Coalgebra Learning via Duality

Simone Barlocco[1], Clemens Kupke[1(✉)], and Jurriaan Rot[2]

[1] University of Strathclyde, Glasgow, UK
{simone.barlocco,clemens.kupke}@strath.ac.uk
[2] Radboud University, Nijmegen, Netherlands
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* [14,26], 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 [12,19,21,22,26] 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 (Sect. 4). Second, we import the abstract notion of *base* of a functor from [8], which allows us to speak about

C. Kupke—Partially supported by EPSRC grant EP/N015843/1.

M. Bojańczyk and A. Simpson (Eds.): FOSSACS 2019, LNCS 11425, pp. 62–79, 2019.
https://doi.org/10.1007/978-3-030-17127-8_4

'successor states' (Sect. 5). In particular, the base allows us to characterise *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 (Sect. 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 [11–13] 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^* algorithm [4] and (ii) to motivate and outline our generalisation.

In the classical L^* algorithm, the learner tries to learn a regular language $\mathcal{L}$ 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

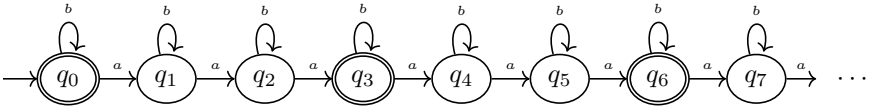
already. The algorithm constructs so-called tables (S, E) where $S, E \subseteq A^*$ are the rows and columns of the table, respectively. The value at position (s, e) of the table is the answer to the membership query "$se \in \mathcal{L}$?".

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*,[1] 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, Ψ) such that S (set of rows) is a selection of states of $\mathcal{A}$ and Ψ (set of columns) is a collection of tests/formulas. Membership queries become checks of tests in Ψ at states in S and equivalence queries verify whether or not the learned structure is logically equivalent to the original one. A table (S, Ψ) 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 Ψ. The clear distinction between states and tests in our algorithm means that counterexamples are formulas that have to be added to Ψ. 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.

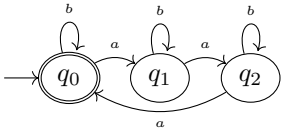
For the DFA case, let $L = \{u \in \{a, b\}^* \mid \text{number of } a\text{'s mod } 3 = 0\}$ and assume that the teacher uses the following (infinite) automaton describing L:



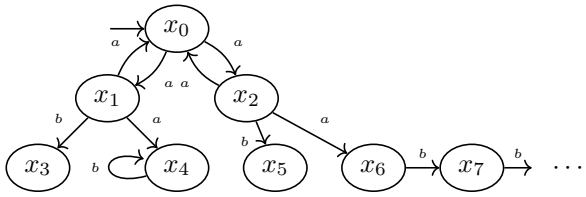
As outlined above, the learner starts to construct tables (S, Ψ) where S is a selection of states of the automaton and Ψ are formulas. For DFAs we will see (Example 1) that our formulas are just words in $\{a, b\}^*$. Our starting table is $(\{q_0\}, \emptyset)$, i.e., we select the initial state and 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

[1] The algorithm additionally requires *consistency*, but this is not needed if counterexamples are added to E. This idea goes back to [22].

second component was generated by closing $\{aa\}$ under suffixes. Suffix closedness features both in the original L* algorithm and in our framework (Sect. 4). The table $(\{q_0\}, \{\varepsilon, a, aa\})$ is not closed as q_1, the a-successor of q_0, does not accept ε 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 $\mathcal{A}$ 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 $q_0 \in S$.



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. Example 3). The semantics is as usual where $\langle a \rangle \phi$ holds at a state if it has an a-successor that makes ϕ 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 an a-loop 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 $\langle a \rangle \langle b \rangle \top$ is true at the root of the original LTS but false in the conjecture. We add the counterexample and all its subformulas to Ψ and obtain a new table $(\{x_0\}, \Psi')$ with $\Psi' = \{\langle a \rangle \langle b \rangle \top, \langle b \rangle \top, \top\}$. 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 (Example 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 [14,26]. We briefly recall the notion of pointed coalgebra, modelling a coalgebra with an initial state. Let $\mathcal{C}$ be a category with a terminal object 1 and let $B\colon \mathcal{C} \to \mathcal{C}$ be a functor. A pointed B-coalgebra is a triple (X, γ, x_0) where $X \in \mathcal{C}$ and $\gamma\colon X \to BX$ and $x_0\colon 1 \to X$, 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 [18,20,21,24].

