Automata Learning: An Algebraic Approach

Henning Urbat*
Friedrich-Alexander-Universität Erlangen-Nürnberg
Erlangen, Germany
henning.urbat@fau.de

Lutz Schröder*
Friedrich-Alexander-Universität Erlangen-Nürnberg
Erlangen, Germany
lutz.schroeder@fau.de

Abstract
We propose a generic categorical framework for learning unknown formal languages of various types (e.g. finite or infinite words, weighted and nominal languages). Our approach is parametric in a monad \( T \) that represents the given type of languages and their recognizing algebraic structures. Using the concept of an automata presentation of \( T \)-algebras, we demonstrate that the task of learning a \( T \)-recognizable language can be reduced to learning an abstract form of algebraic automaton whose transitions are modeled by a functor. For the important case of adjoint automata, we devise a learning algorithm generalizing Angluin’s \( \mathsf{L}^* \). The algorithm is phrased in terms of categorically described extension steps; we provide for a termination and complexity analysis based on a dedicated notion of finiteness. Our framework applies to structures like \( \omega \)-regular languages that are not within the scope of existing categorical accounts of automata learning. In addition, it yields new learning algorithms for several types of languages for which no such algorithms were previously known at all, including sorted languages, nominal languages with name binding, and cost functions.

Keywords Automata Learning, Monads, Algebras

1 Introduction
Active automata learning is the task of inferring a finite representation of an unknown formal language by asking questions to a teacher. Such learning situations naturally arise, e.g., in software verification, where the “teacher” is some reactive system and one aims to construct a formal model of it by running suitable tests [61]. Starting with Angluin’s [8] pioneering work on learning regular languages, active learning algorithms have been developed for countless types of systems and languages, including \( \omega \)-regular languages [9, 32], tree languages [30], weighted languages [12, 62], and nominal languages [47]. Most of these extensions are tailor-made modifications of Angluin’s \( \mathsf{L}^* \) algorithm and thus bear close structural analogies. This has motivated recent work towards a uniform category theoretic understanding of automata learning, based on modelling state-based systems as coalgebras [14, 64]. In the present paper, we propose a novel algebraic approach to automata learning.

Our contributions are two-fold. First, we study the problem of learning an abstract form of automata originally introduced by Arbib and Manes [10] in the context of minimization: given an endofunctor \( F \) on a category \( \mathcal{D} \) and objects \( I, O \in \mathcal{D} \), an \( F \text{-}{\text{automaton}} \) consists of an object \( Q \) of states and morphisms \( \delta_Q, i_Q \) and \( f_Q \) as shown below, representing transitions, initial states and final states (or outputs).

\[
\begin{array}{ccc}
FQ & \xrightarrow{\delta_Q} & O \\
I & \xrightarrow{i_Q} & Q & \xrightarrow{f_Q} & O
\end{array}
\]

Taking \( FQ = \Sigma \times Q \) on \( \mathsf{Set} \) with \( I = 1 \) and \( O = \{0, 1\} \) yields classical deterministic automata, but also several other notions of automata (e.g. weighted automata, residual nondeterministic automata, and nominal automata) arise as instances. As our first main result, we devise a generalized \( \mathsf{L}^* \) algorithm for \( \mathsf{adjoint} \ \mathsf{F-automata} \), i.e. automata whose type functor \( F \) admits a right adjoint \( G \), based on alternating moves along the initial chain for the functor \( I + F \) and the final cochain for the functor \( O \times G \). Our generic algorithm subsumes known \( \mathsf{L}^* \)-type algorithms for all the above classes of automata, and its analysis yields uniform proofs of their correctness and termination. In addition, it also instantiates to a number of new learning algorithms, e.g. for sorted automata and for several versions of nominal automata with name binding.

We subsequently show that learning algorithms for \( \mathsf{F-automata} \) (including our generalized \( \mathsf{L}^* \) algorithm) apply far beyond the realm of automata: they can be used to learn languages representable by \( \mathsf{monads} \) [7, 59]. Given a monad \( T \) on the category \( \mathcal{D} \), we model a \( \mathsf{language} \) as a morphism \( L: TI \to O \) in \( \mathcal{D} \). At this level of generality, one obtains a concept of \( \mathsf{T}-\text{recognizable language} \) (i.e. a language recognized by a finite \( \mathsf{T}\)-algebra) that uniformly captures numerous automata-theoretic classes of languages. For instance, regular and \( \omega \)-regular languages (the languages accepted by classical finite automata and Büchi automata, respectively) correspond precisely to \( \mathsf{T}-\text{recognizable languages} \) for the monads representing semigroups and Wilke algebras,

\[
TI = I^+ \text{ on } \mathsf{Set} \quad \text{and} \quad T(I, J) = (I^+, I^{[\mathsf{up}]} + I^* \times J) \text{ on } \mathsf{Set}_2.
\]

Here \( I^{[\mathsf{up}]} \) denotes the set of ultimately periodic infinite words over the alphabet \( I \). For \( \omega \)-regular languages, Farzan et al. [32] proposed an algorithm that learns a language \( L \subseteq I^\omega \) of infinite words by learning the set of lassos in \( L \), i.e. the
We show that this idea extends to general $T$-recognizable languages, using the concept of an automata presentation. Such a presentation allows for the linearization of $T$-recognizable languages, i.e., a reduction to “regular” languages accepted by finite $F$-automata for suitable $F$.

In combination, our results yield a generic strategy for learning an unknown $T$-recognizable language $L$: (1) find an automata presentation for the free $T$-algebra $TI$; (2) learn the minimal automaton for the linearization of $L$.

This approach turns out to be applicable to a wide range of approaches, using the concept of an $F$-automata for suitable $F$.

Their main contribution is an abstract categorical framework to each state a structured collection of successors. Coalgebra homomorphisms are behaviour-preserving maps, and final coalgebras have abstracted behaviours as states.

**Related work.** A categorical interpretation of several key concepts in Angluin’s L’ algorithm for classical automata was first given by Jacobs and Silva [37], and later extended to $F$-automata in a category, i.e., to similar generality as in the present paper, by van Heerdt, Sammartino, and Silva [63]. Their main contribution is an abstract categorical framework (CALF) for correctness proofs of learning algorithms, while a concrete general algorithm is not given. Van Heerdt et al. [64] also study learning automata with side effects modelled via monads; this use of monads is unrelated to the monad-homorphism $D$-Functor (co-)algebras.

The theory of nominal sets that we will use throughout the paper, by van Heerdt, Sammartino, and Silva [63], is closely related to, e.g., nominal automata. The connections between the two approaches are further discussed in Remark 4.16.

Automata learning can be seen as an interactive version of automata minimization, which has been extensively studied from a (co-)algebraic perspective [5, 10, 16, 24, 35, 62]. In particular, our chain-based iterative learning algorithm resembles the coalgebraic approach to partition refinement [1].

### 2 Preliminaries

We proceed to recall concepts from category theory and the theory of nominal sets that we will use throughout the paper. Readers should be familiar with basic notions such as functors, (co-)limits and adjunctions; see, e.g., Mac Lane [41].

**Functor (co-)algebras.** Let $H: \mathcal{D} \to \mathcal{D}$ be an endofunctor on a category $\mathcal{D}$. An $H$-algebra is a pair $(A, \alpha)$ consisting of an object $A \in \mathcal{D}$ and a morphism $\alpha: HA \to A$. A homomorphism $h: (A, \alpha) \to (B, \beta)$ between $H$-algebras is a morphism $h: A \to B$ such that $h \cdot \alpha = \beta \cdot Hh$. An $H$-algebra $(A, \alpha)$ is initial if for every $H$-algebra $(B, \beta)$ there is a unique homomorphism $(A, \alpha) \to (B, \beta)$; we generally denote the initial algebra of $H$ (unique up to isomorphism if it exists) as $\mu H$.

$\mathcal{D}$ is cocomplete and $H$ preserves filtered colimits, $\mu H$ can be constructed as the colimit of the initial $\alpha$-chain for $H$ [6]:

$$\mu H = \text{colim}(0 \to H0 \xrightarrow{H} H^20 \xrightarrow{H^2} H^30 \to \cdots),$$

where $i$ is the unique morphism from the initial object 0 of $\mathcal{D}$ into $H0$, and $H^n$ means $H$ applied $n$ times. Letting $j_n: H^n0 \to \mu H$ ($n \in \mathbb{N}$) denote the colimit cocone, we obtain the $H$-algebra structure on $\mu H$ as the unique morphism $\alpha: H(\mu H) \to \mu H$ satisfying $\alpha \cdot Hj_n = j_{n+1}$ for all $n \in \mathbb{N}$.

Dually, one has notions of a coalgebra for the endofunctor $H$, a coalgebra homomorphism, and a final coalgebra. Coalgebras provide an abstract notion of state-based transition system: We think of the base object $A$ of an $H$-coalgebra as an object of states, and of its structure map $\alpha: A \to HA$ as assigning to each state a structured collection of successors. Coalgebra homomorphisms are behaviour-preserving maps, and final coalgebras have abstracted behaviours as states.

**Monad algebras.** A monad $T = (T, \mu, \eta)$ on a category $\mathcal{D}$ is given by an endofunctor $T: \mathcal{D} \to \mathcal{D}$ and two natural transformations $\eta: \text{Id}_\mathcal{D} \to T$ and $\mu: TT \to T$ (the unit and multiplication) such that the following diagrams commute:

$$
\begin{array}{ccc}
TTT & \xrightarrow{T\mu} & TT \\
\mu T & \downarrow & \downarrow \mu \\
TT & \xrightarrow{\mu} & T \\
\end{array}
$$

A $T$-algebra is an algebra $(A, \alpha)$ for the endofunctor $T$ for which the following diagrams commute:

$$
\begin{array}{ccc}
TTA & \xrightarrow{T\alpha} & TA \\
\alpha T & \downarrow & \downarrow \alpha \\
TA & \xrightarrow{\alpha} & A \\
\end{array}
$$

A homomorphism of $T$-algebras is just a homomorphism of the underlying $T$-algebras. For each $X \in \mathcal{D}$, the $T$-algebra $TX = (TX, \mu_X)$ is called the free $T$-algebra on $X$.

Monads form a categorical abstraction of algebraic theories [43]. In fact, every algebraic theory (given by a finitary signature $\Gamma$ and a set $E$ of equations between $\Gamma$-terms) induces a monad $T$ on Set where $TX$ is the underlying set of the free $(\Gamma, E)$-algebra on $X$ (i.e., the set of all $\Gamma$-terms over $X$ modulo equations in $E$), and the maps $\eta_X: X \to TX$ and $\mu_X: TTX \to TX$ are given by inclusion of variables and flattening of terms, respectively. Then the categories of $T$-algebras and $(\Gamma, E)$-algebras are isomorphic. Conversely, every monad $T$ on Set with $T$ preserving filtered colimits arises from some algebraic theory $(\Gamma, E)$ in this way.

Similarly, every ordered algebraic theory [17], given by a signature $\Gamma$ and a set $E$ of inequations $s \leq t$ between $\Gamma$-terms, yields a monad $T$ on the category Pos of posets whose algebras are ordered $\Gamma$-algebras (i.e., $\Gamma$-algebras on a poset with monotone operations) satisfying the inequations in $E$. 

\[\text{Henning Urbat and Lutz Schröder}\]
Free monads. Let $H : \mathcal{D} \to \mathcal{D}$ be an endofunctor on a category $\mathcal{D}$ with coproducts, and suppose that, for each $X \in \mathcal{D}$, the initial algebra $\mu(X + H)$ for the functor $X + H$ exists. Then $H$ induces a monad $T_H$, the free monad over $H$ [15]. It is given on objects by $T_HX = \mu(X + H)$; its action on morphisms and the unit and multiplication are defined via initiality of the algebras $\mu(X + H)$. Then the categories of $T_H$-algebras and $H$-algebras are isomorphic: If $B \in H(T_HB) \xrightarrow{[a, a^\alpha]} T_HB$ denotes the $B + H$-algebra structure of $T_HB = \mu(B + H)$, the isomorphism is given on objects by
\[
(T_HB \xrightarrow{\beta} B) \mapsto (HB \xrightarrow{\mu a} H(T_HB) \xrightarrow{\alpha B} T_HB \xrightarrow{\beta} B)
\]
and on morphisms by $h \mapsto h$.

Factorization systems. A factorization system $(\mathcal{E}, \mathcal{M})$ in a category $\mathcal{D}$ is given by two classes $\mathcal{E}$ and $\mathcal{M}$ of morphisms such that (i) $\mathcal{E}$ and $\mathcal{M}$ are closed under composition and contain all isomorphisms, (ii) every morphism $f$ has a factorization $f = m \cdot e$ with $e \in \mathcal{E}$ and $m \in \mathcal{M}$, and (iii) the diagonal fill-in property holds: given a commutative square $m \cdot f = g \cdot e$ with $e \in \mathcal{E}$ and $m \in \mathcal{M}$, there exists a unique morphism $d$ with $f = d \cdot e$ and $g = m \cdot d$. The morphisms $m$ and $e$ in (i) are unique up to isomorphism and are called the image and coimage of $f$. Categories of (co-)algebras typically inherit factorizations from their underlying category:

1. If $H : \mathcal{D} \to \mathcal{D}$ is an endofunctor with $H(\mathcal{E}) \subseteq \mathcal{E}$, the factorization system $(\mathcal{E}, \mathcal{M})$ for $\mathcal{D}$ lifts to the category of $H$-algebras, that is, every $H$-algebra homomorphism uniquely factorizes into a homomorphism in $\mathcal{E}$ followed by a homomorphism in $\mathcal{M}$. Dually, if $H(\mathcal{M}) \subseteq \mathcal{M}$, then the category of $H$-coalgebras has a factorization system lifting $(\mathcal{E}, \mathcal{M})$.

2. If $T$ is a monad on $\mathcal{D}$ with $T(\mathcal{E}) \subseteq \mathcal{E}$, the factorization system $(\mathcal{E}, \mathcal{M})$ for $\mathcal{D}$ lifts to the category of $T$-algebras.

A factorization system $(\mathcal{E}, \mathcal{M})$ is proper if every morphism in $\mathcal{E}$ is epic and every morphism in $\mathcal{M}$ is monic. Whenever a proper factorization system $(\mathcal{E}, \mathcal{M})$ is fixed, quotients and subobjects in $\mathcal{D}$ are represented by morphisms in $\mathcal{E}$ and $\mathcal{M}$, respectively. In particular, in the situation of (1) and (2) above, we represent quotient (co-)algebras and sub(co-)algebras by homomorphisms in $\mathcal{E}$ and $\mathcal{M}$, respectively.

Closed categories. A symmetric monoidal category is a category $\mathcal{D}$ equipped with a functor $\otimes : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ (tensor product), an object $I_\mathcal{D} \in \mathcal{D}$ (tensor unit), and isomorphisms $(X \otimes Y) \otimes Z \cong X \otimes (Y \otimes Z)$, $X \otimes Y \cong Y \otimes X$, $I_\mathcal{D} \otimes X \cong X \cong X \otimes I_\mathcal{D}$, natural in $X, Y, Z \in \mathcal{D}$, satisfying coherence laws [41, Chapter VII]. $\mathcal{D}$ is closed if the endofunctor $X \otimes (-) : \mathcal{D} \to \mathcal{D}$ has a right adjoint (denoted by $[X, -]$) for every $X \in \mathcal{D}$, i.e. there is a natural isomorphism $\mathcal{D}(X \otimes Y, Z) \cong \mathcal{D}(Y, [X, Z])$.

Nominal sets. Fix a countably infinite set $\mathcal{A}$ of names, and let $\text{Perm}(\mathcal{A})$ be the group of all permutations $\pi : \mathcal{A} \to \mathcal{A}$ with $\pi(a) = a$ for all but finitely many $a$. A nominal set [51] is a set $X$ with a group action $\cdot : \text{Perm}(\mathcal{A}) \times X \to X$ subject to the following property: for each $x \in X$ there is a finite set $S \subseteq \mathcal{A}$ (a support of $x$) such that every $\pi \in \text{Perm}(\mathcal{A})$ that leaves all elements of $S$ fixed satisfies $\pi \cdot x = x$. This implies that $x$ has a least support $\text{supp}(x) \subseteq \mathcal{A}$. The idea is that $x$ is a syntactic object with bound and free variables (e.g. a $\lambda$-term modulo $\alpha$-equivalence), and that $\text{supp}(x)$ is its set of free variables. A nominal set $X$ is orbit-finite if the number of orbits (i.e. equivalence classes of the relation $x \equiv y$ iff $x = \pi \cdot y$ for some $\pi$) is finite. A map $f : X \to Y$ between nominal sets is equivariant if $f(\pi \cdot x) = \pi \cdot f(x)$ for $x \in X$ and $\pi \in \text{Perm}(\mathcal{A})$.

3 Automata in a Category

We next develop the abstract categorical notion of automaton that underlies our generic learning algorithm.

Notation 3.1. For the rest of this paper, let us fix
1. a category $\mathcal{D}$ with a proper factorization system $(\mathcal{E}, \mathcal{M})$,
2. an endofunctor $F : \mathcal{D} \to \mathcal{D}$, and
3. two objects $I, O \in \mathcal{D}$.

Definition 3.2 (Automaton (cf. [5, 10])). An $(F)$-automaton is given by an object $Q \in \mathcal{D}$ of states and three morphisms $\delta_Q : FQ \to Q$, $i_Q : I \to Q$, $f_Q : Q \to O$, representing transitions, initial states, and final states (or outputs), respectively. A homomorphism between automata $(Q, \delta_Q, i_Q, f_Q)$ and $(Q', \delta_{Q'}, i_{Q'}, f_{Q'})$ is a morphism $h : Q \to Q'$ in $\mathcal{D}$ such that the following diagrams commute:

\[
\begin{array}{ccc}
FQ & \xrightarrow{\delta_Q} & Q \\
\downarrow{Fh} & & \downarrow{h} \\
FQ' & \xrightarrow{\delta_{Q'}} & Q'
\end{array}
\quad \begin{array}{ccc}
I & \xrightarrow{i_Q} & Q \\
& \downarrow{h} & \downarrow{h} \quad I & \xrightarrow{i_{Q'}} & Q'
\end{array}
\quad \begin{array}{ccc}
Q & \xrightarrow{f_Q} & O \\
\downarrow{h} & & \downarrow{f_{Q'}} \\
Q' & \xrightarrow{f_{Q'}} & O
\end{array}
\]

Example 3.3 (\(\Sigma\)-automata). Suppose that $\mathcal{D} = (\Sigma, \otimes, I_\mathcal{D})$ is a symmetric monoidal closed category. Choosing the data $F = \Sigma \otimes (-)$, $I = I_\mathcal{D}$, and $O \in \mathcal{D}$ (arbitrary) for a fixed input alphabet $\Sigma \subseteq \mathcal{D}$ yields Goguen’s notion of a $\Sigma$-automaton [35]. In our applications, we shall work with the categories Set (sets and functions), Pos (posets and monotone maps), JSL (join-semilattices with $\bot$ and semilattice homomorphisms preserving $\bot$), K-Vec (vector spaces over field $K$ and linear maps) and Nom (nominal sets and equivariant maps). The factorization systems and monoidal structures are given in the table below. In the fourth row, $\otimes$ is the usual tensor product of vector spaces representing bilinear maps. Similarly, in the third row, $\otimes$ is the tensor product of semilattices representing bimorphisms [13], i.e. semilattice morphisms $h : A \otimes B \to C$ correspond to maps $h^* : A \times B \to C$ preserving $\vee$ and $\bot$ in each component. We choose the input alphabet $\Sigma \subset \mathcal{D}$ to be a finite set, a discrete finite poset, a free semilattice on a finite set, a finite-dimensional vector space, and the nominal set $\mathcal{A}$ of atoms, respectively, and the output object $O \in \mathcal{D}$ as shown in the
last column. Then $\Sigma$-automata are precisely classical deterministic automata [53], ordered automata [50], semilattice automata [39], linear weighted automata [31], and nominal automata [18]. See Example 3.9 and 3.10 for further details.

