Coalgebra Learning via Duality

Simone Barlocco\(^1\), Clemens Kupke\(^{1,\ast}\) and Jurriaan Rot\(^2\)

\(^1\) University of Strathclyde, Glasgow, \{simone.barlocco,clemens.kupke\}@strath.ac.uk
\(^2\) Radboud University, Nijmegen, j.rot@cs.ru.nl

Abstract. Automata learning is a popular technique for inferring minimal automata through membership and equivalence queries. In this paper, we generalise learning to the theory of coalgebras. The approach relies on the use of logical formulas as tests, based on a dual adjunction between states and logical theories. This allows us to learn, e.g., labelled transition systems, using Hennessy-Milner logic. Our main contribution is an abstract learning algorithm, together with a proof of correctness and termination.

1 Introduction

In recent years, automata learning is applied with considerable success to infer models of systems and in order to analyse and verify them. Most current approaches to active automata learning are ultimately based on the original algorithm due to Angluin [4], although numerous improvements have been made, in practical performance and in extending the techniques to different models [30].

Our aim is to move from automata to coalgebras [26,14], providing a generalisation of learning to a wide range of state-based systems. The key insight underlying our work is that dual adjunctions connecting coalgebras and tailor-made logical languages [21,12,19,26,22] allow us to devise a generic learning algorithm for coalgebras that is parametric in the type of system under consideration. Our approach gives rise to a fundamental distinction between states of the learned system and tests, modelled as logical formulas. This distinction is blurred in the classical DFA algorithm, where tests are also used to specify the (reachable) states. It is precisely the distinction between tests and states which allows us to move beyond classical automata, and use, for instance, Hennessy-Milner logic to learn bisimilarity quotients of labelled transition systems.

To present learning via duality we need to introduce new notions and refine existing ones. First, in the setting of coalgebraic modal logic, we introduce the new notion of sub-formula closed collections of formulas, generalising suffix-closed sets of words in Angluin’s algorithm (Section 4). Second, we import the abstract notion of base of a functor from [8], which allows us to speak about ‘successor states’ (Section 5). In particular, the base allows us to characterise

\(\ast\) Partially supported by EPSRC grant EP/N015843/1.
reachability of coalgebras in a clear and concise way. This yields a canonical procedure for computing the reachable part from a given initial state in a coalgebra, thus generalising the notion of a generated subframe from modal logic.

We then rephrase coalgebra learning as the problem of inferring a coalgebra which is reachable, minimal and which cannot be distinguished from the original coalgebra held by the teacher using tests. This requires suitably adapting the computation of the reachable part to incorporate tests, and only learn ‘up to logical equivalence’. We formulate the notion of closed table, and an associated procedure to close tables. With all these notions in place, we can finally define our abstract algorithm for coalgebra learning, together with a proof of correctness and termination (Section 6). Overall, we consider this correctness and termination proof as the main contribution of the paper; other contributions are the computation of reachability via the base and the notion of sub-formula closedness. At a more conceptual level, our paper shows how states and tests interact in automata learning, by rephrasing it in the context of a dual adjunction connecting coalgebra (systems) and algebra (logical theories). As such, we provide a new foundation of learning state-based systems.

**Related work.** The idea that tests in the learning algorithm should be formulas of a distinct logical language was proposed first in [6]. However, the work in loc. cit. is quite ad-hoc, confined to Boolean-valued modal logics, and did not explicitly use duality. This paper is a significant improvement: the dual adjunction framework and the definition of the base [8] enables us to present a description of Angluin’s algorithm in purely categorical terms, including a proof of correctness and, crucially, termination. Our abstract notion of logic also enables us to recover exactly the standard DFA algorithm (where tests are words) and the algorithm for learning Mealy machines (where test are many-valued), something that is not possible in [6] where tests are modal formulas. Closely related to our work is also the line of research initiated by [15] and followed up within the CALF project [11,12,13] which applies ideas from category theory to automata learning. Our approach is orthogonal to CALF: the latter focuses on learning a general version of automata, whereas our work is geared towards learning bisimilarity quotients of state-based transition systems. While CALF lends itself to studying automata in a large variety of base categories, our work thus far is concerned with varying the type of transition structures.

## 2 Learning by Example

The aim of this section is twofold: (i) to remind the reader of the key elements of Angluin’s $L^\ast$ algorithm [4] and (ii) to motivate and outline our generalisation.

In the classical $L^\ast$ algorithm, the learner tries to learn a regular language $L$ over some alphabet $A$ or, equivalently, a DFA $\mathcal{A}$ accepting that language. Learning proceeds by asking queries to a teacher who has access to this automaton. Membership queries allow the learner to test whether a given word is in the language, and equivalence queries to test whether the correct DFA has been learned.
already. The algorithm constructs so-called tables \((S, E)\) where \(S, E \subseteq A^*\) are the rows and columns of the table, respectively. The value at position \((s, e)\) of the table is the answer to the membership query “\(se \in \mathcal{L}\)?”.

Words play a double role: On the one hand, a word \(w \in S\) represents the state which is reached when reading \(w\) at the initial state. On the other hand, the set \(E\) represents the set of membership queries that the learner is asking about the states in \(S\). A table is closed if for all \(w \in S\) and all \(a \in A\) either \(wa \in S\) or there is a state \(v \in S\) such that \(wa\) is equivalent to \(v\) w.r.t. membership queries of words in \(E\). If a table is not closed we extend \(S\) by adding words of the form \(wa\) for \(w \in S\) and \(a \in A\). Once it is closed, one can define a conjecture, i.e., a DFA with states in \(S\). The learner now asks the teacher whether the conjecture is correct. If it is, the algorithm terminates. Otherwise the teacher provides a counterexample: a word on which the conjecture is incorrect. The table is now extended using the counterexample. As a result, the table is not closed anymore and the algorithm continues again by closing the table.

Our version of \(L^*\) introduces some key conceptual differences: tables are pairs \((S, \Psi)\) such that \(S\) (set of rows) is a selection of states of \(A\) and \(\Psi\) (set of columns) is a collection of tests/formulas. Membership queries become checks of tests in \(\Psi\) at states in \(S\) and equivalence queries verify whether or not the learned structure is logically equivalent to the original one. A table \((S, \Psi)\) is closed if for all successors \(x'\) of elements of \(S\) there exists an \(x \in S\) such that \(x\) and \(x'\) are equivalent w.r.t. formulas in \(\Psi\). The clear distinction between states and tests in our algorithm means that counterexamples are formulas that have to be added to \(\Psi\). Crucially, the move from words to formulas allows us to use the rich theory of coalgebra and coalgebraic logic to devise a generic algorithm.

We consider two examples within our generic framework: classical DFAs, yielding essentially the \(L^*\) algorithm, and labelled transition systems, which is to the best of our knowledge not covered by standard automata learning algorithms.

For the DFA case, let \(L = \{u \in \{a, b\}^* \mid \text{number of } a\text{'s mod 3 }= 0\}\) and assume that the teacher uses the following (infinite) automaton describing \(L\):

![Diagram of automaton]

As outlined above, the learner starts to construct tables \((S, \Psi)\) where \(S\) is a selection of states of the automaton and \(\Psi\) are formulas. For DFAs we will see (Ex. 1) that our formulas are just words in \(\{a, b\}^*\). Our starting table is \((\{q_0\}, \emptyset)\), i.e., we select the initial state and do not check any logical properties. This table is trivially closed, as all states are equivalent w.r.t. \(\emptyset\). The first conjecture is the automaton consisting of one accepting state \(q_0\) with \(a\)- and \(b\)-loops, whose language is \(\{a, b\}^*\). This is incorrect and the teacher provides, e.g., \(aa\) as counterexample. The resulting table is \((\{q_0\}, \{\varepsilon, a, aa\})\) where the second component was generated by closing \(\{aa\}\) under suffixes. Suffix closedness features both in the original
L∗ algorithm and in our framework (Section 4). The table \((\{q_0\}, \{\varepsilon, a, aa\})\) is not closed as \(q_1\), the \(a\)-successor of \(q_0\), does not accept \(\varepsilon\) whereas \(q_0\) does. Therefore we extend the table to \((\{q_0, q_1\}, \{\varepsilon, a, aa\})\). Note that, unlike in the classical setting, exploring successors of already selected states cannot be achieved by appending letters to words, but we need to locally employ the transition structure on the automaton \(A\) instead. A similar argument shows that we need to extend the table further to \((\{q_0, q_1, q_2\}, \{\varepsilon, a, aa\})\) which is closed. This leads to the (correct) conjecture depicted on the right below. The acceptance condition and transition structure has been read off from the original automaton, where the transition from \(q_2\) to \(q_0\) is obtained by realising that \(q_2\)'s successor \(q_3\) is represented by the equivalent state \(q_0\) ∈ \(S\).

A key feature of our work is that the L∗ algorithm can be systematically generalised to new settings, in particular, to the learning of bisimulation quotients of transition systems. Consider the following labelled transition system (LTS). We would like to learn its minimal representation, i.e., its quotient modulo bisimulation.

Our setting allows us to choose a suitable logical language. For LTSs, the language consists of the formulas of standard multi-modal logic (cf. Ex. 3). The semantics is as usual where \((a) \phi\) holds at a state if it has an \(a\)-successor that makes \(\phi\) true.

As above, the algorithm constructs tables, starting with \((S = \{x_0\}, \Psi = \emptyset)\). The table is closed, so the first conjecture is a single state with no proposition letter true (note that \(x_0\) has no \(b\) or \(c\) successor and no proposition is true at \(x_0\)). It is, however, easy for the teacher to find a counterexample. For example, the formula \((a) \langle b \rangle \top\) is true at the root of the original LTS but false in the conjecture. We add the counterexample and all its subformulas to \(\Psi\) and obtain a new table \((\{x_0\}, \Psi')\) with \(\Psi' = \{(a) \langle b \rangle \top, \langle b \rangle \top, \top\}\). Now, the table is not closed, as \(x_0\) has successor \(x_1\) that satisfies \(\langle b \rangle \top\) whereas \(x_0\) does not satisfy \(\langle b \rangle \top\). Therefore we add \(x_1\) to the table to obtain \((\{x_0, x_1\}, \Psi')\). Similar arguments will lead to the closed table \((\{x_0, x_1, x_3, x_4\}, \Psi')\) which also yields the correct conjecture. Note that the state \(x_2\) does not get added to the table as it is equivalent to \(x_1\) and thus already represented. This demonstrates a remarkable fact: we computed the bisimulation quotient of the LTS without inspecting the (infinite) right-hand side of the LTS.