Our basic setting consists of two categories $\mathcal{C}, \mathcal{D}$ connected by functors P, Q forming a dual adjunction $P \dashv Q\colon \mathcal{C} \leftrightarrows \mathcal{D}^{\mathsf{op}}$. In other words, we have a natural bijection $\mathcal{C}(X, Q\Delta) \cong \mathcal{D}(\Delta, PX)$ for $X \in \mathcal{C}, \Delta \in \mathcal{D}$. Moreover, we assume two functors, $B\colon \mathcal{C} \to \mathcal{C}, L\colon \mathcal{D} \to \mathcal{D}$, see (1). The functor L represents the syntax of the (modalities in the) logic: assuming that L has an initial algebra $\alpha\colon L\Phi \to \Phi$ we think of Φ as the collection of formulas, or tests. In this logical perspective, the functor P maps an object X of $\mathcal{C}$ to the collection of predicates and the functor Q maps an object Δ of $\mathcal{D}$ to the collection $Q\Delta$ of Δ-theories.

$$B \circlearrowleft \mathcal{C} \underset{Q}{\overset{P}{\rightleftarrows}} \mathcal{D}^{\mathsf{op}} \circlearrowright L \qquad (1)$$

The connection between coalgebras and their logics is specified via a natural transformation $\delta\colon LP \Rightarrow PB$, sometimes referred to as the one-step semantics of the logic. The δ is used to define the semantics of the logic on a B-coalgebra (X, γ) by initiality, as in (2). Furthermore, using the bijective correspondence of the dual adjunction between P and Q, the map $[\![\_]\!]$ corresponds to a map $th^\gamma\colon X \to Q\Phi$ that we will refer to as the theory map of (X, γ).

$$\begin{array}{ccccc} L\Phi & \overset{L[\![\_]\!]}{\dashrightarrow} & LPX & \xrightarrow{\delta_X} & PBX \\ {\scriptstyle\alpha}\downarrow & & & & \downarrow{\scriptstyle P\gamma} \\ \Phi & & \overset{\exists![\![\_]\!]}{\dashrightarrow} & & PX \end{array} \qquad (2)$$

The theory map can be expressed directly via a universal property, by making use of the so-called *mate* $\delta^\flat\colon BQ \Rightarrow QL$ of the one-step semantics δ (cf. [18,24]). More precisely, we have $\delta^\flat = QL\varepsilon \circ Q\delta Q \circ \eta BQ$, where η, ε are the unit and counit of the adjunction. Then $th^\gamma\colon X \to Q\Phi$ is the unique morphism making (3) commute.

$$\begin{array}{ccccc} BX & \overset{Bth^\gamma}{\dashrightarrow} & BQ\Phi & \xrightarrow{\delta^\flat_\Phi} & QL\Phi \\ {\scriptstyle\gamma}\uparrow & & & & \uparrow{\scriptstyle Q\alpha} \\ X & & \overset{\exists! th^\gamma}{\dashrightarrow} & & Q\Phi \end{array} \qquad (3)$$

Example 1. Let $\mathcal{C} = \mathcal{D} = \mathsf{Set}$, $P = Q = 2^{-}$ 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 $\alpha = [\varepsilon, \mathrm{cons}] \colon 1 + A \times A^* \to A^*$ where ε 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 $\delta \colon LP \Rightarrow PB$ is defined as follows: $\delta_X(*) = \{(i, f) \in 2 \times X^A \mid i = 1\}$, 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 Φ on a B-coalgebra (X, γ) is as follows: we have $[\![\varepsilon]\!] = \{x \in X \mid \pi_1(\gamma(x)) = 1\}$ and $[\![aw]\!] = \{x \in X \mid \pi_2(\gamma(x))(a) \in [\![w]\!]\}$, where π_1 and π_2 are the projection maps. The theory map th^{γ} sends a state to the language accepted by that state in the usual way.

Example 2. Again let $\mathcal{C} = \mathcal{D} = \mathsf{Set}$ 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 $\delta_X \colon A \times (1 + O^X) \to O^{BX}$ is defined by $\delta_X(a, \mathrm{inl}(*)) = \lambda f.\pi_1(f(a))$ and $\delta_X(a, \mathrm{inr}(g)) = \lambda f.g(\pi_2(f(a)))$. Concretely, formulas are words in A^+; the (O-valued) semantics of $w \in A^+$ at state x is the output $o \in O$ that is produced after processing the input w from state x.