**Example 3.4** (Tree automata). Let $\Gamma$ be a signature and $F_1 Q = \prod_{n \in \mathbb{N}} \prod_{e \in E} Q^n$ on $\text{Set}$ the induced polynomial functor, with $\Gamma_0$, the set of $n$-ary operations in $\Gamma$. Choosing $I = \emptyset$ and $O = \{0, 1\}$, an $F_1$-automaton is a (bottom-up) tree automaton over $\Gamma$ [25], shortly a $\Gamma$-automaton. For the analogous functor $F_1$ on Pos and $O = \{0 < 1\}$, we obtain ordered $\Gamma$-automata.

In the following, we focus on adjoint automata, i.e. automata whose transition type $F$ is a left adjoint:

**Assumptions 3.5.** For the rest of this section and in Section 4, our data is required to satisfy the following conditions:

1. $\mathcal{D}$ is complete and cocomplete; in particular, $\mathcal{D}$ has an initial object $0$ and a terminal object $1$.
2. The unique morphism $! : 0 \to I$ lies in $\mathcal{M}$, and the unique morphism $1 : O \to 1$ lies in $\mathcal{E}$.
3. The functor $F : \mathcal{D} \to \mathcal{D}$ has a right adjoint $G : \mathcal{D} \to \mathcal{D}$.
4. The functor $F$ preserves quotients $(F(E)) \subseteq \mathcal{E}$.

**Example 3.6.** Every symmetric monoidal closed category $\mathcal{D}$ with $F = \Sigma \otimes -$ satisfies Assumption (3): closedness asserts precisely that $F$ has the right adjoint $G = [\Sigma, -]$. The categories $\mathcal{D}$ of Table 1 also satisfy the remaining assumptions.

**Remark 3.7.** The key feature of our adjoint setting is that automata can be dually viewed as algebras and coalgebras for suitable endofunctors. In more detail:

1. An automaton $Q$ corresponds precisely to an algebra

   \[
   (F_1Q \xrightarrow{a_Q} Q) = (I + FQ \xrightarrow{[I_Q, \delta_Q]} Q)
   \]

   for the endofunctor $F_1 = I + F$ equipped with an output morphism $f_Q : Q \to O$. Since $F_1$ preserves filtered colimits (using that the left adjoint $F$ preserves all colimits and the functor $I + (-)$ preserves filtered colimits), the initial algebra $\mu F_1$ for $F_1$ emerges as the colimit of the initial $\omega$-chain:

   \[
   \mu F_1 = \operatorname{colim}(0 \to F_1 0 \xrightarrow{F_1 i} F_1 F_1 0 \xrightarrow{F_1 j_1} F_1^2 0 \to \cdots).
   \]

   The colimit injections and the $F_1$-algebra structure on $\mu F_1$ are denoted by

   \[
   j_n : F_1^n 0 \to \mu F_1 \quad (n \in \mathbb{N}) \quad \text{and} \quad \alpha : F_1(\mu F_1) \to \mu F_1.
   \]

   For any automaton $Q$ (viewed as an $F_1$-algebra), we write

   \[
   e_Q : \mu F_1 \to Q
   \]

   for the unique $F_1$-algebra homomorphism from $\mu F_1$ into $Q$.

2. Dually, replacing $\delta_Q : F_1 Q \to Q$ by its adjoint transpose $\delta_Q^\bigtriangleup : Q \to F_1 GQ$, an automaton can be presented as a coalgebra

   \[
   (Q \xrightarrow{\gamma_Q} G_1 Q) = (Q \xrightarrow{[Q_Q, \delta_Q^\bigtriangleup]} O \times GQ)
   \]

   for the endofunctor $G_1 = O \times G$ equipped with an initial state $i_Q : I \to Q$. Since $G_1$ preserves cofiltered limits, the final coalgebra $v_{G_1}$ arises as the limit of the final $\omega$-cochain:

   \[
   v_{G_1} = \lim(1 \leftarrow G_1 O 1 \xrightarrow{G_1 \delta_1} G_1 G_1 1 \xrightarrow{G_1 \delta_1} G_1 G_1 G_1 1 \xrightarrow{G_1 \delta_1} \cdots).
   \]

   The limit projections and the $G_1$-coalgebra structure on $v_{G_1}$ are denoted by

   \[
   f_k : v_{G_1} \to G_1 k 1 \quad (k \in \mathbb{N}) \quad \text{and} \quad v_{G_1} \xrightarrow{y} G_1(v_{G_1}).
   \]

   For any automaton $Q$ (viewed as a $G_1$-coalgebra), we write

   \[
   m_Q : Q \to v_{G_1}
   \]

   for the unique $G_1$-coalgebra homomorphism into $v_{G_1}$.

**Definition 3.8** (Language). (1) A language is a morphism $L : \mu F_1 \to O$.

2. The language accepted by an automaton $Q$ is defined by

   \[
   L_Q = (\mu F_1 \xrightarrow{e_Q} Q \xrightarrow{f_Q} O).
   \]

**Example 3.9** ($\Sigma$-automata, continued). (1) In the setting of Example 3.3, the initial algebra $\mu F_1$ and the initial chain for the functor $F_1 = I_\emptyset + \Sigma \otimes -$ can be described as follows [35].

Let $\Sigma = \Sigma \otimes \Sigma \otimes \cdots \otimes \Sigma$ denote the $n$th tensor power of $\Sigma$ (where $\Sigma^0 = I_\emptyset$), and put

\[
\Sigma^\leq n = \bigsqcup_{m < n} \Sigma^m \quad (n \in \mathbb{N}) \quad \text{and} \quad \Sigma^* = \bigsqcup_{n \in \mathbb{N}} \Sigma^n.
\]

Then $\mu F_1$ is carried by the object $\Sigma^*$ of words, and the initial chain is given by the coproduct injections

\[
\Sigma^0 \hookrightarrow \Sigma^1 \hookrightarrow \Sigma^2 \hookrightarrow \Sigma^3 \hookrightarrow \cdots.
\]

(2) For the functor $G_1 = O \times [\Sigma, -]$ the final coalgebra $v_{G_1}$ is carried by the object $[\Sigma^*, O]$ of languages and we have the final cochain

\[
[\Sigma^0, O] \leftarrow [\Sigma^1, O] \leftarrow [\Sigma^2, O] \leftarrow [\Sigma^3, O] \leftarrow \cdots
\]

with connecting morphisms given by restriction. To see this, consider the contravariant functor $P = [-, O] : \mathcal{D} \to \mathcal{D}^{\text{op}}$. It is not difficult to verify that $P$ is a left adjoint (with right adjoint $P^{\text{op}}$) and that there is a natural isomorphism

\[
PF_1 \cong G_1^{\text{op}} P.
\]
If $\text{Alg} F_1$ and $\text{Coalg} G_O$ denote the categories of $F_1$-algebras and $G_O$-coalgebras, it follows [36, Theorem 2.4] that $P$ lifts to a left adjoint $\overline{P}$: $\text{Alg} F_1 \rightarrow (\text{Coalg} G_O)^{op}$ given by

$$(F_1 Q \xrightarrow{\alpha} Q) \mapsto (P Q \xrightarrow{P \alpha} PF_1 Q \cong G_O P Q).$$

Since left adjoints preserve initial objects, $\overline{P}$ maps the initial algebra $\mu F_1$ to the final coalgebra $v G_O$, i.e. one has $v G_O = P(\mu F_1)$ with the coalgebra structure $\nu = (v G_O = P(\mu F_1) \xrightarrow{\nu} PF_1(\mu F_1) \cong G_O P(\mu F_1) = G_O (v G_O))$.

Moreover, applying $P$ to the initial chain for $F_1$ yields the final coalchin for $G_O$:

$$(1 \xleftarrow{1} G_O \xrightarrow{G_O^1} G_O^2 \ldots) = (P 0 \xleftarrow{P 0} PF_1 0 \xrightarrow{PF_1i} PF_1^2 0 \ldots).$$

Since $\mu F_1 = \Sigma^*$ and $P = [-, O]$, we obtain the above description of $v G_O$ and of the final coalchin for $G_O$.

(3) For the categories of Table 1, the categorical notion of (accepted) language given in Definition 3.8 thus specializes to the familiar ones. For illustration, let us spell out the case $\mathcal{D} = \text{Set}$. A $\Sigma$-automaton in Set is precisely a classical deterministic automaton: it is given by a set $Q$ of states, a transition map $\delta_Q: \Sigma \times Q \rightarrow Q$, a map $\iota_Q: 1 \rightarrow Q$ (representing an initial state $q_0 = \iota_Q(*)$), and a map $\nu_Q: Q \rightarrow 2$ (representing a set $f_Q^{-1}(1)$ of final states). From (1) and (2) we obtain the well-known description of the initial algebra for $F_1 = 1 + \Sigma \times -$ as the set $\Sigma^*$ of finite words over $\Sigma$ (with algebra structure $\alpha: 1 + \Sigma \times \Sigma^* \rightarrow \Sigma^*$ given by $\ast \mapsto \epsilon$ and $(a, w) \mapsto aw$) and of the final coalgebra for $G_O = 2 \times [\Sigma, -]$ as the set $[\Sigma^*, 2] \cong \mathcal{P}\Sigma^*$ of all languages $L \subseteq \Sigma^*$ [55]. The unique $F_1$-algebra homomorphism $e_Q: \Sigma^* \rightarrow Q$ maps a word $w \in \Sigma^*$ to the state of $Q$ reached on input $w$. Thus, the language $L_Q = f_Q \circ e_Q$ accepted by $Q$ is the usual concept: $w$ lies in $L_Q$ if and only if $Q$ reaches a final state on input $w$.

**Example 3.10 (Nominal automata).** Our notion of automaton (Definition 3.2) has several natural instantiations to the category $\text{Nom}$ of nominal sets and equivariant maps.

(1) The simplest instance was already mentioned in Example 3.3: a $\Sigma$-automaton in $\text{Nom}$ corresponds precisely to a nominal deterministic automaton [18]. For simplicity, we choose the alphabet $\Sigma = \#$. A nominal automaton is given by a nominal set $Q$ of states, an equivariant transition map $\delta_Q: \# \times Q \rightarrow Q$, an equivariant map $\iota_Q: 1 \rightarrow Q$ (representing an equivariant initial state $q_0 \in Q$), and an equivariant map $\nu_Q: Q \rightarrow 2$ (representing an equivariant subset $F \subseteq Q$ of final states). The initial algebra $\mathbb{A}^\Sigma$ is the nominal set of words over $\#$ with group action $(a_1 \ldots a_n) = (\pi \cdot a_1) \ldots (\pi \cdot a_n)$ for $a_1 \ldots a_n \in \mathbb{A}^\Sigma$ and $\pi \in \text{Perm}(\#)$. Thus, a language $L: \mathbb{A}^\Sigma \rightarrow 2$ corresponds to an equivariant set of words over $\#$.

Nominal automata with orbit-finite state space are known to be expressively equivalent to Kaminisky and Francez’ [38] deterministic finite memory automata.

(2) Now $\text{Nom}$ carries a further symmetric monoidal closed structure, the separated product $\ast$ given on objects by

$$X \ast Y = \{ (x, y) \in X \times Y : x \neq y \},$$

where $x \neq y$ means that $\text{supp}(x) \cap \text{supp}(y) = \emptyset$. The right adjoint of $F = \# \ast (-)$ is the abstraction functor $G = [\#](-)$ [51] which maps a nominal set $X$ to the quotient of $\# \times X$ modulo the equivalence relation $\sim$ defined by $(a, x) \sim (b, y)$ iff $(a \cdot x) = (b \cdot y)$ for some (equivalently, all) $c \in \#$ with $c \neq a, b, x, y$. We write $(a \cdot x)$ for the equivalence class of $(a, x)$, which we think of as the result of binding the name $a$ in $x$. $F$-automata are precisely the separated automata recently introduced by Moerman and Rot [46]. (3) By combining the adjunctions of (1) and (2), we obtain the adjoint pair of functors $F + G$ with

$$F = \# \times (-) + \# \ast (-), \quad G = [\#, -] \times [\#](-).$$

The ensuing notion of automaton coincides with one used in Kozen et al.’s [40] coalgebraic representation of nominal Kleene algebra [33]. Such automata have two types of transitions, free transitions ($[\#, -]$) and bound transitions ($[\#](-)$). They accept bar languages [56]; putting $\mathbb{A} = \# \cup \{ \{a = \# \} \}$ (changing the original notation from $a$ to $\langle a \rangle$ for compatibility with dynamic sequences as discussed next) a bar string is just a word over $\mathbb{A}$. We consider $\langle a \rangle$ as binding $a$ to the right. This gives rise to the expected notions of free names and $\alpha$-equivalence $\equiv_\alpha$. A bar string is clean if its bound names are mutually distinct and distinct from all its free names. Simplifying slightly, we define a bar language to be an equivariant set of bar strings modulo $\alpha$-equivalence, i.e. an equivariant subset of $\mathbb{A}^\Sigma / \equiv_\alpha$. The initial algebra $\mu F_1$ is the nominal set of clean bar strings. A language in our sense is thus an equivariant set of clean bar strings; such languages are in bijective correspondence with bar languages [56].

(4) We note next that $[\#](-)$ is itself a left adjoint, our first example of a left adjoint that is not of the form $\Sigma \circ -$ for a closed structure $\Sigma$. The right adjoint $R$ is given on objects by $R X = \{ f \in [\#] X : a \# f(a) \text{ for all } a \in \# \}$ [51]. We extend the above notion of automaton with this feature, i.e. we now work with the adjoint pair $F + G$ given by

$$F = \# \times (-) + \# \ast (-) + [\#](-), \quad G = [\#, -] \times [\#](-) \times R.$$
We conclude with a discussion of minimal automata. In our applications in Section 5, we shall encounter a generalized version of $\Sigma$-automata where (1) the input object $I$ is arbitrary, not necessarily equal to the tensor unit $I_\otimes$, and (2) the automaton has a sorted object of states and consumes sorted words. This reflects the fact that the algebraic structures arising in algebraic language theory are often sorted. For brevity, we only treat the case of sorted automata in Set. Fix a set $S$ of sorts and a family of sets $\Sigma = (\Sigma_i, i \in S)$; we think of the elements of $\Sigma_i$ as letters with domain sort $s$ and codomain sort $t$. We instantiate our setting to the adjoint pair $F \dashv G$: $\text{Set}^S \to \text{Set}^S$ defined as follows for $Q \in \text{Set}^S$ and $s, t \in S$:

\[(FQ)_s = \prod_{s \in S} \Sigma_{s,t} \times Q_s, \quad (GQ)_s = \prod_{s \in S} \Sigma_{s,t} \times Q_t.\]

Choosing $I \in \text{Set}^S$ arbitrary and the output object $O = 2$, the $S$-sorted set with two elements in each component, an $F$-automaton is a \textit{sorted} $\Sigma$-automaton. It is given by an $S$-sorted set of states $Q$, transitions $\delta_{Q,s,t} : \Sigma_{s,t} \times Q_t \to Q_t$ ($s, t \in S$), initial states $i : I \to Q$ and an output map $f_Q : Q \to 2$ (representing an $S$-sorted set of final states). The initial algebra $\mu F I$ is the $S$-sorted set of all well-sorted words over $\Sigma$ with an additional first letter from $I$. More precisely, $(\mu F I)_s$ consists of all words $x a_1 \ldots a_n$ with $x \in \prod_{s \in S} I_s$ and $a_1, \ldots, a_n \in \prod_{s \in S} \Sigma_{s,t}$ such that the sorts of consecutive letters match, i.e., there exist sorts $s = s_0, s_1, \ldots, s_n = t \in S$ such that $x \in I_s$ and $a_i \in \Sigma_{s_i+1,s_i}$ for $i = 1, \ldots, n$. In particular, in the single-sorted case we have $\mu F I = I \times \Sigma^*$. For any well-sorted input word $w = x a_1 \ldots a_n$ one obtains the run

\[x \xrightarrow{a_1} q_0 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n\]

in $Q$ where $q_0 = i_{\Sigma^*}(x)$ and $q_i = \delta_{Q,s_i+1,s_i}(a_i, q_{i-1})$ for $i = 1, \ldots, n$, and $w$ is accepted if and only if $q_n$ is a final state.

We conclude with a discussion of minimal automata.

**Definition 3.12** (Minimal automaton). An automaton $Q$ is called (1) \textit{reachable} if the unique $F_I$-algebra homomorphism $e_Q : \mu F I \to Q$ lies in $\mathcal{E}$, and (2) \textit{minimal} if it is reachable and for every reachable automaton $Q'$ with $L_Q = L_{Q'}$, there exists a unique automata homomorphism from $Q'$ to $Q$.

**Theorem 3.13.** For every language $L$ there exists a minimal automaton $\text{Min}(L)$ accepting $L$, unique up to isomorphism.

**Proof sketch.** We describe the construction of the minimal automaton. By equipping $F_I$ with the final states $L : \mu F I \to O$, we can view $\mu F I$ as a $G_O$-coalgebra. Consider the $(\mathcal{E}, M)$-factorization of the unique coalgebra homomorphism $\mu F I$:

\[m_{\mu F I} = (\mu F I \xrightarrow{\text{min}(L)} \text{Min}(L) \xrightarrow{m_{\text{min}(L)}} \nu G_O).\]

The object $\text{Min}(L)$ can be uniquely equipped with a automaton structure for which $\text{min}(L)$ is an $F_I$-algebra homomorphism and $m_{\text{min}(L)}$ is a $G_O$-coalgebra homomorphism. This automaton is the minimal acceptor for $L$. $\square$

The minimization theorem and its proof are closely related to the classical work of Arbib and Manes [10] on the minimal realization of dynamorphisms, i.e., $F$-algebra homomorphisms from $\mu F I$ into $\nu G_O$. Under different assumptions on the type functor $F$ and the base category $\mathcal{D}$ (e.g., co-wellpoweredness), minimization results were also established by Adámek and Trnková [3] and, recently, by van Heerdt et al. [62].

## 4 A Categorical $L^*$ Algorithm

To motivate our learning algorithm for adjoint automata, we recall Angluin’s classical $L^*$ algorithm [8] for learning an unknown $\Sigma$-automaton $Q$ in Set. The algorithm assumes that the learner has access to an oracle (the \textit{teacher}) that can be asked two types of questions:

1. \textit{Membership queries}: given a word $w \in \Sigma^*$, is $w \in L_Q$?
2. \textit{Equivalence queries}: given an automaton $H$, is $L_H = L_Q$?

If the answer in (2) is “no”, the teacher discloses a counterexample, i.e., a word $w \in L_Q \setminus L_H$ or $L_H \setminus L_Q$, to the learner.

