, it is convenient to use the curried version $\Lambda(\text{row}_A): S \rightarrow (2^E)^A$ of row_A given by $\Lambda(\text{row}_A)(s)(a) = \text{row}_A(sa)$.

Lemma 1. *Given S and E , the observation table is defined by $\text{row} = k \circ l \circ n: S \rightarrow 2^E$ and $\Lambda(\text{row}_A) = k^A \circ \partial \circ l \circ n: S \rightarrow (2^E)^A$.*

Proof. For all $s \in S$ and $e \in E$, we have

$$\begin{aligned} (k \circ l \circ n)(s)(e) &= (k \circ l)(s)(e) && \text{definition of } n \\ &= l(s)(e) && \text{definition of } k \\ &\stackrel{(1)}{=} \mathcal{L}(se) \end{aligned}$$

and for each $a \in A$,

$$\begin{aligned} (k^A \circ \partial \circ l \circ n)(s)(a)(e) &= k((\partial \circ l \circ n)(s)(a))(e) \\ &= (\partial \circ l \circ n)(s)(a)(e) && \text{definition of } k \\ &= (l \circ n)(s)(ae) && \text{definition of } \partial \\ &= l(s)(ae) && \text{definition of } n \\ &\stackrel{(1)}{=} \mathcal{L}(sae). \end{aligned}$$

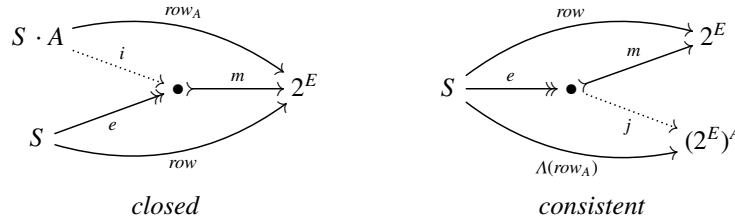
Thus, this yields $\text{row}(s)(e) = \mathcal{L}(se)$ and $\text{row}_A(sa)(e) = \Lambda(\text{row}_A)(s)(a)(e) = \mathcal{L}(sae)$, which is precisely the original definition. \square

There are two crucial properties of the observation table that play a key role in the algorithm of [2], allowing for the construction of a deterministic automaton from an observation table: closedness and consistency.

Definition 1 (Closed and Consistent Table [2]). *An observation table (S, E) is closed if for all $t \in S \cdot A$ there exists an $s \in S$ such that $\text{row}_A(t) = \text{row}(s)$. An observation table (S, E) is consistent if whenever s_1 and s_2 are elements of S such that $\text{row}(s_1) = \text{row}(s_2)$, for all $a \in A$, $\text{row}_A(s_1a) = \text{row}_A(s_2a)$.*

In many categories each map $f: A \rightarrow B$ 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 necessarily unique map i (resp. j) such that the diagram on the left (resp. right) commutes.*



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 $row(s) = row_A(t)$. We define i by $i(t) = e(s)$. It remains to show that $m \circ i = row_A$.

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

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

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

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

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 $row_A(s_1a) = row_A(s_2a)$. We define j by $j(e(s)) = \Lambda(row_A)(s)$, using that e is epi. By definition, $j \circ e = \Lambda(row_A)$. 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 $\Lambda(row_A)(s_1) = \Lambda(row_A)(s_2)$.

$$\begin{aligned} e(s_1) = e(s_2) &\Rightarrow row(s_1) = row(s_2) && \text{definition of } row \\ &\Rightarrow \forall a \in A. row_A(s_1a) = row_A(s_2a) && \text{consistency assumption} \\ &\Rightarrow \Lambda(row_A)(s_1) = \Lambda(row_A)(s_2) && \text{definition of } \Lambda. \end{aligned}$$

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

Conversely, suppose that there exists j such that $j \circ e = \Lambda(row_A)$, and let $s_1, s_2 \in S$ be such that $row(s_1) = row(s_2)$. Note that this is equivalent to $e(s_1) = e(s_2)$ because m is monic. We need to show $row_A(s_1a) = row_A(s_2a)$ for all $a \in A$, or, equivalently, $\Lambda(row_A)(s_1) = \Lambda(row_A)(s_2)$.

$$\begin{aligned} \Lambda(row_A)(s_1) &= (j \circ e)(s_1) && \text{assumption } \Lambda(row_A) = j \circ e \\ &= (j \circ e)(s_2) && \text{assumption } e(s_1) = e(s_2) \\ &= \Lambda(row_A)(s_2) && \text{assumption } \Lambda(row_A) = j \circ e. \quad \square \end{aligned}$$

Closed and consistent observation tables are important in the algorithm of [2] because they can be translated into a deterministic automaton. We first describe the construction concretely and subsequently more abstractly using our categorical reformulation.

Definition 2 (Automaton associated with an observation table [2]). *Given a closed and consistent observation 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, $q_0 \in Q$ is the initial state, and $\delta: Q \times A \rightarrow Q$ is the transition function. These are given by:*

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

To see that this is a well-defined automaton we need to check two facts: that F is a well-defined subset and that δ is a well-defined function.

Suppose s_1 and s_2 are elements of S such that $(\star) \text{row}(s_1) = \text{row}(s_2)$. We must show

$$\text{row}(s_1) \in F \iff \text{row}(s_2) \in F \text{ and} \quad (2)$$

$$\delta(\text{row}(s_1), a) = \delta(\text{row}(s_2), a) \in Q, \text{ for all } a \in A. \quad (3)$$

We have:

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

This concludes the proof of (2) above. Since the observation table is consistent, we have for each $a \in A$ that (\star) implies $\text{row}_A(s_1 a) = \text{row}_A(s_2 a)$, and hence we can calculate

$$\delta(\text{row}(s_1), a) = \text{row}_A(s_1 a) = \text{row}_A(s_2 a) = \delta(\text{row}(s_2), a).$$

It remains to show that $\text{row}_A(s_1 a) \in Q$. Since the table is closed, there exists an $s \in S$ such that $\text{row}(s) = \text{row}_A(s_1 a)$. Hence, $\text{row}_A(s_1 a) \in Q$ and (3) 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. [5]). This allows us to use the diagonal-fill-in property in the next result.