Example 3. Let $\mathcal{C} = \mathsf{Set}$ and $\mathcal{D} = \mathsf{BA}$, 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 $B = (\mathcal{P}-)^A$ where $\mathcal{P}$ 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 $L \colon \mathsf{BA} \to \mathsf{BA}$ is defined as $LY := F_{\mathsf{BA}}(\{\langle a \rangle\, y \mid a \in A, y \in Y\})/\equiv$ where $F_{\mathsf{BA}} \colon \mathsf{Set} \to \mathsf{BA}$ denotes the free Boolean algebra functor and where, roughly speaking, $\equiv$ is the congruence generated from the axioms $\langle a \rangle \perp \equiv \perp$ and $\langle a \rangle (y_1 \vee y_2) \equiv \langle a \rangle (y_1) \vee \langle a \rangle (y_2)$ 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 $(\phi ::= \perp \mid \phi \vee \phi \mid \neg\phi \mid \langle a \rangle\, \phi)$ quotiented by logical equivalence. The definition of an appropriate δ can be found in, e.g., [21]—the semantics $[\![\_]\!]$ 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 $\mathsf{Sub}(X)$ the collection of subobjects of an object $X \in \mathcal{C}$. Let $\leq$ be the order on subobjects $s \colon S \rightarrowtail X, s' \colon S' \rightarrowtail X$ given by $s \leq s'$ iff there is $m \colon S \to S'$ s.t. $s = s' \circ m$. The *intersection* $\bigwedge J \rightarrowtail X$ of a family $J = \{s_i \colon S_i \to X\}_{i \in I}$ 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 $x_i \colon \bigwedge J \rightarrowtail S_i$.

For a functor $B \colon \mathcal{C} \to \mathcal{D}$, 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 $\{Bs_i \colon BS_i \to BX\}_{i \in I}$. By [2, Lemma 3.53] (building on [29]), *finitary* functors on 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, γ)* minimal w.r.t. logical equivalence *if the theory map $th^\gamma \colon X \to Q\Phi$ is a monomorphism.*

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

For expressive logics [27], behavioural equivalence coincides 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 expressive.

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 $\mathsf{Sub}(X)$ 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 $e \colon X \to Y$ 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.

$$\begin{array}{ccc} X & \xrightarrow{e} & Y \\ {\scriptstyle h}\downarrow & \swarrow^{d} & \downarrow{\scriptstyle g} \\ U & \xrightarrow[m]{} & Z \end{array} \tag{4}$$

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 $(\Phi, \alpha \colon L\Phi \to \Phi)$. To define a notion of partial correctness we need to consider subobjects of Φ to which we can restrict the theory map. This is formalised via the notion of "subformula closed" subobject of Φ.

The definition of such subobjects is based on the notion of *recursive coalgebra*. For $L\colon \mathcal{D} \to \mathcal{D}$ an endofunctor, a coalgebra $f\colon X \to LX$ is called *recursive* if for every L-algebra $g\colon LY \to Y$ there is a unique 'coalgebra-to-algebra' map $g^\dagger$ making (5) commute.

$$\begin{array}{ccc} LX & \xrightarrow{Lg^\dagger} & LY \\ {\scriptstyle f}\uparrow & & \downarrow{\scriptstyle g} \\ X & \xrightarrow{g^\dagger} & Y \end{array} \tag{5}$$

Definition 6. *A subobject $j\colon \Psi \to \Phi$ is called a* subformula closed collection *(of formulas) if there is a unique L-coalgebra structure $\sigma\colon \Psi \to L\Psi$ such that (Ψ, σ) is a recursive L-coalgebra and j is the (necessarily unique) coalgebra-to-algebra map from (Ψ, σ) to the initial algebra (Φ, α).*

Remark 7. The uniqueness of σ in Definition 6 is implied if L preserves monomorphisms. This is the case in our examples. The notion of recursive coalgebra goes back to [23,28]. 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 (Ψ, σ) 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, $\mathcal{C} = \mathsf{Set}$ and $\mathcal{D} = \mathsf{BA}$, 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 Ψ, 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 Δ of modal formulas that is closed under taking subformulas, we define a Boolean algebra $\Psi_\Delta \subseteq \Phi$ as the smallest Boolean subalgebra of Φ that is generated by the set $\hat{\Delta} = \{[\phi]_\Phi \mid \phi \in \Delta\}$ where for a formula ϕ we let $[\phi]_\Phi \in \Phi$ denote its equivalence class in Φ.