Another important example that fits smoothly into our framework is the well-known variant of Angluin’s algorithm to learn Mealy machines (Ex. 2). Thanks to our general notion of logic, our framework allows to use an intuitive language, where a formula is simply an input word \(w\) whose truth value at a state \(x\) is the observed output after entering \(w\) at \(x\). This is in contrast to [6] where formulas
had to be Boolean valued. Multi-valued logics fit naturally in our setting; this is expected to be useful to deal with systems with quantitative information.

3 Preliminaries

The general learning algorithm in this paper is based on the theory of coalgebras, which provides an abstract framework for representing state-based transition systems. In what follows we assume that the reader is familiar with basic notions of category theory and coalgebras \cite{14,26}. We briefly recall the notion of pointed coalgebra, modelling a coalgebra with an initial state. Let $C$ be a category with a terminal object $1$ and let $B: C \to C$ be a functor. A pointed $B$-coalgebra is a triple $(X, \gamma, x_0)$ where $X \in C$ and $\gamma: X \to BX$ and $x_0: 1 \to X$, specifying the coalgebra structure and the point (“initial state”) of the coalgebra, respectively.

Coalgebraic modal logic. Modal logics are used to describe properties of state-based systems, modelled here as coalgebras. The close relationship between coalgebras and their logics is described elegantly via dual adjunctions \cite{20,18,24,21}. Our basic setting consists of two categories $C, D$ connected by functors $P, Q$ forming a dual adjunction $P \dashv Q: C \leftrightarrow D^{op}$. In other words, we have a natural bijection $C(X, Q\Delta) \cong D(\Delta, PX)$ for $X \in C, \Delta \in D$. Moreover, we assume the two functors, $B: C \to C, L: D \to D$, see (1). The functor $L$ represents the syntax of the (modalities in the) logic: assuming that $L\Phi \Rightarrow \Phi$ we think of $\Phi$ as the collection of formulas, or tests. In this logical perspective, the functor $P$ maps an object $X$ of $C$ to the collection of predicates and the functor $Q$ maps an object $\Delta$ of $D$ to the collection $Q\Delta$ of $\Delta$-theories.

The connection between coalgebras and their logics is specified via a natural transformation $\delta: LP \Rightarrow PB$, sometimes referred to as the one-step semantics $\Phi \Rightarrow \exists \Phi$ (cf. \cite{18,24}). More precisely, we have $\delta^\gamma = Q\varepsilon \circ Q\delta \circ \eta BQ$, where $\eta, \varepsilon$ are the unit and counit of the adjunction. Then $\text{th}^\gamma: X \to Q\Phi$ is the unique morphism making (3) commute.

Example 1. Let $C = D = \text{Set}, P = Q = 2^{-}$ the contravariant power set functor, $B = 2 \times -A$ and $L = 1 + A \times -$ . In this case $B$-coalgebras can be thought of as...
deterministic automata with input alphabet $A$ (e.g., [25]). It is well-known that the initial $L$-algebra is $\Phi = A^*$ with structure $\alpha = [\varepsilon, \text{cons}]: 1 + A \times A^* \to A^*$ where $\varepsilon$ selects the empty word and cons maps a pair $(a, w) \in A \times A^*$ to the word $aw \in A^*$, i.e., in this example our tests are words with the intuitive meaning that a test succeeds if the word is accepted by the given automaton. For $X \in \mathcal{C}$, the $X$-component of the (one-step) semantics $\delta: LP \Rightarrow PB$ is defined as follows: $\delta_X(\varepsilon) = \{(i, f) \in 2 \times X^A \mid i = 1\}$, and $\delta_X(a, U) = \{(i, f) \in 2 \times X^A \mid f(a) \in U\}$. It is matter of routine checking that the semantics of tests in $\Phi$ on a $B$-coalgebra $(X, \gamma)$ is as follows: we have $[\varepsilon] = \{x \in X \mid \pi_1(\gamma(x)) = 1\}$ and $[aw] = \{x \in X \mid \pi_1(\gamma(x))(a) \in [w]\}$, where $\pi_1$ and $\pi_2$ are the projection maps. The theory map $\theta^2$ sends a state to the language accepted by that state in the usual way.

**Example 2.** Again let $\mathcal{C} = \mathcal{D} = \text{Set}$ and consider the functors $P = Q = O^-$, $B = (O \times -)^A$ and $L = A \times (1 + -)$, where $A$ and $O$ are fixed sets, thought of as input and output alphabet, respectively. Then $B$-coalgebras are Mealy machines and the initial $L$-algebra is given by the set $A^+$ of finite non-empty words over $A$. For $X \in \mathcal{C}$, the one-step semantics $\delta_X: A \times (1 + O^X) \to O^{BX}$ is defined by $\delta_X(a, \text{inl}(*) = \lambda f.\pi_1(f(a))$ and $\delta_X(a, \text{inr}(g)) = \lambda f.g(\pi_2(f(a)))$. Concretely, formulas are words in $A^+$; the (O-valued) semantics of $w \in A^+$ at state $x$ is the output $a \in O$ that is produced after processing the input $w$ from state $x$.

**Example 3.** Let $\mathcal{C} = \text{Set}$ and $\mathcal{D} = \text{BA}$, where the latter denotes the category of Boolean algebras. Again $P = 2^-$, but this time $2^X$ is interpreted as a Boolean algebra. The functor $Q$ maps a Boolean algebra to the collection of ultrafilters over it [7]. Furthermore $B = (P-)^A$ where $P$ denotes covariant power set and $A$ a set of actions. Coalgebras for this functor correspond to labelled transition systems, where a state has a set of successors that depends on the action/input from $A$. The dual functor $L: \text{BA} \to \text{BA}$ is defined as $LY := F_{\text{BA}}(\{(a) y \mid a \in A, y \in Y\})/\equiv$ where $F_{\text{BA}}: \text{Set} \to \text{BA}$ denotes the free Boolean algebra functor and where, roughly speaking, $\equiv$ is the congruence generated from the axioms $\langle a \rangle \bot \equiv \bot \equiv \bot$ and $\langle a \rangle \langle y_1 \lor y_2 \equiv \langle a \rangle (y_1 \lor (a) y_2\rangle$ for each $a \in A$. This is explained in more detail in [21]. The initial algebra for this functor is the so-called Lindenbaum-Tarski algebra [7] of modal formulas $(\phi ::= \bot \mid \phi \lor \phi \mid \neg \phi \mid \langle a \rangle \phi \rangle$ quotiented by logical equivalence. The definition of an appropriate $\delta$ can be found in, e.g., [21]—the semantics $\llbracket \cdot \rrbracket$ of a formula then amounts to the standard one [7].

Different types of probabilistic transition systems also fit into the dual adjunction framework, see, e.g., [17].

**Subobjects and intersection-preserving functors.** We denote by $\text{Sub}(X)$ the collection of subobjects of an object $X \in \mathcal{C}$. Let $\leq$ be the order on subobjects $s: S \to X, s': S' \to X$ given by $s \leq s'$ iff there is $m: S \to S'$ s.t. $s = s' \circ m$. The intersection $\bigwedge J \rightsquigarrow X$ of a family $J = \{s_i: S_i \to X\}_{i \in I}$ is defined as the greatest lower bound w.r.t. the order $\leq$. In a complete category, it can be computed by (wide) pullback. We denote the maps in the limiting cone by $x_i: \bigwedge J \rightsquigarrow S_i$. 

---

6
For a functor \( B: \mathcal{C} \to \mathcal{D} \), we say \( B \) preserves (wide) intersections if it preserves these wide pullbacks, i.e., if \( (B(\bigwedge J), \{BX_i\}_{i \in I}) \) is the pullback of \( \{BS_i: BS_i \to BX\}_{i \in I} \). By [2, Lemma 3.53] (building on [29]), finitary functors on \( \text{Set} \) ‘almost’ preserve wide intersections: for every such functor \( B \) there is a functor \( B' \) which preserves wide intersections and agrees with \( B \) on all non-empty sets. Finally, if \( B \) preserves intersections, then it preserves monos.

**Minimality notions.** The algorithm that we will describe in this paper learns a minimal and reachable representation of an object. The intuitive notions of minimality and reachability are formalised as follows.

**Definition 4.** We call a \( B \)-coalgebra \((X, \gamma)\) minimal w.r.t. logical equivalence if the theory map \( \text{th}: X \to Q\Phi \) is a monomorphism.

**Definition 5.** We call a pointed \( B \)-coalgebra \((X, \gamma, x_0)\) reachable if for any subobject \( s: S \to X \) and \( s_0: 1 \to S \) with \( x_0 = s \circ s_0 \), if \( S \) is a subcoalgebra of \((X, \gamma)\) then \( s \) is an isomorphism.

For expressive logics [27], behavioural equivalence concides with logical equivalence. Hence, in that case, our algorithm learns a “well-pointed coalgebra” in the terminology of [2], i.e., a pointed coalgebra that is reachable and minimal w.r.t. behavioural equivalence. All logics appearing in this paper are expressiv.

**Assumption on \( \mathcal{C} \) and Factorisation System.** Throughout the paper we will assume that \( \mathcal{C} \) is a complete and well-powered category. Well-powered means that for each \( X \in \mathcal{C} \) the collection \( \text{Sub}(X) \) of subobjects of a given object forms a set. Our assumptions imply [10, Proposition 4.4.3] that every morphism \( f \) in \( \mathcal{C} \) factors uniquely (up to isomorphism) as \( f = m \circ e \) with \( m \) a mono and \( e \) a strong epi. Recall that an epimorphism \( e: X \to Y \) is strong if for every commutative square in (4) where the bottom arrow is a monomorphism, there exists a unique diagonal morphism \( d \) such that the entire diagram commutes.

4 Subformula Closed Collections of Formulas

Our learning algorithm will construct conjectures that are “partially” correct, i.e., correct with respect to a subobject of the collection of all formulas/tests. Recall this collection of all tests are formalised in our setting as the initial \( L \)-algebra \((\Phi, \alpha: L\Phi \to \Phi)\). To define a notion of partial correctness we need to consider subobjects of \( \Phi \) to which we can restrict the theory map. This is formalised via the notion of “subformula closed” subobject of \( \Phi \).

The definition of such subobjects is based on the notion of recursive coalgebra. For \( L: \mathcal{D} \to \mathcal{D} \) an endofunctor, a coalgebra \( f: X \to LX \) is called recursive if for every \( L \)-algebra \( g: LY \to Y \) there is a unique ‘coalgebra-to-algebra’ map \( g^! \) making (5) commute.
Definition 6. A subobject \( j : \Psi \to \Phi \) is called a subformula closed collection (of formulas) if there is a unique \( L \)-coalgebra structure \( \sigma : \Psi \to L\Psi \) such that \((\Psi, \sigma)\) is a recursive \( L \)-coalgebra and \( j \) is the (necessarily unique) coalgebra-to-algebra map from \((\Psi, \sigma)\) to the initial algebra \((\Phi, \alpha)\).