The idea of $L^*$ is to compute a sequence of approximations of the unknown automaton $Q$ by considering finite (co-)restrictions of the morphism $m_Q : \nu G_O$, as indicated by the diagram below. Note that the kernel of $m_Q : \nu G_O$ is precisely the well-known Nerode congruence of $L_Q$.

\[
\begin{array}{cccccccc}
\Sigma^0 & \to & \ldots & 
\Sigma^N & \to & \Sigma^{N+1} & \to & \ldots & \Sigma^* \\
S & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
H_{S,T} & = & H_{S,T} & \to & H_{S,T} & \to & H_{S,T} & \to & H_{S,T} \\
\quad & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow \\
[T, 2] & \leftrightarrow & [\Sigma^0, 2] & \leftrightarrow & [\Sigma^K, 2] & \leftrightarrow & [\Sigma^{K+1}, 2] & \leftrightarrow & [\Sigma^*, 2] \\
\end{array}
\]

In more detail, the algorithm maintains a pair $(S, T)$ of finite sets $S, T \subseteq \Sigma^*$ (“states” and “tests”). For any such pair, the restriction of $m_Q : \nu G_O$ to the domain $S$ and codomain $[T, 2]$, $h_{S,T} : S \to [T, 2]$, $h_{S,T}(s)(t) = L_Q(st)$ for $s \in S, t \in T$, is called the \textit{observation table} for $(S, T)$. It is usually represented as an $|S| \times |T|$-matrix with binary entries. The learner can compute $h_{S,T}$ via membership queries. The pair $(S, T)$ is \textit{closed} if for each $s \in S$ and $a \in \Sigma$ there exists $s' \in S$ with

$$h_{S,\Sigma a, T}(sa) = h_{S,T}(s').$$

It is \textit{consistent} if, for all $s, s' \in S$,

$$h_{S,T}(s) = h_{S,T}(s') \text{ implies } h_{S,\Sigma a, T}(s) = h_{S,\Sigma a, T}(s').$$

Initially, one puts $S = T = \{e\}$. If at some stage the pair $(S, T)$ is not closed or not consistent, either $S$ or $T$ can be extended by invoking one of the following two procedures:
We shall now develop all ingredients of $L$ on top of our Assumptions 3.5, we re-
Assumptions 4.1.

The algorithm runs in polynomial time w.r.t. the size of $h$, i.e. $h(S,T)$ is closed and consistent. Then, one constructs an automaton $H_{S,T}$, the hypothesis for $(S,T)$. Its set of states is the image $h(S,T)$, the transitions $\delta(S,T): \Sigma \times H_{S,T} \rightarrow H_{S,T}$ are given by $\delta(sa,a) = h(S,T)(sa)$ for $s \in S$ and $a \in \Sigma$, the initial state is $h(S,T)(\epsilon)$, and a state $h(S,T)(s)$ is final if $s \in L_Q$ (i.e. $h(S,T)(s) = 1$). Note that the well-definedness of $h(S,T)$ is equivalent to $(S,T)$ being closed and consistent.

The learner now tests whether $L_{H_{S,T}} = L_Q$ by asking an equivalence query. If the answer is "yes", the algorithm terminates successfully; otherwise, the teacher’s counterexample and all its prefixes are added to $S$. In summary:

**L' Algorithm**

**Goal:** Learn an automaton equivalent to an unknown automaton $Q$.

(0) Initialize $S = T = \{\epsilon\}$.

(1) While $(S,T)$ is not closed or not consistent:
   (a) If $(S,T)$ is not closed: Extend $S$.
   (b) If $(S,T)$ is not consistent: Extend $T$.

(2) Construct the hypothesis $H_{S,T}$.
   (a) If $L_{H_{S,T}} = L_Q$: Return $H_{S,T}$.
   (b) If $L_{H_{S,T}} \neq L_Q$: Put $S := S \cup C$, where $C$ is the set of prefixes of the teacher’s counterexample.

(3) Go to (1).

The algorithm runs in polynomial time w.r.t. the size of the minimal automaton $\text{Min}(L_Q)$ and the length of the longest counterexample provided by the teacher. The learned automaton (i.e. the correct hypothesis returned in Step (2a)) is isomorphic to $\text{Min}(L_Q)$. Correctness and termination rest on the invariant that $S$ is prefix-closed and $T$ is suffix-closed. Note that if $T \subseteq \Sigma^*, 2 \rightarrow [T,2]$ given by restriction. In the following, $T$ is represented via this quotient.

We shall now develop all ingredients of $L'$ for adjoint $F$-automata. This requires additional assumptions, which hold for all the functors discussed in Example 3.3, 3.10 and 3.11:

**Assumptions 4.1.** On top of our Assumptions 3.5, we require for the rest of this section that $F_I = I + F$ preserves subobjects ($F_I(M) \subseteq M$) and pullbacks of $M$-morphisms, and that $G_O = O \times G$ preserves quotients ($G_O(E) \subseteq E$).

Our categorical learning algorithm generalizes (1) to the diagram shown below, where the upper and lower part are given by the initial chain for $F_I$ and the final cochain for $G_O$: 

$$
\begin{array}{cccc}
F^0_I & \cdots & F^N_I & \cdots \\
\downarrow j & & \downarrow j & \\
F^0_I & \cdots & F^{N+1}_I & \cdots \\
\downarrow i & & \downarrow i & \\
F^N_I & \cdots & F^{N+1}_I & \cdots \\
\downarrow \epsilon & & \downarrow \epsilon & \\
F^N_I & \cdots & F^{N+1}_I & \cdots \\
\downarrow \mu & & \downarrow \mu & \\
F^N_I & \cdots & F^{N+1}_I & \cdots \\
\end{array}
$$

The algorithm maintains a pair $(s,t)$ of an $F_I$-subcoalgebra and a $G_O$-quotient algebra $s: (S,\sigma) \rightarrow (F^N_I, F^N_I)$, $t: (G^K_O, G^K_O) \rightarrow (T,\tau)$, with $N,K > 0$. For $\Sigma$-automata in Set, this means precisely that $S$ is a prefix-closed subset of $\Sigma^\leq N$, and that $T$ represents a suffix-closed subset of $\Sigma^{< K}$.

Initially, one takes $N = K = 1$, $s = \text{id}_I$ and $t = \text{id}_O$, which corresponds to Step (0) of the original $L'$ algorithm.

**Remark 4.2.** By Assumptions 3.5(2) and 4.1, every subcoalgebra $s: (S,\sigma) \rightarrow (F^N_I, F^N_I)$ induces the two subcoalgebras $$(s, \sigma) \rightarrow (F^{N+1}_I, F^{N+1}_I) \rightarrow (F^N_I, F^N_I).$$ In the case of $\Sigma$-automata in Set, the construction of these two subcoalgebras corresponds to viewing a prefix-closed subset $S \subseteq \Sigma^\leq N$ as a subset of $\Sigma^{< N+1}$, and to extending $S$ to the prefix-closed subset $S \cup \{\epsilon\} = S \cup S \subseteq \Sigma^{< N+1}$. A dual remark applies to quotient algebras of $(G^K_O, G^K_O)$.

**Definition 4.3 (Observation table).** Let $(s,t)$ be a pair as in (3), and let $Q$ be an automaton. The observation table for $(s,t)$ w.r.t. $Q$ is the morphism

$$
h^Q_{s,t} = (S \xrightarrow{\epsilon} F^0_I \xrightarrow{j} F^N_I \xrightarrow{\mu} Q \xrightarrow{\epsilon} Q \xrightarrow{\mu} G_O \xrightarrow{\nu} G_O \xrightarrow{j} T).$$

Its $(E,M)$-factorization is denoted by

$$
h^Q_{s,t} = (S \xrightarrow{\epsilon} H^Q_{s,t} \xrightarrow{\mu} T).$$

In the following, we fix $Q$ (the unknown automaton to be learned) and omit the superscripts $(-)^Q$.

**Remark 4.4.** In our categorical setting, membership queries are replaced by the assumption that the learner can compute the observation table $h^Q_{s,t}$ for each pair $(s,t)$. Importantly,
this morphism depends only on the language of \( Q \): one can show that for every automaton \( Q' \) with \( L_Q = L_{Q'} \) one has \( m_Q \cdot e_Q = m_{Q'} \cdot e_{Q'} \), whence \( h_{s,t}^Q = h_{s,t}^{Q'} \).

**Definition 4.5** (Closed/Consistent pair). For any pair \((s, t)\) as in (3), let \( cl_{s,t} \) and \( cs_{s,t} \) be the unique diagonal fill-ins making all parts of the diagram below commute:

\[
\begin{array}{ccc}
S & \xrightarrow{\sigma} & T \\
\downarrow{cl_{s,t}} & & \uparrow{m_{cs_{s,t}}}
\end{array}
\]

The pair \((s, t)\) is closed if \( cl_{s,t} \) is an isomorphism, and consistent if \( cs_{s,t} \) is an isomorphism.

If \((s, t)\) is not closed or not consistent, at least one of the two dual procedures below applies. "Extend \( s \)" replaces \( S \rightarrow F_0 T \) by a new subalgebra \( S' \rightarrow F_0 T \), i.e. it moves to the right in the initial chain for \( F_1 \). Analogously, "Extend \( t \)" replaces \( G_0 T \rightarrow T \) by a new quotient algebra \( G_0 T \rightarrow T \), and thus moves to the right in the finial cochain for \( G_0 \).

**Extend \( s \)**

**Input:** A pair \((s, t)\) as in (3) that is not closed.

0. Choose an object \( S' \) and \( M \)-morphisms \( s_0 : S \rightarrow S' \) and \( s_1 : S' \\rightarrow F_1 S \) such that

\[ \sigma = s_1 \cdot s_0 \quad \text{and} \quad e_{F_1 s, t} \cdot s_1 \in \mathcal{E}. \]

1. Replace \( s : (S, \sigma) \rightarrow (F_0 T, F_1 T) \) by the subcoalgebra \( F_1 S \cdot s_1 : (S', F_1 s_0 \cdot s_1) \rightarrow (F_0 T, F_1 T) \).

**Remark 4.6.** (1) One trivial choice in Step (0) is \( S' = F_1 S \) so \( s_0 = \sigma \) and \( s_1 = id \).

To get an efficient implementation of the algorithm, one aims to choose the subobject \( s_1 : S' \\rightarrow F_1 S \) as small as possible.

(2) The update of \( s \) in Step (1) is well-defined, i.e. \( F_1 S \cdot s_1 \) is a subcoalgebra. Indeed, the commutative diagram below shows that \( F_1 S \cdot s_1 \) is a coalgebra homomorphism:

\[
\begin{array}{cccc}
F_1 S & \xrightarrow{F_1 s_0} & F_1 T & \xrightarrow{F_1 e_{F_1 s, t}} \\
\downarrow{s_1} & & \uparrow{F_1 F_1 s} & \\
S & \xrightarrow{s} & F_1 S & \xrightarrow{F_1 \sigma} S'
\end{array}
\]

Moreover, since \( s, s_1 \in M \) and \( F_1 \) preserves \( M \) (see Assumptions 4.1), we have \( F_1 S \cdot s_1 \in M \).

(3) In the case of \( \Sigma \)-automata in \( \text{Set} \), the condition \( \sigma = s_1 \cdot s_0 \) states that \( S \subseteq S' \subseteq S \cup \Sigma S = S \Sigma \cup \{ t \} \). The condition \( e_{F_1 s, t} \cdot s_1 \in \mathcal{E} \) asserts that given \( s \in S \) and \( a \in \Sigma \) such that \( h_{S,S', T}(sa) \neq h_{S, T}(r) \) for all \( r \in S \), there exists \( s' \in S' \) with \( h_{S,S',T}(sa) = h_{S,S',T}(s') \). Thus, “Extend \( s \)” subsumes several executions of “Extend \( S \)” in the original \( L \)-algorithm.

**Extend \( t \)**

**Input:** A pair \((s, t)\) as in (3) that is not consistent.

0. Choose an object \( T' \) and \( E \)-morphisms \( t_0 : G_0 T \rightarrow T' \) and \( t_1 : T' \rightarrow T \) such that

\[ \tau = t_1 \cdot t_0 \quad \text{and} \quad t_0 \cdot m_{cs_{s,t}} \in M. \]

1. Replace \( t : (G_0 T, Q_0 T) \rightarrow (T, \tau) \) by the quotient algebra \( t_0 \cdot G_0 T : (G_0 T, Q_0 T) \rightarrow (T', t_0 \cdot G_0 T) \).

**Remark 4.7.** (1) Dually to Remark 4.6, a trivial choice in Step (0) is given by \( T' = G_0 T \), \( t_0 = id \), \( t_1 = \tau \), and Step (1) is well-defined, i.e. \( t_0 \cdot G_0 T \) is a quotient algebra.

(2) In the case of \( \Sigma \)-automata in \( \text{Set} \), we view the quotients \( T \) and \( T' \) as subsets of \( \Sigma^{< \infty} \) and \( \Sigma^{< \infty} \), respectively, using the above identification between subsets and quotients. The condition \( \tau = t_1 \cdot t_0 \) then states that \( T \subseteq T' \subseteq T \cup \Sigma T \). The condition \( t_0 \cdot m_{cs_{s,t}} \in M \) states that every inconsistency admits a witness in \( T' \): given \( s, s' \in S \) with \( h_{S, T}(s) = h_{S, T}(s') \) but \( h_{S, T}(s') \neq h_{S, T}(t'(s')) \), there exists \( t' \in T' \) with \( h_{S, T}(s'(t')) \neq h_{S, T}(t'(s')) \). Thus, “Extend \( t \)” subsumes several executions of “Extend \( T \)” in the original \( L \)-algorithm.

If \((s, t)\) is both closed and consistent, then we can define an automaton structure on \( H_{s,t} \):

**Definition 4.8** (Hypothesis). Let the pair \((s, t)\) be closed and consistent. The **hypothesis** for \((s, t)\) is the automaton

\[
(H_{s,t}, \delta_{s,t}, i_{s,t}, F_{s,t})
\]

with states \( H_{s,t} \) and structure defined below. Here, \( \text{inl}/\text{inr} \) are coproduct injections, \( \text{outl}/\text{outr} \) are product projections, and \(^*\) denotes adjoint transpose along the adjunction \( F \vdash G \):

(1) The transitions \( \delta_{s,t} : FH_{s,t} \rightarrow H_{s,t} \) are given by the diagonal fill-in of the commutative square

\[
\begin{array}{ccc}
FH_{s,t} & \xrightarrow{F_{s,t}} & FH_{s,t} \\
\downarrow{\delta_{s,t}} & & \uparrow{\delta_{s,t}} \\
H_{s,t} & \xrightarrow{i_{s,t}} & T
\end{array}
\]

with the two vertical morphisms defined by

\[
\begin{array}{ccc}
l_{s,t} = (FS \xrightarrow{\text{inl}} I + FS = F_1 S \xrightarrow{e_{F_1 s, t}} H_{s,t} \xrightarrow{m_{cs_{s,t}}} H_{s,t}),
\end{array}
\]

\[
\begin{array}{ccc}
r_{s,t} = (H_{s,t} \xrightarrow{cs_{s,t}^{-1}} H_{s,t}G_0 T \xrightarrow{m_{cs_{s,t}}} G_0 T = O \times GT \xrightarrow{\text{outl}} OT).
\end{array}
\]

(2) The initial states are

\[
\begin{array}{ccc}
i_{s,t} = (I \xrightarrow{\text{inl}} I + FS = F_1 S \xrightarrow{e_{F_1 s, t}} H_{s,t} \xrightarrow{m_{cs_{s,t}^{-1}}} H_{s,t}).
\end{array}
\]

(3) The final states are

\[
\begin{array}{ccc}
f_{s,t} = (H_{s,t} \xrightarrow{cs_{s,t}^{-1}} H_{s,t}G_0 T \xrightarrow{m_{cs_{s,t}}} G_0 T = O \times GT \xrightarrow{\text{outl}} O).
\end{array}
\]
Remark 4.9. The square defining \( \delta_{s,t} \) commutes: both legs can be shown to be equal to \( FS \xrightarrow{in} I + FS = F_I S \xrightarrow{h_F \ast_{s,t}} T \).

The idea of constructing the \( F \)-algebra structure of a hypothesis via diagonal fill-in originates in the abstract framework of CALF \([63]\). An important difference is that in the latter the existence of the two vertical morphisms of the corresponding square is postulated, while in our present setting features a concrete description of \( l_{s,t} \) and \( r_{s,t} \).

Recall that in \( L^* \), if a hypothesis \( H_{S,T} \) is not correct (i.e. \( L_{H_{S,T}} \neq L_Q \)), the learner receives a counterexample \( w \in \Sigma \) from the teacher and adds the set \( C \) of all its prefixes to \( S \). Identifying the word \( w \) with this set, the concept of a counterexample has the following categorical version:

**Definition 4.10** (Counterexample). Let \((s,t)\) be closed and consistent. A counterexample for \( H_{s,t} \) is a subcoalgebra \( c : (C,\gamma) \to (F^M_I 0, F^M_I i) \) for some \( M > 0 \) such that \( H_{s,t} \) and \( Q \) do not agree on inputs from \( C \), that is,

\[
L_{H_{s,t}} \cdot j_M \cdot c \neq L_Q \cdot j_M \cdot c.
\]

**Remark 4.11.** (1) If \( L_{H_{s,t}} \neq L_Q \), then a counterexample always exists. Indeed, since the colimit injections \( j_M : F^M_I 0 \to \mu F_I \) are jointly epimorphic, one has \( L_{H_{s,t}} \cdot j_M \neq L_Q \cdot j_M \) for some \( M > 0 \) and thus \( (C,\gamma) = (F^M_I 0, F^M_I i) \) is a counterexample. To obtain an efficient algorithm, it is often assumed that the teacher delivers a minimal counterexample, i.e. \( M \) is minimal and no proper subcoalgebra is a counterexample. (2) Given a counterexample \( c : (C,\gamma) \to (F^M_I 0, F^M_I i) \), one can add \( c \) to the subcoalgebra \( s : (S,\sigma) \to (F^N_J 0, F^N_J i) \) as follows: by Remark 4.2, we can assume that \( M = N \), and then form the supremum \( s \lor c : (S \lor C,\sigma \lor \gamma) \to (F^N_J 0, F^N_J i) \) of \( s \) and \( c \) in the lattice of subcoalgebras of \( (F^N_J 0, F^N_J i) \), viz. the image of the homomorphism \( [s,c] : S + C \to F^I_I 0 \).

With all these ingredients at hand, we obtain the following abstract learning algorithm for adjoint \( F \)-automata:

**Generalized \( L^* \) Algorithm**

**Goal:** Learn an automaton equivalent to an unknown automaton \( Q \).

(0) Initialize \( N = K = 1, s = id_I \) and \( t = id_O \).

(1) While \((s,t)\) is not closed or not consistent:

(a) If \((s,t)\) is not closed: Extend \( s \).

(b) If \((s,t)\) is not consistent: Extend \( t \).

(2) Construct the hypothesis \( H_{s,t} \).

(a) If \( L_{H_{s,t}} = L_Q \): Return \( H_{s,t} \).

(b) If \( L_{H_{s,t}} \neq L_Q \): Replace the subcoalgebra \( s \) by \( s \lor c \), where \( c \) is the teacher’s counterexample.

(3) Go to (1).

To prove the termination and correctness of Generalized \( L^* \), we need a finiteness assumption on the unknown automaton \( Q \). We call a \( \mathcal{D} \)-object \( Q \) Noetherian if both its poset of subobjects (ordered by \( m \leq m' \) iff \( m = m' \cdot p \) for some \( p \)) and that of its quotients (ordered by \( e \leq e' \) iff \( e = q \cdot e' \) for some \( q \)) contain no infinite strictly ascending chains.

**Theorem 4.12.** If \( Q \) is Noetherian, then the generalized \( L^* \) algorithm terminates and returns \( \text{Min}(L_Q) \).

**Remark 4.13.** Under a slightly stronger finiteness condition on \( Q \), we obtain a complexity bound. Suppose that \( Q \) has finite height \( n \), that is, \( n \) is the maximum length of any strictly ascending chain of subobjects or quotients of \( Q \). Then Steps (1a), (1b) and (2b) are executed \( O(n) \) times.

**Example 4.14.** In \( \mathcal{D} = \text{Set}, \text{Pos}, \text{JSL},\ K\text{-Vec}, \text{and Nom} \), the Noetherian objects are precisely the finite sets, finite posets, finite semilattices, finite-dimensional vector spaces and orbit-finite nominal sets. The height of \( Q \) is equal to the number of elements of \( Q \) (for \( \mathcal{D} = \text{Set}, \text{Pos} \)) or the dimension (for \( \mathcal{D} = K\text{-Vec} \)). For \( \mathcal{D} = \text{Nom} \), the height of an orbit-finite set \( Q \) can be shown to be polynomial in the number of orbits of \( Q \) and \( \max \{| \text{supp}(q) | \mid q \in Q \} \), using upper bounds on the length of subgroup chains in symmetric groups \([11]\).