It is then not difficult to define a suitable $\sigma\colon \Psi_\Delta \to L\Psi_\Delta$. As Ψ_Δ is generated by closing $\hat{\Delta}$ under Boolean operations, any two states x_1, x_2 in a given coalgebra (X, γ) satisfy $(\forall b \in \Psi_\Delta . x_1 \in [\![b]\!] \Leftrightarrow x_2 \in [\![b]\!])$ iff $\left(\forall b \in \hat{\Delta} . x_1 \in [\![b]\!] \Leftrightarrow x_2 \in [\![b]\!]\right)$. In other words, equivalence w.r.t. Ψ_Δ coincides with equivalence w.r.t. the *set* of formulas Δ. 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 Ψ is that we can restrict our attention to the so-called Ψ-theory map. Intuitively, subformula closedness is what allows us to define this theory map inductively.

$$\begin{array}{ccccc} X & & \xrightarrow{th^\gamma_\Psi} & & Q\Psi \\ {\scriptstyle \gamma}\downarrow & & & & \uparrow{\scriptstyle Q\sigma} \\ BX & \xrightarrow{Bth^\gamma_\Psi} & BQ\Psi & \xrightarrow{\delta^\flat_\Psi} & QL\Psi \end{array} \tag{6}$$

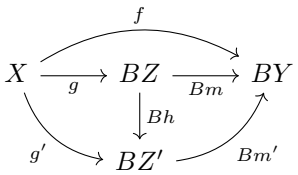
Lemma 10. *Let $\Psi \overset{j}{\rightarrowtail} \Phi$ be a sub-formula closed collection, with coalgebra structure $\sigma\colon \Psi \to L\Psi$. Then $th^{\gamma}_{\Psi} = Qj \circ th^{\gamma}_{\Phi}$ is the unique map making (6) commute. We call th^{γ}_{Ψ} the Ψ-theory map, and omit the Ψ if it is 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 $B\colon \mathcal{C} \to \mathcal{C}$ be an endofunctor. We say B* has a base *if for every arrow $f\colon X \to BY$ there exist $g\colon X \to BZ$ and $m\colon Z \rightarrowtail Y$ with m a monomorphism such that $f = Bm \circ g$, and for any pair $g'\colon X \to BZ', m'\colon Z' \rightarrowtail Y$ with $Bm' \circ g' = f$ and m' a monomorphism there is a unique arrow $h\colon Z \to Z'$ such that $Bh \circ g = g'$ and $m' \circ h = m$, see Diagram (7). We call (Z, g, m) the (B)-*base *of the morphism f.*

We sometimes refer to $m\colon Z \rightarrowtail Y$ 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.



(7)

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

For $B = (\mathcal{P}-)^A\colon \mathsf{Set} \to \mathsf{Set}$, the B-*base* of $f\colon X \to Y$ is given by the inclusion $m\colon Z \rightarrowtail Y$, 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 $B\colon \mathcal{C} \to \mathcal{C}$ preserves (wide) intersections. Then B has a base.*

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

Lemma 13. *Let $B\colon \mathcal{C} \to \mathcal{C}$ be a functor that has a base and that preserves preimages. Let $f\colon S \to BX$ and $h\colon X \to Y$ be morphisms, let (Z, g, m) be the base of f and let $e\colon Z \to W, m'\colon W \to Y$ be the (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 Sect. 3, the collection of subobjects $(\mathsf{Sub}(X), \leq)$ ordered as usual (cf. Section 3) forms a complete lattice. Recall that the meet on $\mathsf{Sub}(X)$ (intersection) is defined via pullbacks. In categories with coproducts, the join $s_1 \vee s_2$ of subobjects $s_1, s_2 \in \mathsf{Sub}(X)$ is defined as the mono part of the factorisation of the map $[s_1, s_2] \colon S_1 + S_2 \to X$, i.e., $[s_1, s_2] = (s_1 \vee s_2) \circ e$ for a strong epi e. In Set, this amounts to taking the union of subsets.