Remark 7. The uniqueness of \( \sigma \) in Definition 6 is implied if \( L \) preserves monomorphisms. This is the case in our examples. The notion of recursive coalgebra goes back to [28,23]. The paper [1] contains a claim that the first item of our definition of subformula closed collection is implied by the second one if \( L \) preserves preimages. In our examples both properties of \((\Psi, \sigma)\) are verified directly, rather than by relying on general categorical results.

Example 8. In the setting of Example 1, where the initial \( L \)-algebra is based on the set \( A^* \) of words over the set (of inputs) \( A \), a subset \( \Psi \subseteq A^* \) is subformula-closed if it is suffix-closed, i.e., if for all \( aw \in \Psi \) we have \( w \in \Psi \) as well.

Example 9. In the setting that \( B = (\mathcal{P} -)^A \) for some set of actions \( A \), \( C = \text{Set} \) and \( D = BA \), the logic is given as a functor \( L \) on Boolean algebras as discussed in Example 3. As a subformula closed collection is an object in \( \Psi \), we are not simply dealing with a set of formulas, but with a Boolean algebra. The connection to the standard notion of being closed under taking subformulas in modal logic [7] can be sketched as follows: given a set \( \Delta \) of modal formulas that is closed under taking subformulas, we define a Boolean algebra \( \Psi \Delta \subseteq \Phi \) as the smallest Boolean subalgebra of \( \Phi \) that is generated by the set \( \widehat{\Delta} = \{ [\phi]_{\Phi} | \phi \in \Delta \} \) where for a formula \( \phi \) we let \( [\phi]_{\Phi} \in \Phi \) denote its equivalence class in \( \Phi \).

It is then not difficult to define a suitable \( \sigma : \Psi_\Delta \to L\Psi_\Delta \). As \( \Psi_\Delta \) is generated by closing \( \Delta \) under Boolean operations, any two states \( x_1, x_2 \) in a given coalgebra \((X, \gamma)\) satisfy (\( \forall b \in \Psi_\Delta, x_1 \in [b] \Leftrightarrow x_2 \in [b] \)) iff (\( \forall b \in \widehat{\Delta}, x_1 \in [b] \Leftrightarrow x_2 \in [b] \)). In other words, equivalence w.r.t. \( \Psi_\Delta \) coincides with equivalence w.r.t. the set of formulas \( \Delta \). This explains why in the concrete algorithm, we do not deal with Boolean algebras explicitly, but with subformula closed sets of formulas instead.

The key property of subformula closed collections \( \Psi \) is that we can restrict our attention to the so-called \( \Psi \)-theory map. Intuitively, subformula closedness is what allows us to define this theory map inductively.

Lemma 10. Let \( \Psi \to \Phi \) be a sub-formula closed collection, with coalgebra structure \( \sigma : \Psi \to L\Psi \). Then \( \theta^\Psi_\Psi = Qj \circ \theta^\Psi_\Phi \) is the unique map making (6) commute. We call \( \theta^\Psi_\Psi \) the \( \Psi \)-theory map, and omit the \( \Psi \) if it is clear from the context.

5 Reachability and the Base

In this section, we define the notion of base of an endofunctor, taken from [8]. This allows us to speak about the (direct) successors of states in a coalgebra, and about reachability, which are essential ingredients of the learning algorithm.
Definition 11. Let $B : \mathcal{C} \to \mathcal{C}$ be an endofunctor. We say $B$ has a base if for every arrow $f : X \to BY$ there exist $g : X \to BZ$ and $m : Z \to Y$ with $m$ a monomorphism such that $f = Bm \circ g$, and for any pair $g' : X \to BZ'$, $m' : Z' \to Y$ with $Bm' \circ g' = f$ and $m' \circ h = m$, see Diagram (7). We call $(Z, g, m)$ the \textit{(B)}-base of the morphism $f$.

We sometimes refer to $m : Z \to Y$ as the base of $f$, omitting the $g$ when it is irrelevant, or clear from the context. Note that the terminology ‘the’ base is justified, as it is easily seen to be unique up to isomorphism.

For example, let $B : \text{Set} \to \text{Set}$, $BX = 2 \times X^A$. The base of a map $f : X \to BY$ is given by $m : Z \to Y$, where $Z = \{(\pi_2 \circ f)(x)(a) \mid x \in X, a \in A\}$, and $m$ is the inclusion. The associated $g : X \to BZ$ is the corestriction of $f$ to $BZ$.

For $B = (P-)^A : \text{Set} \to \text{Set}$, the $B$-base of $f : X \to Y$ is given by the inclusion $m : Z \to Y$, where $Z = \{y \in Y \mid \exists x \in X, \exists a \in A \text{ s.t. } y \in f(x)(a)\}$. 

Proposition 12. Suppose $\mathcal{C}$ is complete and well-powered, and $B : \mathcal{C} \to \mathcal{C}$ preserves (wide) intersections. Then $B$ has a base.

If $\mathcal{C}$ is a locally presentable category, then it is complete and well-powered \cite{3} Remark 1.56]. Hence, in that case, any functor $B : \mathcal{C} \to \mathcal{C}$ which preserves intersections has a base. The following lemma will be useful in proofs.

Lemma 13. Let $B : \mathcal{C} \to \mathcal{C}$ be a functor that has a base and that preserves pre-images. Let $f : S \to BX$ and $h : X \to Y$ be morphisms, let $(Z, g, m)$ be the base of $f$ and let $e : Z \to W$, $m' : W \to Y$ be the \textit{(strong epi, mono)}-factorisation of $h \circ m$. Then $(W, Be \circ g, m')$ is the base of $Bh \circ f$.

The $B$-base provides an elegant way to relate reachability within a coalgebra to a monotone operator on the (complete) lattice of subobjects of the carrier of the coalgebra. Moreover, we will see that the least subcoalgebra that contains a given subobject of the carrier can be obtained via a standard least fixpoint construction. Finally, we will introduce the notion of prefix closed subobject of a coalgebra, generalising the prefix closedness condition from Angluin’s algorithm.

By our assumption on $\mathcal{C}$ at the end of Section 3, the collection of subobjects $(\text{Sub}(X), \leq)$ ordered as usual (cf. page 3) forms a complete lattice. Recall that the meet on $\text{Sub}(X)$ (intersection) is defined via pullbacks. In categories with coproducts, the join $s_1 \lor s_2$ of subobjects $s_1, s_2 \in \text{Sub}(X)$ is defined as the mono part of the factorisation of the map $[s_1, s_2] : S_1 + S_2 \to X$, i.e., $[s_1, s_2] = (s_1 \lor s_2) \circ e$ for a strong epi $e$. In Set, this amounts to taking the union of subsets.

For a binary join $s_1 \lor s_2$ we denote by $\text{inl}_i : S_i \to (S_1 \lor S_2)$ and $\text{inr}_i : S_2 \to (S_1 \lor S_2)$ the embeddings that exist by $s_i \leq s_1 \lor s_2$ for $i = \{1, 2\}$. Let us now define the key operator of this section.

\[ \begin{array}{ccc} S & \xrightarrow{\gamma} & X \\
\downarrow g & & \downarrow \gamma \\
B\Gamma(S) & \xrightarrow{B\Gamma^B(\gamma)} & BX \end{array} \]
**Definition 14.** Let $B$ be a functor that has a base, $s: S \rightarrow X$ a subobject of some $X \in C$ and let $(X, \gamma)$ be a $B$-coalgebra. Let $(\Gamma(S), g, \Gamma_B(s))$ be the $B$-base of $\gamma \circ s$, see Diagram 5. Whenever $B$ and $\gamma$ are clear from the context, we write $\Gamma(s)$ instead of $\Gamma_B^B(s)$.

**Lemma 15.** Let $B: C \rightarrow C$ be a functor with a base and let $(X, \gamma)$ be a $B$-coalgebra. The operator $\Gamma: \text{Sub}(X) \rightarrow \text{Sub}(X)$ defined by $s \mapsto \Gamma(s)$ is monotone.

Intuitively, $\Gamma$ computes for a given set of states $S$ the set of “immediate successors”, i.e., the set of states that can be reached by applying $\gamma$ to an element of $S$.

We will see that pre-fixpoints of $\Gamma$ correspond to subcoalgebras. Furthermore, $\Gamma$ is the key to formulate our notion of closed table in the learning algorithm.

**Proposition 16.** Let $s: S \rightarrow X$ be a subobject and $(X, \gamma) \in \text{Coalg}(B)$ for $X \in C$ and $B: C \rightarrow C$ a functor that has a base. Then $s$ is a subcoalgebra of $(X, \gamma)$ if and only if $\Gamma(s) \leq s$. Consequently, the collection of subcoalgebras of a given $B$-coalgebra forms a complete lattice.

Using this connection, reachability of a pointed coalgebra (Definition 5) can be expressed in terms of the least fixpoint $\text{lfp}$ of an operator defined in terms of $\Gamma$.

**Theorem 17.** Let $B: C \rightarrow C$ be a functor that has a base. A pointed $B$-coalgebra $(X, \gamma, x_0)$ is reachable iff $X \cong \text{lfp}(\Gamma \lor x_0)$.

This justifies defining the reachable part from an initial state $x_0$: $1 \rightarrow X$ as the least fixpoint of the monotone operator $\Gamma \lor x_0$. Standard means of computing the least fixpoint by iterating this operator then give us a way to compute this subcoalgebra. Further, $\Gamma$ provides a way to generalise the notion of “prefixed closedness” from Angluin’s L” algorithm to our categorical setting.

**Definition 18.** Let $s_0, s \in \text{Sub}(X)$ for some $X \in C$ and let $(X, \gamma)$ be a $B$-coalgebra. We call $s$ $s_0$-prefix closed w.r.t. $\gamma$ if $s = \bigvee_{i=0}^n s_i$ for some $n \geq 0$ and a collection $\{s_i \mid i = 1, \ldots, n\}$ with $s_{j+1} \leq \Gamma(\bigvee_{i=0}^j s_i)$ for all $j$ with $0 \leq j < n$.