Lemma 3. *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,*

$$\begin{array}{ccc} S & \xrightarrow{e} & Q \\ \Lambda(i) \downarrow & \delta \swarrow & \downarrow j \\ Q^A & \xrightarrow{m^A} & (2^E)^A \end{array}$$

Proof. The function δ obtained by diagonalization above satisfies:

$$\delta(e(s))(a) = \Lambda(i)(s)(a) = i(sa).$$

This is the same as the above definition of δ , since $e(s)$ and $i(sa)$ represent, respectively, $\text{row}(s)$ and $\text{row}_A(sa)$. \square

Finally, the definitions of the initial and final states can be recovered from properties reminiscent of our reformulations of closedness and consistency.

Lemma 4. *The initial and final states can be obtained as the necessarily unique maps init and final making the diagrams below commute.*

$$\begin{array}{ccc} 1 & \xrightarrow{k \circ l \circ \lambda} & 2^E \\ \text{init} \swarrow & & \searrow m \\ S & \xrightarrow{e} & Q \\ \text{row} \swarrow & & \searrow m \\ & & 2^E \end{array} \quad \begin{array}{ccc} & \xrightarrow{\text{row}} & 2^E \\ S & \xrightarrow{e} & Q \\ & \searrow \text{final} & \searrow \\ & & 2 \\ \lambda' \circ l \circ n \swarrow & & \end{array}$$

Proof. We define $init(*) = e(\lambda)$ and $final(e(s)) = row(s)(\lambda)$ for all $s \in S$. These are equivalent to q_0 and F given in Definition 2 and satisfy

$$\begin{aligned}
(m \circ init)(*) &= (m \circ e)(\lambda) && \text{definition of } init \\
&= row(\lambda) && \text{factorization of } row \\
&= (k \circ l \circ n)(\lambda) && \text{Lemma 1} \\
&= (k \circ l \circ \lambda)(*) && n(\lambda) = \lambda = \lambda(*)
\end{aligned}$$

and for all $s \in S$,

$$\begin{aligned}
(final \circ e)(s) &= row(s)(\lambda) && \text{definition of } final \\
&= (k \circ l \circ n)(s)(\lambda) && \text{Lemma 1} \\
&= (l \circ n)(s)(\lambda) && \text{definition of } k \\
&= (\lambda? \circ l \circ n)(s) && \text{definition of } \lambda?.
\end{aligned}$$

Uniqueness is again necessary because m is monic and e is epic. \square

2.2 Minimal Conjectures

So far we have ignored the prefix-closedness of S and the suffix-closedness of E as they were not relevant for the construction of the automaton. The minimality result of [2], however, does depend on them. We encode these properties in two maps:

$$\begin{aligned}
\rho: S &\rightarrow 1 + S \times A && \sigma: 2 \times (2^E)^A \rightarrow 2^E \\
\rho(\lambda) &= * && \sigma(v, f)(\lambda) = v \\
\rho(sa) &= (s, a) && \sigma(v, f)(ae) = f(a)(e).
\end{aligned}$$

The main point of these maps is that they come equipped with inductive principles corresponding to induction on the length of prefix- and suffix-closed words.

Lemma 5. *For each algebra $[\chi_1, \chi_A]: 1 + X \times A \rightarrow X$ there exists a unique function $f: S \rightarrow X$ making the diagram below on the left commute, and for every coalgebra $\langle v_2, v_A \rangle: Y \rightarrow 2 \times Y^A$ there is a unique function $g: Y \rightarrow 2^E$ making the diagram below on the right commute.*

$$\begin{array}{ccc}
1 + S \times A & \xrightarrow{\text{id}_1 + f \times \text{id}_A} & 1 + X \times A \\
\rho \uparrow & & \downarrow [\chi_1, \chi_A] \\
S & \xrightarrow{f} & X
\end{array}
\qquad
\begin{array}{ccc}
Y & \xrightarrow{g} & 2^E \\
\langle v_2, v_A \rangle \downarrow & & \uparrow \sigma \\
2 \times Y^A & \xrightarrow{\text{id}_2 \times g^A} & 2 \times (2^E)^A
\end{array}$$

Proof. The requirement $f = [\chi_1, \chi_A] \circ (\text{id}_1 + f \times \text{id}_A) \circ \rho = [\chi_1, \chi_A \circ (f \times \text{id}_A)] \circ \rho$ corresponds to a definition of f by induction on the length of words in S , thus providing existence and uniqueness:

$$\begin{aligned}
f(\lambda) &= ([\chi_1, \chi_A \circ (f \times \text{id}_A)] \circ \rho)(\lambda) = [\chi_1, \chi_A \circ (f \times \text{id}_A)](*) = \chi_1(*) \\
f(sa) &= ([\chi_1, \chi_A \circ (f \times \text{id}_A)] \circ \rho)(sa) = [\chi_1, \chi_A \circ (f \times \text{id}_A)](s, a) = \chi_A(f(s), a).
\end{aligned}$$