For a binary join $s_1 \vee s_2$ we denote by $inl_\vee \colon S_1 \to (S_1 \vee S_2)$ and $inr_\vee \colon S_2 \to (S_1 \vee S_2)$ the embeddings that exist by $s_i \leq s_1 \vee s_2$ for $i = \{1, 2\}$. Let us now define the key operator of this section.

$$\begin{array}{ccc} S & \xrightarrow{\ s\ } & X \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle \gamma} \\ B\Gamma(S) & \xrightarrow{B\Gamma^B_\gamma(s)} & BX \end{array} \tag{8}$$

Definition 14. *Let B be a functor that has a base, $s \colon S \rightarrowtail X$ a subobject of some $X \in \mathcal{C}$ and let (X, γ) be a B-coalgebra. Let $(\Gamma(S), g, \Gamma^B_\gamma(s))$ be the B-base of $\gamma \circ s$, see Diagram (8). Whenever B and γ are clear from the context, we write $\Gamma(s)$ instead of $\Gamma^B_\gamma(s)$.*

Lemma 15. *Let $B \colon \mathcal{C} \to \mathcal{C}$ be a functor with a base and let (X, γ) be a B-coalgebra. The operator $\Gamma \colon \mathsf{Sub}(X) \to \mathsf{Sub}(X)$ defined by $s \mapsto \Gamma(s)$ is monotone.*

Intuitively, Γ 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 γ to an element of S. We will see that pre-fixpoints of Γ correspond to subcoalgebras. Furthermore, Γ is the key to formulate our notion of closed table in the learning algorithm.

Proposition 16. *Let $s \colon S \rightarrowtail X$ be a subobject and $(X, \gamma) \in \mathsf{Coalg}(B)$ for $X \in \mathcal{C}$ and $B \colon \mathcal{C} \to \mathcal{C}$ a functor that has a base. Then s is a subcoalgebra of (X, γ) 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 lfp of an operator defined in terms of Γ.

Theorem 17. *Let $B \colon \mathcal{C} \to \mathcal{C}$ be a functor that has a base. A pointed B-coalgebra (X, γ, x_0) is reachable iff $X \cong \mathsf{lfp}(\Gamma \vee x_0)$ (isomorphic as subobjects of X, i.e., equal).*

This justifies defining the reachable part from an initial state $x_0 \colon 1 \rightarrowtail X$ as the least fixpoint of the monotone operator $\Gamma \vee x_0$. Standard means of computing the least fixpoint by iterating this operator then give us a way to compute this subcoalgebra. Further, Γ provides a way to generalise the notion of "prefixed closedness" from Angluin's L* algorithm to our categorical setting.

Definition 18. *Let $s_0, s \in \mathsf{Sub}(X)$ for some $X \in \mathcal{C}$ and let (X, γ) be a B-coalgebra. We call s s_0-*prefix closed w.r.t. γ *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, γ, 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 Φ of all tests, i.e., the theory of (X, γ) 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 1, 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 $\gamma \colon X \to BX$. Concretely, the teacher gives us:

- for any subobject $S \rightarrowtail X$ and sub-formula closed subobject Ψ of Φ, the composite theory map $S \rightarrowtail X \xrightarrow{th^{\gamma}_{\Psi}} Q\Psi$;
- for $(S, \hat{\gamma}, \hat{s}_0)$ a pointed coalgebra, whether or not it is correct w.r.t. the collection Φ of all tests;
- in case of a negative answer to the previous question, a *counterexample*, which essentially is a subobject Ψ' of Φ 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 $S \rightarrowtail X \xrightarrow{\gamma} BX$.

The first three points correspond respectively to the standard notions of membership query ('filling in' the table with rows S and columns Ψ), 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, γ) is used, which allows to represent next states in a fixed manner. It is required in our setting of an arbitrary coalgebra (X, γ).

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 (Sect. 6.1). Based on these notions, the actual algorithm is presented (Sect. 6.2), followed by proofs of correctness and termination (Sect. 6.3).

Assumption 19. *Throughout this section, we assume*

- *that we deal with coalgebras over the base category* $\mathcal{C} = \mathsf{Set}$*;*
- *a functor* $B \colon \mathcal{C} \to \mathcal{C}$ *that preserves pre-images and wide intersections;*
- *a category* $\mathcal{D}$ *with an initial object* 0 *s.t. arrows with domain* 0 *are monic;*
- *a functor* $L \colon \mathcal{D} \to \mathcal{D}$ *with an initial algebra* $L\Phi \xrightarrow{\cong} \Phi$*;*
- *an adjunction* $P \dashv Q \colon \mathcal{C} \leftrightarrows \mathcal{D}^{\mathsf{op}}$*, and a logic* $\delta \colon LP \Rightarrow PB$*.*