### 6 Learning Algorithm

We define a general learning algorithm for $B$-coalgebras. First, we describe the setting, in general and slightly informal terms. The teacher has a pointed $B$-coalgebra $(X, \gamma, s_0)$. Our task is to ‘learn’ a pointed $B$-coalgebra $(S, \hat{\gamma}, \hat{s}_0)$ s.t.:

- $(S, \hat{\gamma}, \hat{s}_0)$ is correct w.r.t. the collection $\Phi$ of all tests, i.e., the theory of $(X, \gamma)$ and $(S, \hat{\gamma})$ coincide on the initial states $s_0$ and $\hat{s}_0$, (Definition 25);
- $(S, \hat{\gamma}, \hat{s}_0)$ is minimal w.r.t. logical equivalence;
- $(S, \hat{\gamma}, \hat{s}_0)$ is reachable.

The first point means that the learned coalgebra is ‘correct’, that is, it agrees with the coalgebra of the teacher on all possible tests from the initial state. For instance, in case of deterministic automata and their logic in Example 4, this just means that the language of the learned automaton is the correct one.

In the learning game, we are only provided limited access to the coalgebra $\gamma: X \rightarrow BX$. Concretely, the teacher gives us:
for any subobject \( S \hookrightarrow X \) and sub-formula closed subobject \( \Psi \) of \( \Phi \), the composite theory map \( S \xhookrightarrow{\theta} X \xrightarrow{\gamma} Q\Psi \);
- for \((S, \hat{\gamma}, s_0)\) a pointed coalgebra, whether or not it is correct w.r.t. the collection \( \Phi \) of all tests;
- in case of a negative answer to the previous question, a counterexample, which essentially is a subobject \( \Psi' \) of \( \Phi \) representing some tests on which the learned coalgebra is wrong (defined more precisely below);
- for a given subobject \( S \) of \( X \), the ‘next states’; formally, the computation of the \( B \)-base of the composite arrow \( S \xhookrightarrow{\gamma} X \xrightarrow{\gamma} BX \).

The first three points correspond respectively to the standard notions of membership query (‘filling in’ the table with rows \( S \) and columns \( \Psi \)), equivalence query and counterexample generation. The last point, about the base, is more unusual: it does not occur in the standard algorithm, since there a canonical choice of \((X, \gamma)\) is used, which allows to represent next states in a fixed manner. It is required in our setting of an arbitrary coalgebra \((X, \gamma)\).

In the remainder of this section, we describe the abstract learning algorithm and its correctness. First, we describe the basic ingredients needed for the algorithm: tables, closedness, counterexamples and a procedure to close a given table (Section 6.1). Based on these notions, the actual algorithm is presented (Section 6.2), followed by proofs of correctness and termination (Section 6.3).

**Assumption 19** Throughout this section, we assume
- that we deal with coalgebras over the base category \( C = \text{Set} \);
- a functor \( B: C \to C \) that preserves pre-images and wide intersections;
- a category \( D \) with an initial object \( 0 \) s.t. arrows with domain \( 0 \) are monic;
- a functor \( L: D \to D \) with an initial algebra \( L\Phi \xrightarrow{\sim} \Phi \);
- an adjunction \( P \dashv Q: C \leftrightarrow D^{\text{op}} \), and a logic \( \delta: LP \Rightarrow PB \).

Moreover, we assume a pointed \( B \)-coalgebra \((X, \gamma, s_0)\).

**Remark 20.** We restrict to \( C = \text{Set} \), but see it as a key contribution to state the algorithm in categorical terms: the assumptions cover a wide class of functors on \( \text{Set} \), which is the main direction of generalisation. Further, the categorical approach will enable future generalisations. The assumptions on the category \( C \) are: it is complete, well-powered and satisfies that for all (strong) epis \( q: S \to S' \in C \) and all monos \( i: S' \to S \) such that \( q \circ i \) is mono there is a morphism \( q^{-1}: S \to S' \) such that (i) \( q \circ q^{-1} = \text{id} \) and \( q^{-1} \circ q \circ i = i \).

### 6.1 Tables and counterexamples

**Definition 21.** A table is a pair \((S \hookrightarrow X, \Psi \hookrightarrow \Phi)\) consisting of a subobject \( S \) of \( X \) and a subformula-closed subobject \( i \) of \( \Phi \).
To make the notation a bit lighter, we sometimes refer to a table by \((S, \Psi)\), using \(s\) and \(i\) respectively to refer to the actual subobjects. The pair \((S, \Psi)\) represents ‘rows’ and ‘columns’ respectively, in the table; the ‘elements’ of the table are given abstractly by the map \(\text{th}_\gamma \circ s\). In particular, if \(\mathcal{C} = \mathcal{D} = \text{Set}\) and \(Q = 2^\omega\), then this is a map \(S \to 2^\Psi\), assigning a Boolean value to every pair of a row (state) and a column (formula).

For the definition of closedness, we use the operator \(\Gamma(S)\) from Definition 14 which characterises the successors of a subobject \(S \rightharpoonup X\).

**Definition 22.** A table \((S, \Psi)\) is closed if there exists a map \(k: \Gamma(S) \to S\) such that Diagram (9) commutes. A table \((S, \Psi)\) is sharp if the composite map \(S \xrightarrow{s} X \xrightarrow{\text{th}_\gamma} Q\Psi\) is monic.

Thus, a table \((S, \Psi)\) is closed if all the successors of states (elements of \(\Gamma(S)\)) are already represented in \(S\), up to equivalence w.r.t. the tests in \(\Psi\). In other terms, the rows corresponding to successors of existing rows are already in the table. Sharpness amounts to minimality w.r.t. logical equivalence: every row has a unique value. The latter will be an invariant of the algorithm (Theorem 32).

A conjecture is a coalgebra on \(S\), which is not quite a subcoalgebra of \(X\); instead, it is a subcoalgebra ‘up to equivalence w.r.t. \(\Psi\)’, that is, the successors agree up to logical equivalence.

**Definition 23.** Let \((S, \Psi)\) be a table. A coalgebra structure \(\hat{\gamma}: S \to BS\) is called a conjecture (for \((S, \Psi)\)) if Diagram (10) commutes.

It is essential to be able to construct a conjecture from a closed table. The following, stronger result is a variation of Proposition 16.

**Theorem 24.** A sharp table is closed iff there exists a conjecture for it. Moreover, if the table is sharp and \(B\) preserves monos, then this conjecture is unique.

Our goal is to learn a pointed coalgebra which is correct w.r.t. all formulas. To this aim we ensure correctness w.r.t. an increasing sequence of subformula closed collections \(\Psi\).

**Definition 25.** Let \((S, \Psi)\) be a table, and let \((S, \hat{\gamma}, \hat{s}_0)\) be a pointed \(B\)-coalgebra on \(S\). We say \((S, \hat{\gamma}, \hat{s}_0)\) is correct w.r.t. \(\Psi\) if Diagram (11) commutes.

All conjectures constructed during the learning algorithm will be correct w.r.t. the subformula closed collection \(\Psi\) of formulas under consideration.
Lemma 26. Suppose \((S, \Psi)\) is closed, and \(\hat{\gamma}\) is a conjecture. Then \(th_{\hat{\gamma}} \circ s = th_{\hat{\gamma}}^{\Psi} : S \to Q\Psi\). If \(\hat{s}_0 : 1 \to S\) satisfies \(s \circ \hat{s}_0 = s_0\) then \((S, \hat{\gamma}, \hat{s}_0)\) is correct w.r.t. \(\Psi\).

We next define the crucial notion of counterexample to a pointed coalgebra: a subobject \(\Psi'\) of \(\Psi\) on which it is ‘incorrect’.

Definition 27. Let \((S, \Psi)\) be a table, and let \((S, \hat{\gamma}, \hat{s}_0)\) be a pointed \(B\)-coalgebra on \(S\). Let \(\Psi'\) be a subformula closed subobject of \(\Phi\), such that \(\Psi\) is a subcoalgebra of \(\Psi'\). We say \(\Psi'\) is a counterexample (for \((S, \hat{\gamma}, \hat{s}_0)\), extending \(\Psi\)) if \((S, \hat{\gamma}, \hat{s}_0)\) is not correct w.r.t. \(\Psi'\).

The following elementary lemma states that if there are no more counterexamples for a coalgebra, then it is correct w.r.t. the object \(\Phi\) of all formulas.

Lemma 28. Let \((S, \Psi)\) be a table, and let \((S, \hat{\gamma}, \hat{s}_0)\) be a pointed \(B\)-coalgebra on \(S\). Suppose that there are no counterexamples for \((S, \hat{\gamma}, \hat{s}_0)\) extending \(\Psi\). Then \((S, \hat{\gamma}, \hat{s}_0)\) is correct w.r.t. \(\Phi\).

The following describes, for a given table, how to extend it with the successors (in \(X\)) of all states in \(S\). As we will see below, by repeatedly applying this construction, one eventually obtains a closed table.

Definition 29. Let \((S, \Psi)\) be a sharp table. Let \((S, q, r)\) be the \((\text{strong epi}, \text{mono})\)-factorisation of the map \(th_{\hat{\gamma}} \circ (s \lor \Gamma(s))\), as in the diagram:

\[
\begin{array}{ccc}
S \lor \Gamma(S) & \xrightarrow{\lor \Gamma(s)} & X \\
\downarrow{q} & & \downarrow{th_{\hat{\gamma}}} \\
\overline{S} & & Q\Psi \\
\end{array}
\]

We define \(\text{close}(S, \Psi) := \{ \overline{s} : \overline{S} \to X \mid th_{\hat{\gamma}} \circ \overline{s} = r, s \leq \overline{s} \leq s \lor \Gamma(s) \}\). For each \(\overline{s} \in \text{close}(S, \Psi)\) we have \(s \leq \overline{s}\) and thus \(s = \overline{s} \circ \kappa\) for some \(\kappa : S \to \overline{S}\).

Lemma 30. In Definition 29, for each \(\overline{s} \in \text{close}(S, \Psi)\), we have \(\kappa = q \circ \text{inl}_\lor\).

We will refer to \(\kappa = q \circ \text{inl}_\lor\) as the connecting map from \(s\) to \(\overline{s}\).

Lemma 31. In Definition 29, if there exists \(q^{-1} : \overline{S} \to S \lor \Gamma(S)\) such that \(q \circ q^{-1} = \text{id}\) and \(q^{-1} \circ q = \text{inl}_\lor\), then \(\text{close}(S, \Psi)\) is non-empty.

By our assumptions, the hypothesis of Lemma 31 is satisfied (Remark 20), hence \(\text{close}(S, \Psi)\) is non-empty. It is precisely (and only) at this point that we need the strong condition about existence of right inverses to epimorphisms.