The condition $g = \sigma \circ (\text{id}_2 \times g^A) \circ \langle v_2, v_A \rangle = \sigma \circ \langle v_2, g^A \circ v_A \rangle$ gives a definition of g by induction on the length of words in E :

$$\begin{aligned} g(y)(\lambda) &= (\sigma \circ \langle v_2, g^A \circ v_A \rangle)(y)(\lambda) = v_2(y) \\ g(y)(ae) &= (\sigma \circ \langle v_2, g^A \circ v_A \rangle)(y)(ae) = (g^A \circ v_A)(y)(a)(e) = g(v_A(y)(a))(e). \quad \square \end{aligned}$$

One might wonder what the unique such maps into the initial algebra and out of the final coalgebra are. Our next result will be that these are precisely n and k , respectively, but first we need to be more explicit about what the initial algebra is. The reason that we can identify it with the maps λ and c is that the currying operation extends from concatenated languages to arbitrary products and has an inverse Ψ (uncurrying). Moreover, given $f: W \rightarrow X$, $g: X \times A \rightarrow Y$, and $h: Y \rightarrow Z$, we have

$$h^A \circ \Lambda(g) \circ f = \Lambda(h \circ g \circ (f \times \text{id}_A)). \quad (4)$$

Explicitly, the initial algebra is the set A^* together with $[\lambda, \Psi(c)]: 1 + A^* \times A \rightarrow A^*$, where $\Psi(c)(u, a) = c(u)(a)$ for all $u \in A^*$ and $a \in A$.

Lemma 6. *The following diagrams commute.*

$$\begin{array}{ccc} 1 + S \times A & \xrightarrow{\text{id}_1 + n \times \text{id}_A} & 1 + A^* \times A \\ \rho \uparrow & & \downarrow [\lambda, \Psi(c)] \\ S & \xrightarrow{n} & A^* \end{array} \quad \begin{array}{ccc} 2^{A^*} & \xrightarrow{k} & 2^E \\ \langle \lambda?, \partial \rangle \downarrow & & \uparrow \sigma \\ 2 \times (2^{A^*})^A & \xrightarrow{\text{id}_2 \times k^A} & 2 \times (2^E)^A \end{array}$$

Proof. Suppose f is the unique coalgebra-to-algebra morphism $S \rightarrow A^*$ provided by Lemma 5. Using the calculations in that lemma (\star) we prove by induction on the length of words in S that $f = n$:

$$\begin{aligned} f(\lambda) &\stackrel{(\star)}{=} \lambda(*) &= \lambda &= n(\lambda) \\ f(sa) &\stackrel{(\star)}{=} \Psi(c)(f(s), a) &\stackrel{(\text{IH})}{=} \Psi(c)(n(s), a) &= n(s)a = n(sa). \end{aligned}$$

Similarly, let g be the unique coalgebra-to-algebra morphism $2^{A^*} \rightarrow 2^E$. Using induction on the length of words in E , we find that $g = k$:

$$\begin{aligned} g(L)(\lambda) &\stackrel{(\star)}{=} \lambda?(L) &= L(\lambda) &= k(L)(\lambda) \\ g(L)(ae) &\stackrel{(\star)}{=} g(\partial(L)(a))(e) &\stackrel{(\text{IH})}{=} k(\partial(L)(a))(e) &= \partial(L)(a)(e) = L(ae) = k(L)(ae). \end{aligned} \quad \square$$

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 this characterization that goes back to Kalman and was subsequently generalized by Arbib and Manes [9,3], these two properties can be nicely captured in the following diagram, where in the middle we have our automaton constructed from the

observation table.

$$\begin{array}{ccccc}
 1 & & & & 2 \\
 \lambda \downarrow & \nearrow \text{init} & & \nearrow \text{final} & \uparrow \lambda? \\
 A^* & \xrightarrow{r} & Q & \xrightarrow{o} & 2^{A^*} \\
 c \downarrow & & \delta \downarrow & & \downarrow \partial \\
 (A^*)^A & \xrightarrow{r^A} & Q^A & \xrightarrow{o^A} & (2^{A^*})^A
 \end{array} \tag{5}$$

Recall that the structure on the left is the initial algebra. The map r thus exists and is unique by initiality; it sends every word to the state it reaches. The map o exists and is unique by finality; it 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 (5): the automaton Q is *reachable* if $r: A^* \rightarrow Q$ is epic/surjective and it is *observable* if $o: Q \rightarrow 2^{A^*}$ is monic/injective.

Theorem 1. *The automaton associated with a closed and consistent observation table is minimal.*

Proof. We prove observability first because it is relatively straightforward. Note first that

$$m^A \circ \delta \circ e = j \circ e = \Lambda(\text{row}_A) = k^A \circ \partial \circ l \circ n, \tag{6}$$

using, from left to right, Lemma 3, Lemma 2, and Lemma 1. Now observe that the diagram below on the left commutes by Lemma 6 and the definition of o ,

$$\begin{array}{ccccc}
 Q & \xrightarrow{o} & 2^{A^*} & \xrightarrow{k} & 2^E \\
 \langle \text{final}, \delta \rangle \downarrow & & \downarrow \langle \lambda?, \partial \rangle & & \uparrow \sigma \\
 2 \times Q^A & \xrightarrow{\text{id}_2 \times o^A} & 2 \times (2^{A^*})^A & \xrightarrow{\text{id}_2 \times k^A} & 2 \times (2^E)^A
 \end{array} \quad \begin{array}{ccccc}
 Q & \xrightarrow{m} & 2^E \\
 \langle \text{final}, \delta \rangle \downarrow & & \uparrow \sigma \\
 2 \times Q^A & \xrightarrow{\text{id}_2 \times m^A} & 2 \times (2^E)^A
 \end{array}$$

and that the diagram on the right commutes because

$$\begin{aligned}
 & \sigma \circ (\text{id}_2 \times m^A) \circ \langle \text{final}, \delta \rangle \circ e \\
 &= \sigma \circ \langle \text{final} \circ e, m^A \circ \delta \circ e \rangle \\
 &= \sigma \circ \langle \lambda? \circ l \circ n, m^A \circ \delta \circ e \rangle && \text{Lemma 4} \\
 &\stackrel{(6)}{=} \sigma \circ \langle \lambda? \circ l \circ n, k^A \circ \partial \circ l \circ n \rangle \\
 &= \sigma \circ (\text{id}_2 \times k^A) \circ \langle \lambda?, \partial \rangle \circ l \circ n \\
 &= k \circ l \circ n && \text{Lemma 6} \\
 &= \text{row} && \text{Lemma 1} \\
 &= m \circ e && \text{factorization of row}
 \end{aligned}$$

and e is epic. From Lemma 5 it then follows that $k \circ o = m$, and hence o is monic.