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

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

6.1 Tables and Counterexamples

Definition 21. *A* table *is a pair $(S \overset{s}{\rightarrowtail} X, \Psi \overset{i}{\rightarrowtail} \Phi)$ consisting of a subobject s of X and a subformula-closed subobject i of Φ.*

To make the notation a bit lighter, we sometimes refer to a table by (S, Ψ), using s and i respectively to refer to the actual subobjects. The pair (S, Ψ) represents 'rows' and 'columns' respectively, in the table; the 'elements' of the table are given abstractly by the map $th^\gamma_\Psi \circ s$. In particular, if $\mathcal{C} = \mathcal{D} = \mathsf{Set}$ and $Q = 2^-$, 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 $S \rightarrowtail X$.

$$\begin{array}{ccccc} S & \overset{s}{\rightarrowtail} & X & \xrightarrow{th^\gamma} & Q\Psi \\ {\scriptstyle k}\uparrow & & & \nearrow_{th^\gamma} & \\ \Gamma(S) & & \xrightarrow[\Gamma(s)]{} & X & \end{array} \tag{9}$$

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

Thus, a table (S, Ψ) 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 Ψ. 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. Ψ', that is, the successors agree up to logical equivalence.

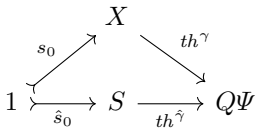
$$\begin{array}{ccccc} S & \overset{s}{\rightarrowtail} & X & \xrightarrow{\gamma} & BX \\ {\scriptstyle \hat{\gamma}}\downarrow & & & & \downarrow{\scriptstyle Bth^\gamma} \\ BS & \xrightarrow[Bs]{} & BX & \xrightarrow[Bth^\gamma]{} & BQ\Psi \end{array} \tag{10}$$

Definition 23. *Let (S, Ψ) be a table. A coalgebra structure $\hat{\gamma}\colon S \to BS$ is called a* conjecture *(for (S, Ψ)) 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 Ψ.



(11)

Definition 25. *Let (S, Ψ) 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. Ψ if Diagram (11) commutes.*

All conjectures constructed during the learning algorithm will be correct w.r.t. the subformula closed collection Ψ of formulas under consideration.

Lemma 26. *Suppose (S, Ψ) is closed, and $\hat{\gamma}$ is a conjecture. Then $th^{\gamma}_{\Psi} \circ s = th^{\hat{\gamma}}_{\Psi} \colon S \to Q\Psi$. If $\hat{s}_0 \colon 1 \to S$ satisfies $s \circ \hat{s}_0 = s_0$ then $(S, \hat{\gamma}, \hat{s}_0)$ is correct w.r.t. Ψ.*

We next define the crucial notion of *counterexample* to a pointed coalgebra: a subobject Ψ' of Ψ on which it is 'incorrect'.

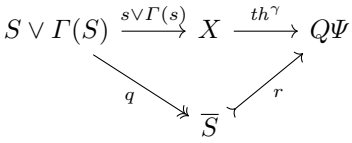
Definition 27. *Let (S, Ψ) be a table, and let $(S, \hat{\gamma}, \hat{s}_0)$ be a pointed B-coalgebra on S. Let Ψ' be a subformula closed subobject of Φ, such that Ψ is a subcoalgebra of Ψ'. We say Ψ' is a* counterexample (for $(S, \hat{\gamma}, \hat{s}_0)$, extending Ψ) *if $(S, \hat{\gamma}, \hat{s}_0)$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

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

Lemma 28. *Let (S, Ψ) 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 Ψ. Then $(S, \hat{\gamma}, \hat{s}_0)$ is correct w.r.t. Φ.*

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, Ψ) be a sharp table. Let $(\overline{S}, q, r)$ be the (strong epi, mono)-factorisation of the map $th^{\gamma} \circ (s \vee \Gamma(s))$, as in the diagram:*



We define $\mathsf{close}(S, \Psi) := \{\overline{s} \colon \overline{S} \rightarrowtail X \mid th^{\gamma} \circ \overline{s} = r, s \leq \overline{s} \leq s \vee \Gamma(s)\}$. *For each $\overline{s} \in \mathsf{close}(S, \Psi)$ we have $s \leq \overline{s}$ and thus $s = \overline{s} \circ \kappa$ for some $\kappa \colon S \to \overline{S}$.*

Lemma 30. *In Definition 29, for each $\overline{s} \in \mathsf{close}(S, \Psi)$, we have $\kappa = q \circ inl_{\vee}$.*

We will refer to $\kappa = q \circ inl_{\vee}$ as the connecting map from s to $\overline{s}$.