6.2 The algorithm

Having defined closedness, counterexamples and a procedure for closing a table, we are ready to define the abstract algorithm. In the algorithm, the teacher has access to a function \(\text{counter}((S, \hat{\gamma}, \hat{s}_0), \Psi)\), which returns the set of all counterexamples (extending \(\Psi\)) for the conjecture \((S, \hat{\gamma}, \hat{s}_0)\). If this set is empty, the
coalgebra \((S, \hat{\gamma}, \hat{s}_0)\) is correct (see Lemma 28), otherwise the teacher picks one of its elements \(\Psi'\). We also make use of \(\text{close}(S, \Psi)\), as given in Definition 29.

**Algorithm 1** Abstract learning algorithm

1: \((S \xrightarrow{a} X) \leftarrow (1 \xrightarrow{a} X)
2: \hat{s}_0 \leftarrow \text{id}_1,
3: \Psi \leftarrow 0
4: \text{while true do}
5: \quad \text{while } (S \xrightarrow{a} X, \Psi) \text{ is not closed do}
6: \quad \quad \text{let } (S \xrightarrow{a} X) \in \text{close}(S, \Psi), \text{ with connecting map } \kappa: S \xrightarrow{a} \overline{S}
7: \quad \quad (S \xrightarrow{a} X) \leftarrow (S \xrightarrow{a} X)
8: \quad \quad \hat{s}_0 \leftarrow \kappa \circ \hat{s}_0
9: \quad \text{end while}
10: \quad \text{let } (S, \hat{\gamma}) \text{ be a conjecture for } (S, \Psi)
11: \quad \text{if } \text{counter}((S, \hat{\gamma}, \hat{s}_0), \Psi) = \emptyset \text{ then}
12: \quad \quad \text{return } (S, \hat{\gamma}, \hat{s}_0)
13: \quad \text{else}
14: \quad \quad \Psi \leftarrow \Psi' \text{ for some } \Psi' \in \text{counter}((S, \hat{\gamma}, \hat{s}_0), \Psi)
15: \quad \text{end if}
16: \text{end while}

The algorithm takes as input the coalgebra \((X, \gamma, s_0)\) (which we fixed throughout this section). In every iteration of the outside loop, the table is first closed by repeatedly applying the procedure in Definition 29. Then, if the conjecture corresponding to the closed table is correct, the algorithm returns it (Line 12). Otherwise, a counterexample is chosen (Line 14), and the algorithm continues.

### 6.3 Correctness and Termination

Correctness is stated in Theorem 33. It relies on establishing loop invariants:

**Theorem 32.** The following is an invariant of both loops in Algorithm 32.

1. \((S, \Psi)\) is sharp.
2. \(s \circ \hat{s}_0 = s_0\), and
3. \(s\) is \(s_0\)-prefix closed w.r.t. \(\gamma\).

**Theorem 33.** If Algorithm 32 terminates, then it returns a pointed coalgebra \((S, \hat{\gamma}, \hat{s}_0)\) which is minimal w.r.t. logical equivalence, reachable and correct w.r.t. \(\Phi\).

In our termination arguments, we have to make an assumption about the coalgebra which is to be learned. It does not need to be finite itself, but it should be finite up to logical equivalence—in the case of deterministic automata, for instance, this means the teacher has a (possibly infinite) automaton representing a regular language. To speak about this precisely, let \(\Psi\) be a subobject of \(\Phi\). We take a (strong epi, mono)-factorisation of the theory map, i.e.,
Theorem 34. Let \((S, \Psi)\) be a sharp table, let \(S \in \text{close}(S, \Psi)\) and let \(\kappa : S \to S\) be the connecting map. If \(\kappa\) is an isomorphism, then \((S, \Psi)\) is closed.

Lemma 35. Consider a sequence of sharp tables \((S_i \xrightarrow{s_i} X, \Psi)\) such that \(s_{i+1} \in \text{close}(S_i, \Psi)\) for all \(i\). Moreover, let \((\kappa_i : S_i \to S_{i+1})\) be the connecting maps (Definition 29). If the logical quotient \(|X|_\Phi\) has finitely many subobjects, then \(\kappa_i\) is an isomorphism for some \(i \in \mathbb{N}\).

Corollary 36. If the \(\Phi\)-logical quotient \(|X|_\Phi\) has finitely many subobjects, then the inner while loop of Algorithm 1 terminates.

For the outer loop, we assume that \(|X|_\Phi\) has finitely many quotients, ensuring that every sequence of counterexamples proposed by the teacher is finite.

Theorem 37. If the \(\Phi\)-logical quotient \(|X|_\Phi\) has finitely many quotients and finitely many subobjects, then Algorithm 1 terminates.

7 Future Work

We showed how duality plays a natural role in automata learning, through the central connection between states and tests. Based on this foundation, we proved correctness and termination of an abstract algorithm for coalgebra learning. The generality is not so much in the base category (which, for the algorithm, we take to be \(\text{Set}\)) but rather in the functor used; we only require a few mild conditions on the functor, and make no assumptions about its shape. The approach is thus considered coalgebra learning rather than automata learning.

Returning to automata, an interesting direction is to extend the present work to cover learning of, e.g., non-deterministic or alternating automata \([9,15]\) for a regular language. This would require explicitly handling branching in the type of coalgebra. One promising direction would be to incorporate the forgetful logics of \([19]\), which are defined within the same framework of coalgebraic logic as the current work. It is not difficult to define in this setting what it means for a table to be closed ‘up to the branching part’, stating, e.g., that even though the table is not closed, all the successors of rows are present as combinations of other rows.

Another approach would be to integrate monads into our framework, which are also used to handle branching within the theory of coalgebras \([15]\). It is an intriguing question whether the current approach, which allows to move beyond automata-like examples, can be combined with the CALF framework \([13]\), which is very far in handling branching occurring in various kinds of automata.
Acknowledgments. We are grateful to Joshua Moerman, Nick Bezhanishvili, Gerco van Heerdt, Aleks Kissinger and Stefan Milius for valuable discussions and suggestions.

References

1. Jiří Adámek, Dominik Lücke, and Stefan Milius. Recursive coalgebras of finitary functors. ITA, 41(4):447–462, 2007.
2. Jiří Adámek, Stefan Milius, Lawrence S. Moss, and Lurdes Sousa. Well-pointed coalgebras. *Logical Methods in Computer Science*, 9(3), 2013.
3. Jiří Adámek and Jiří Rosický. *Locally Presentable and Accessible Categories*. Cambridge Tracts in Mathematics. Cambridge University Press, 1994.
4. Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.
5. Dana Angluin, Sarah Eisenstat, and Dana Fisman. Learning regular languages via alternating automata. In Qiang Yang and Michael Wooldridge, editors, *IJCAI 2015*, pages 3308–3314. AAAI Press, 2015.
6. Simone Barlocco and Clemens Kupke. Angluin learning via logic. In Sergei N. Artemov and Anil Nerode, editors, *LFCS 2018, Proceedings*, pages 72–90, 2018.
7. Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Number 53 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.
8. Alwin Blok. Interaction, observation and denotation. Master’s thesis, ILLC Amsterdam, 2012.
9. Benedikt Bollig, Peter Habermehl, Carsten Kern, and Martin Leucker. Angluin-style learning of NFA. In Craig Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence 2009*, pages 1004–1009, 2009.
10. Francis Borceux. *Handbook of Categorical Algebra*, volume 1 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1994.
11. Gerco van Heerdt. An abstract automata learning framework. Master’s thesis, Radboud Universiteit Nijmegen, 2016.
12. Gerco van Heerdt, Matteo Sammartino, and Alexandra Silva. CALF: categorical automata learning framework. In Valentin Goranko and Mads Dam, editors, *26th EACSL Annual Conference on Computer Science Logic, CSL 2017*, volume 82 of *LIPIcs*, pages 29:1–29:24. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
13. Gerco van Heerdt, Matteo Sammartino, and Alexandra Silva. Learning automata with side-effects. *CoRR*, abs/1704.08055, 2017.
14. Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*, volume 59 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2016.
15. Bart Jacobs and Alexandra Silva. Automata learning: A categorical perspective. In Franck van Breugel, Elham Kashefi, Catuscia Palamidessi, and Jan Rutten, editors, *Horizons of the Mind. A Tribute to Prakash Panangaden - Essays Dedicated to Prakash Panangaden on the Occasion of His 60th Birthday*, volume 8464 of *LNCS*, pages 384–406. Springer, 2014.
16. Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. *J. Comput. Syst. Sci.*, 81(5):859–879, 2015.
17. Bart Jacobs and Ana Sokolova. Exemplaric Expressivity of Modal Logics. *Journal of Logic and Computation*, 20(5):1041–1068, 2009.

18. Bartek Klin. Coalgebraic modal logic beyond sets. *Electr. Notes Theor. Comput. Sci.*, 173:177–201, 2007.

19. Bartek Klin and Jurriaan Rot. Coalgebraic trace semantics via forgetful logics. *Logical Methods in Computer Science*, 12(4), 2016.

20. Clemens Kupke, Alexander Kurz, and Dirk Pattinson. Algebraic semantics for coalgebraic logics. *Electr. Notes Theor. Comput. Sci.*, 106:219–241, 2004.

21. Clemens Kupke and Dirk Pattinson. Coalgebraic semantics of modal logics: An overview. *Theor. Comput. Sci.*, 412(38):5070–5094, 2011.

22. Oded Maler and Amir Pnueli. On the learnability of infinitary regular sets. *Inf. Comput.*, 118(2):316–326, 1995.

23. Gerhard Osius. Categorical set theory: A characterization of the category of sets. *Journal of Pure and Applied Algebra*, 4(1):79 – 119, 1974.

24. Dusko Pavlovic, Michael W. Mislove, and James Worrell. Testing semantics: Connecting processes and process logics. In Michael Johnson and Varmo Vene, editors, *Algebraic Methodology and Software Technology, 11th International Conference, AMAST 2006, Proceedings*, volume 4019 of *LNCS*, pages 308–322. Springer, 2006.

25. Jan J. M. Rutten. Automata and coinduction (an exercise in coalgebra). In Davide Sangiorgi and Robert de Simone, editors, *CONCUR '98: Concurrency Theory, 9th International Conference, Proceedings*, volume 1466 of *LNCS*, pages 194–218. Springer, 1998.

26. Jan J. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000.

27. Lutz Schröder. Expressivity of coalgebraic modal logic: The limits and beyond. *Theor. Comput. Sci.*, 390(2-3):230–247, 2008.

28. Paul Taylor. *Practical Foundations of Mathematics*. Cambridge University Press, 1999.

29. Věra Trnková. On descriptive classification of set-functors. i. *Commentationes Mathematicae Universitatis Carolinae*, 12(1):143–174, 1971.