For reachability we would like to do a similar proof, but as a result of a coalgebraic bias in some of our definitions the proof of

$$m \circ \Psi(\delta) \circ (e \times \text{id}_A) = k \circ l \circ c \circ (n \times \text{id}_A) \tag{7}$$

needs a little more work. In particular, it follows by using several times that Ψ and Λ are inverse to each other:

$$\begin{aligned}
m \circ \Psi(\delta) \circ (e \times \text{id}_A) &= \Psi(\Lambda(m \circ \Psi(\delta) \circ (e \times \text{id}_A))) \\
&\stackrel{(4)}{=} \Psi(m^A \circ \Lambda(\Psi(\delta)) \circ e) \\
&= \Psi(m^A \circ \delta \circ e) \\
&\stackrel{(6)}{=} \Psi(k^A \circ \partial \circ l \circ n) \\
&= \Psi(k^A \circ l^A \circ \Lambda(c) \circ n) && \text{definition of } l \\
&\stackrel{(4)}{=} \Psi(\Lambda(k \circ l \circ c \circ (n \times \text{id}_A))) \\
&= k \circ l \circ c \circ (n \times \text{id}_A).
\end{aligned}$$

The diagram below on the left commutes by Lemma 6 and the definition of r ,

$$\begin{array}{ccc}
1 + S \times A & \xrightarrow{\text{id}_1 + n \times \text{id}_A} & 1 + A^* \times A & \xrightarrow{\text{id}_1 + r \times \text{id}_A} & 1 + Q \times A \\
\rho \uparrow & & \downarrow [\lambda, \Psi(c)] & & \downarrow [\lambda, \Psi(\delta)] \\
S & \xrightarrow{n} & A^* & \xrightarrow{r} & Q
\end{array}
\quad
\begin{array}{ccc}
1 + S \times A & \xrightarrow{\text{id}_1 + e \times \text{id}_A} & 1 + Q \times A \\
\rho \uparrow & & \downarrow [\text{init}, \Psi(\delta)] \\
S & \xrightarrow{e} & Q
\end{array}$$

and for the diagram on the right we simply note that

$$\begin{aligned}
&m \circ [\text{init}, \Psi(\delta)] \circ (\text{id}_1 + e \times \text{id}_A) \circ \rho \\
&= [m \circ \text{init}, m \circ \Psi(\delta) \circ (e \times \text{id}_A)] \circ \rho \\
&\stackrel{(7)}{=} [m \circ \text{init}, k \circ l \circ c \circ (n \times \text{id}_A)] \circ \rho \\
&= [k \circ l \circ \lambda, k \circ l \circ c \circ (n \times \text{id}_A)] \circ \rho && \text{Lemma 4} \\
&= k \circ l \circ [\lambda, c] \circ (\text{id}_1 + n \times \text{id}_A) \circ \rho \\
&= k \circ l \circ n && \text{Lemma 6} \\
&= \text{row} && \text{Lemma 1} \\
&= m \circ e && \text{factorization of row}
\end{aligned}$$

and m is monic. From Lemma 5 it follows that $r \circ n = e$, so r must be epic. \square

2.3 The Learning Algorithm

We present the algorithm of [2] 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:    $S \leftarrow \{\lambda\}; E \leftarrow \{\lambda\}$ .
3:   repeat
4:     while  $(S, E)$  is not closed or not consistent do
5:       if  $(S, E)$  is not consistent then
6:         find  $s_1, s_2 \in S, a \in A$ , and  $e \in E$  such that
7:            $row(s_1) = row(s_2)$  and  $\mathcal{L}(s_1ae) \neq \mathcal{L}(s_2ae)$ 
8:            $E \leftarrow E \cup \{ae\}$ .
9:       end if
10:      if  $(S, E)$  is not closed then
11:        find  $s_1 \in S, a \in A$  such that
12:           $row_A(s_1a) \neq row_A(s)$ , for all  $s \in S$ 
13:           $S \leftarrow S \cup \{s_1a\}$ .
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:       $S \leftarrow S \cup \downarrow t$ .
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 [2]

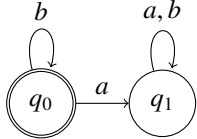
Imagine the Learner receives as input a Teacher for the master language

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

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

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

We extend row index set S and we again check for closedness and consistency.

Step 2	λ λ 1 a 0 b 1 aa 0 ab 0	(S, E) consistent? \checkmark (S, E) closed? \checkmark 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 . $S \leftarrow S \cup \{\lambda, a, aa, aaa\}$ 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.

	λ	
	λ	1
	a	0
	aa	0
	aaa	1
Step 3	b	1
	ab	0
	aab	0
	$aaaa$	0
	$aaab$	1

(S, E) consistent?
 No, $row(a) = row(aa)$ but $row_A(aa) \neq row_A(aaa)$.
 Then $E \leftarrow E \cup \{a\}$ and we go to **(Step 4)**.

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.