**Remark 4.15.** In the generalized \( L^* \) algorithm, counterexamples are added to \( S \). Dually, one may opt to add them to \( T \) instead; for \( \Sigma \)-automata in \( \text{Set} \), this corresponds to a modification of Angluin’s algorithm due to Maler and Pnueli \([42]\) that makes it possible to avoid inconsistent observation tables, i.e. all tables constructed in the modified algorithm are consistent. In this dual approach, the accepted language of an automaton \( Q \) is defined coalgebraically as the morphism

\[
L'_{Q} = (I \xrightarrow{i_Q} Q \xrightarrow{m_Q} \nu G_O),
\]

and a counterexample is a quotient algebra \( c : (G_{O1}^M, G_{O1}^M !) \to (C,\gamma) \) for some \( M > 0 \) such that \( c \cdot j_{\gamma} \cdot L_{H_{s,t}} \neq c \cdot j_{\gamma} \cdot L'_{Q} \). In Step (2b), a counterexample \( c \) is added to the quotient algebra \( t : (G_{O1}^M, G - O^M) \to (T,\tau) \) by forming the supremum of \( t \) and \( c \). To guarantee termination, our original requirement that \( F_I \) preserves pullbacks of \( M \)-morphisms (see Assumptions 4.1) needs to be replaced by the dual requirement that \( G_O \) preserves pushouts of \( G \)-morphisms.

**Remark 4.16.** We elaborate on the connection between Generalized \( L^* \) and the learning algorithm for coalgebras due to Barlocco et al. \([14]\). The latter is concerned with coalgebras whose semantics is given in terms of a \textit{coalgebraic logic}, i.e. a natural transformation \( \delta : L^\text{op} P \to PB \) where \( L : \mathcal{A} \to \mathcal{A} \) and \( B : \mathcal{B} \to \mathcal{B} \) are endofunctors and \( P : \mathcal{B} \to \mathcal{A}^\text{op} \) is a left adjoint (see the left-hand square below).

```
\[
\begin{array}{ccc}
L^\text{op} P & \xrightarrow{P} & B \\
\xrightarrow{id} & \downarrow & \xrightarrow{id} \\
(\mathcal{A}^\text{op} P) & \xrightarrow{P^\text{op}} & G_O \\
\end{array}
\]
```

Here, \( L \) represents the syntax (usually modalities over a propositional base logic embodied by \( \mathcal{A} \)), and \( B \) the behaviour (defining the branching type of coalgebras on \( \mathcal{B} \)).
The coalgebraic semantics of \( F \)-automata corresponds to the trivial logic shown in the right-hand square. In this sense, \( F \)-automata are formally covered by the framework of [14].

While Generalized \( L^* \) is based on Angluin’s \( L^* \) algorithm, the coalgebraic learning algorithm in op. cit. generalizes Maler and Pnueli’s approach, and thus needs to keep observation tables consistent (Remark 4.15). To this end, tables are required to satisfy a property called \textit{sharpness}, which entails that the existence of extensions of non-closed tables is nontrivial and can only be guaranteed under strong assumptions on epimorphisms in the base category (e.g., all epimorphisms must split). Thus, the algorithm is effectively limited to coalgebras in \( \text{Set} \) and does not apply, e.g., to \( \Sigma \)-automata in \( \Nom \); see Appendix. In our Generalized \( L^* \), no such assumptions are needed since table extensions always exist (Remark 4.6). This makes our algorithm applicable in categories beyond \( \text{Set} \), including the ones in Example 3.3.

Generalized \( L^* \) provides a unifying perspective on known learning algorithms for several notions of deterministic automata, including classical \( \Sigma \)-automata (\( \mathcal{D} = \text{Set} \) [8]), linear weighted automata (\( \mathcal{D} = \text{K-Vec} \) [12]) and nominal automata (\( \mathcal{D} = \Nom \) [21, 47]). For \( \mathcal{D} = \text{JSL} \), finite semilattices automata can be interpreted as \textit{nondeterministic} finite automata by means of an equivalence between the category of finite semilattices and a suitable category of finite closure spaces and relational morphisms [3, 48]. For any regular language \( L \), the minimal \( \Sigma \)-automaton \( \text{Min}(L) \) in \( \text{JSL} \) corresponds under this equivalence to the minimal \textit{residual finite state automaton} (RFSA) [28], a canonical nondeterministic acceptor for \( L \) whose states are the join-irreducible elements of \( \text{Min}(L) \). Consequently, the NL* algorithm for learning RFSA due to Bollig et al. [20] is also subsumed by our categorical setting. We note that although NL* learns a minimal RFSA, the intermediate hypotheses arising in the algorithm are not necessarily RFSA, but general nondeterministic finite automata. Our categorical perspective provides an explanation of this phenomenon: it shows that NL* implicitly computes deterministic finite automata over \( \text{JSL} \), and not every such automaton corresponds to an RFSA.

Finally, our algorithm instantiates to new learning algorithms for nominal languages with name binding, including languages of dynamic sequences (Example 3.10), and for sorted languages (Example 3.11). A special instance of sorted automata where all transitions are sort-preserving (i.e., \( \Sigma_{s,t} = \emptyset \) for \( s \neq t \)) appeared in the work of Moerman [45] on learning product automata.

In each of the above settings, in order to turn Generalized \( L^* \) into a concrete algorithm, one only needs to provide a suitable data structure for representing observation tables \( h_{s,t} \) by finite means, and a strategy for choosing the objects \( S' \) and \( T' \) in the procedures "Extend \( s \)" and "Extend \( t \". We emphasize that these design choices can be non-trivial and depend on the specific structure of the underlying category \( \mathcal{D} \). The typical approach is to represent the map \( h_{s,t} : S \rightarrow T \) by restricting the objects \( S \) and \( T \) to finite sets of generators. For instance, finite-dimensional vector spaces can be represented by their bases (\( \mathcal{D} = \text{K-Vec} \)), finite semilattices by their join-irreducible elements (\( \mathcal{D} = \text{JSL} \)) and orbit-finite sets by subgroups of finite symmetric groups (\( \mathcal{D} = \Nom \)).

Our above results demonstrate, however, that the core of our learning algorithm is independent from such implementation details; in particular, its correctness and termination, and parts of the complexity analysis, always come for free as instances of the general results in Theorem 4.12 and Remark 4.13. In this way, the categorical approach provides a clean separation between generic structures and design choices tailored to a specific application. This leads to a simplified derivation of learning algorithms in new settings.

## 5 Learning Monad-Recognizable Languages

In this section, we investigate languages recognizable by monad algebras and show that the task of learning them can be reduced to learning \( F \)-automata.

**Notation 5.1.** Fix a monad \( T = (T, \mu, \eta) \) on \( \mathcal{D} \) that preserves quotients \( (T(E) \subseteq E) \). We continue to work with the fixed objects \( I, O \in \mathcal{D} \) of inputs and outputs (with \( I \) now thought of as an input alphabet, so not normally the monoidal unit). Finally, we fix a full subcategory \( \mathcal{D}_f \subseteq \mathcal{D} \) closed under subobjects and quotients, and call the objects of \( \mathcal{D}_f \) the \textit{finite} objects of \( \mathcal{D} \).

**Example 5.2.** Choose \( \text{Set}_f \), \( \text{Pos}_f \), \( \text{JSL}_f \), \( \text{K-Vec}_f \) and \( \Nom_f \) to be the class of all Noetherian objects (see Example 4.14). Our monads of interest model formal languages:

| \( \mathcal{D} \) | \( T \) |
|---|---|
| \( \text{Set} \) | \( T_s X = X^* \) |
| \( \text{Set}_f \) | \( T_{\omega}(X, Y) = (X^*, X^{up} + X^* Y) \) |
| \( \text{Set} \) | \( T_I X = \Gamma \)-trees over \( X \) |
| \( \text{JSL} \) | \( T_X = \text{free idempotent semiring on } X \) |
| \( \text{K-Vec} \) | \( T_X = \text{free } K \text{-algebra on } X \) |
| \( \text{Pos} \) | \( T_S X = \text{free stabilization algebra on } X \) |
| \( \Nom \) | \( T_X X = X^* \) |

In the second row, \( X^{up} = \{ vw^\omega \mid v \in X^*, w \in X^* \} \) denotes the set of ultimately periodic words over \( X \), and in the third row, \( \Gamma \) is a finitary algebraic signature. Finite algebras for the above seven monads correspond to finite semigroups, finite Wilke algebras [65], finite \( \Gamma \)-algebras, finite-dimensional \( k \)-algebras, finite stabilization algebras [26], and orbit-finite nominal monoids [19], respectively.

In the present setting, we shall consider the following generalized concept of a language:

**Definition 5.3** (Language). A language is a morphism \( L : TI \rightarrow O \) in \( \mathcal{D} \).
It is called recognizable if there exists a T-homomorphism \( e : (T_I, \mu_I) \rightarrow (A, \alpha) \) into a finite T-algebra \((A, \alpha)\) and a morphism \( p : A \rightarrow O \) in \( \mathcal{D} \) with \( L = p \cdot e \).

In this case, we say that \( e \) recognizes \( L \) (via \( p \)).

**Remark 5.4.** The above definition generalizes the concepts of the previous sections. Indeed, if \( F \) is functor for which the free monad \( T_F \) (see Section 2) exists, then a language \( L : T_F I \rightarrow O \) in the sense of Definition 5.3 is precisely a language \( L : \mu F_1 \rightarrow O \) in the sense of Definition 3.8. Moreover, since the categories of F-algebras and \( T_F \)-algebras are isomorphic, \( L \) is \( T_F \)-recognizable if and only if \( L \) is regular, i.e. accepted by some finite T-automaton.

**Example 5.5.** Many important automata-theoretic classes of languages can be characterized algebraically as recognizable languages for a monad. For the monads of Example 5.2 we obtain the following languages:

| \( \mathcal{D} \) | \( T \) | T-recognizable languages |
|-----------------|-------|------------------------|
| \text{Set} \* | \( T_\alpha \) | regular languages \[50\] |
| \text{Set}^2 | \( T_\omega \) | \( \omega \)-regular languages \[49\] |
| \text{Set} | \( T_\Gamma \) | tree languages over \( \Gamma \) \[25\] |
| \text{JSL} | \( T_\nu \) | regular languages \[52\] |
| \text{K-Vec} | \( T_\nu \) | recognizable weighted languages \[54\] |
| \text{Pos} | \( T_\lambda \) | regular cost functions \[23\] |
| \text{Nom} | \( T_\nu \) | monoid-recognizable data languages \[19\] |

In the following, we focus on \( (\omega) \)-regular languages and cost functions; see \[58, 60\] for details on the remaining examples.

1. For the semigroup monad \( T_\alpha \) on \text{Set} we obtain the classical concept of algebraic language recognition: a language \( L \subseteq \Gamma^* \) is recognizable if there exists a semigroup morphism \( e : \Gamma^* \rightarrow S \) into a finite semigroup \( S \) and a subset \( P \subseteq S \) with \( L = e^{-1}[P] \). Recognizable languages are exactly the \((\epsilon\text{-free})\) regular languages \[50\]. In fact, the expressive equivalence between \( \Sigma \)-automata in \text{Set} and semigroups generalizes to \( \Sigma \)-automata in symmetric monoidal closed categories \[4\].

2. Languages of infinite words can be captured algebraically as follows. A Wilke algebra \[65\] is a two-sorted set \((S_x, S_o)\) with a product \( \cdot : S_x \times S_x \rightarrow S_x \), a mixed product \( \cdot : S_x \times S_o \rightarrow S_o \) and a unary operation \((\cdot)^o : S_x \rightarrow S_o\) subject to the laws

\[
(st)u = s(tu), \quad (st)z = s(tz), \quad (s(t))^o = (st)^o, \quad (s^n)^o = s^n^o,
\]

for all \( s, t, u \in S_x, z \in S_o \) and \( n > 0 \). The free Wilke algebra generated by the two-sorted set \((X, Y)\) is \( T_w(X, Y) = (X^+, X^o + X^o Y) \) with the two products given by concatenation of words, and \( w^n^o = \underbrace{ww \ldots w}_{n} \) for \( w \in X^+ \). In particular, choosing the input object \((I, \theta)\) for some set \( I \) and the output object \( O = \{(0, 1), (0, 1)\} \), we have \( T_w(I, \theta) = (I^+, I^o) \), and thus a language \( L : T_w(I, \theta) \rightarrow O \) specifies a set of finite or ultimately periodic infinite words. Languages recognizable by Wilke algebras correspond to \( \omega \)-regular languages, i.e. languages accepted by Büchi automata \[49, 65\].

3. Regular cost functions were introduced by Colcombet \[23\] as a quantitative extension of regular languages that provides a unifying framework for studying limitedness problems. A cost function over the alphabet \( I \) is a function \( f : I^* \rightarrow \mathbb{N} \cup \{\infty\} \). Two cost functions \( f \) and \( g \) are identified if, for every subset \( A \subseteq \mathbb{N} \), the function \( f \) is bounded on \( A \) iff \( g \) is bounded on \( A \). Regular cost functions correspond to languages recognizable by finite stabilization algebras. The latter are ordered algebras over the signature \( \Gamma = \{0/0, \cdot/2, (-)^o/1, (-)^o/1\} \), with \(-/n\) denoting arities, subject to suitable inequations; see \[26, 58\]. We let \( T_S \) denote the monad on \text{Pos} induced by this ordered algebraic theory.

Our generic approach to learning \( T \)-recognizable languages is based on the idea of presenting the free algebra \( T I = (T_I, \mu_I) \) and its finite quotient algebras as automata:

**Definition 5.6** (\( T \)-refinable). A quotient \( e : TI \rightarrow A \) in \( \mathcal{D} \) is \( T \)-refinable if there exists a finite quotient algebra \( e' : T I \rightarrow (B, \beta) \) of \( TI \) and a morphism \( f : B \rightarrow A \) with \( e = f \cdot e' \).

**Definition 5.7** (Automata presentation). An automata presentation of the free \( T \)-algebra \( TI \) is given by an endofunctor \( F \) on \( \mathcal{D} \) and an \( F \)-algebra structure \( \delta : F TI \rightarrow TI \) such that

1. \( F(E) \subseteq E \), the initial algebra \( \mu F_1 \) exists, and every regular language \( L : \mu F_1 \rightarrow O \) admits a minimal automaton \( \text{Min}(L) \);
2. the \( F_1 \)-algebra \((TI, [\eta_1, \delta])\) is reachable (i.e. \( e TI \in E \));
3. a \( T \)-refinable quotient \( e : TI \rightarrow A \) carries a \( T \)-algebra quotient iff \( e \) carries an \( F \)-algebra quotient; that is, there exists \( \alpha_A \) making the left-hand square below commute iff there exists \( \delta_A \) making the right-hand square commute.

\[
\begin{align*}
TTI \xrightarrow{\mu} TI & \quad \Longleftrightarrow \quad F TI \xrightarrow{\delta} TI \\
T A \xrightarrow{\tau e} A & \quad \Longleftrightarrow \quad F A \xrightarrow{f e} A \\
\end{align*}
\]

If in (3) only the implication \( \Rightarrow \) is required, \((F, \delta)\) is called a weak automata presentation.

**Remark 5.8.** (1) Examples of functors \( F \) for which the first condition is satisfied include all functors satisfying the Assumptions 3.5, see Remark 3.7(1) and Theorem 3.13, and polynomial functors \( F = F_T \) on \text{Set} or \text{Pos} for a signature \( \Gamma \). Recall from Example 3.4 that \( F_T \)-automata are \( \Gamma \)-automata.

(2) Presentations of \( T \)-algebras as (sorted) \( \Sigma \)-automata were previously studied by Urbat, Adámek, Chen, and Milus \[59\] for the special case where \( \mathcal{D} \) is a variety of algebras and \( \Sigma \in \mathcal{D} \) is a free algebra, and called unary presentations.

**Example 5.9.** For all monads of Example 5.2, free algebras admit an automata presentation (in fact, a presentation as (sorted) \( \Sigma \)-automata \[58–60\]). Here we consider three cases:
The homomorphism $e : \Sigma \times I^* \rightarrow I^*$ as a list of instructions for forming a word in $I^*$.

From now on, we fix a weak automata presentation $\Gamma = (\Sigma, \delta, \mu)$ of $\Gamma$, see Section 2. Then $\Gamma I$ can be presented as the $I$-terms over $\Sigma$, a concept recently employed by Angluin and Fisman to the finite subalphabets $\Sigma^I, \Sigma^+I$ and adding letters $w$ for $w \in I^*$, $a \in I$.

Recall from Example 3.11 that the initial algebra $\mu F_1$ consists of sorted words over $\Sigma$ with an additional first letter from $I$. The homomorphism $e_{(1, \varphi)} : \mu F_1 \rightarrow (I^*, I^\varphi)$ views such a word as an instruction for forming a word in $(I^*, I^\varphi)$, e.g.,

$$e_{(1, \varphi)}(a b a b b a) = baaba.$$

To obtain a weak automata presentation, it suffices to restrict $\Sigma^I$ and $\Sigma^+I$ to the finite subalphabets $\Sigma^I, \Sigma^+I$ and $\Sigma^\varphi$. A $\Sigma$-automaton is similar to a family of DFAs, a concept recently employed by Angluin and Fisman [9] for learning $\omega$-regular languages.

(3) Stabilization algebras. Suppose that $T$ is a monad on $\mathsf{Set}$ or $\mathsf{Pos}$ induced by a finitary signature $\Gamma$ and (in-)equations $E$; see Section 2. Then $T I$ can be presented as the $\Gamma$-automaton $\delta : F_2(T I) \rightarrow TI$ given by the $\Gamma$-algebra structure on the free $(\Gamma, E)$-algebra $TI$. The initial algebra $\mu F_2$ is the algebra $T_1 I$ of $\Gamma$-terms over $\Sigma$ and the unique homomorphism $e_{T_1} : T_1 I \rightarrow TI$ interprets $\Gamma$-terms in $TI$. In particular, for the monad $T = T_3$ on $\mathsf{Pos}$, the free stabilization algebra $T_3 I$ admits a $\Gamma$-automata presentation for the signature $\Gamma$ of Example 5.5(3).

From now on, we fix a weak automata presentation $(F, \delta)$ of the free $T$-algebra $TI$.

**Definition 5.10** (Linearization). The linearization of a language $L : TI \rightarrow O$ is given by

$$\text{lin}(L) = (\mu F_1 e_{T_1} \xrightarrow{\varepsilon_T} TI \xrightarrow{L} O).$$

**Example 5.11.** (1) Semigroups. Take the $\Sigma$-automata presentation of Example 5.9(1). Given $L \subseteq I^*$, the language $\text{lin}(L) \subseteq I \times \Sigma^*$ consists of all possible ways of generating words in $L$ by starting with a letter $a \in I$ and adding letters on the left or on the right. For instance, if $L$ contains the word $abc$, then $\text{lin}(L)$ contains the words $a b c, b a c, b c a, c b a$.

(2) Wilke algebras. Take the weak presentation of Example 5.9(2). Given $L \subseteq (I^*, I^\varphi)$, the language $\text{lin}(L)$ consists of all possible ways of generating words in $L$ by starting with a letter $a \in I$ and repeatedly applying any of the following operations: (i) right concatenation of a finite word with a letter; (ii) left concatenation of an infinite word with a letter; (iii) taking the $\omega$-power of a finite word. For instance, if $L$ contains the word $(ab)^n$, then $\text{lin}(L)$ contains $a b \omega ã, b a \omega ã, a b \omega b a, b a \omega a b \omega b a, \ldots$. Thus, $\text{lin}(L)$ is a two-sorted version of the language lasso($L$) mentioned in the Introduction.