30. Frits W. Vaandrager. Model learning. *Commun. ACM*, 60(2):86–95, 2017.
A Proofs of Section 5

Proof (Proof of Proposition 12). Let \( f: X \rightarrow B(Y) \). Consider the collection of all pairs of maps \( g_k: X \rightarrow B(U_k), m_k: U_k \rightarrow Y \) such that \( B(m_k) \circ g_k = f_k \) and \( m_k \) is a subobject, indexed by \( k \in K \). Let \( m: \bigwedge\{m_k\}_{k \in K} \rightarrow Y \) be the intersection of all the \( m_k \) – this is a (small) set since \( \mathcal{C} \) is well-powered. We abbreviate \( \bigwedge\{m_k\}_{k \in K} \) by \( I \).

Since \( B \) preserves intersections, \( B(m): B(I) \rightarrow B(Y) \) is the intersection of all the subobjects \( B(m_k) \). Now the \( g_k \)'s form a cone over the \( B(m_k) \)'s, so we get a unique \( g: X \rightarrow B(I) \) from the universal property of the pullback \( B(I) \).

We claim that \((I, g, m)\) is the base of \( f \). To see this, first of all, note that \( i \) is mono, and \( B(m) \circ g = f \) by definition of \( m \) and \( g \). Further, if there is any \( g': X \rightarrow B(U), m': U \rightarrow Y \) with \( B(m') \circ g' = f \) and \( m' \) monic then it is (up to isomorphism) one of the \( g_k,m_k \) pairs. Hence, there is the map \( x_k: I \rightarrow U_k \) in the limiting cone, i.e., \( m_k \circ x_k = i \), and we have \( B(x_k) \circ g = g_k \). Finally \( x_k \) is unique among such maps, since \( B \) preserves monos (as it preserves intersections).

Proof (Proof of Lemma 13). By our assumption on \( Z \) there exists a morphism \( g: S \rightarrow BZ \) such that \( Bm \circ g = f \). Therefore we have \( Bm' \circ Be \circ g = Bh \circ f \) which shows that \( m' \) is a candidate for the base of \( Bh \circ f \). We still need to check the universal property of the base. To this aim let \( g': S \rightarrow BU \) and \( n: U \rightarrow Y \) be the base of \( Bh \circ f \): 

\[
\begin{array}{ccc}
S & \xrightarrow{g} & BZ \\
\downarrow & & \downarrow \leftarrow \\
BZ & \xleftarrow{g'} & BU
\end{array}
\]

\[
\begin{array}{ccc}
& & BX \\
& \xrightarrow{Bm} & \\
& \downarrow & \downarrow Bm' \\
& \downarrow & \downarrow Bf \\
BY & \xrightarrow{Bh} & BW
\end{array}
\]

By the universal property of the base there is a morphism \( j: U \rightarrow W \) making the lower right diagram commute. Now, consider the following pullback:

\[
\begin{array}{ccc}
P & \xrightarrow{p_n} & X \\
\downarrow & & \downarrow \leftarrow \\
U & \xrightarrow{n} & Y
\end{array}
\]

This is a preimage because \( n \) is mono and by assumption on \( B \) we have that this pullback is preserved under application of \( B \). \( S \) forms a cone over the diagram
with $S \to BX, S \to BU$. So there exists a map from $S$ to the pullback.

$p_n$ is mono, so $Bp_n \circ w$ is a base factorization. So, we get an arrow from $BZ$ to $BP$, i.e., $k : Z \to P$ such that:

(i) $Bk \circ g = w$
(ii) $p_n \circ k = m$

Consider the following diagram. According to the diagonal filling property, we have $\exists d : W \to U$ such that (i) $n \circ d = m'$ and (ii) $d \circ e = p_h \circ k$.

By the universal property of the base we have $d \circ j = id_U$. Moreover, $m' \circ j \circ d = n \circ d = m'$, and because $m'$ is monic we have $j \circ d = id_W$.

**Lemma 38.** If $\mathcal{C}$ is complete and well-powered, then for each $X \in \mathcal{C}$, we have $\text{Sub}(X)$ has arbitrary meets. Consequently, $\text{Sub}(X)$ is a complete lattice.

**Proof (Proof of Lemma 38).** Consider some $X \in \mathcal{C}$ and an arbitrary family of subobjects $\{m_i : S_i \to X\}_{i \in I}$. Let $P$ be the pullback with pullback maps $\{p_i : P \to S_i\}_{i \in I}$. As the $m_i$ are mono, the $p_i$ are mono as well, so let define $p := m_i \circ p_i : P \to X \in \text{Sub}(X)$. Obviously, we have $p \leq m_i \forall i \in I$. So, $P$ is a lower bound. To see that $P$ is the greatest lower bound, consider an arbitrary $P'$ with $p' : P' \to X$ that is a lower bound of the same family of subobjects. By definition of lower bound we have for each $i \in I$ a map $p'_i : P' \to S_i$ s.t. $m_i \circ p'_i = p$. By universal property of the (wide) pullback, there exists a unique map $c : P' \to P$ s.t. $p \circ c = p'$, i.e., $p' \leq p$. As $P'$ was an arbitrary lower bound, we showed that $P$ is the greatest lower bound. This finishes the proof of the fact that $\text{Sub}(X)$ has arbitrary meets. Completeness of the lattice can now be proven in a standard way by defining the join of an arbitrary collection of subobjects as the meet of all upper bounds of this collection.
Proof (Proof of Lemma \[15\]). It is obvious that $\Gamma$ is well-defined. To check monotonicity, we consider the following diagram for subobjects $s : S \to X$ and $s' : S' \to X$ such that $s \leq s'$.

![Diagram](image)

Here $j$ exists by the definition of $\leq$ and $h$ exists by the universal property of the base of $\gamma \circ s$. Therefore we have $\Gamma(s) \leq \Gamma(s')$ as required.

Proof (Proof of Proposition \[16\]).

$\Rightarrow$ Consider the following diagram

![Diagram](image)

As $s$ is a subcoalgebra there exists $\sigma : S \to BS$ s.t. the outer square commutes. By the universal property of the base there exists $j : \Gamma(S) \to S$ s.t. $s \circ j = \Gamma(s)$. In other words, $\Gamma(s) \leq s$ as required.

$\Leftarrow$ By assumption there exists a $j : \Gamma(S) \to S$ such that $s \circ j = \Gamma(s)$. We define a $B$-coalgebra structure on $S$ by putting $\sigma := j \circ e$. We have to show that the outer square in the above diagram commutes, but this is easy to show because the inner square commutes by definition of the base, the left triangle commutes by definition of $\sigma$ and the right one by assumption on $j$.

Finally, that the collection of subcoalgebras of $(X, \gamma)$ forms a complete lattice is now a direct consequence of the fact that the collection of pre-fixpoints of the monotone operator $\Gamma$ forms a complete lattice (Knaster-Tarski theorem).
Proof (Proof of Theorem 22). Suppose $X \cong \text{lfp}(\Gamma \lor x_0)$, and let $s : S \to X$ be a subcoalgebra together with an arrow $s_0 : 1 \to S$ with $x_0 = s \circ s_0$. The latter implies that $x_0 \leq s$. Further, since $S$ is a subcoalgebra, by Proposition 16 we get $\Gamma(s) \leq s$. Hence $\Gamma(s) \lor x_0 \leq s$, i.e., $s$ is a pre-fixed point of $\Gamma \lor x_0$. By the Knaster-Tarski theorem (using that $\text{Sub}(X)$ is a complete lattice), $\text{lfp}(\Gamma \lor x_0)$ is the least pre-fixed point, so it now suffices to prove that $s \leq \text{lfp}(\Gamma \lor x_0)$. But this follows easily, since $s$ is a subobject of $X$ and $X \cong \text{lfp}(\Gamma \lor x_0)$.

Conversely, suppose $(X, \gamma, x_0)$ is reachable. We have that $\Gamma(\text{lfp}(\Gamma \lor x_0)) \leq \Gamma(\text{lfp}(\Gamma \lor x_0) \lor x_0) = \text{lfp}(\Gamma \lor x_0)$, so by Proposition 16 $\text{lfp}(\Gamma \lor x_0)$ is a subcoalgebra of $(X, \gamma)$. Moreover, we have $x_0 \leq \Gamma(\text{lfp}(\Gamma \lor x_0) \lor x_0) = \text{lfp}(\Gamma \lor x_0)$, so there exists a map $s_0 : 1 \to \text{lfp}(\Gamma \lor x_0)$ such that $x_0 = s_0 \circ s_0$, where $s : \text{lfp}(\Gamma \lor x_0) \to X$ is the inclusion. Hence, by definition of reachability, we get that $s$ is an isomorphism. 

\[ \square \]

B Proofs of Section 6

Proof (Proof of Theorem 24). Given a table $(S, \Psi)$ that is closed, it is straightforward to construct a conjecture $\hat{\gamma}$ as composite of $g : S \to B\Gamma(S)$ and the arrow $Bk : B\Gamma(S) \to S$, where $g$ is part of the base $(\Gamma(S), g, \Gamma(s))$ of $\gamma \circ s$ and $k : \Gamma(S) \to S$ is the morphism that exists by closedness of $(S, \Psi)$.