	λ	a	
	λ	1	0
	a	0	0
	aa	0	1
	aaa	1	0
Step 4	b	1	0
	ab	0	0
	aab	0	1
	$aaaa$	0	0
	$aaab$	1	0

(S, E) consistent? \checkmark (S, E) closed? \checkmark
 We make another guess:

```

    graph TD
      q0(((q0))) -- b --> q0
      q1(((q1))) -- b --> q1
      q0 -- a --> q1
      q1 -- a --> q2((q2))
      q2 -- a --> q0
      q2 -- b --> q2
    
```

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 Application to Reo automata

We now apply the categorical generalization of Angluin’s algorithm derived in the previous sections to learn coordination protocols. Specifically, the type of automata that we learn will be Reo automata [6], one of the many semantics for the Reo coordination language. This presents a different generalization made possible by our formulation of Angluin’s algorithm, namely by varying the base category for the coalgebra.

We need to find a suitable encoding of Reo automata; let us first recall their definition.

Definition 3. Let Σ be a finite set of ports. A Reo automaton is a tuple (Q, \rightarrow, q_0) where

- Q is a finite set of states, with $q_0 \in Q$ the initial state
- $\rightarrow \subseteq Q \times 2^\Sigma \times 2^\Sigma \times Q$ is a relation

When $(q, U, V, q') \in \rightarrow$, we write $q \xrightarrow{U|V} q'$. Furthermore,

- if $q \xrightarrow{U|V} q'$, then $U \subseteq V$ (reactivity)
- if $q \xrightarrow{U|V} q'$ and $U \subseteq V' \subseteq V$, then $q \xrightarrow{U|V'} q'$ (uniformity)

For the sake of simplicity, we work with *normalized* Reo automata; every Reo automaton can be transformed into an equivalent normalized Reo automaton [6].

Each transition of a Reo automaton represents an interaction allowed by the Reo circuit, as well as the necessary change in (internal) state. In a transition $q \xrightarrow{U|V} q'$, the set U represents the *ports fired* in the interaction, while V represents the *ports available* at that point in time. Reactivity guarantees that only available ports are fired, while uniformity ensures that unfired but available ports becoming unavailable will not change the availability of the interaction, or the resulting state change [6].

A Reo automaton $M = (Q, \rightarrow, q_0)$ defines a language $L_M(q)$ for every state $q \in Q$ as follows: for all $q \in Q$, it holds that $\lambda \in L_M(q)$; furthermore, $(U, V)w \in L_M(q)$ if and only if there exists a $q' \in Q$ such that $q \xrightarrow{U|V} q'$ and $w \in L_M(q')$. We write L_M for $L_M(q_0)$.

To encode Reo automata coalgebraically, we switch the base category of the discussion in this section to **Posets**, the category of partially ordered sets and monotone functions. If X is a poset, we write \leq_X for the accompanying partial order. We use 2 to denote the two-element poset $\{0, 1\}$, in which $0 \leq_2 1$. It is not hard to see that this category is *Cartesian closed*, that is, it comes with a terminal object (the singleton poset 1), products (the product poset $X \times Y$) and exponentials (the poset of monotone functions X^Y , ordered pointwise). We also have an analogue of the powerset functor: 2^X is the set of all monotone functions from X to 2 , or, equivalently, the poset of all \leq_X -upclosed subsets of X , ordered by inclusion. We can verify that for a fixed poset A , the mappings $FX = A \times X$ and $FX = X^A$ are functorial, as is $FX = 2^X$; the actions of the former two on monotone functions can be lifted from **Sets**, while the latter sends $f : X \rightarrow Y$ to the function 2^f , mapping \leq_X -upclosed subsets of X to \leq_Y -upclosed subsets of Y :

$$2^f(U) = \{y \in Y : \exists x \in U. f(x) \leq y\}$$

We now have the ingredients to define a coalgebraic version of Reo automata, which takes the form of a deterministic automaton in **Posets** over a special alphabet, as follows.

Definition 4. The poset of interactions, denoted \mathbb{A} , is $\{(U, V) : \emptyset \neq U \subseteq V \subseteq \Sigma\}$, ordered by the partial order $\leq_{\mathbb{A}}$, in which $(U, V) \leq_{\mathbb{A}} (U', V')$ if and only if $U = U'$ and $V' \subseteq V$.

A Reo coalgebra is a coalgebra for the functor $FX = 2 \times X^{\mathbb{A}}$.

In what follows, we denote the elements (U, V) of \mathbb{A} by writing $U|V$. We further abbreviate by denoting U and V as strings, i.e., when $U = \{A, B\}$ and $V = \{A, B, C\}$, we write $AB|ABC$ for (U, V) . Words over \mathbb{A} are written as letters separated by semicolons, e.g., we write $A|A;B|B$ for the word $(\{A\}, \{A\})(\{B\}, \{B\}) \in \mathbb{A}^*$.

Let us fix a Reo automaton $M = (Q, \rightarrow, q_0)$. We can represent \rightarrow as a function $\delta : Q \rightarrow (2^Q)^\mathbb{A}$, by setting $\delta(q)(a) = \{q' \in Q : q \xrightarrow{a} q'\}$. If we equip Q with the discrete order, δ is monotone. Furthermore, for $q \in Q$, $\delta(q) : \mathbb{A} \rightarrow 2^Q$ is monotone as a consequence of uniformity. Now, δ gives rise to $\delta^\sharp : 2^Q \rightarrow (2^Q)^\mathbb{A}$, given by

$$\delta^\sharp(U)(a) = \{q' \in Q : \exists q \in U. q \xrightarrow{a} q'\}$$

We note that δ^\sharp is again monotone. In general, a monotone $\delta^\sharp : 2^Q \rightarrow (2^Q)^\mathbb{A}$ can be constructed from any monotone $\delta : Q \rightarrow (2^Q)^\mathbb{A}$, by recognizing that $2^{(-)}$ is a monad on **Posets**, and choosing for δ^\sharp the Kleisli extension of δ .