(3) Stabilization algebras. Take the presentation of Example 5.9(3). Given a language $L \subseteq T_3 I$, the set $\text{lin}(L) \subseteq T_3 I$ consists of all $\Gamma$-trees whose interpretation in $T_3 I$ lies in $L$.

As demonstrated by the above examples, the linearization allows us to identify a language $L : TI \rightarrow O$ with a language $\text{lin}(L) : \mu F_1 \rightarrow O$ of finite words or trees. Since the morphism $e_{T_1} : \mu F_1 \rightarrow TI$ is assumed to be epic by Definition 5.7(2), this identification is unique; that is, $\text{lin}(L)$ uniquely determines $L$. In particular, in order to learn $L$, it is sufficient to learn $\text{lin}(L)$. This approach is supported by the following result:

**Theorem 5.12.** If $L : TI \rightarrow O$ is a $T$-recognizable language, then its linearization $\text{lin}(L) : \mu F_1 \rightarrow O$ is regular, i.e. accepted by some finite $F$-automaton.

**Proof sketch.** Let $e : TI \rightarrow (A, a)$ be a $T$-homomorphism recognizing $L$ via $p : A \rightarrow O$. By replacing $e$ with its coinage, we may assume that $e \in E$. The weak automata presentation yields an $F$-algebra structure on $A$ making $e$ an $F$-algebra homomorphism. Then $A$, viewed as an automaton with initial states $e \cdot \eta : I \rightarrow A$ and final states $p$, accepts $\text{lin}(L)$. □

In view of this theorem, one can apply any learning algorithm for finite $F$-automata (e.g. Generalized $L^*$ for the case of adjoint automata, or a learning algorithm for tree automata [30] if $F$ is a polynomial functor) to learn the minimal automaton $Q_L$ for $\text{lin}(L)$. This automaton, together with the epimorphism $e_{T_1}$, constitutes a finite representation of the unknown language $L : TI \rightarrow O$. If the given automata presentation for $TI$ is non-weak, we can go one step further and infer from $Q_L$ a minimal algebraic representation of $L$.

**Definition 5.13** (Syntactic $T$-algebra). Let $L : TI \rightarrow O$ be recognizable. A syntactic $T$-algebra for $L$ is a quotient $T$-algebra $e_L : TI \rightarrow \text{Syn}(L)$ of $TI$ such that (1) $e_L$ recognizes $L$, and (2) $e_L$ factorizes through every finite quotient $T$-algebra.
We have presented a general algorithm (Generalized \(L^*\) -type algorithms, e.g. for various notions of nominal automata with name binding. Moreover, we have shown how to extend the scope of Generalized \(L^*\)-type algorithms, and other learning algorithms for \(F\)-automata, to languages recognizable by monad algebras. This gives rise to a generic approach to learning numerous types of languages, including cases for which no learning algorithms are known (e.g. cost functions).

The next step is to turn our high-level categorical approach into an implementation-level algorithm, parametric in the monad \(T\) and its automata presentation, with corresponding tool support. We expect that the recent work on coalgebraic minimization algorithms and their implementation\([27, 29]\) can provide guidance. It should be illuminating to experimentally compare the performance of the generic algorithm with tailor-made algorithms for specific types of automata.

Our generalized \(L^*\) -algorithm is concerned with adjoint \(F\)-automata and applies to a wide variety of automata on finite words (including weighted, residual nondeterministic, and nominal automata), but presently not to tree automata. To deal with the latter, the adjointness of the type functor \(F\) needs to be relaxed, which entails that a coalgebraic semantics is no longer directly available. We leave a categorical understanding of tree automata learning for future work.

References

1. Jiří Adámek, Filippo Bonchi, Mathias Hülsbusch, Barbara König, Stefan Milius, and Alexandra Silva. 2012. A Coalgebraic Perspective on Minimization and Determinization. In Foundations of Software Science and Computational Structures, Lars Birkedal (Ed.). Springer Berlin Heidelberg, 58–73.
2. Jiří Adámek, Horst Herrlich, and George Strecker. 2004. Abstract and Concrete Categories: The Joy of Cats. Dover Publications. 528 pages.
3. Jiří Adámek, Stefan Milius, Robert S. R. Myers, and Henning Urbat. 2014. On Continuous Nondeterminism and State Minimality. In Proc. Mathematical Foundations of Programming Semantics (MFPS XXX) (Electron. Notes Theor. Comput. Sci.), Bart Jacobs, Alexandra Silva, and Sam Staton (Eds.), Vol. 308. Elsevier, 3–23.
4. J. Adámek, S. Milius, and H. Urbat. 2015. Syntactic Monoids in a Category. In Proc. CALCO’15 (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
5. Jiří Adámek and Vera Trnková. 1989. Automata and Algebras in Categories. Springer.
6. Jiří Adámek. 1974. Free algebras and automata realizations in the language of categories. Commentationes Mathematicae Universitatis Carolinae 15, 4 (1974), 589–602. http://eudml.org/doc/16649
7. Mikołaj Bojańczyk. 2015. Recognisable languages over monads. In Proc. DLT 2015, Igor Potapov (Ed.), LNCS, Vol. 9168. Springer, 1–13. http://arxiv.org/abs/1502.04898.
8. Dana Angluin. 1987. Learning Regular Sets from Queries and Counterexamples. Inf. Comput. 75, 2 (1987), 87–106.
9. Dana Angluin and Dana Fisman. 2016. Learning regular omega languages. Theoretical Computer Science 650 (2016), 57 – 72.
10. Michael A. Arbib and Ernest G. Manes. 1975. Adjoint machines, state-behavior machines, and duality. Journal of Pure and Applied Algebra 6, 3 (1975), 313 – 344.
11. László Babai. 1986. On the length of subgroup chains in the symmetric group. Comm. Alg. 14, 9 (1986), 1729–1736. https://doi.org/10.1080/00927878608823393
Automata Learning: An Algebraic Approach

[55] Jan J. M. M. Rutten. 2000. Universal coalgebra: a theory of systems. Theoret. Comput. Sci. 249, 1 (2000), 3–80.

[56] Lutz Schröder, Dexter Kozen, Stefan Milius, and Thorsten Wißmann. 2017. Nominal Automata with Name Binding. In Foundations of Software Science and Computation Structures, FOSSACS 2017 (LNCS), Vol. 10203. Springer, 124–142. https://doi.org/10.1007/978-3-662-54458-7

[57] Paul Taylor. 1999. Practical Foundations of Mathematics. Cambridge University Press.

[58] Henning Urbat, Jiří Adámek, Liang-Ting Chen, and Stefan Milius. 2017. Eilenberg Theorems for Free. CoRR abs/1602.05831 (2017). http://arxiv.org/abs/1602.05831

[59] Henning Urbat, Jiří Adámek, Liang-Ting Chen, and Stefan Milius. 2017. Eilenberg Theorems for Free. In Proc. MFCS 2017 (LIPIcs), Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin (Eds.), Vol. 83. Schloss Dagstuhl.

[60] Henning Urbat and Stefan Milius. 2019. Varieties of Data Languages. In Proc. 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019) (LIPIcs) Christel Baier, Ioannis Chatzigiannakis, Paolo Flocchini, and Stefano Leonardi (Eds.), Vol. 132. 130:1–130:14.

(Presents the first Eilenberg-type correspondence for data languages and a nominal Eilenberg-Schützenberger theorem characterizing pseudovarieties of nominal monoids.)

[61] Frits Vaandrager. 2017. Model Learning. Commun. ACM 60, 2 (2017), 86–95.

[62] Gerco van Heerdt, Tobias Kappé, Jurriaan Rot, Matteo Sammartino, and Alexandra Silva. 2019. Tree Automata as Algebras: Minimisation and Determinisation. CoRR abs/1904.08802 (2019). http://arxiv.org/abs/1904.08802

[63] Gerco van Heerdt, Matteo Sammartino, and Alexandra Silva. 2017. CALF: Categorical Automata Learning Framework. In Proc. CSL 2017. 29:1–29:24.

[64] Gerco van Heerdt, Matteo Sammartino, and Alexandra Silva. 2017. Learning Automata with Side-Effects. CoRR abs/1704.08055 (2017). http://arxiv.org/abs/1704.08055

[65] T. Wilke. 1991. An Eilenberg Theorem for \(\infty\)-Languages. In Proc. ICALP’91 (LNCS), Vol. 510. Springer, 588–599.
A Appendix: Omitted Proofs and Details

In this appendix, we provide full proofs of all our results and more detailed treatment of some examples omitted due to space restrictions.

Discussion of the Assumptions 3.5 and 4.1

We comment on some technical consequences of our Assumptions 3.5 and 4.1.

Remark A.1. The assumption $F(E) \subseteq E$ implies that the factorization system $(E, M)$ of $\mathcal{D}$ lifts to automata: given an automata homomorphism $h: Q \to Q'$ and its $(E, M)$-factorization $h = (Q \xrightarrow{e} Q'' \xrightarrow{m} Q')$ in $\mathcal{D}$, there exists a unique automata structure $(Q'', \delta_{Q''}, i_{Q''}, f_{Q''})$ on $Q''$ such that both $e$ and $m$ are automata homomorphisms. Indeed, the transitions $\delta_Q'$ are given by diagonal fill-in

\[
\begin{array}{c}
\xymatrix{ 
FQ \ar[r]^\delta_Q \ar[d]_{Fe} & Q \\
FQ'' \ar[r]_{\delta_{Q''}} & Q'' \\
FQ' \ar[u]^{f_m} & 
}
\end{array}
\]

and the initial and final states by

\[
i_{Q''} = \left( I \xrightarrow{i_0} Q \xrightarrow{e} Q'' \right),
\]

\[
f_{Q''} = \left( Q'' \xrightarrow{m} Q' \xrightarrow{f_{Q'}} O \right).
\]

Remark A.2. The condition $F_I(M) \subseteq M$ makes sure that the factorization system $(E, M)$ lifts from $\mathcal{D}$ to Coalg $F_I$, the category of $F_I$-coalgebras: given an $F_I$-coalgebra homomorphism $h: (C, \gamma) \to (C', \gamma')$ and its $(E, M)$-factorization $h = (C \xrightarrow{e} C'' \xrightarrow{m} C')$ in $\mathcal{D}$, there is a unique $F_I$-coalgebra structure $(C'', \gamma'')$ on $C''$ such that both $e$ and $m$ are coalgebra homomorphisms. The structure $\gamma''$ is defined via diagonal fill-in in analogy to Remark A.1.

Dually, the condition $G_O(E) \subseteq E$ implies that Alg $G_O$, the category of $G_O$-algebras, has a factorization system lifting $(E, M)$.

Details for Example 3.3

We show that for each of the five categories $\mathcal{D}$ of Table 1 and the endofunctors $F$ and $G$ on $\mathcal{D}$ given by

\[
F = \Sigma \otimes (-) \quad \text{and} \quad G = \Sigma, -,
\]

the Assumptions 3.5(1)–(4) and 4.1 are satisfied.

Clearly, all the categories $\mathcal{D}$ with the corresponding choices of $I$ and $O$ satisfy the Assumptions 3.5(1)(2). Moreover, (3) holds because $\mathcal{D}$ is closed. For (4), note that in all cases $\mathcal{E}$ coincides with the class of all epimorphisms. Since every left adjoint $F$ preserves epimorphisms, it follows that $F(E) \subseteq E$. It remains to verify the Assumptions 4.1. We consider the cases $\mathcal{D} = \text{Set}, \text{Pos}, \text{JSL}, \text{K-Vec}$; for $\mathcal{D} = \text{Nom}$, see the details for Example 3.10.

$F_I$ preserves $M$ and intersections of $M$-morphisms. This is clear for $\mathcal{D} = \text{Set}, \text{Pos}$ since in these categories coproducts commute with intersections, i.e. one has

\[
(A + B) \cap (C + D) \equiv (A \cap C) + (B \cap D).
\]

For $\mathcal{D} = \text{JSL}$ recall that we have chosen $\Sigma$ to be the free semilattice $P_I \Sigma_0$ over a finite set $\Sigma_0$ of generators, i.e. the $\cup$-semilattice of finite subsets of $\Sigma_0$. It follows that

\[
F_I X = I + \Sigma \otimes X = I + \left( \bigsqcup_{a \in \Sigma_0} I \otimes X \right) \equiv I + \bigsqcup_{a \in \Sigma_0} I \otimes X \equiv I + \bigsqcup_{a \in \Sigma_0} X
\]

using that $I = P_I 1$, $I \otimes X \equiv X$, and the left adjoint $(-) \otimes X$ preserves coproducts. Now note that the coproduct $X + Y$ of two semilattices coincides with the product $X \times Y$, with injections given by

\[
\text{inl}: X \to X \times Y, \quad x \mapsto (x, \bot)
\]

\[
\text{inr}: Y \to X \times Y, \quad y \mapsto (\bot, y)
\]

This implies that monomorphisms in $\text{JSL}$ are stable under coproducts, and that intersections commute with coproducts. It thus follows from the above formula for $F_I X$ that $F_I$ preserves monomorphisms and intersections.

For $\mathcal{D} = \text{K-Vec}$, the proof is analogous, using again the product/coproduct coincidence.

$G_O$ preserves epimorphisms. We first show that the functor $\Sigma, -$ preserves epimorphisms (i.e. surjections). Note first that in $\mathcal{D} = \text{Set}, \text{Pos}, \text{JSL}, \text{K-Vec}$, the object $[\Sigma, X]$ is carried by the set $\mathcal{D}(\Sigma, X)$ with the $\mathcal{D}$-structure inherited from $X$ (i.e. defined pointwise), and that for any morphism $e: X \to Y$ the morphism $[\Sigma, e]: [\Sigma, X] \to [\Sigma, Y]$ is given by $f \mapsto e \cdot f$. We need to prove that $[\Sigma, e]$ is surjective provided that $e$ is surjective; that is, for every morphism $g: \Sigma \to Y$ there exists a morphism $f: \Sigma \to X$ making the following triangle commute:

\[
\Sigma \xrightarrow{f} X \xrightarrow{e} Y.
\]

This follows from the fact that in each case, $\Sigma$ has been chosen as a projective object of $\mathcal{D}$. For instance, for $\mathcal{D} = \text{JSL}$ we construct $f$ as follows. Recall that $\Sigma$ is the free semilattice on a finite set $\Sigma_0$, and denote by $\eta: \Sigma_0 \to \Sigma$ the universal map. For each $a \in \Sigma_0$, choose $x_a \in X$ with $e(x_a) = g(\eta(a))$, using that $e$ is surjective. This gives a map

\[
f_0: \Sigma_0 \to X, \quad a \mapsto x_a.
\]

Let $f: \Sigma \to X$ be the unique semilattice homomorphism extending $f_0$, i.e. with $f \cdot \eta = f_0$. Then $e \cdot f = g$ since this
We verify that the functors of Example 3.10(1)–(4), see the table below, satisfy our Assumptions 3.5(4) and 4.1. Recall that we have chosen \( I = 1 \) and \( O = 2 \), and that the factorization system of \( \text{Nom} \) is the one given by epimorphisms (= surjective equivariant maps) and monomorphisms (= injective equivariant maps).

| \( F \) | \( G \) |
|-------|-------|
| (1) \( \mathbb{A} \times (–) \) | \([\mathbb{A}, –]\) |
| (2) \( \mathbb{A} \ast (–) \) | \([\mathbb{A}, –]\) |
| (3) \( \mathbb{A} \times (–) + \mathbb{A} \ast (–) \) | \([\mathbb{A}, –] \times [\mathbb{A}](-)\) |
| (4) \( \mathbb{A} \times (–) + \mathbb{A} \ast (–) + [\mathbb{A}](-) \) | \([\mathbb{A}, –] \times [\mathbb{A}](-) \times R\) |

**F preserves epimorphisms.** This follows from the fact that \( F \) is a left adjoint.

**\( F_1 \) preserves monomorphisms.** The functors \( \mathbb{A} \times (–) \) and \( \mathbb{A} \ast (–) \) preserve monomorphisms by definition, recalling that for an equivariant map \( e: X \rightarrow Y \) the map \( \mathbb{A} \ast e \) is given by \( \mathbb{A} \ast e: \mathbb{A} \times X \rightarrow \mathbb{A} \times Y, \quad (a, x) \mapsto (a, e(x)) \).

The functor \([\mathbb{A}][-]\) preserves monomorphisms because it is a right adjoint. Since coproducts in \( \text{Nom} \) are formed at the level of \( \text{Set} \), it follows that monomorphisms in \( \text{Nom} \) are stable under coproducts. This implies that for all the functors \( F \) in (1)–(4), the functor \( F_1 = I + F \) preserves monomorphisms.

**\( F_1 \) preserves intersections.** Note that intersections of subobjects (i.e. equivariant subsets) in \( \text{Nom} \) are just set-theoretic intersections. Thus, the functors \( \mathbb{A} \times (–) \) and \( \mathbb{A} \ast (–) \) clearly preserve intersections by definition. The functor \([\mathbb{A}][-]\) preserves them because it is right adjoint and thus preserves all limits. Since intersections commute with coproducts in \( \text{Set} \) and thus also in \( \text{Nom} \), it follows that for all the functors \( F \) in (1)–(4), the functor \( F_1 = I + F \) preserves intersections.

**\( G_O \) preserves epimorphisms.** The functor \([\mathbb{A}][-]\) preserves epimorphisms because it is a left adjoint. Moreover, we have

**Lemma A.3.** The functors \([\mathbb{A}, –]\): \( \text{Nom} \rightarrow \text{Nom} \) and \( R: \text{Nom} \rightarrow \text{Nom} \) preserve epimorphisms.

**Proof.** (1) We first show that \([\mathbb{A}, –]\) preserves epimorphisms (i.e. surjections). This can be deduced from the fact that every polynomial functor on \( \text{Nom} \) preserves epimorphisms (like in \( \text{Set} \)) and that \([\mathbb{A}, –]\) can be expressed as a quotient functor of a polynomial functor \([44, \text{Lemma } 6.9]\). In the following, we give a direct proof for the convenience of the reader.

Recall from \([51, \text{Theorem } 2.19]\) that \([\mathbb{A}, X] \) is the nominal set of finitely supported maps \( f: \mathbb{A} \rightarrow X \); here \( f \) is finitely supported if there exists a finite subset \( S \subseteq \mathbb{A} \) such that for all permutations \( \pi \in \text{Perm}(\mathbb{A}) \) that fix \( S \) and all \( a \in \mathbb{A} \) one has \( f(\pi \cdot a) = \pi \cdot f(a) \). In particular, equivariant maps are finitely supported maps with support \( S = \emptyset \). For any equivariant map \( e: X \rightarrow Y \), the map \([\mathbb{A}, e]\) is given by

\[ [\mathbb{A}, e]: [\mathbb{A}, X] \rightarrow [\mathbb{A}, Y], \quad f \mapsto e \cdot f. \]

We need to show that \([\mathbb{A}, e]\) is surjective provided that \( e \) is surjective; in other words, for every finitely supported map \( g: \mathbb{A} \rightarrow Y \), there exists a finitely supported map \( f: \mathbb{A} \rightarrow X \) making the following triangle commute:

\[ \mathbb{A} \xrightarrow{f} X \xrightarrow{e} Y \]
Fix an arbitrary atom \( a \notin A \setminus \supp g \). Moreover, choose \( x \in X \) with \( e(x) = g(a) \), and choose \( x_b \in X \) with \( e(x_b) = g(b) \) for every \( b \in \supp g \cup \supp x \), using that \( e \) is surjective. Define the map \( f : A \to X \) as follows:
\[
f(b) = \begin{cases} 
(b \cdot a) \cdot x & \text{for } b \in A \setminus (\supp g \cup \supp x); \\
x_b & \text{for } b \in \supp g \cup \supp x.
\end{cases}
\]
We claim that (i) the map \( f \) is finitely supported and (ii) it satisfies \( e \cdot f = g \).