For the converse, consider a conjecture $(\hat{S}, \hat{\gamma})$ for a sharp table $(S, \Psi)$, let $(\Gamma(S), g, \Gamma(s))$ be the base of $\gamma \circ s$ and let $(h : \Gamma(S) \to Y, m : Y \to Q\Psi)$ be the factorisation of $\hat{\gamma} \circ \Gamma(s)$. By Lemma 13 as $h$ is epi, we have that $(Y, B\hat{h} \circ g, m)$ is the base of $B\hat{h} \circ \gamma \circ s$. The situation is depicted in the (commuting) upper square of the diagram below.

\[ \begin{array}{ccc}
S & \xrightarrow{s} & X \\
 & & \xrightarrow{\gamma} \\
 & & BX \\
 & & \xrightarrow{B\hat{h}} BQ(\Psi)
\end{array} \begin{array}{c}
\begin{array}{c}
\downarrow Bm \\
B(\Gamma(S)) \\
BS
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
\downarrow Bg \\
B(Y) \\
BX
\end{array}
\end{array} \]

The bigger outer square commutes by the fact that $\gamma$ is assumed to be a conjecture. As the table is sharp, we have $\hat{\gamma} \circ s$ is mono. Therefore the universal property of the base yields existence of a morphism $j : Y \to S$ such that $\hat{\gamma} \circ s \circ j = m$. We define $k := j \circ h$ and claim that this $k$ is a witness for $(S, \Psi)$, i.e., that $k$ makes the relevant diagram from Definition 22 commute. To see this, we calculate:

\[ th^\gamma \circ \Gamma(s) = m \circ h = th^\gamma \circ s \circ j \circ h = th^\gamma \circ s \circ k. \]

This finishes the proof. 

\[ \square \]
Proof (Proof of Lemma 26). The map $\hat{\gamma}$ is, by definition, the unique map making the following diagram commute.

\[
\begin{array}{ccc}
S & \xrightarrow{\gamma} & Q\Psi \\
\downarrow & & \uparrow d \\
BS & \xrightarrow{B\hat{\gamma}} & BQ\Psi \\
& \xrightarrow{B\theta} & QL\Psi \\
\end{array}
\]

where $d : \Psi \to L\Psi$ is the coalgebra structure from subformula closedness of $\Psi$. Consider the following diagram:

\[
\begin{array}{ccc}
S & \xrightarrow{s} & X \\
\downarrow & & \uparrow d \\
BS & \xrightarrow{ Bs } & BX \\
& \xrightarrow{B\gamma} & BQ\Psi \\
& \xrightarrow{B\theta} & QL\Psi \\
\end{array}
\]

The rectangle on the right commutes by definition of $\hat{\gamma}$. Together with $\hat{\gamma}$ being a conjecture, it follows that the outside of the diagram commutes. Since $\hat{\gamma}_{\Phi}$ is the unique such map, we have $\hat{\gamma}_{\Phi} \circ s = \hat{\gamma}_{\Phi}$. \hfill $\square$

Proof (Proof of Lemma 28). If there is no counterexample, then in particular $\Phi$ is not a counterexample. The object $\Phi$ is subformula-closed subobject of itself, and $\Psi$ is a subcoalgebra of $\Phi$. Hence, by definition of counterexamples, it must be the case that $(S, \hat{\gamma}, \hat{s}_0)$ correct w.r.t. $\Phi$. \hfill $\square$

Proof (Proof of Lemma 30). Let $\pi \in \text{closed}(S, \Psi)$. We calculate:

\[
\begin{align*}
th^\gamma \circ \pi \circ \kappa &= th^\gamma \circ s = th^\gamma \circ (s \vee \Gamma(s)) \circ \text{inl}_\vee = r \circ q \circ \text{inl}_\vee \\
&= th^\gamma \circ \pi \circ q \circ \text{inl}_\vee
\end{align*}
\]

which implies $\kappa = q \circ \text{inl}_\vee$ as $r = th^\gamma \circ \pi$ is a mono.

Proof (Proof of Lemma 31). Given the assumption of the lemma we are able to define a morphism $\pi : S \to X$ by putting $\pi := (s \vee \Gamma(s)) \circ q^{-1}$. Obviously $\pi$ is a mono as it is defined as composition of monos. Furthermore, by definition, we have $\pi \leq s \vee \Gamma(s)$. To see that $s \leq \pi$, we calculate

\[
\pi \circ q \circ \text{inl}_\vee = (s \vee \Gamma(s)) \circ q^{-1} \circ q \circ \text{inl}_\vee = (s \vee \Gamma(s)) \circ \text{inl}_\vee = s.
\]

Finally, the condition concerning the theory map also follows easily:

\[
th^\gamma \circ \pi = th^\gamma \circ (s \vee \Gamma(s)) \circ q^{-1} = r \circ q \circ q^{-1} = r.
\]

This finishes the proof of the lemma.

We need a few auxiliary lemma’s in the proofs below.
Lemma 39. If \((S, \Psi)\) is sharp, then \((\overline{S} \rightarrow X, \Psi)\) is sharp for any \(\overline{\pi} \in \text{close}(S, \Psi)\).

Proof. This follows immediately from Definition 29, since \(\text{th}^\gamma \circ \overline{\pi} = r\), where \(r\) is monic. \(\square\)

Lemma 40. Let \(\Psi\) and \(\Psi'\) be subformula closed, with \(\Psi\) a subcoalgebra of \(\Psi'\), witnessed by a mono \(i: \Psi \rightarrow \Psi'\). Then we have \(Q_i \circ \text{th}^\gamma_{\Psi'} = \text{th}^\gamma_{\Psi}\).

Proof (Proof of Lemma 40). Let \(\sigma': \Psi' \rightarrow L\Psi'\) and \(\sigma: \Psi \rightarrow L\Psi\) be the coalgebra structures from subformula closedness of \(\Psi'\) and \(\Psi\) respectively. Consider the following diagram.

\[
\begin{array}{ccc}
X & \xrightarrow{\gamma} & Q(\Psi') \\
\downarrow & & \downarrow Q_i \\
BX & \xrightarrow{\text{th}^\gamma_{\Psi'}} & BQ(\Psi') \\
\downarrow BQ_i & & \downarrow \text{th}^\gamma_{\Psi'} \\
BQ\Psi & & BQ\Psi
\end{array}
\]

By definition, \(\text{th}^\gamma_{\Psi'}\) is the unique map making the left rectangle commute. The (right) square commutes by assumption that \(i\) is a coalgebra homomorphism, and the (lower) triangle by naturality. Since \(\text{th}^\gamma_{\Psi'}\) is the unique map such that \(\text{th}^\gamma_{\Psi'} = Q\sigma \circ \rho'_{\Psi'} \circ \text{th}^\gamma_{\Psi} \circ \gamma\), we have \(\text{th}^\gamma_{\Psi'} = Q_i \circ \text{th}^\gamma_{\Psi}\). \(\square\)

Lemma 41. Suppose \((S, \Psi)\) is sharp, and \(\Psi'\) is a counterexample. Then \((S, \Psi')\) is again sharp.

Proof. Let \(i: \Psi \rightarrow \Psi'\) be the inclusion of the coalgebra \((\Psi, \sigma)\) into \((\Psi', \sigma')\). By Lemma 10 we have \(Q_i \circ \text{th}^\gamma_{\Psi} = \text{th}^\gamma_{\Psi'}\). Hence \(Q_i \circ \text{th}^\gamma_{\Psi} \circ s = \text{th}^\gamma_{\Psi'} \circ s\), and since \(\text{th}^\gamma_{\Psi'} \circ s\) is monic, it follows that \(\text{th}^\gamma_{\Psi} \circ s\) is monic. \(\square\)

Proof (Proof of Theorem 32). To show that each of these is an invariant of both loops, it suffices to prove that they hold at once we enter the first iteration of the outer loop, and that both loops preserve them (that it holds at the start of each first iteration of the inner loop then follows).

1. (Holds at entry of the outer loop.) At this point, \((S, \Psi) = (S_0, 0)\). Since \(Q\) is a right adjoint, it maps 0 to the terminal object \(Q0 = 1\) of \(C\). Hence, the map from \(S_0 = 1\) to \(Q\Psi\) is of the form \(1 \xrightarrow{s_0} X \xrightarrow{\gamma} Q\Psi = 1\) which is an iso, so in particular monic.

(Preserved by the inner loop.) This follows from Lemma 39.

(Provided by the loop body.) If \((S, \Psi)\) is sharp on entry of the body of the outer loop, and the inner loop terminates, then \((S, \Psi)\) is again sharp at Line 10. It only remains to show that if \(\Psi'\) is a counterexample (extending \(\Psi'\)) for a conjecture for \((S, \Psi)\), then \((S, \Psi')\) is sharp. This follows, in turn, from Lemma 41.
2. (Holds at entry of the outer loop.) Follows immediately from the first two lines of the algorithm.

(Preserved by the inner loop.) Suppose \( \hat{s}_0 \circ s = s_0 \), and let \( \overline{s} \in \text{close}(S, \Psi) \).

We need to prove that \( \overline{s}_0 \circ \kappa \circ \hat{s}_0 = s_0 \) where \( \kappa : S \to \overline{s} \) is the connecting map.

Indeed, we have \( \overline{s}_0 \circ \kappa \circ \hat{s}_0 = s \circ \hat{s}_0 = s_0 \), by definition of \( \kappa \) and assumption, respectively.

(Preserved by the outer loop.) This follows immediately from preservation by the inner loop.

3. Clearly the initial configuration \((s_0, 0)\) is \( s_0 \)-prefix closed. Suppose now that \((S, \Psi)\) is a table with \( s \) being \( s_0 \)-prefix closed. We need to check that any \( \overline{s} \in \text{close}(S, \Psi) \) is \( s_0 \)-prefix closed as well. By assumption on \((S, \Psi)\) we have \( s = \bigvee_{i=0}^n s_i \) for a suitable family of subobjects \( s_0, \ldots, s_n \). Let \( \overline{s} \in \text{close}(S, \Psi) \).

Then by definition we have \( \overline{s} \leq s \lor \Gamma(s) \), so we put \( s_{n+1} := \Gamma(s) \land \overline{s} \). It is then easy to check that \( \overline{s} \) is \( s_0 \)-prefix closed:

\[
\bigvee_{i=0}^{n+1} s_i = \bigvee_{i=0}^n s_i \lor s_{n+1} = \overline{s} \lor (\Gamma(s) \land \overline{s}) = (s \lor \Gamma(s)) \land (s \lor \overline{s}) = \overline{s}
\]

where the last equality follows from \( s \leq \overline{s} \leq s \lor \Gamma(s) \). By definition we have \( s_{n+1} \leq \Gamma(s) = \bigvee_{i=0}^n s_i \) as required.

**Proof (Proof of Theorem 24).** Minimality w.r.t logical equivalence follows from the fact that sharpness of the table is maintained throughout. As the algorithm terminated there is no counterexample, which means by Lemma 28 that the coalgebra is correct w.r.t. \( \Phi \). For reachability we show that the pointed coalgebra that is returned by the algorithm is reachable by showing that any conjecture that is constructed during the run of the algorithm is reachable. While running the algorithm we will only encounter conjectures that are built from tables that are both sharp and closed. Therefore we consider an arbitrary sharp and closed table \((S, \Psi)\) together with the conjecture \((S, \hat{\gamma})\) that exists according to Theorem 24. We are going to prove that \((S, \hat{\gamma}, \hat{s}_0)\) is reachable.

By Theorem 22 we know that \((S, \Psi)\) is \( s_0 \)-prefix closed. This means that \( s = \bigvee_{i=0}^n s_i \) for suitable subobjects \( s_1, \ldots, s_n \in \text{Sub}(X) \). Suppose now that \((\overline{s}, \overline{\gamma}, \overline{s}_0)\) is a subcoalgebra of \((S, \hat{\gamma}, s_0)\) with inclusion \( j : \overline{s} \to S \) such that \( j \circ s_0 = \hat{s}_0 \). We prove by induction on \( i \) that \( s_i \leq \overline{s} \) for all \( i \in \{0, \ldots, n\} \) and thus \( s \leq \overline{s} \) - this will imply \( s = \overline{s} \) and thus, as \( \overline{s} \) was assumed to be an arbitrary (pointed) subcoalgebra, reachability of \((S, \hat{\gamma}, s_0)\).