Lemma 31. *In Definition 29, if there exists $q^{-1} \colon \overline{S} \to S \vee \Gamma(S)$ such that $q \circ q^{-1} = \mathsf{id}$ and $q^{-1} \circ q \circ inl_{\vee} = inl_{\vee}$, then $\mathsf{close}(S, \Psi)$ is non-empty.*

By our assumptions, the hypothesis of Lemma 31 is satisfied (Remark 20), hence $\mathsf{close}(S, \Psi)$ 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 $\mathsf{counter}((S, \hat{\gamma}, \hat{s}_0), \Psi)$, which returns the set of all counterexamples (extending Ψ) 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 Ψ'. We also make use of $\mathsf{close}(S, \Psi)$, as given in Definition 29.

Algorithm 1. Abstract learning algorithm

1: $(S \overset{s}{\rightarrowtail} X) \leftarrow (1 \overset{s_0}{\rightarrowtail} X)$
2: $\hat{s}_0 \leftarrow \mathsf{id}_1$
3: $\Psi \leftarrow 0$
4: **while** `true` **do**
5: **while** $(S \overset{s}{\rightarrowtail} X, \Psi)$ is not closed **do**
6: let $(\overline{S} \overset{\overline{s}}{\rightarrowtail} X) \in \mathsf{close}(S, \Psi)$, with connecting map $\kappa \colon S \rightarrowtail \overline{S}$
7: $(S \overset{s}{\rightarrowtail} X) \leftarrow (\overline{S} \overset{\overline{s}}{\rightarrowtail} X)$
8: $\hat{s}_0 \leftarrow \kappa \circ \hat{s}_0$
9: **end while**
10: let $(S, \hat{\gamma})$ be a conjecture for (S, Ψ)
11: **if** $\mathsf{counter}((S, \hat{\gamma}, \hat{s}_0), \Psi) = \emptyset$ **then**
12: **return** $(S, \hat{\gamma}, \hat{s}_0)$
13: **else**
14: $\Psi \leftarrow \Psi'$ for some $\Psi' \in \mathsf{counter}((S, \hat{\gamma}, \hat{s}_0), \Psi)$
15: **end if**
16: **end while**

The algorithm takes as input the coalgebra (X, γ, 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 1 in Sect. 6.2: 1.* (S, Ψ) *is sharp, 2.* $s \circ \hat{s}_0 = s_0$*, and 3.* s *is* s_0*-prefix closed w.r.t.* γ*.*

Theorem 33. *If Algorithm 1 in Sect. 6.2 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.* Φ*.*

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 Ψ be a subobject of Φ. We take a (strong epi, mono)-factorisation of the theory map, i.e., $th^{\gamma}_{\Psi} = \left(X \xrightarrow{e_\Psi} |X|_\Psi \xrightarrowtail{m_\Psi} Q\Psi \right)$ for some strong epi e and mono m. We call the object $|X|_\Psi$ in the middle the *Ψ-logical quotient*. For the termination result (Theorem 37), $|X|_\Phi$ is assumed to have finitely many quotients and subobjects, which just amounts to finiteness, in Set.

We start with termination of the inner while loop (Corollary 36). This relies on two results: first, that once the connecting map κ is an iso, the table is closed, and second, that—under a suitable assumption on the coalgebra (X, γ)—during execution of the inner while loop, the map κ will eventually be an iso.

Theorem 34. *Let (S, Ψ) be a sharp table, let $\overline{S} \in \mathsf{close}(S, \Psi)$ and let $\kappa \colon S \to \overline{S}$ be the connecting map. If κ is an isomorphism, then (S, Ψ) is closed.*