\textbf{Ad (i).} We show that the finite set of atoms
\[
S = \supp g \cup \supp x \cup \bigcup_{b \in \supp g \cup \supp x} \supp x_b
\]
supports the map \( f \). Thus, let \( \pi \in \Perm(A) \) be a permutation fixing \( S \); we need to prove that \( f(\pi \cdot b) = \pi \cdot f(b) \) for all \( b \in A \). For \( b \in \supp g \cup \supp x \), we have
\[
f(\pi \cdot b) = f(b) = x_b = \pi \cdot x_b = \pi \cdot f(b).
\]
For \( b \in A \setminus (\supp g \cup \supp x) \), we get
\[
f(\pi \cdot b) = (\pi(b) \cdot a) \cdot x = \pi \cdot (b \cdot a) \cdot x = \pi \cdot f(b).
\]
Here the first and last equation use the definition of \( f \). The middle equation holds because the two permutations \( \pi(b) a \) and \( \pi \cdot (b \cdot a) \) are equal on \( \supp x \). Indeed, both permutations send \( a \) to \( \pi(b) \cdot a \), and all elements of \( \supp x \setminus \{a\} \) are fixed by both permutations because \( b, \pi(b) \notin \supp x \) and \( \pi \) fixes \( \supp x \).

\textbf{Ad (ii).} We show that \( e(f(b)) = g(b) \) for all \( b \in A \). For \( b \in \supp g \cup \supp x \) we have
\[
e(f(b)) = e(x_b) = g(b)
\]
by definition of \( f \) and \( x_b \). For \( b \in A \setminus (\supp g \cup \supp x) \),
\[
e(f(b)) = e((b \cdot a) \cdot x) \quad \text{def. } f \\
= (b \cdot a) \cdot e(x) \quad \text{equivariant} \\
= (b \cdot a) \cdot g(a) \quad \text{def. } x \\
= g((b \cdot a) \cdot a) \quad a, b \notin \supp g \\
= g(b)
\]
where the last equation uses that \( a, a' \# g(a) \). Thus, we can replace \( x \) by \( x' \). Analogously for \( x_b \).

Part (1) now shows that \( f \) is finitely supported and satisfies \( e \cdot f = g \). Moreover, we clearly have \( b \# f(b) \) for every \( b \in A \) by definition of \( f \) and the above choices of \( x \) and \( x_b \), i.e. \( f \in RX \).

Since epimorphisms in \( 
\text{Nom} \) are stable under products (which follows from the corresponding property in \( \text{Set} \)), we conclude that for all the functors \( G \) in (1)–(4), the functor \( G_O = 2 \times G \) preserves epimorphisms.

\textbf{Details for Example 3.11}

We describe sorted \( \Sigma \)-automata for the case of general base categories \( D \). Suppose that \( (D, \emptyset, I_{\emptyset}) \) is a symmetric monoidal closed category satisfying our Assumptions 3.5(1)–(2), and let \( S \) be a set of sorts. Then the category \( D^S \) (equipped with the monoidal structure and the factorization system inherited sortwise from \( D \)) is also symmetric monoidal closed and satisfies the Assumptions 3.5(1)–(2).

Fix an arbitrary object \( l \in D^S \) inputs (not necessarily the tensor unit), an arbitrary object \( O \in D^S \) of outputs, and a family of objects \( \Sigma = (\Sigma_s, t)_{s \in S} \) in \( D \); we think of \( \Sigma_{s, t} \) as a set of letters with input sort \( s \) and output sort \( t \). Take the functors
\[
F : D^S \to D^S, \quad (FQ)_t = \prod_{s \in S} \Sigma_{s, t} \otimes Q_s \quad (t \in S),
\]
\[
G : D^S \to D^S, \quad (GQ)_s = \prod_{t \in S} [\Sigma_{s, t}, Q_t] \quad (s \in S).
\]
The functor \( F \) is a left adjoint of \( G \): we have the isomorphisms (natural in \( P, Q \in D^S \))
\[
D^S(FQ, P) = \prod_{s \in S} D((FQ)_s, P_s) = \prod_{s \in S} D(\Sigma_{s, t} \otimes Q_s, P_t) = \prod_{s \in S} D(\Sigma_{s, t}, Q_s, P_t) = D(Q_s, \prod_{t \in S} [\Sigma_{s, t}, P_t]) = D(Q_s, (GP)_s). = D(S, GP).
\]
Instantiating Definition 3.2 to the above data, we obtain the concept of a sorted \( \Sigma \)-automaton. It is given by an \( S \)-sorted object of states \( Q \in D^S \) together with morphisms \( \delta_{Q,s,t}, i_{Q,s,t} \).
and \( f_{Q,t} \) as in the diagram below for \( s, t \in S \):

\[
\begin{array}{c}
\Sigma_{s,t} \otimes Q_t \\
\downarrow \delta_{Q,s,t} \\
I_t \xrightarrow{i_{Q,t}} Q_t \xrightarrow{f_{Q,t}} O_t
\end{array}
\]

In generalization of the single-sorted case (see Example 3.9), the initial algebra for \( F_1 \) can be described as follows. For \( n \in \mathbb{N} \) and \( s, t \in S \) define the object \( \Sigma^0_{s,t} \in \mathcal{D} \) inductively by

\[
\Sigma^0_{s,t} = I_\emptyset, \quad \Sigma^{n+1}_{s,t} = \bigsqcup_{r \in \mathbb{N}} \Sigma_r \otimes \Sigma^n_{s,r,t}.
\]

and put

\[
\Sigma_{s,t} = \bigsqcup_{n \in \mathbb{N}} \Sigma^n_{s,t}.
\]

The initial algebra for the functor \( F_1 \) is given by

\[
(\mu F_1)_I = \bigsqcup_{t \in S} I_s \otimes \Sigma_{s,t} \quad (t \in S).
\]

**Proof of Theorem 3.13**

We first establish some basic observations about automata homomorphisms and languages:

**Proposition A.4.** For each automata homomorphism \( h: Q \rightarrow Q' \) one has \( L_Q = L_{Q'} \)

**Proof.** This follows from the commutative diagram below. The upper triangle commutes by initiality of \( \mu F_1 \), and all remaining parts commute by definition.

\[
\begin{array}{c}
Q \\
\downarrow \downarrow \\
Q' \\
\downarrow \downarrow \\
O \\
\downarrow \downarrow \\
L_Q = L_{Q'}
\end{array}
\]

**Remark A.5.** Every \( F \)-algebra homomorphism \( h: (Q, \delta) \rightarrow (Q', \delta') \) is also a \( G \)-coalgebra homomorphism \( h: (Q, \delta^0) \rightarrow (Q', \delta'^0) \), and vice versa. Indeed, the corresponding commutative squares are just adjoint transposes of each other.

\[
\begin{array}{c}
FQ \xrightarrow{\delta} Q \\
\downarrow Fh \\
FQ' \xrightarrow{\delta'} Q'
\end{array}
\quad
\begin{array}{c}
Q \xrightarrow{\delta^0} GQ \\
\downarrow h \\
Q' \xrightarrow{(\delta')^0} GQ'
\end{array}
\]

**Proposition A.6.** For all automata \( Q \) and \( Q' \), we have

\[
L_Q = L_{Q'} \iff m_Q \cdot e_Q = m_{Q'} \cdot e_{Q'}.
\]

**Proof.** (1) For the “if” direction, suppose that \( m_Q \cdot e_Q = m_{Q'} \cdot e_{Q'} \). Then the following diagram (where \( \text{outl} : G_Q \rightarrow O \) denotes the left product projection) commutes by the definition of \( \text{outl} \) in Remark 3.7 and because \( m_Q \) is a \( G_Q \)-coalgebra homomorphism.

\[
\begin{array}{c}
Q \xrightarrow{\text{outl}} G_Q \xrightarrow{\text{outl}} O \\
\downarrow m_Q \xrightarrow{G_{\text{outl}}} G_Q \xrightarrow{\text{outl}} O
\end{array}
\]

Thus \( f_Q = \text{outl} \cdot \gamma \cdot m_Q \) and analogously \( f_{Q'} = \text{outl} \cdot \gamma \cdot m_{Q'} \). This implies

\[
L_Q = f_Q \cdot e_Q = \text{outl} \cdot \gamma \cdot m_Q \cdot e_Q = \text{outl} \cdot \gamma \cdot m_{Q'} \cdot e_{Q'} = \cdots = L_{Q'}.
\]

(2) For the “only if” direction, suppose that \( L = L_Q = L_{Q'} \). By equipping \( \mu F_1 \) with final states \( L : \mu F_1 \rightarrow O \), we can view \( \mu F_1 \) as a \( G_Q \)-coalgebra, and thus \( e_Q : \mu F_1 \rightarrow Q \) as a \( G_Q \)-coalgebra homomorphism (see Remark A.5). It follows that \( m_Q \cdot e_Q : \mu F_1 \rightarrow G_O Q \) is a \( G_Q \)-coalgebra homomorphism. Analogously, \( m_Q \cdot e_Q \) is a coalgebra homomorphism. Thus, \( m_Q \cdot e_Q = m_{Q'} \cdot e_{Q'} \) by finality of \( \text{outl} \).

**Remark A.7.** For every language \( L : \mu F_1 \rightarrow O \) there exists an automaton \( Q \) accepting \( L \). Indeed, one can choose \( Q = \mu F_1 \) with output morphism \( L : \mu F_1 \rightarrow O \).

We are prepared to prove the minimization theorem:

**Proof of Theorem 3.13.** Fix an arbitrary automaton \( Q \) with \( L_Q = L \) (see Remark A.7). Viewing \( \mu F_1 \) as an automaton with output morphism \( L_Q = f_Q \cdot e_Q : \mu F_1 \rightarrow O \), the unique \( F_1 \)-algebra homomorphism \( e_Q \) is an automata homomorphism. Analogously, equipping \( v G_Q \) with the initial states \( m_Q \cdot i_Q : I \rightarrow v G_Q \) makes the unique \( G_Q \)-coalgebra homomorphism \( m_Q : Q \rightarrow v G_Q \) an automata homomorphism. Thus \( m_Q \cdot e_Q \) is an automata homomorphism. Form its \((E, M)\)-factorization, see Remark A.1:

\[
\begin{array}{c}
Q \\
\downarrow \downarrow \\
\text{Min}(L)
\end{array}
\]

We claim that \( \text{Min}(L) \) is the minimal automaton for \( L \). To this end, note first that \( \text{Min}(L) \) is reachable by the “if” direction of Proposition A.6. Thus, \( \text{Min}(L) \) accepts the language \( L \). Moreover, \( \text{Min}(L) \) is reachable because \( e_{\text{Min}(L)} \in E \).

To establish the universal property of \( \text{Min}(L) \), suppose that \( R \) is a reachable automaton accepting \( L \); we need to show that there is a unique homomorphism from \( R \) into \( \text{Min}(L) \). From
Given another automata homomorphism $h': R \to \text{Min}(L)$, we have $h' \cdot e_R = \epsilon_{\text{Min}(L)}$ by initiality of $\mu F_I$. Thus $h' \cdot e_R = h \cdot e_R$, which implies $h' = h$ because $e_R$ is epic. This proves the desired universal property of $\text{Min}(L)$.

The uniqueness of $\text{Min}(L)$ up to isomorphism follows immediately from its universal property. \hfill \Box

The construction of $\text{Min}(L)$ is the above proof also shows:

**Corollary A.8.** An automaton $Q$ is minimal if and only if it is both reachable ($e_Q \in E$) and simple ($m_Q \in M$).

**Details for Remark 4.4**
That $L_Q = L_{Q'}$ implies $h_{s,t}^Q = h_{s,t}^{Q'}$, follows immediately from the "only if" direction of Proposition A.6 and the definition of $h_{s,t}^{(\cdot)}$.

**Details for Definition 4.8**
For the diagonal fill-in $\delta_{s,t}$ to exist, we need to verify that for each pair $(s, t)$ as in (3), the square below is commutative:

$$
\begin{array}{ccc}
FS & \xrightarrow{F_{s,t}} & FH_{s,t} \\
| & & | \\
\mu F_I & \xrightarrow{\epsilon_{s,t}} & \mu F_I \\
\downarrow{m_{s,t}} & & \downarrow{m_{s,t}} \\
H_{s,t} & \xrightarrow{\epsilon_{s,t}} & T
\end{array}
$$

where

$$
l_{s,t} = (FS \xrightarrow{\text{intr}} I + FS = F_I \xrightarrow{\epsilon_{s,t}} H_{F_I, s,t} \xrightarrow{\epsilon_{s,t}} T)
$$

and

$$
r_{s,t} = (H_{s,t} \xrightarrow{\epsilon_{s,t}^{-1}} H_{s,\text{tr}GOT} \xrightarrow{m_{s,\text{tr}GOT}} \text{GOT} \xrightarrow{\epsilon_{s,t}} GT).
$$

**Proof:** By definition of $\text{cl}_{s,t}$ and $\text{cs}_{s,t}$, the lower path of the square is equal to

$$FS \xrightarrow{\text{intr}} F_I \xrightarrow{h_{F_I, s,t}} T$$

and the upper path is equal to

$$FS \xrightarrow{F_{s,t}} FH_{s,t} \xrightarrow{\epsilon_{s,t}} T.$$

We therefore need to verify that the outside of the following diagram commutes:

All parts except $(\ast)$ clearly commute either by definition or by naturality of $\text{intr}: F \to F_I$ and $\text{outr}: G_O \to G$. For $(\ast)$, note that the lower path is the adjoint transpose of $v_{G_O} \xrightarrow{\gamma} G_{O_0} \xrightarrow{\text{outr}} G_T \xrightarrow{\alpha} GT$, the upper path is the adjoint transpose of $v_{G_O} \xrightarrow{\gamma} G_{O_0} \xrightarrow{\text{outr}} G_T \xrightarrow{\alpha} GT$, and the commutative diagram below shows that these two morphisms are equal:

This concludes the proof. \hfill \Box

**Proof of Theorem 4.12**
The proof of the correctness and termination of the generalized $L^*$ algorithm requires some preparation. First, recall that for any endofunctor $H$, an $H$-coalgebra $C \xrightarrow{\gamma} HC$ is recursive [57] if for each $H$-algebra $HA \xrightarrow{\alpha} A$ there exists a unique coalgebra-to-algebra homomorphism $h$ from $(C, \gamma)$ into $(A, \alpha)$; that is, $h$ makes the square below commute.
Dually, an $H$-algebra $HA \xrightarrow{\alpha} A$ is corecursive if for each $H$-coalgebra $C \xrightarrow{\gamma} HC$ there exists a unique coalgebra-to-algebra homomorphism $h$ from $(C, \gamma)$ into $(A, \alpha)$.

**Lemma A.9** (see [22], Prop. 6). For each recursive coalgebra $C \xrightarrow{\gamma} HC$, the coalgebra $HC \xrightarrow{\text{coker } \gamma} HHC$ is also recursive.

Barolocco et al. [14] model prefix-closed sets as recursive subcoalgebras of an initial algebra $\mu H$. In our present setting, recursivity comes for free:

**Proposition A.10.** Every subcoalgebra of $(F^N_1 0, F^N_1 i)$, $N \geq 0$, is recursive.

In particular, this result applies to the subcoalgebras $(S, \sigma)$ in the generalized $\mathcal{L}^*$ algorithm.

**Proof.** Suppose that $s: (S, \sigma) \rightarrow (F^N_1 0, F^N_1 i)$ is a subcoalgebra for some $N \geq 0$. We prove that $(S, \sigma)$ is recursive by induction on $N$.

For $N = 0$, note first that in any category $\mathcal{D}$ the initial object 0 has no proper subobjects. (Indeed, suppose that $m: S \rightarrow 0$ is a subobject. Then the unique morphism $i: 0 \rightarrow S$ satisfies $m \cdot i = id_0$ by initiality of 0, so $m$ is both monic and split epic, i.e., an isomorphism.) Consequently, we have $(S, \sigma) = (0, i)$, and this coalgebra is trivially recursive by initiality of 0.

For the induction step, let $N > 0$, and let $(A, \alpha)$ be an arbitrary $F_1$-algebra. We need to prove that there is a unique coalgebra-to-algebra homomorphism $h: (S, \sigma) \rightarrow (A, \alpha)$.

(1) **Existence.** Since $(F^N_1 0, F^N_1 i)$ is a recursive coalgebra by Lemma A.9, we have a unique coalgebra-to-algebra homomorphism $h'$ from $(F^N_1 0, F^N_1 i)$ to $(A, \alpha)$. Thus $h = h' \cdot s$ is a coalgebra-to-homomorphism from $(S, \sigma)$ to $(A, \alpha)$.

(2) **Uniqueness.** Suppose that $h: (S, \sigma) \rightarrow (A, \alpha)$ is a coalgebra-to-algebra homomorphism. Form the pullback of $s$ and $F^N_1 i$:

$$
\begin{array}{ccc}
F^N_1 i & \rightarrow & F^N_1 0 \\
\downarrow & & \downarrow \\
S' & \rightarrow & S
\end{array}
$$

Note that $F^N_1 i \in M$ because $i: 0 \rightarrow F_1 0 = I$ lies in $M$ by Assumption 3.5(2) and $F_1$ preserves $M$ by Assumptions 4.1. Since in any factorization system $(E, M)$ the class $M$ is stable under pullbacks [2, Prop. 14.15], it follows that $m, s' \in M$. Since $F_1$ preserves pullbacks of $M$-morphisms by Assumptions 4.1, the upper right square in the diagram below is a pullback, and the outer part commutes because $s$ is a coalgebra homomorphism. Thus, there is a unique morphism $n$ making the two triangles commute:

It follows that $m: (S', n \cdot m) \rightarrow (S, \sigma)$ and $s': (S', n \cdot m) \rightarrow (F^N_1 0, F^N_1 i)$ are coalgebra homomorphisms, as shown by the two commutative diagrams below:

$$
\begin{array}{ccc}
S & \xrightarrow{\sigma} & F_1 S \\
\downarrow & & \downarrow \\
S' & \xrightarrow{\sigma} & F_1 S'
\end{array}
$$

By induction we know that the coalgebra $(S', n \cdot m)$ is recursive, that is, we have a unique coalgebra-to-algebra homomorphism $g: (S', n \cdot m) \rightarrow (A, \alpha)$. Since also $h \cdot m: (S', n \cdot m) \rightarrow (A, \alpha)$ is coalgebra-to-algebra homomorphism (being the composite of a coalgebra homomorphism with a coalgebra-to-algebra homomorphism), we get $h \cdot m = g$. Then the commutative diagram below shows that $h = \alpha \cdot F_1 g \cdot n$, i.e., $h$ is uniquely determined by $g$.

Note that the proof of Proposition A.10 uses our assumption that $F_1$ preserves pullbacks in $M$-morphisms. Since we do not require $G_0$ to preserve pushouts of $E$-morphisms, the corresponding statement that every $G_0$-quotient algebra of $(G^K_0 1, G^K_0 i)$ is corecursive does not hold. However, we have the following weaker result:

**Proposition A.11.** At each stage of Generalized $\mathcal{L}^*$, the algebra $(T, \tau)$ is corecursive.

**Proof.** Recall that $(T, \tau)$ is a quotient algebra $t: (G^K_0 1, G^K_0 i) \rightarrow (T, \tau)$ for some $K > 0$. We need to show that (1) $(T, \tau)$ is corecursive after its initialization in Step 0 of the algorithm, and that (2) every application of "Extend $t" preserve corecursivity.