**Case** \( i = 0 \). Then \( \overline{s} \circ \overline{s}_0 = s \circ j \circ \overline{s}_0 = s \circ \hat{s}_0 = s_0 \) and thus \( s_0 \leq \overline{s} \) as required.

**Case** \( i = j + 1 \). Then

\[
\begin{align*}
th_\overline{\gamma} \circ s_{j+1} & \leq th_\overline{\gamma} \circ \Gamma(\bigvee_{i=0}^j s_j) \overset{\text{I.H.}}{\leq} th_\overline{\gamma} \circ \Gamma(\overline{s}) \overset{\text{Thm.22}}{\leq} \overline{s}
\end{align*}
\]

where we slightly abuse notation by writing \( f \leq g \) for arbitrary morphisms \( f : X_1 \to Y \) and \( g : X_2 \to Y \). There exists a morphism \( m : X_1 \to X_2 \) such that
\( g \circ m = f \). The inequality implies that there is a map \( k_{j+1} : S_i \to S \) such that 
\[
\theta^\gamma \circ \pi \circ k_{j+1} = \theta^\gamma \circ s_{j+1}.
\]
This implies
\[
\theta^\gamma \circ s \circ j \circ k_{j+1} = \theta^\gamma \circ s_{j+1} \tag{13}
\]
On the other hand, we have 
\[
s \circ \text{in}_{j+1} = s_{j+1}
\]
where \( \text{in}_{j+1} = s_{j+1} \) denotes the inclusion of \( s_{j+1} \) into \( s \). Therefore we have 
\[
\theta^\gamma \circ s \circ \text{in}_{j+1} = \theta^\gamma \circ s_{j+1}.
\]
Together with (13) this implies 
\[
\theta^\gamma \circ s \circ \text{in}_{j+1} = \theta^\gamma \circ s \circ j \circ k_{j+1}.
\]
By sharpness of the table \( s \) we obtain \( \text{in}_{j+1} = j \circ k_{j+1} \) and finally 
\[
s_{j+1} = s \circ \text{in}_{j+1} = s \circ j \circ k_{j+1} = s' \circ k_{j+1}
\]
which shows that \( s_{j+1} \leq s' \). This finishes the induction proof.

**Proof (Proof of Theorem 34).** Suppose \( \kappa \) is an isomorphism. Consider the following diagram (using the notation from Definition 29), where 
\( g : S \to BX \) is the map which forms the base of \( \gamma \circ s \) together with \( \Gamma(s) : \Gamma(S) \to X \).

The inner shapes commute, from top to bottom: (1) by definition the base, (2) by definition of \( \text{inr}_\vee \), (3) by definition of \((q, r)\); for the bottom triangle (4), we have 
\[
\theta^\gamma \circ s \overset{\text{Def. of } \kappa}{=} \theta^\gamma \circ \pi \circ \kappa \overset{\text{in char}(S, \Psi)}{=} r \circ \kappa
\]
which suffices since \( \kappa \) is an iso. Since the entire diagram commutes, the coalgebra structure on \( S \) gives a conjecture for \((S, \Psi)\). Hence, by Theorem 24 the table is closed.

**Proof (Proof of Lemma 35).** First, observe that the \( S_i \)’s form an increasing chain of subobjects of \( X \). Since all these tables \((S_i, \Psi)\) are sharp, they give rise to an increasing chain of subobjects of \( Q(\Psi) \), by composition with \( \theta^\gamma \), given by 
\[
\theta^\gamma \circ s_i : S_i \to Q(\Psi).
\]
By Lemma 40 it follows that each \( \theta^\gamma \circ s_i : S_i \to Q(\Psi) \) is
monic, and we obtain a sequence of subobjects of $\Phi$:

$$
\begin{align*}
S_0 & \xrightarrow{\kappa_0} S_1 \xrightarrow{\kappa_1} S_2 \xrightarrow{\kappa_2} \ldots \\
\downarrow s_0 & \downarrow s_1 \downarrow s_2 & \quad \downarrow X & \downarrow X & \downarrow X \\
X & \xrightarrow{t_\Phi} X & \xrightarrow{t_\Phi} X & \xrightarrow{t_\Phi} Q\Phi
\end{align*}
$$

It follows that this induces a chain of subobjects of $|X|_\Phi$:

$$
\begin{align*}
S_0 & \xrightarrow{\kappa_0} S_1 \xrightarrow{\kappa_1} S_2 \xrightarrow{\kappa_2} \ldots \\
\downarrow s_0 & \downarrow s_1 \downarrow s_2 & \quad \downarrow |X|_\Phi & \downarrow |X|_\Phi & \downarrow |X|_\Phi \\
X & \xrightarrow{e_\Phi} X & \xrightarrow{e_\Phi} X & \xrightarrow{e_\Phi} |X|_\Phi
\end{align*}
$$

By assumption, $|X|_\Phi$ has finitely many subobjects, so $\kappa_i$ must be an isomorphism for some $i$. \qed

**Proof (Proof of Corollary 36).** The while loop computes a chain of subobjects of $X$ as in Lemma 35; in particular, each of these forms a sharp table (with $\Psi$), since sharpness is an invariant (Theorem 32). Hence, after a finite number of iterations, $\kappa$ is an iso. By Theorem 34 this implies that $(S, \Psi)$ is closed, which means the guard of the while loop is false. \qed

For termination of the outer loop, we need several auxiliary lemmas.

**Lemma 42.** Let $(S, \Psi)$ be table, and let $\Psi'$ be a subformula-closed subobject of $\Phi$, such that $\Psi$ is a subcoalgebra of $\Psi'$. Then there is a unique map $q$ making the following diagram commute:

$$
\begin{align*}
X & \xrightarrow{e_{\Psi'}} |X|_{\Psi'} \xrightarrow{m_{\Psi'}} Q\Psi' \\
\downarrow e_\Psi & \downarrow q & \quad \downarrow |X|_\Psi & \downarrow Qi \\
|X|_\Psi & \xrightarrow{m_\Psi} Q\Psi
\end{align*}
$$

Moreover, this map $q$ is an epimorphism.

**Proof.** The outside of the diagram commutes by Lemma 40. The map $q$ arises by the unique fill-in property. That $q$ is an epi follows since $e_\Psi$ is an epi, and $e_\Psi = q \circ e_{\Psi'}$. \qed

**Lemma 43.** Let $(S \xrightarrow{\gamma} X, \Psi)$ be a closed table, and $(S, \hat{\gamma}, \hat{s}_0)$ a pointed coalgebra, such that $(S, \hat{\gamma})$ is a conjecture and $s \circ \hat{s}_0 = s_0$. If $\Psi'$ is a countexample for $(S, \hat{\gamma}, \hat{s}_0)$, then the map $q: |X|_{\Psi} \to |X|_{\Psi'}$ from Lemma 42 is not an isomorphism.
Proof. Suppose that \( q \) is an iso; we prove that, in that case, \((S, \hat{\gamma}, \hat{s}_0)\) is correct w.r.t. \( \Psi' \). Let \( q^{-1} \) be the inverse of \( q \). Since \( q \circ e_{\Psi'} = e_{\Phi} \) we also have \( e_{\Psi'} = q^{-1} \circ e_{\Phi} \). Hence, the two shapes on the lower right in the following diagram commute:

\[
\begin{array}{ccc}
S & \xrightarrow{s} & X \\
\downarrow \hat{\gamma} & & \downarrow B e_{\Phi} \\
BS & \xrightarrow{B e_{\Phi}} & BX \\
\end{array}
\]

The rectangle commutes since \( \hat{\gamma} \) is a conjecture for the closed table \((S, \Psi)\). Since the entire diagram commutes, it shows that \((S, \hat{\gamma}, \hat{s}_0)\) is a conjecture for the closed table \((S, \Psi')\) as well. Together with \( s \circ \hat{s}_0 = s_0 \), by Lemma 26, we obtain that \((S, \hat{\gamma}, \hat{s}_0)\) is correct w.r.t. \( \Psi' \).

□

Proof (Proof of Theorem 37). The inner while loop terminates in each iteration of the outer loop by Corollary 36. The outer loop generates a sequence \( \Psi_0, \Psi_1, \Psi_2, \ldots \) of subobjects, such that for each \( i \), there is a pointed coalgebra \((S_i, \hat{\gamma}, \hat{s}_0)\) such that
- \((S_i \hookrightarrow X, \Psi_i)\) is a closed table,
- \((S_i, \hat{\gamma})\) is a conjecture for this table,
- \( s_i \circ \hat{s}_0 = s_0 \), and
- \( \Psi_{i+1} \) is a counterexample for \((S_i, \hat{\gamma}, \hat{s}_0)\).

We will show that such a sequence is necessarily finite.

By the last point and Lemma 42, for each \( i \), there exists a map \( q_{i+1,i} \) making the diagram on the left-hand side commute:

\[
\begin{array}{ccc}
X & \xrightarrow{c_{\Psi_i}} & X \\
\downarrow c_{\Psi_{i+1}} & & \downarrow c_{\Psi_{i+1}} \\
|X|_{\Psi_i} & \xrightarrow{q_{i+1,i}} & |X|_{\Psi_{i+1}} \\
\end{array}
\]

Moreover, again by Lemma 42, for each \( i \), there is a map \( q_i \) making the diagram on the right-hand side above commute. For each \( i \), we have

\[
q_{i+1,i} \circ q_{i+1} \circ c_{\Phi} = q_{i+1,i} \circ c_{i+1} = c_{\Psi_i} = q_i \circ c_{\Phi}
\]

and since \( c_{\Phi} \) is epic, we obtain \( q_{i+1,i} \circ q_{i+1} = q_i \). Hence, we get the following sequence of quotients:

\[
\begin{array}{ccc}
|X|_{\Psi_0} & \xrightarrow{q_0} & |X|_{\Psi_1} \\
\downarrow q_1 & & \downarrow q_2 \\
|X|_{\Psi_1} & \xrightarrow{q_2} & |X|_{\Psi_2} \\
\end{array}
\]

...
It follows from Lemma 43 and the previous assumptions that none of the quotients $q_{i+1}$ can be an iso. But since for each $i$, $|X|_{\phi_i}$ is a quotient of $|X|_{\psi}$, and the latter has only finitely many quotients, the sequence of counterexamples must be finite. $\square$