Our Reo automaton M now gives rise to a Reo coalgebra $(2^Q, \langle \epsilon^\sharp, \delta^\sharp \rangle)$, where 2^Q and δ^\sharp are as above, and $\epsilon^\sharp : 2^Q \rightarrow 2$ is given by $\epsilon(U) = 1$ if and only if $U \neq \emptyset$.

We proceed to recover the language semantics of Reo automata from a Reo coalgebra, by finality. For this, we equip \mathbb{A}^* with the pointwise extension of $\leq_{\mathbb{A}}$, i.e., $a_0 a_1 \cdots a_{n-1} \leq_{\mathbb{A}^*} a'_0 a'_1 \cdots a'_{m-1}$ if and only if $n = m$, and for $i \in \{0, 1, \dots, n-1\}$ it holds that $a_i \leq_{\mathbb{A}} a'_i$. This is equivalent to defining $\mathbb{A}^* = \coprod_{i \in \mathbb{N}} \mathbb{A}^i$, using the coproducts and products of **Posets**. We know from the work of Arbib and Manes [3, Section 2.2] that \mathbb{A}^* (resp. $2^{\mathbb{A}^*}$) defined as such provides the desired notion of reachability (resp. observability). Specifically, we have the exact same situation as in (5), but with \mathbb{A} instead of A and with Q being a Reo coalgebra with an initial state in **Posets**. Reachability and observability maps are defined in complete analogy to their definition for a DFA.

The relation between the the language semantics of a Reo automaton and its encoding into a Reo coalgebra can now be formulated as follows.

Lemma 7. *Let $M = (Q, \rightarrow, q_0)$ be a Reo automaton, and let $(2^Q, \langle \epsilon^\sharp, \delta^\sharp \rangle)$ be the Reo coalgebra obtained from it. Furthermore, let h be the unique homomorphism into the final Reo coalgebra $(2^{\mathbb{A}^*}, \langle \lambda?, \delta \rangle)$. If $U \in 2^Q$, then*

$$h(U) = \bigcup_{q \in U} L_M(q)$$

Proof. Let $w \in \mathbb{A}^*$; we proceed by induction on $|w|$. In the base, where $w = \lambda$, we have that $\lambda \in h(U)$ if and only if $U \neq \emptyset$, which holds precisely when $\lambda \in \bigcup_{q \in U} L_M(q)$.

For the inductive step, let $w = av$ for $a \in \mathbb{A}$ and $v \in \mathbb{A}^*$. In that case, $av \in h(U)$ if and only if $v \in h(d(U)(a))$; by induction, the latter holds if and only if $v \in L_M(q')$ for some $q' \in d(U)(a)$, i.e., $v \in L_M(q')$ for some $q \in U$ with $q \xrightarrow{a} q'$, which is equivalent to $av \in L_M(q)$, which in turn is equivalent to $av \in \bigcup_{q \in U} L_M(q)$.

To learn Reo coalgebras using the framework outlined earlier, we define an observation table in this setting. Consider finite subsets S and E of \mathbb{A}^* , with all ordering inherited from \mathbb{A}^* ,³ S prefix-closed, and E suffix-closed. Note that if $s, s' \in S$ and $e, e' \in E$ are such that $s \leq_S s'$ and $e \leq_E e'$, then $\text{row}(s)(e) \leq_2 \text{row}(s')(e')$ by monotonicity. Thus, if $\text{row}(s)(e) = 1$, we can immediately conclude $\text{row}(s')(e') = 1$ without having to query the latter; similarly, $\text{row}(s)(e) = 0$ whenever $\text{row}(s')(e') = 0$. A similar optimization applies

³ What follows works also for discrete orders on S and E . Acknowledging the additional structure, however, allows us to explicitly save queries by exploiting the monotonicity of the row function.

to computing the function row_A . Note that the assumption of a prefix-closed language enables another optimization for the computation of these functions: if $s, s' \in S$ and $e, e' \in E$ are such that $s'e'$ is a prefix of se , then $row(s')(e') = 1$ whenever $row(s)(e) = 1$, and $row(s)(e) = 0$ whenever $row(s')(e') = 0$.

Since **Posets** is locally finitely presentable, it admits strong epi-mono factorizations [1,10]. There is no difference between closedness and consistency for learning a regular language over the alphabet \mathbb{A} and closedness and consistency in the present setting. Furthermore, like the notions of reachability and observability, the encodings of prefix-closedness and suffix-closedness as performed in Section 2.2 translate directly, as do Lemmas 5 and 6. Theorem 1 is therefore valid also in this setting.

It remains to show how we can obtain a Reo automaton from an observation table.

Definition 5. For (S, E) closed and consistent, we define $M = (Q^+, \rightarrow, q_0)$ as follows

$$Q^+ = \{row(s) : s \in S, row(s)(\lambda) = 1\} \quad q_0 = row(\lambda)$$

$$row(s) \xrightarrow{a} row(t) \iff row(t) \leq_{2^E} row_A(sa)$$

The fact that M is well-defined is a consequence of closedness and consistency as before. Additionally, we remark that reactivity follows from the definition of \mathbb{A} , and uniformity is a consequence of the monotonicity of row . It remains to show that the translation above is faithful, i.e., that the languages of the states of this Reo automaton correspond to the interpretation of $(Q, \langle final, \delta \rangle)$ in the final Reo-coalgebra.