**Proof of (1).** Initially, we have $(T, \tau) = (G_0 1, G_0 i)$. Since the algebra $(1, !)$ is trivially corecursive by terminality of 1, the dual of Lemma A.9 shows that $(T, \tau)$ is corecursive.
**Proof of (2).** Suppose that \((T, \tau)\) is corecursive. Applying “Extend \(t'\)” replaces \((T, \tau)\) by the algebra \((T', t_0 \cdot G_O t_1)\), where \(\tau = t_1 \cdot t_0\). Then \(t_0 : (G_O T, G_O \tau) \rightarrow (T', t_0 \cdot G_O t_1)\) and \(t_1 : (T', t_0 \cdot G_O t_1) \rightarrow (T, \tau)\) are \(G_O\)-algebra homomorphisms, as shown by the diagram below.

![Diagram](image)

To show that \((T', t_0 \cdot G_O t_1)\) is corecursive, let \((C, \gamma)\) be a \(G_O\)-coalgebra. We need to prove that there is a unique coalgebra-to-algebra homomorphism \(h\) from \((C, \gamma)\) into \((T', t_0 \cdot G_O t_1)\).

**Existence.** Since \((T, \tau)\) is corecursive, the algebra \((G_O T, G_O \tau)\) is also corecursive by the dual of Lemma A.9. Thus, there exists a unique coalgebra-to-algebra homomorphism \(h'\) from \((C, \gamma)\) into \((G_O T, G_O \tau)\). It follows that \(h = t_0 \cdot h'\) is a coalgebra-to-algebra homomorphism from \((C, \gamma)\) into \((T', t_0 \cdot G_O t_1)\), being the composite of the coalgebra-to-algebra homomorphism \(h'\) with the algebra homomorphism \(t_0\).

**Uniqueness.** Let \(h\) be a coalgebra-to-algebra homomorphism from \((C, \gamma)\) into \((T', t_0 \cdot G_O t_1)\), and denote by \(g\) the unique coalgebra-to-algebra homomorphism from \((C, \gamma)\) into the corecursive algebra \((T, \tau)\). Since also \(t_1 \cdot h\) is such a homomorphism (being the composite of a coalgebra-to-algebra homomorphism with an algebra homomorphism), we have \(t_1 \cdot h = g\). From the commutative diagram below it then follows that \(h = t_0 \cdot G_O g \cdot \gamma\), which shows that \(h\) is uniquely determined by \(g\).

![Diagram](image)

**Lemma A.12.** Let \((s, t)\) be closed and consistent, and suppose that the algebra \((T, \tau)\) is corecursive. Then the associated hypothesis automaton \(H_{s, t}\) (see Definition 4.8) is minimal. Moreover, the two diagrams below commute:

![Diagram](image)

In particular, by Proposition A.11, this lemma applies to the pairs \((s, t)\) constructed in the generalized \(L^*\) algorithm.

**Proof.** (1) We first prove that the left-hand diagram commutes. Consider the \(F_I\)-algebra structure on \(H_{s, t}\) given by

\[
[i_{s, t}, \delta_{s, t}] : F_I H_{s, t} \rightarrow H_{s, t}
\]

Then \(\varepsilon_{s, t} : (S, \sigma) \rightarrow (H_{s, t}, [i_{s, t}, \delta_{s, t}])\) is a coalgebra-to-algebra homomorphism, as shown by the commutative diagram below:

![Diagram](image)

Indeed, the upper left part commutes by the definition of \(c_{s, t}\), and the lower right part commutes by definition of \(i_{s, t}\) and \(\delta_{s, t}\) (consider the two coproduct components of \(F_I S = I + FS\) separately).

Since also \(\varepsilon_{H_{s, t}, t} \cdot j_N \cdot s : (S, \sigma) \rightarrow (H_{s, t}, [i_{s, t}, \delta_{s, t}])\) is a coalgebra-to-algebra homomorphism (being the composite of the \(F_I\)-coalgebra homomorphism \(s\), the coalgebra-to-algebra homomorphism \(j_N\) and the \(F_I\)-algebra homomorphism \(\varepsilon_{H_{s, t}}\)) and the coalgebra \((S, \sigma)\) is recursive by Proposition A.10, we conclude that \(\varepsilon_{s, t} = \varepsilon_{H_{s, t}, t} \cdot j_N \cdot s\).

(2) The proof that the right-hand diagram commutes is completely analogous: one views \(H_{s, t}\) as a \(G_O\)-coalgebra

\[
(f_{s, t}, \delta_{s, t}^\oplus) : H_{s, t} \rightarrow G_O H_{s, t},
\]

where \(\delta_{s, t}^\oplus : H_{s, t} \rightarrow G_O H_{s, t}\) denotes the adjoint transpose of \(\delta_{s, t} : F_I H_{s, t} \rightarrow H_{s, t}\), and shows that both \(m_{s, t}\) and \(t \cdot j_K \cdot m_{H_{s, t}}\) are coalgebra-to-algebra homomorphisms from \((H_{s, t}, (f_{s, t}, \delta_{s, t}^\oplus))\) into the corecursive algebra \((T, \tau)\).

(3) Since \(\varepsilon_{s, t} \in \mathcal{E}\) and \(m_{s, t} \in \mathcal{M}\), it follows from the two commutative diagrams that \(\varepsilon_{H_{s, t}} \in \mathcal{E}\) and \(m_{H_{s, t}} \in \mathcal{M}\) (see [2, Prop. 14.11]). Thus, the automaton \(H_{s, t}\) is minimal by Corollary A.8.

An important invariant of the generalized \(L^*\) algorithm is that the subcoalgebra \(s\) is pointed and that the quotient algebra \(t\) is co-pointed:

**Definition A.13.** An \(F_I\)-coalgebra \((R, \rho)\) is pointed if there is a morphism \(i_R\) such that the left-hand triangle below commutes. A \(G_O\)-algebra \((B, \beta)\) is co-pointed if there is a morphism \(f_B\) such that the right-hand triangle below commutes:
Note that if \((R, \phi)\) is a subcoalgebra of \((F_i M_0, F_i M_1)\), then \(i_R\) is necessarily unique because \(F_i M_1\) is monic by Assumptions 3.5(2) and Assumptions 4.1. Dually for co-pointed quotient algebras of \((G_O^M 0, G_O^M 1)\).

**Lemma A.14.** At each stage of the generalized \(L^*\) algorithm, the coalgebra \((S, \sigma)\) is pointed and the algebra \((T, \tau)\) is co-pointed.

**Proof.** We proceed by induction on the number of steps of the algorithm required to construct the pair \((s, t)\). Initially, after Step \((0)\), \((S, \sigma)\) is equal to \((I, F_i i)\), and thus pointed via \(i_S = \text{id}_I\).

![Diagram](image.png)

Dually, \((T, \tau)\) is co-pointed via \(f_T = \text{id}_O\).

Now suppose that at some stage of the algorithm, \((S, \sigma)\) is pointed and \((T, \tau)\) is co-pointed. We need to show that \((S, \sigma)\) remains pointed after executing "Extend \(s\)" or adding a counterexample to \(s\), and that \((T, \tau)\) remains co-pointed after executing "Extend \(t\).

1. **Extend \(s\).** When calling "Extend \(s\)\", the coalgebra \((S, \sigma)\) is replaced by the coalgebra \((S', F_i s_0 \cdot s_i)\). This coalgebra is pointed via \(i_{S'} = s_0 \cdot i_S\), as witnessed by the commutative diagram below:

![Diagram](image.png)

2. **Extend \(t\).** Symmetric to (1).

3. **Adding a counterexample.** Let \((C, \gamma)\) be the counterexample added to \((S, \sigma)\), and denote by \(i: (S, \sigma) \rightarrow (S \vee C, \sigma \vee \gamma)\) the embedding. Then the coalgebra \((S \vee C, \sigma \vee \gamma)\) is pointed via \(i_{S\vee C} = i \cdot i_S\), as shown by the commutative diagram below:

![Diagram](image.png)

**Lemma A.15.** Let \(A\) be an automaton. For any pointed subcoalgebra \(r: (R, \phi) \rightarrow (F_i M_0, F_i M_1)\), we have

\[
i_A = (I \xrightarrow{i_R} R \xrightarrow{F_i M_0} \mu F_i \xrightarrow{\epsilon_A} A)
\]

Dually, for any co-pointed quotient algebra \(b: (G_O^M 1, G_O^M 1) \rightarrow (B, \beta)\), we have

\[
f_A = (A \xrightarrow{m_A} v G_O \xrightarrow{j_M} G_O^M 1 \xrightarrow{\nu b} O).
\]

**Proof.** The first statement follows from the commutative diagram below, all of whose parts either commute trivially or by definition.

![Diagram](image.png)

The proof of the second statement is dual. \(\square\)

**Proposition A.16.** Let \((s, t)\) be a closed and consistent pair as in (3), and suppose that \(t\) is co-pointed. Then the hypothesis \(H = H_{s,t}\) and the unknown automaton \(Q\) have the same observation tables for \((s, t)\):

\[h^H_{s,t} = h^Q_{s,t}.
\]

In particular, \(H\) and \(Q\) agree on inputs from \(S\), that is,

\[L_H \cdot j_N \cdot s = L_Q \cdot j_N \cdot s.
\]

**Proof.** (1) For the first equality, consider the following diagram:

![Diagram](image.png)

The outward commutes by definition of \(h_{s,t}\) and since \(h_{s,t} = m_{s,t} \cdot e_{s,t}\). The upper left and lower left parts commute by Lemma A.12. It follows that the remaining part commutes.
when precomposed with \(j_N \cdot s\) and postcomposed with \(t \cdot j'_K\), which gives \(h^H_{s,t} = h^Q_{s,t}\).

The second equality follows by postcomposing both sides of the equality \(h^H_{s,t} = h^Q_{s,t}\) with \(f_T : T \to O\) and applying Lemma A.15.

---

**The key to the termination of the learning algorithm lies in the following result.**

**Lemma A.17.** Let \((s, t)\) be a closed and consistent pair as in (3), and suppose that \(t\) is co-pointed. Then for every counterexample \(c\) for \(H_{s,t}\), the pair \((s \vee c, t)\) is not closed or not consistent.

**Proof:** Suppose for the contrary that the pair \((s \vee c, t)\) is closed and consistent. Denote by

\[
i : S \hookrightarrow S \vee C \quad \text{and} \quad i' : C \to S \vee C
\]

the two embeddings, satisfying \((s \vee c) \cdot i = s\) and \((s \vee c) \cdot i' = c\).

Via diagonal fill-in we obtain a unique \(j : H_{s,t} \to H_{s \vee c, t}\) such that the following diagram commutes:

\[
\begin{array}{ccc}
S & \xrightarrow{i} & S \vee C \\
\downarrow{e_{s,t}} & & \downarrow{e_{s \vee c,t}} \\
H_{s,t} & \xrightarrow{j} & H_{s \vee c,t} \\
\downarrow{m_{s,t}} & & \downarrow{m_{s \vee c,t}} \\
T & \xrightarrow{j} & T
\end{array}
\]

We shall show below that \(j\) is an automata homomorphism. In particular, \(H_{s,t}\) and \(H_{s \vee c,t}\) accept the same language by Proposition A.4. Letting \(H = H_{s \vee c,t}\), we compute

\[
L_{H_{s,t}} \cdot j_N \cdot c = L_H \cdot j_N \cdot c
\]

\[
= f_{H} \cdot e_{H} \cdot j_N \cdot c
\]

\[
= f_{H} \cdot e_{H} \cdot j_N \cdot c
\]

\[
= f_{H} \cdot j'_K \cdot m_{H} \cdot e_{H} \cdot j_N \cdot c
\]

by Lemma A.15

\[
= f_{H} \cdot j'_K \cdot m_{H} \cdot e_{H} \cdot j_N \cdot (s \vee c) \cdot i'
\]

\[
= f_{H} \cdot j'_K \cdot m_{H} \cdot e_{H} \cdot j_N \cdot (s \vee c) \cdot i'
\]

\[
= f_{H} \cdot h^H_{s \vee c,t} \cdot i'
\]

\[
= f_{H} \cdot h^Q_{s \vee c,t} \cdot i'
\]

\[
= \ldots
\]

\[
= L_Q \cdot j_N \cdot c
\]

compute backwards

This contradicts the fact that \(c\) is a counterexample for \(H_{s,t}\).

To conclude the proof, it only remains to verify our above claim that \(j\) is an automata homomorphism.

1. \(j\) preserves transitions. Observe first that we have

\[
m_{s,t} \cdot i_{s,t} = m_{s \vee c,t} \cdot i_{s \vee c,t} \cdot Fi
\]

as shown by the commutative diagram below:

Here the left-hand part commutes by naturality of \(\text{inr}\), the central triangle commutes by definition of \(h_{s,t}\) (using that \((s \vee c) \cdot i = s\), and all remaining parts commute by definition.

Now, consider the following diagram:

The outward commutes by (6), and all parts except the central square commute by definition. It follows that also the central square commutes, because it commutes when precomposed with the epimorphism \(F_{e_{s,t}}\) and postcomposed with the monomorphism \(m_{s \vee c,t}\). Thus, \(j\) preserves transitions.

2. \(j\) preserves the initial state. Observe first that we have

\[
m_{s,t} \cdot i_{s,t} = m_{s \vee c,t} \cdot i_{s \vee c,t}
\]
as shown by the commutative diagram below:

Now consider the following diagram:

The outward commutes by (7), and the right-hand triangle by the definition of \( j \). Thus the left-hand part commutes, since it does when postcomposed with the monomorphism \( m_{s \vee c} \). This proves that \( j \) preserves the initial state.

(3) \( j \) preserves final states. The proof is analogous to (2). \( \square \)

With the above results at hand, we are ready to prove Theorem 4.12:

**Proof of Theorem 4.12.** The algorithm only terminates if a hypothesis \( H_{s,t} \) constructed in Step (2) is correct (i.e. it accepts the same language as the unknown automaton \( Q \)), in which case \( H_{s,t} \) is returned. This automaton is minimal by Lemma A.12, so \( H_{s,t} = \text{Min}(L_Q) \).

Thus, we only need to verify that the algorithm eventually finds a correct hypothesis. For any \( F_I \)-subcoalgebra \( r : (R, Q) \rightarrow (F^M_I 0, F^M_I) \), let \( e_r \) and \( m_r \) denote the \((E, M)\)-factorizations of \( e_Q \cdot j_M \cdot r \).

Similarly, for any \( G_Q \)-quotient algebra \( b : (G^M_Q 1, G^M_Q) \rightarrow (B, \beta) \), let \( \bar{e}_b \) and \( \bar{m}_b \) be the \((\bar{E}, \bar{M})\)-factorization of \( b \cdot j_M \cdot m_Q \).

Let \( (s, t) \) and \( (s', t') \) be two consecutive pairs appearing in an execution of the algorithm. We show below that the following statements hold:

1. If \( (s', t') \) emerges from \( (s, t) \) via "Extend \( s' \)" then \( m_s < m_{s'} \) and \( \bar{e}_t = \bar{e}_{t'} \).
2. If \( (s', t') \) emerges from \( (s, t) \) via "Extend \( t' \)" then \( m_s = m_{s'} \) and \( \bar{e}_s < \bar{e}_{s'} \).
3. If \( (s', t') \) emerges from \( (s, t) \) by adding a counterexample, then \( m_s \leq m_{s'} \) and \( \bar{e}_t = \bar{e}_{t'} \).

Letting \( (s^1, t^1), (s^2, t^2), \ldots \) denote the sequence of pairs constructed in an execution of the algorithm, it follows that we obtain two ascending chains

\[
m_{s^0} \leq m_{s^1} \leq \cdots \quad \text{and} \quad \bar{e}_{t^0} \leq \bar{e}_{t^1} \leq \cdots.
\]

of subobjects and quotients of \( Q \), respectively. By our assumption that \( Q \) is Noetherian, both chains must stabilize, i.e. all but finitely many of the relations \( \leq \) are equalities. By (1) and (2), this implies that "Extend \( s' \)" and "Extend \( t' \)" are called only finitely often. Moreover, whenever a counterexample is added to \( s \), this must be immediately followed by a call of "Extend \( s' \)" or "Extend \( t' \)" by Lemma A.17. Thus also Step (2b) is executed only finitely often. This proves that the algorithm necessarily terminates.

It remains to establish the above statements (1)–(3).

1. An application of "Extend \( s' \)" to \( (s, t) \) yields the new pair \( (s', t') \) with

\[
s' = F_I s \cdot s_1 \quad \text{and} \quad t' = t.
\]

Thus, we trivially have \( \bar{e}_t = \bar{e}_{t'} \). Moreover, \( m_s \leq m_{s'} \) holds by the right-hand triangle in the diagram below, where the morphism \( n_{s, s'} \) is obtained via diagonal fill-in:

To prove \( m_s < m_{s'} \), we need to show that \( n_{s, s'} \) is not an isomorphism. To this end, consider the unique morphisms \( d_s \) and \( d_{s'} \) (defined via diagonal fill-in) such that the diagrams
below commute:

\[
\begin{array}{ccc}
S & \xrightarrow{s} & F_1^N \xrightarrow{ij_N} \mu F_I \\
& \downarrow{e_s} & \downarrow{\mu F_I} \\
H_t, t & \xrightarrow{Q_t, m_t} & Q \\
& \downarrow{Q_t} & \downarrow{m_Q} \\
T & \xleftarrow{t} & G_{O_1} ^K \xleftarrow{v_{G_O}} \\
\end{array}
\quad \quad
\begin{array}{ccc}
S' & \xrightarrow{s'} & F_1^{N+1} \xrightarrow{ij_{N+1}} \mu F_I \\
& \downarrow{e_{s'}} & \downarrow{\mu F_I} \\
H_{t'}, t' & \xrightarrow{Q_{t'}, m_{t'}} & Q \\
& \downarrow{Q_{t'}} & \downarrow{m_{Q_{t'}}} \\
T & \xleftarrow{t'} & G_{O_1} ^K \xleftarrow{v_{G_O}} \\
\end{array}
\]

Moreover, observe that we have the following commutative diagram:

\[
\begin{array}{ccc}
S' & \xrightarrow{s'} & Q' \\
& \downarrow{h_{s', t'}} & \downarrow{m_{s', t'}} \\
T & \xleftarrow{t'} & Q' \\
\end{array}
\]

By the choice of \(s_1\) in "Extend \(s\)", we have \(e_{F_J s, t} \cdot s_1 \in \mathcal{E}\). The uniqueness of \((\mathcal{E}, \mathcal{M})\)-factorizations thus implies that, up to isomorphism,

\[H_{s', t'} = H_{F_J s, t}, \quad e_{s', t} = e_{F_J s, t} \cdot s_1, \quad m_{s', t} = m_{F_J s, t}.\]

We now claim that the following diagram commutes:

\[
\begin{array}{ccc}
Q & \xrightarrow{n_{s, s'}} & Q' \\
& \downarrow{e_{s, t}} & \downarrow{e_{s', t}} \\
S & \xrightarrow{s_0} & S' \\
& \downarrow{h_{s, t}} & \downarrow{h_{s', t}} \\
H_t, t & \xrightarrow{c_{s, t}} & H_{F_J s, t} = H_{s', t'} \\
\end{array}
\]

All inner parts commute by definition. Thus also the outward commutes, since it does when precomposed with the epimorphism \(e_s\) and postcomposed with the monomorphism \(m_{s', t}\).

We are ready to prove our claim that \(n_{s, s'}\) is not an isomorphism. Suppose for the contrary that it is. Since \(d_{s'} \in \mathcal{E}\), the diagram (8) yields \(c_{s, t} \cdot d_{s'} = d_{s'} \cdot n_{s, s'} \in \mathcal{E}\). Thus \(c_{s, t} \in \mathcal{E}\). One the other hand, by definition of \(c_{s, t}\) we have \(m_{F_J s, t} \cdot c_{s, t} = m_{s, t} \in \mathcal{M}\) and thus \(c_{s, t} \in \mathcal{M}\). But from \(c_{s, t} \in \mathcal{E} \cap \mathcal{M}\) it follows that that \(c_{s, t}\) is an isomorphism [2, Prop. 14.6], contradicting the fact that the input pair \((s, t)\) of "Extend \(s\)" is not closed.