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

Corollary 36. *If the Φ-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 Φ-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 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 [5,9] 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 [16]. 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. Adámek, J., Lücke, D., Milius, S.: Recursive coalgebras of finitary functors. ITA **41**(4), 447–462 (2007)
2. Adámek, J., Milius, S., Moss, L.S., Sousa, L.: Well-pointed coalgebras. Logical Methods Comput. Sci. **9**(3) (2013)
3. Adámek, J., Rosický, J.: Locally Presentable and Accessible Categories. Cambridge Tracts in Mathematics. Cambridge University Press, Cambridge (1994)
4. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987)
5. Angluin, D., Eisenstat, S., Fisman, D.: Learning regular languages via alternating automata. In: Yang, Q., Wooldridge, M. (eds.) IJCAI 2015, pp. 3308–3314. AAAI Press (2015)
6. Barlocco, S., Kupke, C.: Angluin learning via logic. In: Artemov, S., Nerode, A. (eds.) LFCS 2018. LNCS, vol. 10703, pp. 72–90. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72056-2_5
7. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press, Cambridge (2001)
8. Blok, A.: Interaction, observation and denotation. Master's thesis, ILLC Amsterdam (2012)
9. Bollig, B., Habermehl, P., Kern, C., Leucker, M.: Angluin-style learning of NFA. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009, pp. 1004–1009 (2009)
10. Borceux, F.: Handbook of Categorical Algebra. Encyclopedia of Mathematics and its Applications, vol. 1. Cambridge University Press, Cambridge (1994)
11. van Heerdt, G.: An abstract automata learning framework. Master's thesis, Radboud Universiteit Nijmegen (2016)
12. van Heerdt, G., Sammartino, M., Silva, A.: CALF: categorical automata learning framework. In: Goranko, V., Dam, M. (eds.) 26th EACSL Annual Conference on Computer Science Logic, CSL 2017. LIPIcs, vol. 2, pp. 29:1–29:24. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
13. van Heerdt, G., Sammartino, M., Silva, A.: Learning automata with side-effects. CoRR, abs/1704.08055 (2017)
14. Jacobs, B.: Introduction to Coalgebra: Towards Mathematics of States and Observation. Cambridge Tracts in Theoretical Computer Science, vol. 59. Cambridge University Press, Cambridge (2016)

15. Jacobs, B., Silva, A.: Automata learning: a categorical perspective. In: van Breugel, F., Kashefi, E., Palamidessi, C., Rutten, J. (eds.) Panangaden Festschrift. LNCS, vol. 8464, pp. 384–406. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06880-0_20
16. Jacobs, B., Silva, A., Sokolova, A.: Trace semantics via determinization. J. Comput. Syst. Sci. **81**(5), 859–879 (2015)
17. Jacobs, B., Sokolova, A.: Exemplaric expressivity of modal logics. J. Logic Comput. **20**(5), 1041–1068 (2009)
18. Klin, B.: Coalgebraic modal logic beyond sets. Electr. Notes Theor. Comput. Sci. **173**, 177–201 (2007)
19. Klin, B., Rot, J.: Coalgebraic trace semantics via forgetful logics. Logical Methods Comput. Sci. **12**(4) (2016)
20. Kupke, C., Kurz, A., Pattinson, D.: Algebraic semantics for coalgebraic logics. Electr. Notes Theor. Comput. Sci. **106**, 219–241 (2004)
21. Kupke, C., Pattinson, D.: Coalgebraic semantics of modal logics: an overview. Theor. Comput. Sci. **412**(38), 5070–5094 (2011)
22. Maler, O., Pnueli, A.: On the learnability of infinitary regular sets. Inf. Comput. **118**(2), 316–326 (1995)
23. Osius, G.: Categorical set theory: a characterization of the category of sets. J. Pure Appl. Algebra **4**(1), 79–119 (1974)
24. Pavlovic, D., Mislove, M., Worrell, J.B.: Testing semantics: connecting processes and process logics. In: Johnson, M., Vene, V. (eds.) AMAST 2006. LNCS, vol. 4019, pp. 308–322. Springer, Heidelberg (2006). https://doi.org/10.1007/11784180_24
25. Rutten, J.J.M.M.: Automata and coinduction (an exercise in coalgebra). In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 194–218. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055624
26. Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. Theor. Comput. Sci. **249**(1), 3–80 (2000)
27. Schröder, L.: Expressivity of coalgebraic modal logic: the limits and beyond. Theor. Comput. Sci. **390**(2–3), 230–247 (2008)
28. Taylor, P.: Practical Foundations of Mathematics. Cambridge University Press, Cambridge (1999)
29. Trnková, V.: On descriptive classification of set-functors. I. Comment. Math. Univ. Carolinae **12**(1), 143–174 (1971)
30. Vaandrager, F.W.: Model learning. Commun. ACM **60**(2), 86–95 (2017)