Lemma 8. Let (S, E) be a closed and consistent observation table, obtained by learning the language of a Reo automaton, and let $M = (Q^+, \rightarrow, q_0)$ be the Reo automaton obtained from this table. Let h be the unique homomorphism from $(Q, \langle final, \delta \rangle)$ into the final Reo coalgebra. For $q \in Q^+$, we have that $L_M(q) = h(q)$.

Proof. We start by proving that if $w \in h(q)$, then $final(q) = 1$; the proof proceeds by induction on $|w|$. In the base, where $w = \lambda$, we know that $1 = h(q)(\lambda) = final(q)$. For the inductive step, write $w = av$ and assume the claim holds for v . Since $v \in h(\delta(q)(a))$, we find that $final(\delta(q)(a)) = 1$ by induction. But then

$$1 = final(\delta(q)(a)) = row_A(sa)(\lambda) = \mathcal{L}(sa) \leq_2 \mathcal{L}(s) = row(s)(\lambda) = final(q)$$

in which $\mathcal{L}(sa) \leq_2 \mathcal{L}(s)$ follows from prefix-closure of \mathcal{L} . Consequently, $final(q) = 1$.

For the main claim, we should verify that for $w \in \mathbb{A}^*$ and $q \in Q^+$ it holds that $L_M(q)(w) = h(q)(w)$. Note that there exists an $s \in S$ such that $q = row(s)$. The proof again proceeds by induction on $|w|$. In the base, where $w = \lambda$, we have that $\lambda \in L_M(q)$, as well as $\lambda \in h(q)$, since $q \in Q^+$ and therefore $final(q) = row(s)(\lambda) = 1$.

For the inductive step, let $w = av$ for $a \in \mathbb{A}$ and $v \in \mathbb{A}^*$. On the one hand, if $w \in L_M(q)$, then there exists a $q' \in Q$ such that $q \xrightarrow{a}_d q'$ and $v \in L_M(q')$. By definition of \rightarrow_d , we find that $q' \in Q^+$ and $q' \leq_{2^E} row_A(sa)$. By induction and monotonicity:

$$v \in h(q') \subseteq h(row_A(sa)) = h(\delta(row(s))(a)) = \partial(h(row(s)))(a)$$

allowing us to conclude that $w = av \in h(row(s)) = h(q)$.

On the other hand, if $w \in h(q)$, then $v \in \partial(h(q))(a) = h(\delta(q)(a))$. By the first part of this proof, $final(\delta(q)(a)) = 1$, and so $\delta(q)(a) \in Q^+$. We then find by induction that $v \in L_M(\delta(q)(a))$. Since $q \xrightarrow{a}_d \delta(q)(a)$, we conclude that $w = av \in L_M(q)$.

3.1 Learning a Reo circuit

We now review how the algorithm in Figure 1 would work as applied to Reo automata. Imagine that the Learner receives as input a Teacher for the Reo circuit in Figure 2, which represents what is called a “lossy FIFO” in Reo literature. The intended behavior of this circuit is as follows. When the buffer is empty, A can fire, and fill the buffer. If the buffer is full, three possibilities exist:

- (i) the buffer is emptied by firing B, or
- (ii) the buffer is emptied and immediately filled by firing A and B concurrently, or
- (iii) A fires, but the input is discarded (and the first token remains in the buffer).

Important to note is that the last option should not be available when B is enabled; that is, data is only discarded when the buffer is full *and* the token in the buffer cannot be handed off through B. This distinction makes the circuit in Figure 2 a prime example for learning a Reo automaton, since transitions carry information about ports fired and ports available [6]; if we compute the semantics of this circuit using Constraint Automata [4], this behavior cannot be modeled.

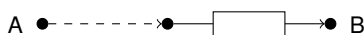


Fig. 2. The Reo circuit being learned

Before we dive into learning the circuit, let us first take a brief look at the poset of interactions we will be working with. Given that $\Sigma = \{A, B\}$, we can compute

$$\mathbb{A} = \{A|A, A|AB, B|B, B|AB, AB|AB\}$$

As for $\leq_{\mathbb{A}}$, there are only two (non-trivially) related pairs of letters; they are:

$$A|AB \leq_{\mathbb{A}} A|A \qquad B|AB \leq_{\mathbb{A}} B|B$$

The algorithm starts off by building a table for $S = \{\lambda\}$ and $E = \{\lambda\}$. Here, a membership query of $U_1|V_1; U_2|V_2; \dots; U_n|V_n$ should be interpreted as “can I fire these ports, while these ports are available, in this order?”. In this case, the entry for $row_A(A|A)(\lambda)$ is 1 because, in the initial configuration, we can fire A if it is available, while the entry for $row_A(B|B)(\lambda)$ is 0 because B cannot be fired in the initial configuration even if it is available.

	λ	
	λ	1
	A A	1
	A AB	1
	B B	0
	B AB	0
	AB AB	0

(S, E) consistent? \checkmark (S, E) closed?
 No, $row_A(B|B) \neq row(\lambda)$

Then, $S \leftarrow S \cup \{B|B\}$.
 We continue with **Step 2**.

Step 1

At this point, we note that we have made a membership query that could in principle have been skipped: the fact that $row_A(A|A)(\lambda) = 1$ follows from the fact that $row_A(A|AB)(\lambda) = 1$, since $A|AB \leq_A A|A$, and row_A is monotone.

To make the table closed, we add $B|B$ to S , and end up with the following table. Here, the row label “ $B|B;-$ ” represents all rows labeled $B|B;a$ for $a \in \mathbb{A}$; the value for entries in these rows is always 0, by prefix closure of the target language.