(2) The proof is symmetric to (1).

(3) Adding a counterexample \(c\) means to to replace the pair \((s, t)\) by the pair \((s', t')\) with

\[s' = s \cup c \quad \text{and} \quad t' = t.\]

Thus \(\tau_{t} = \tau_{t'}\). Letting \(i: (S, \sigma) \mapsto (S \cup C, \sigma \cup \gamma) = (S', \sigma')\) denote the embedding with \(s = (s \cup c) \cdot \iota, \) diagonal fill-in yields a morphism \(n_{s, s'}\) making the diagram below commute:

\[
\begin{array}{ccc}
S & \xrightarrow{e_s} & Q \\
& \downarrow{c_{s, t}} & \downarrow{m_{s}} \\
T & \xleftarrow{t} & G_{O_1} ^K \xleftarrow{v_{G_O}} \\
\end{array}
\quad \quad
\begin{array}{ccc}
S' & \xrightarrow{e_{s'}} & Q' \\
& \downarrow{c_{s', t'}} & \downarrow{m_{s'}} \\
T & \xleftarrow{t'} & G_{O_1} ^K \xleftarrow{v_{G_O}} \\
\end{array}
\]

This proves that \(m_s \leq m_{s'}\). \(\Box\)

**Details for Remark 4.13**

Let \(m\) and \(n\) be the height (i.e. the length of the longest strictly ascending chain) of the poset of subobjects and quotients of \(Q\), respectively. The proof of Theorem 4.12 shows that

(1) "Extend \(s\)" is executed at most \(m\) times;

(2) "Extend \(t\)" is executed at most \(n\) times;

(3) Step (2b) is executed at most \(m + n\) times.

Thus, Steps (1a), (1b) and (2b) are executed at most \(2m + 2n = O(m + n)\) times.

**Details for Example 4.14**

(1) The statements for \(\mathcal{D} = \text{Set, Pos, K-Vec}\) are clear.

(2) \(\mathcal{D} = \text{JSL}\): clearly every finite semilattice is Noetherian. Conversely, if \(Q\) is a infinite semilattice, choose a sequence \(q_0, q_1, q_2, \ldots\) of elements of \(Q\) such that \(q_{n+1}\) is not an element of the sub-semilattice \(\langle q_0, \ldots, q_n \rangle\) of \(Q\) generated by \(q_0, \ldots, q_n\). Since this subsemilattice is finite (of cardinality at most \(2^{n+1}\)), such a \(q_{n+1}\) can always be chosen. Then

\[\langle q_0 \rangle \rightarrow \langle q_0, q_1 \rangle \rightarrow \langle q_0, q_1, q_2 \rangle \rightarrow \ldots\]

is an infinite strictly ascending chain of subsemilattices of \(Q\), showing that \(Q\) is not Noetherian.

(3) \(\mathcal{D} = \text{Nom}\): We show that orbit-finite sets have the claimed polynomial bound. Let \(X\) be an orbit-finite nominal set with \(n\) orbits. It is clear that chains of subobjects, i.e. equivariant subsets, of \(X\) have length at most \(n\). It remains to show the polynomial bound on chains of quotients. The number of orbits decreases non-strictly along such a chain, and can strictly decrease at most \(n\) times, so it suffices to consider chains of quotients that retain the same number of orbits. Such quotients are sums of quotients of single-orbit sets, so it suffices to consider the case where \(X\) has only one orbit. Then, all elements of \(X\) have supports of the same size \(k\); since this number decreases non-strictly along a chain...
of quotients, and can strictly decrease at most $k$ times, it suffices to consider chains of quotients that retain the same support size.

We now use the standard fact that $X$ is a quotient of $A^k$, the $k$-fold separated product of $A$; the same, of course, holds for all quotients of $X$. A quotient of $A^k$ whose elements retain supports of size $k$ is determined by a subgroup $G$ of the symmetric group $S_k$. (Specifically, the quotient determined by $G$ identifies $(a_1, \ldots, a_k)$ and $(a_{\pi(1)}, \ldots, a_{\pi(k)})$ for all $(a_1, \ldots, a_k) \in A^k$ and $\pi \in G$. Conversely, from a given quotient $e : X \to Y$, we obtain $G$ as consisting of all $\pi \in S_k$ such that $e$ identifies $(a_1, \ldots, a_k)$ and $(a_{\pi(1)}, \ldots, a_{\pi(k)})$ for all $(a_1, \ldots, a_k) \in A^k$.) The given chain of quotients thus corresponds to a chain of subgroups of $S_k$, which for $k \geq 2$ has length at most $2k - 3$ [11].

**Details for Remark 4.16**

We demonstrate that the coalgebraic learning algorithm in [14] gets stuck when applied to the setting of $\Sigma$-automata in $\text{Nom}$. In the following, we assume some familiarity with the algorithm and the notation introduced in op. cit.

A coalgebraic logic giving the semantics of nominal automata can be described in complete analogy to the $\text{Set}$ case [14, Example 1]. We instantiate the logical framework to

\[
\begin{array}{ccc}
\text{Nom}^\text{op} & \xrightarrow{P} & \text{Nom} \\
L^P & \xrightarrow{\gamma} & \text{Nom} \\
\text{Nom} & \xleftarrow{\nu} & \text{Nom}
\end{array}
\]

where

\[
LX = 1 + A \times X, \quad BX = 2 \times [A, X], \quad P = [-, -]
\]

The right adjoint of $P$ is $Q = [-, 2] : \text{Nom}^\text{op} \to \text{Nom}$. For each $X \in \text{Nom}$, the map

\[
\delta_X : 1 + A \times [X, 2] \to [2 \times [A, X], 2]
\]

sends the unique element of 1 to the left product projection, and $(a, f) \in A \times [X, 2]$ to $\delta_X(a, f) \in [2 \times [A, X], 2]$ with

\[
\delta_X(a, f)(b, g) = f(g(a)) \quad \text{for} \quad b \in 2, g \in [A, X].
\]

We have the initial algebra for $L$ given by $\Phi = \mu L = A^*$, and the theory map

\[
\text{th}^\nu : X \to Q\Phi = [A^*, 2]
\]

for a nominal automaton (i.e. $B$-coalgebra) $X$ is just the unique coalgebra homomorphism from $X$ into the final coalgebra $vB = [A^*, 2]$ (cf. Example 3.9).

Now consider the nominal language $K : A^* \to 2$ with $K(\omega) = 1$ iff $\omega$ has even length. We assume that the unknown coalgebra is given by

\[
(X \xrightarrow{\gamma} BX) = (A^* \xrightarrow{(K, \gamma')} 2 \times [A, A^*])
\]

with $\gamma'(\omega)(a) = wa$ for $w \in A^*$, $a \in A$. (The state set $X$ is effectively made known to the learner in advance since the learning algorithm computes subobjects of $X$. Thus, in the typical scenario $X$ will be orbit-infinite like in the present example, although of course the language $K$ can be accepted by an orbit-finite automaton.) The algorithm starts with the trivial observation table

\[
S = \{\epsilon\} \to X \quad \text{and} \quad \Psi = \emptyset \to \Phi,
\]

This table is closed and the induced conjecture is the trivial one-state automaton accepting all words in $A^*$. Since $a \notin K$ for $a \in A$, the teacher provides the (minimal) counterexample $\{\epsilon\} + A \to \Phi$. After adding it to $\Psi$, the new table is

\[
S = \{\epsilon\} \to X \quad \text{and} \quad \Psi = \{\epsilon\} + A \to \Phi.
\]

The next reachability step computes the set $\Gamma(S)$ of elements of $X$ reachable from $S = \{\epsilon\}$ in a single transition step:

\[
\Gamma(S) = A.
\]

Thus

\[
S \lor \Gamma(S) = S \cup \Gamma(S) = \{\epsilon\} + A \to X.
\]

Viewing the elements of $Q\Psi = \{\{\epsilon\} + A, 2\}$ as finitely supported subsets of $\Psi = \{\epsilon\} + A$, we can describe the map

\[
S \lor \Gamma(S) \to X \xrightarrow{\text{th}^\nu} Q\Psi
\]

as sending $\epsilon$ to $\{\epsilon\} \subseteq \Psi$ and every $a \in A$ to $A \subseteq \Psi$, i.e. the image of this map is the discrete nominal set

\[
\overline{S} = \{\{\epsilon\}, A\} \equiv 2.
\]

In order to close the table, Step 6 of the algorithm now requires to choose a monomorphism $\overline{S} \to X$ subject to certain conditions. But clearly there exists no monomorphism from $\overline{S} = 2 \to X = A^*$ in $\text{Nom}$, i.e. the algorithm cannot make the required choice.

**Details for Example 5.9**

Our categorical notion of automata presentation involves quotients of $T$-algebras. For practical purposes, it is sometimes more convenient to work with the equivalent concept of a congruence:

**Remark A.18.** (1) Recall that for a monad $T$ on $\text{Set}$ given by a finitary signature $\Gamma$ and equations $E$ between $\Gamma$-terms, quotient algebras of a $T$-algebra (i.e. $(\Gamma, E)$-algebra) $A$ correspond bijectively to congruences on $A$. Here a congruence $\equiv$ is an equivalence relation $\equiv$ on $A$ respecting all $\Gamma$-operations: for all $a, a' \in A$ with $a \equiv a'$, one has

\[
y(a_1, \ldots, a_{i-1}, a, a_{i+1}, \ldots, a_n) \equiv y(a_1, \ldots, a_{i-1}, a', a_{i+1}, \ldots, a_n)
\]

for $n > 0$, $y \in \Gamma_n$, $i \in \{1, \ldots, n\}$ and $a_j \in A$ ($j \neq i$). The bijection identifies a quotient $e : A \to B$ with its kernel, i.e. the congruence given by

\[
a \equiv a' \iff e(a) = e(a')
\]

Thus, if the object $TI$ is equipped with some $\Sigma$-automata structure $\Sigma \times TI \xrightarrow{\delta} TI$, the equivalence in Definition 5.7(3)
states precisely that an equivalence relation \( \equiv \) on \( T \) corresponding to \( T \)-refinable quotients is a congruence on \( T \) if and only if \( w \equiv w' \) implies \( \delta(a, w) \equiv \delta(a, w') \).

(2) An analogous remark applies to monads \( T \) on Set\(^{5} \) corresponding to a finitary \( S \)-sorted signature \( \Gamma \) and equations between \( \Gamma \)-terms: quotient algebras of a \((T, E)\)-algebra \( A \) correspond to \( S \)-sorted congruence relations, i.e. families of equivalence relations \( \equiv \) (\( \equiv_{S} \subseteq A_{x} \times A_{x} \)) respecting all operations. Thus, if \( T \) is equipped with the structure of a sorted \( \Sigma \)-automaton \( \delta_{\Sigma,t} : \Sigma_{\times t} \times (TI)_{s} \rightarrow (TI)_{t} \) \((s, t \in S)\), the equivalence in Definition 5.7(3) holds for arbitrary, not only \( T \)-refinable, quotients.

**Semigroups.** The free semigroup \( T_{+} I = I^{+} \) has a \( \Sigma \)-automata presentation \( \delta : \Sigma \times I^{+} \rightarrow I^{+} \) given by the alphabet

\[ \Sigma = \{ \tilde{a} : a \in I \} \cup \{ \tilde{a} : a \in I \} \]

and the transitions

\[ \delta(\tilde{a}, w) = wa \quad \text{and} \quad \delta(\tilde{a}, w) = aw \quad \text{for} \quad w \in I^{+}, a \in I. \]

We show that (1)–(3) of Definition 5.7 (with \( F = \Sigma \times I^{+} \) on Set) are satisfied. (1) is clear by Remark 5.4. For (2), recall from Example 3.11 that the initial algebra \( \mu F_{I} \) consists of sorted words over \( \Sigma \) with an additional first letter from \( I \). The homomorphism \( e_{(I^{+}, I^{op})} : \mu F_{I} \rightarrow (I^{+}, I^{op}) \) views such a word as an instruction for forming a word in \( (I^{+}, I^{op}) \), e.g.

\[ e_{(I^{+}, I^{op})}(a b a a a a a) = aa(aba)^{op}. \]

Thus \( e_{(I^{+}, I^{op})} \) is surjective: every finite word \( w \in I^{+} \) is in the image of \( e_{(I^{+}, I^{op})} \) as in the case of semigroups, and for an ultimately periodic word \((a_{1}, \ldots, a_{n})(b_{1}, \ldots, b_{m})^{op} \in I^{op} \) we have

\[ (a_{1}, \ldots, a_{n})(b_{1}, \ldots, b_{m})^{op} = e_{(I^{+}, I^{op})}(b_{1} b_{2} \cdots b_{m} o \tilde{a}_{n} \cdots \tilde{a}_{1}). \]

To show (3), we use Remark A.18(2): we need to verify that a two-sorted equivalence relation \( \equiv \) on \((I^{+}, I^{op}) \) is a congruence w.r.t. the Wilke algebra structure iff, for each \( w, w' \in I^{+} \) with \( w \equiv w' \) and \( a \in I \), one has

\[ aw \equiv aw', \quad wa \equiv w'a, \quad w^{op} \equiv (w')^{op}, \quad w^{op} \equiv w'^{op}, \]

and for each \( z, z' \in I^{op} \) with \( z \equiv z' \) and \( a \in I \) one has \( a z \equiv a z' \). The "only if" direction is clear. For the "if" direction, we need to show that for all \( w, w' \equiv w^{op} \) and \( v \in I^{+} \),

- \( w \equiv w' \) implies \( uvw \equiv uvw' \),
- \( w \equiv w' \) implies \( wv \equiv w'^{op} \),
- \( z \equiv z' \) implies \( wz \equiv wz' \).

Let us show that \( w \equiv w' \) implies \( wz \equiv wz' \); the proofs of the other statements are similar. We have \( z = a_{1}, \ldots, a_{n} y^{op} \) with \( a_{1}, \ldots, a_{n} \in I \) and \( y \in I^{+} \). From \( w \equiv w' \) it follows that

\[ wz = w_{1} a_{1}, \quad wa_{1} a_{2} \equiv w'_{1} a_{1} a_{2}, \quad \cdots, \quad wa_{1} a_{2} \equiv w'_{1} a_{1} a_{2}, \quad \cdots, \quad wa_{1} a_{n} \equiv w'_{1} a_{1} a_{n} \]

and thus

\[ wz = w_{1} a_{1} a_{n} y^{op} \equiv w'_{1} a_{1} a_{n} y^{op} = w'_{z}. \]
Stabilization algebras. Suppose that $T$ is a monad on Set or Pos induced by a finitary signature $\Gamma$ and (in-)equations $E$; see Section 2. Then $TI$ can be presented as the $\Gamma$-automaton $\delta: F_T(I) \to TI$ given by the $\Gamma$-algebra structure on the free $(\Gamma, E)$-algebra $TI$. We show that (1)–(3) of Definition 5.7 are satisfied.

(1) is clear by Remark 5.4. For (2), observe that the initial algebra $\mu(F_T I)$ is the algebra $T_I$ of $\Gamma$-terms over $I$, and that the unique $\Gamma$-homomorphism $\epsilon_{T_I}: T_I \to TI$ interprets $\Gamma$-terms in $TI$. Since the $\Gamma$-algebra $TI$ is generated by the set $I$ as a $\Gamma$-algebra, every element of $TI$ can be expressed as a $\Gamma$-term over $I$, i.e. $\epsilon_{T_I}$ is surjective. (3) is clear: the equivalence just amounts to the statement that if $e$ is a surjective homomorphism of (ordered) $\Gamma$-algebras and its domain satisfies all (in-)equations in $E$, then so does its codomain.

By instantiating to the monad $T = T_S$ on Pos, we see that the free stabilization algebra $T_S I$ has a $\Gamma$-automata presentation for the signature $\Gamma$ of Example 5.5(3).

Proof of Theorem 5.12

Suppose that $L$ is recognized via $e: TI \to (A, \alpha)$ and $p: A \to O$, where $(A, \alpha)$ is a finite $T$-algebra. We may assume that $e \in E$. (Otherwise consider the $(E, \mathcal{M})$-factorization

$$TI \xrightarrow{e'} (A', \alpha') \xrightarrow{m} (A, \alpha)$$

of $e$. Since $\mathcal{D}_f$ is closed under subobjects, $L$ is recognized by the finite $T$-algebra $(A', \alpha')$ via $e'$ and $p \cdot m$, i.e. we can replace $e$ by $e'$.

Since $(F, \delta)$ forms a weak automata presentation, the object $A$ can be equipped with an $F$-algebra structure $\delta_A: FA \to A$ such that $e: (TI, \delta) \to (A, \delta_A)$ is an $F$-algebra homomorphism. Equipping $TI$ and $A$ with the initial states $\eta_I: I \to TI$ and $e \cdot \eta_I: I \to A$, respectively, we can view $TI$ and $A$ as $F_I$-algebras and $e$ as an $F_I$-algebra homomorphism. By initiality of $\mu F_I$, it follows that $\epsilon_A = e \cdot \epsilon_{T_I}$. It follows that the diagram below commutes, which proves that the automaton $(A, \delta_A, e \cdot \eta_I, p)$ accepts the language $\text{lin}(L) = L \cdot \epsilon_{T_I}$.

$$\mu F_I \xrightarrow{\epsilon_{T_I}} TI \xrightarrow{e} A \xrightarrow{e} B$$

Proof of Theorem 5.14

The proof is illustrated by the diagram below:

Let $A = \text{Min}(\text{lin}(L))$ be the minimal automaton for the language $\text{lin}(L)$. Equipping $TI$ with the initial states $\eta_I: I \to TI$ and the final states $L: TI \to O$, we can view $TI$ as an automaton accepting $\text{lin}(L) = L \cdot \epsilon_{T_I}$. Since $\epsilon_{T_I} \in E$ (that is, the automaton $TI$ reachable) and $A$ is minimal, there exists a unique automata homomorphism $e: TI \to A$. We now prove the theorem by establishing the following claims:

Claim 1. For every finite quotient $T$-algebra $e': TI \to (B, \beta)$ that recognizes $L$, there exists a unique $h: B \to A$ with $e = h \cdot e'$.

Proof. As in the proof of Theorem 5.12, $B$ can be viewed as a reachable automaton recognizing $\text{lin}(L)$. By minimality of $A$, there is an automata homomorphism $h: B \to A$. We have $h \cdot e' \cdot \epsilon_{T_I} = e \cdot \epsilon_{T_I}$ because both sides are $F_I$-algebra homomorphisms from $\mu F_I$ to $B$ and $\mu F_I$ is initial. Thus $h \cdot e' = e$ because $\epsilon_{T_I}$ is epic.

Claim 2. The automaton $A$ can be equipped with $T$-algebra structure $(A, \alpha_A)$ such that $e: TI \to (A, \alpha_A)$ is a $T$-homomorphism.

Proof. Since $L$ is $T$-recognizable, we have $L = p' \cdot e'$ for some finite quotient $T$-algebra $e': T \to (B, \beta)$ and some $p': A \to O$. By Claim 1, $e = h \cdot e'$ for some $h$, which shows that $e$ is $T$-refinable. Since $(F, \delta)$ is an automata presentation, we obtain the desired $\alpha_A$.

Claim 3. $e: TI \to (A, \alpha_A)$ is a syntactic $T$-algebra for $L$.

Proof. The homomorphism $e$ recognizes $L$ via $f_A$: we have $L \cdot e_{T_I} = \text{lin}(L) = f_A \cdot e_A = f_A \cdot e \cdot \epsilon_{T_I}$ and thus $L = f_A \cdot e$ because $e_{T_I}$ is epic. The universal property of $e$ follows from Claim 1.

Since $A$ is finite, we conclude that $\text{lin}(L)$ is regular. \qed