	λ
λ	1
$B B$	0
$A A$	1
$A AB$	1
$B AB$	0
$AB AB$	0
$B B;-$	0

(S, E) consistent? \checkmark (S, E) closed? \checkmark
 Then, we guess the automaton

Teacher replies with counter-example: $A|A;A|AB$
 Then, $S \leftarrow S \cup \{\lambda, A|A, A|A;A|AB\}$.
 We continue with **Step 3**.

The counterexample given by the Teacher here tells us that, after firing $A|A$, our circuit ends up in a different state, since firing $A|AB$ should not be possible (i.e., we should not be able to fire A but not B while both are available). Our current automaton does not account for this possibility: $A|AB$ can be fired after $A|A$, and this brings us to an accepting state; the counterexample is therefore justified.

Extending the table with the new contents of S , we find the following

	λ
λ	1
$A A$	1
$B B$	0
$A A;A AB$	0
$A AB$	1
$B AB$	0
$AB AB$	0
$B B;-$	0
$A A;A A$	1
$A A;B B$	1
$A A;B AB$	1
$A A;AB AB$	1
$A A;A AB;-$	0

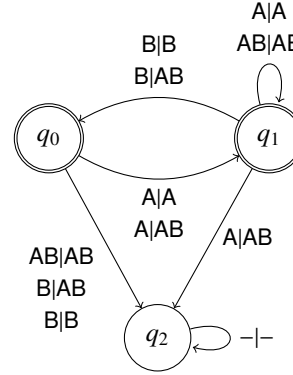
(S, E) consistent?
 No, because $row(\lambda) = row(A|A)$,
 while $row_A(A|AB) \neq row_A(A|A;A|AB)$.
 Then, $E \leftarrow \{A|AB\}$.
 We continue with **Step 4**.

Filling in the table with the updated E , we arrive at the following.

Step 4

	λ	A AB
λ	1	1
A A	1	0
B B	0	0
A A;A AB	0	0
A AB	1	0
B AB	0	0
AB AB	0	0
B B;-	0	0
A A;A A	1	0
A A;B B	1	1
A A;B AB	1	1
A A;AB AB	1	0
A A;A AB;-	0	0

(S, E) consistent? \checkmark (S, E) closed? \checkmark
Then, we guess the automaton



The Teacher replies **yes**.

In the last step, the algorithm stops, as the conjectured automaton faithfully represents the description of the lossy FIFO given at the start of this example. In particular, q_1 models a circuit with a full buffer: the edge towards q_0 represents firing B (emptying the buffer), while the loop back to q_1 represents the possibility of firing A (discarding an incoming token) or firing both A and B (shifting the incoming token into the emptied buffer); lastly, the edge towards q_2 encodes that A cannot be fired when B is also available.

4 Discussion

We have presented a categorical reformulation 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). In a previous paper [7] we explored the former avenue and derived algorithms for Moore and Mealy machines, which generalize the output set of DFAs. The application we concretely considered in the present paper explored the latter avenue, yielding an algorithm for Reo automata — essentially, finite automata in the category **Posets**. What makes the change in category interesting is that it precisely captures the algebraic structure of the actions (signal flows) in Reo and highlights the fact that interaction is a first-class concept by defining a poset of interactions to be used as the alphabet for the Reo automaton.

We would like to explore linking this algorithm to other work of Farhad on compilation [8] to learn Reo patterns. The work in this paper enables us to learn a Reo automaton of the global connector. However, in order to learn the components of the connector we would have to compile the *global* automaton into the composition of smaller Reo automata that would in turn correspond to basic Reo connectors. For scalability, it would be ideal to integrate part of the splitting into the learning algorithm.

References

1. Adámek, J., Rosický, J.: Locally presentable and accessible categories. Cambridge University Press (1994)
2. Angluin, D.: Learning regular sets from queries and counterexamples. *Inf. Comput.* 75(2), 87–106 (1987)
3. Arbib, M.A., Manes, E.G.: Adjoint machines, state-behavior machines, and duality. *Journal of Pure and Applied Algebra* 6(3), 313–344 (1975)
4. Baier, C., Sirjani, M., Arbab, F., Rutten, J.J.M.M.: Modeling component connectors in reo by constraint automata. *Sci. Comput. Program.* 61(2), 75–113 (2006), <https://doi.org/10.1016/j.scico.2005.10.008>
5. Barr, M., Wells, C.: *Toposes, Triples and Theories*. Springer, Berlin (1985), revised and corrected version available from www.cwru.edu/artsci/math/wells/pub/ttt.html
6. Bonsangue, M.M., Clarke, D., Silva, A.: Automata for context-dependent connectors. In: *Proc. Coordination Models and Languages*. pp. 184–203 (2009)
7. Jacobs, B., Silva, A.: Automata learning: A categorical perspective. In: van Breugel, F., Kashefi, E., Palamidessi, C., Rutten, J. (eds.) *Horizons of the Mind. A Tribute to Prakash Panangaden - Essays Dedicated to Prakash Panangaden on the Occasion of His 60th Birthday*. *Lecture Notes in Computer Science*, vol. 8464, pp. 384–406. Springer (2014), https://doi.org/10.1007/978-3-319-06880-0_20
8. Jongmans, S.T.Q., Arbab, F.: Global consensus through local synchronization: A formal basis for partially-distributed coordination. *Sci. Comput. Program.* 115-116, 199–224 (2016), <https://doi.org/10.1016/j.scico.2015.09.001>
9. Kalman, R.: On the general theory of control systems. *IRE Transactions on Automatic Control* 4(3), 110–110 (1959)
10. Milius, S.: A sound and complete calculus for finite stream circuits. In: *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*. pp. 421–430 (2010), <https://doi.org/10.1109/LICS.2010.11>
11. Vaandrager, F.W.: Model learning. *Commun. ACM* 60(2), 86–95 (2017), <http://doi.acm.org/10.1145/2967606>